PTO/AIA/123 (08-12)
Approved for use through 11/30/2014. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

CHANGE OF CORRESPONDENCE ADDRESS
Patent

Address to:
Mail Stop Post Issue
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Patent Number	8,572,138 B2
Issue Date	October 29, 2013
Application Number	11/694,483
Filing Date	March 30, 2007
First Named Inventor	Jagane SUNDAR
Attorney Docket Number	063170.9310

Please change the Correspondence Address for the above-identified patent to:

☑ The address associated with Customer Number: 107280

OR

☐ **Firm *or* Individual Name**

Address

City | **State** | **ZIP**

Country

Telephone | **Email**

This form cannot be used to change the data associated with a Customer Number. To change the data associated with an existing Customer Number use "Request for Customer Number Data Change" (PTO/SB/124).

This form will not affect any "fee address" provided for the above-identified patent. To change a "fee address" use the "Fee Address Indication Form" (PTO/SB/47).

I am the:

☐ Patentee.

☐ If the Patentee was not the applicant for patent (37 CFR 1.42), then a Statement under 37 CFR 3.73(c) (Form PTO/AIA/96 or equivalent) is enclosed or was filed on _____. See 37 CFR 3.71.

☑ Attorney or agent of record. Registration Number 45003.

☐ Patent practitioner acting in a representative capacity whose correspondence address is the correspondence address of record. Notice has been given to the patentee or owner. Registration Number 45003.

Signature /luke k. pedersen/

Typed or Printed Name Luke K. PEDERSEN

Date November 21, 2013 | Telephone 202.639.7730

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. Submit multiple forms if more than one signature is required, see below*.

☐ *Total of _____ forms are submitted.

This collection of information is required by 37 CFR 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop Post Issue, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	17466168
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	106095
Filer:	Jenni R. Moen/Janet Daddona
Filer Authorized By:	Jenni R. Moen
Attorney Docket Number:	063170.9310
Receipt Date:	21-NOV-2013
Filing Date:	30-MAR-2007
Time Stamp:	13:58:54
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Change of Address	9310.pdf	206906 cf59fecce7477a49bb3fefd355c92f2a2204a648	no	2

Warnings:	
Information:	

Total Files Size (in bytes):	206906

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	10/29/2013	8572138	063170.9310	1040

106095 7590 10/09/2013

Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 1339 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

Jagane Sundar, Saratoga, CA;
Sanjay Radia, Fremont, CA;
David A. Henseler, Maplewood, MN;

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The USA offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to encourage and facilitate business investment. To learn more about why the USA is the best country in the world to develop technology, manufacture products, and grow your business, visit SelectUSA.gov.

Netflix, Inc. - Ex. 1002, Page 000005
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

106095 7590 08/20/2013
Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

TITLE OF INVENTION: DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	$890	$300	$0	$1190	11/20/2013

EXAMINER	ART UNIT	CLASS-SUBCLASS
RUIZ, ANGELICA	2158	707-104000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- ☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- ☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 Baker Botts, L.L.P.
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: CA, Inc.

(B) RESIDENCE: (CITY and STATE OR COUNTRY): Islandia, New York

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:
- ☒ Issue Fee
- ☒ Publication Fee (No small entity discount permitted)
- ☐ Advance Order - # of Copies .....

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**
- ☐ A check is enclosed.
- ☐ Payment by credit card. Form PTO-2038 is attached.
- ☒ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number 02-0384 (enclose an extra copy of this form).

Netflix, Inc. - Ex. 1002, Page 000006
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

5. **Change in Entity Status** (from status indicated above)

☐ Applicant certifying micro entity status. See 37 CFR 1.29

NOTE: Absent a valid certification of Micro Entity Status (see form PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.

☐ Applicant asserting small entity status. See 37 CFR 1.27

NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.

☒ Applicant changing to regular undiscounted fee status.

NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature /luke k. pedersen/ Date September 23, 2013

Typed or printed name Luke K. PEDERSEN Registration No. 45,003

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Netflix, Inc. - Ex. 1002, Page 000007
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.:	11/694483	Confirmation No.	1040
Applicants:	Jagane SUNDAR		
Filed:	March 30, 2007		
Art Unit:	2158		
Examiner:	Angelica RUIZ		
Docket No.:	063170.9310		
Customer No.	**106095**		

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

RESPONSE TO REASONS FOR ALLOWANCE

Applicants appreciate the Examiner's allowance of Claims 1, 2, 4-21 and 23-28. Pursuant to 37 C.F.R. § 1.104, Applicants respectfully issue a statement commenting on the Examiner's reasons for allowance. Applicants respectfully disagree with the Examiner's reasons for allowance to the extent that they are inconsistent with applicable case law, statutes, and regulations. Furthermore, Applicants do not admit to any characterization or limitation of the claims or to any characterization of a reference by the Examiner, particularly any that are inconsistent with the language of the claims considered in their entirety and including all of their constituent limitations.

Respectfully submitted,
BAKER BOTTS L.L.P.



Attorneys for Applicants
Luke K. Pedersen
Registration No. 45,003

Date: September 23, 2013
CUSTOMER NO. 106095

Netflix, Inc. - Ex. 1002, Page 000008
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Electronic Patent Application Fee Transmittal

Application Number:	11694483
Filing Date:	30-Mar-2007
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Filer:	Jenni R. Moen/Janet Daddona
Attorney Docket Number:	063170.9310

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Utility Appl Issue Fee	1501	1	1780	1780
Publ. Fee- Early, Voluntary, or Normal	1504	1	300	300

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**2080**

Electronic Acknowledgement Receipt

EFS ID:	16923504
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	106095
Filer:	Jenni R. Moen/Janet Daddona
Filer Authorized By:	Jenni R. Moen
Attorney Docket Number:	063170.9310
Receipt Date:	23-SEP-2013
Filing Date:	30-MAR-2007
Time Stamp:	11:44:07
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$2080
RAM confirmation Number	12966
Deposit Account	020384
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)
Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)
Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Issue Fee Payment (PTO-85B)	9310.PDF	147661 ff9da4fa82dc9c2be180f6e90e86c2e40ceec524	no	2
Warnings:					
Information:					
2	Post Allowance Communication - Incoming	9310i.PDF	40420 6efc686c83bf90429249a11b022582c8bd4d7408	no	1
Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	32113 970826a3b5ae888100121458 67d5acedb879083f	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	220194		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

106095 7590 08/20/2013

Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

EXAMINER
RUIZ, ANGELICA

ART UNIT	PAPER NUMBER
2158	

DATE MAILED: 08/20/2013

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

TITLE OF INVENTION: DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	$890	$300	$0	$1190	11/20/2013

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the ENTITY STATUS shown above. If the ENTITY STATUS is shown as SMALL or MICRO, verify whether entitlement to that entity status still applies.

If the ENTITY STATUS is the same as shown above, pay the TOTAL FEE(S) DUE shown above.

If the ENTITY STATUS is changed from that shown above, on PART B - FEE(S) TRANSMITTAL, complete section number 5 titled "Change in Entity Status (from status indicated above)".

For purposes of this notice, small entity fees are 1/2 the amount of undiscounted fees, and micro entity fees are 1/2 the amount of small entity fees.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PTOL-85 (Rev. 02/11)

Netflix, Inc. - Ex. 1002, Page 000013
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

106095 7590 08/20/2013
Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

TITLE OF INVENTION: DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	SMALL	$890	$300	$0	$1190	11/20/2013

EXAMINER	ART UNIT	CLASS-SUBCLASS
RUIZ, ANGELICA	2158	707-104000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____
2 _____
3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:

❑ Issue Fee

❑ Publication Fee (No small entity discount permitted)

❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

❑ A check is enclosed.

❑ Payment by credit card. Form PTO-2038 is attached.

❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

PTOL-85 (Rev. 02/11)

Netflix, Inc. - Ex. 1002, Page 000014
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

5. **Change in Entity Status** (from status indicated above)

❑ Applicant certifying micro entity status. See 37 CFR 1.29	NOTE: Absent a valid certification of Micro Entity Status (see form PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.
❑ Applicant asserting small entity status. See 37 CFR 1.27	NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.
❑ Applicant changing to regular undiscounted fee status.	NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Netflix, Inc. - Ex. 1002, Page 000015
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

106095 7590 08/20/2013

Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

DATE MAILED: 08/20/2013

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 1020 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 1020 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Netflix, Inc. - Ex. 1002, Page 000016
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No. 11/694,483	Applicant(s) SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	Art Unit 2158	AIA (First Inventor to File) Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *RCE Filed 7/31/2013*.
   ☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on_____.
2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.
3. ☒ The allowed claim(s) is/are *1, 2, 4-21, and 23-28 (Renumbered 1-26)*. As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov .
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   Certified copies:

   a) ☐ All b) ☐ Some *c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
4. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
5. ☐ Examiner's Amendment/Comment
6. ☒ Examiner's Statement of Reasons for Allowance
7. ☐ Other _____.

/Angelica Ruiz/
Primary Examiner, Art Unit 2158

Netflix, Inc. - Ex. 1002, Page 000018
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

DETAILED ACTION

1. This office action is in response to applicant's request for continued examination filed on July 31 2013, in which claims 1-2, 4-21, and 23-28 are presented for further examination.

Reasons for Allowance

2. The following is an examiner's statement of reasons for allowance: the prior art of record does not appear to teach or suggest the subject matter of independent claims 1, 20, and 28. The dependent claims 2, 4-19, 21, and 23-27 further limit the independent claims and are therefore allowed as well (Claims renumbered 1-26).

An updated search for the prior art in EAST database and on domains (Google Scholar and Google Patents) has been conducted. The prior art searched and examined in the database and domains do not fairly teach or suggest the limitations of the newly amended claimed subject matter.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

3. Any inquiry concerning this communication or earlier communications from the examiner should be directed to ANGELICA RUIZ whose telephone number is (571)270-3158. The examiner can normally be reached on 8:00 a.m. to 4:30 p.m., ET.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jacob Betit can be reached on (571) 272-4105. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Angelica Ruiz/
Primary Examiner, Art Unit 2158

Notice of References Cited	Application/Control No. 11/694,483	Applicant(s)/Patent Under Reexamination SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	Art Unit 2158	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2003/0192035 A1	10-2003	Duesterwald ald, Evelyn	717/138
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Netflix, Inc. - Ex. 1002, Page 000021
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	4	(US-20030135658-$ or US-20060259292-$ or US-20050138370-$).did. or (US-20050138370-$).did.	US-PGPUB; DERWENT	OR	ON	2013/08/12 11:31
L2	16	(emulat$4 with platform) same (virtual near4 machine).ab.	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:31
L3	2	L2 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:31
L4	2	L3 and (emulat$4 with platform) same (virtual near4 machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:31
L5	56	(virtual near2 machine) same application same node same (control$5) and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L6	24	L5 and (software near3 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L7	1	L6 and (node same x86)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L8	15386	(server same software same application) and operating near4 system and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L9	3216	L8 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L10	271	L9 and (control$4 near4 node)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L11	201	L10 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L12	131	L11 and (server near4 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L13	2	"20050005200"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L14	15386	(server same software same application) and operating near4 system and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L15	3216	L14 and @py<="2006"	US-PGPUB; USPAT;	OR	ON	2013/08/12 11:32

Netflix, Inc. - Ex. 1002, Page 000022
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

			IBM_TDB			
L16	271	L15 and (control$4 near4 node)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L17	201	L16 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L18	131	L17 and (server near4 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L19	30	L18 and "707"/$.ccls.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L20	125	"707"/$.ccls. and (enterprise near4 comput$5) same (tier hierarchical)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L21	35	L20 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L22	8	L21 and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:32
L24	536	(emulat$4 with platform) same (virtual near4 machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:32
L25	101	L24 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:32
L26	45	L25 and (replica image instance) same virtual with machine	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:32
L27	12	L26 and (node)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:32
L28	16	(emulat$4 with platform) same (virtual near4 machine).ab.	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:32
L29	2	L2 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:32
L30	2	L3 and (emulat$4 with platform) same (virtual near4 machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:32
L39	29	(emulat$4 with platform) same (virtual near4 machine).clm.	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:35
L40	4	39 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/08/12 11:36

Netflix, Inc. - Ex. 1002, Page 000023
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

L41	4	40 and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:36
L42	1	41 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/08/12 11:36

EAST Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L31	4175	718/1.ccls.	US-PGPUB; USPAT; UPAD	OR	ON	2013/08/12 11:33
L32	526	31 and @py<="2006"	US-PGPUB; USPAT; UPAD	OR	ON	2013/08/12 11:33
L34	359	32 and (virtual near4 machine)	US-PGPUB; USPAT; UPAD	OR	ON	2013/08/12 11:34
L36	1	34 and ((emulat$4 with platform) same (virtual near4 machine)).clm.	US-PGPUB; USPAT; UPAD	OR	ON	2013/08/12 11:34
L37	29	((emulat$4 with platform) same (virtual near4 machine)).clm.	US-PGPUB; USPAT; UPAD	OR	ON	2013/08/12 11:35
L38	4	37 and @py<="2006"	US-PGPUB; USPAT; UPAD	OR	ON	2013/08/12 11:35

8/12/2013 11:37:23 AM
C:\Users\aruiz1\Documents\EAST\Workspaces\11694483_RCE.wsp

Netflix, Inc. - Ex. 1002, Page 000024
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Web Images More... Sign in

Google virtual machine automation infrastructure image repository emulates "computer platform" software image "operating system" "application nodes"

Scholar 1 result (0.05 sec)

Articles
Legal documents

Any time
Since 2013
Since 2012
Since 2009
Custom range...

Sort by relevance
Sort by date

include patents
include citations

Create alert

DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF **VIRTUAL MACHINE** DISK **IMAGES**

J Sundar, S Radia, DA Henseler - US Patent App. 11/694,483, 2007 - Google Patents

... of the **image** instances of the **software** applica- tions on the **virtual machines**. **...** I CONFIGURE **VIRTUAL MACHINE** MANAGER WITH DESIRED NUMBER OF **VM** INSTANCES I **...** A NODE THAT IS RUNNING **VIRTUAL MACHINE** MANAGER CREATE A **VIRTUAL MACHINE** ON THE **...**

Cited by 15 Related articles All 2 versions Cite

Create alert

About Google Scholar All About Google Privacy & Terms Give us feedback

+You Search Images Maps Play YouTube News Gmail More Sign in

Search

Web Images Maps Shopping Patents More Search tools

Before Mar 30, 2006 Any Patent Office Any filing status Any patent type Sorted by relevance Clear

Method and apparatus for storing policies for policy-based ...



www.google.com/patents/US6718380

Grant - Filed Aug 8, 2002 - Issued Apr 6, 2004 - Shai Mohaban - Cisco Technology, Inc.

Policy statements are stored in a **Repository**, which may be a Directory Server or .... requesting an **operating system** function to modify a packet of the traffic flows ...

Overview - Related - Discuss

Advanced search Search Help Send feedback

Google Home Advertising Programs Business Solutions Privacy & Terms
About Google

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	ANGELICA RUIZ	2158

| CPC | | |

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	ANGELICA RUIZ	2158

US ORIGINAL CLASSIFICATION	
CLASS	**SUBCLASS**
707	828

CROSS REFERENCE(S)

CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				
707	831				
718	1				

INTERNATIONAL CLASSIFICATION					
CLAIMED					**NON-CLAIMED**
G	0	6	F	12 / 00 (2006.01.01)	
G	0	6	F	17 / 30 (2006.01.01)	

NONE		**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	26	
/ANGELICA RUIZ/ Primary Examiner.Art Unit 2158	08/12/2013	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Netflix, Inc. - Ex. 1002, Page 000028
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	ANGELICA RUIZ	2158

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	16	17												
2	2	17	18												
	3	18	19												
3	4	19	20												
4	5	20	21												
5	6		22												
6	7	21	23												
7	8	22	24												
8	9	23	25												
9	10	24	26												
10	11	25	27												
11	12	26	28												
12	13														
13	14														
14	15														
15	16														

NONE (Assistant Examiner)	(Date)	**Total Claims Allowed:** 26	
/ANGELICA RUIZ/ Primary Examiner.Art Unit 2158 (Primary Examiner)	08/12/2013 (Date)	O.G. Print Claim(s) 1	O.G. Print Figure 1

U.S. Patent and Trademark Office Part of Paper No. 20130811

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	Art Unit
	ANGELICA RUIZ	2158

CPC- SEARCHED		
Symbol	Date	Examiner

CPC COMBINATION SETS - SEARCHED		
Symbol	Date	Examiner

US CLASSIFICATION SEARCHED			
Class	Subclass	Date	Examiner
707	600-831, 899, and 999.001-999.206, by keyword	8/26/2010	A. Ruiz
709	223-227, by keyword	8/26/2010	A. Ruiz
711	6, by keyword	8/26/2010	A. Ruiz
719	315 -324, by keyword	8/26/2010	A. Ruiz

SEARCH NOTES		
Search Notes	Date	Examiner
Consulted Rehana Perveen for (QAS/WQAS) for 101 issues	8/26/2010	A. Ruiz
See ACM Portal Search strategy, attached	8/26/2010	A. Ruiz
See EAST Search history, including inventor	8/26/2010	A. Ruiz
See Google scholar including patents strategy, attached	8/26/2010	A. Ruiz
See EAST Search history updated, attached	2/9/2011	A. Ruiz
See EAST Search history updated, attached	9/21/2012	A. Ruiz
See Google scholar including patents strategy, attached	9/21/2012	A. Ruiz
See EAST Search history updated, attached	5/6/2013	A. Ruiz
See EAST Search history updated, attached	8/12/2013	A. Ruiz
See Google Scholar Search, attached	8/12/2013	A. Ruiz
See Google Patents search, attached	8/12/2013	A. Ruiz

INTERFERENCE SEARCH

	/ANGELICA RUIZ/ Primary Examiner.Art Unit 2158

US Class/ CPC Symbol	US Subclass / CPC Group	Date	Examiner
718	1, by keyword (See PGPub)	8/12/2013	A. Ruiz

	/ANGELICA RUIZ/ Primary Examiner.Art Unit 2158

Netflix, Inc. - Ex. 1002, Page 000031
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

vPTO/SB/30 (10-01)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL

Address to:
Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	11/694,483
Filing Date	March 30, 2007
First Named Inventor	Jagane Sundar et al.
Art Unit	2158
Examiner Name	Angelica Ruiz
Attorney Docket Number	063170.9310

This is a Request for Continued Examination (RCE) under 37 C.F.R. 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. Submission required under 37 C.F.R. § 1.114
   - a. ☐ Previously submitted
     - i. ☐ Consider the amendment(s)/reply under 37 C.F.R. 1.116 previously filed on _____ (Any unentered amendment(s) referred to above will be entered).
     - ii. ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____
     - iii. ☐ Other _____
   - b. ☒ **Enclosed**
     - i. ☒ **Response Accompanying RCE**
     - ii. ☐ Affidavit(s)/Declaration(s)
     - iii. ☐ Information Disclosure Statement (IDS)
2. Miscellaneous
   - a. ☐ Suspension of action on the above-identified application is requested under 37 C.F.R. 1.103(c) for a period of _____ months. (Period of suspension shall not exceed 3 months; Fee under 37 C.F.R. 1.17(i) required)
   - b. ☐ Other _____
3. Fees The RCE fee under 37 C.F.R. 1.17(e) is required by 37 C.F.R 1.114 when the RCE is filed.
   - a. ☒ **The Director is hereby authorized to charge any fees or credit any overpayments to Deposit Account No. 02-0384**
     - i. ☒ **RCE fee required under 37 C.F.R. § 1.17(e) in the amount of $1,700.00**
     - ii. ☐ _____ Month Extension of time fee (37 C.F.R. 1.136 and 1.17) in the amount of $_____
     - iii. ☐ Additional Claim Fees in the amount of $_____
   - b. ☐ Check in the amount of $ _____ (RCE Fee) is enclosed.
   - c. ☐ Payment by credit card (Form PTO-2038 enclosed)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Name (Print / Type)	Roshan S. Mansinghani	Registration No. (Attorney/Agent)	62,429
Signature		Date	07/31/13

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND Fees and Completed Forms to the following address: Assistant Commissioner for Patents, Box RCE, Washington, DC 20231.

Netflix, Inc. - Ex. 1002, Page 000032
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.:	11/694,483	Confirmation No.	1040
Applicant:	Jagane Sundar et al.		
Filed:	March 30, 2007		
Art Unit:	2158		
Examiner:	Angelica Ruiz		
Docket No.:	CAS-013		
Customer No.	106095		

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

RESPONSE ACCOMPANYING REQUEST FOR CONTINUED EXAMINATION

This paper is being filed in response to the final Office Action mailed May 10, 2013 ("Final Office Action"). Reconsideration and reexamination are respectfully requested in light of the amendments and remarks below.

Amendments to the Claims are reflected in the listing of claims which begins on page 2 of this paper.

Remarks/Arguments begin on page 9 of this paper.

In the Claims

1. (Currently Amended) A distributed computing system comprising:

a software image repository comprising non-transitory, computer-readable media operable to store: (i) a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide a plurality of virtual machines, each of the plurality of virtual machine operable to provide an environment that emulates a computer platform, and (ii) a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines; and

a control node that comprises an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes by causing the plurality of image instances of the virtual machine manager to be copied from the software image repository to the application nodes and to provide autonomic deployment of the plurality of image instances of the software applications on the virtual machines by causing the plurality of image instances of the software applications to be copied from the software image repository to the application nodes.

2. (Previously Presented) The distributed computing system of claim 1, wherein the image instances of the plurality of software applications comprise virtual disk image files.

3. (Canceled)

4. (Previously Presented) The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises an operating system.

5. (Previously Presented) The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises a server application.

6. (Original) The distributed computing system of claim 1, wherein a first one of the virtual machines provides an environment that emulates an x86 platform.

7. (Previously Presented) The distributed computing system of claim 1, wherein the automation infrastructure captures a first image instance of the image instances of the virtual machine manager from a first application node of the plurality of application nodes.

8. (Previously Presented) The distributed computing system of claim 7, wherein the first application node executes the virtual machine manager.

9. (Previously Presented) The distributed computing system of claim 1, wherein the control node maintains the plurality of image instances of the virtual machine manager and the plurality of image instances of the plurality of software applications as persistent or non-persistent based on a configuration.

10. (Original) The distributed computing system of claim 1, wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.

11. (Previously Presented) The distributed computing system of claim 10, wherein the one or more rule engines are forward-chaining rule engines.

12. (Previously Presented) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the one or more rules engines to autonomically control the deployment of the software applications to the application nodes in accordance with a current state of an application matrix.

13. (Original) The distributed computing system of claim 1, wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.

14. (Original) The distributed computing system of claim 13, further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.

15. (Previously Presented) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the one or more rules engines to monitor the execution of the software applications when deployed to the application nodes in accordance with a current state of the application matrix.

16. (Previously Presented) The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles rules into one or more discrimination networks.

17. (Previously Presented) The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein one or more rule engines analyze the status data from the monitoring subsystem and apply a set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

18. (Previously Presented) The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein status data is collected based on the tiers.

19. (Previously Presented) The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein one or more rule engines automatically produce action requests to control the deployment of the software applications in accordance with the hierarchical organization of the distributed computing system defined by the model.

20. (Currently Amended) A method comprising:

storing a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide a plurality of virtual machines, each virtual machine operable to provide an environment that emulates a computer platform;

storing a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines;

executing a rules engine to perform operations to autonomically deploy the image instances of the virtual machine manager on the application nodes by causing the image instances of the virtual machine manager to be copied to the application nodes; and

executing the rules engine to perform operations to autonomically deploy the image instances of the software applications on the virtual machines by causing the image instances of the software applications to be copied to the application nodes.

21. (Previously Presented) The method of claim 20, wherein storing the image instances of the plurality of software applications comprises storing virtual disk image files.

22. (Canceled)

23. (Previously Presented) The method of claim 20, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises an operating system.

24. (Previously Presented) The method of claim 20,

further comprising generating an application matrix that specifies data for controlling the deployment of the software applications within the distributed computing system; and

wherein performing operations to provide autonomic control comprises providing autonomic control over the deployment, undeployment, and execution of the software applications on the virtual machines within the distributed computing system in accordance with parameters of the application matrix.

25. (Previously Presented) The method of claim 20, wherein performing operations to autonomically deploy the image instances of the software applications on the virtual machines comprises:

receiving status data for the distributed computing system, wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the application nodes; and

performing the operations to provide autonomic control over the deployment of the software applications on the virtual machines within the distributed computing system in accordance with the rules.

26. (Previously Presented) The method of claim 20, further comprising accessing parameters of an application matrix and determining one or more rules for use in autonomically controlling the deployment of the software applications on the virtual machines.

27. (Previously Presented) The method of claim 26, further comprising processing the one or more rules with a plurality of forward-chaining rule engines that compile the one or more rules into one or more discrimination networks.

28. (Currently Amended) A non-transitory, computer-readable medium comprising instructions stored thereon, that when executed by a programmable processor:

store a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide a plurality of virtual machines, each virtual machine operable to provide an environment that emulates a computer platform;

store a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines;

perform operations to autonomically deploy the image instances of the virtual machine manager on the application nodes by causing the image instances of the virtual machine manager to be copied to the application nodes; and

perform operations to autonomically deploy the image instances of the software applications on the virtual machines by causing the image instances of the software applications to be copied to the application nodes.

REMARKS

This Application has been carefully reviewed in light of the final Office Action. Claims 1-28 are pending and rejected. Applicants amend Claims 1, 20, and 28. Applicants cancel Claims 3 and 22. Applicants respectfully request reconsideration and favorable action in this case.

Summary of Interview

Applicants thank the Examiner for the telephonic interviews held with Applicants' attorney Roshan S. Mansinghani (Reg. No. 62,429) on July 9, 2013 and July 25, 2013. During the interviews, the rejection of Claim 1 was discussed. The Examiner, during the telephonic interview dated July 25, 2013, agreed that the present amendments would overcome the rejection of Claim 1.

Request for Interview

If the Examiner intends to issue a new Office Action in response to Applicants' present submission, Applicants respectfully request that the Examiner contact Applicants' attorney prior to issuing the new Office Action to discuss a possible resolution to any outstanding issues.

Rejections under 35 U.S.C. § 103

The Examiner rejects Claims 1-16, 18-24, and 26-28 under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent Publication No. 2005/0005200 by Matena et al. ("*Matena*") in view of U.S. Patent Publication No. 2003/0110173 by Marsland et al. ("*Marsland*"). Claims 17 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Matena* in view of *Marsland,* and further in view of U.S. Patent Publication No. 2004/0181794 by Coleman et al. ("*Coleman*"). For at least the reasons provided below, Applicants respectfully traverse these rejections.

At a minimum, the cited portions of the proposed *Matena-Marsland* combination do not disclose, teach, or suggest the following limitations of amended Claim 1 which Applicant discusses as an example:

> a control node that comprises an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes by causing the plurality of image instances of the virtual machine manager to be copied from the software image repository to the application nodes and to provide autonomic deployment of the plurality of image instances of the software applications on the virtual machines by causing the plurality of image instances of the software applications to be copied from the software image repository to the application nodes.

The Examiner has cited to, inter alia, ¶¶ 81, 193, 195 of *Matena*:

> The disclosure further describes how an application controller controls remote invocation and creation of server objects located in the execution models. [¶ 81]
>
> JAC manages how the containers are distributed over the nodes of the distributed system and how the application execution modules are distributed over the containers. . . . The container 2102 is a process that includes a Java Virtual Machine ("JVM") 2104 and application server classes 2105. The application server classes 2105 include Java classes that implement the operations that JAC uses to manage the lifecycle of the execution models (such as the classes that implement the creation of execution models) and Java classes that are provided as runtime library classes for the execution models (such as classes that implement the Enterprise JavaBeans container functions). [¶ 193]
>
> An application definition 2201 includes Java classes 2202 and deployment descriptors 2203. The Java classes 2202 include the program instructions of the application's execution modules. The deployment descriptors 2203 include the definitions of one or more execution module ("EM definitions") 2204. An EM definition 2204 is a template for one or more execution modules created from the EM definition 2204. In one embodiment of the invention, the distribution policies specify how many instances of the execution module should be created, the name of the container group in which the execution modules should be created, and how the execution modules should be distributed over the containers. [¶ 195]

Matena further discloses at ¶ 233:

> A system management tool sends the "Start Application" message to JAC 2501. In response to the message, JAC reads the application definition files and creates the Application object 2601, EM definition object 2602, and EM Distribution Policy object 2603 in its state model 2502. The exemplary EM Distribution Policy is the EMContainerDistribution(2) distribution policy object, which indicates that there shall be two execution modules in each container. The EM definition included in the application definition may specify the name of the container group to associate with the execution modules.

That is, the cited portions of *Matena* disclose that a JAC manages the creation of execution modules (i.e., Java objects) from Java classes as well as the creation of containers in which the execution modules are to operate. Figure 26 of *Matena* further discloses that the JAC causes the operation of execution modules on nodes by sending commands to create execution modules to the nodes. The nodes send messages to the JAC indicating whether the attempt to create the execution modules was successful. *See Matena* at ¶ 239. As the Examiner agreed to during the interview, this does not disclose, teach, or suggest the following limitations recited in amended Claim 1:

> an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes by causing the plurality of image instances of the virtual machine manager to be copied from the software image repository to the application nodes and to provide autonomic deployment of the plurality of image instances of the software applications on the virtual machines by causing the plurality of image instances of the software applications to be copied from the software image repository to the application nodes.

For at least these reasons, independent Claim 1 and its dependent claims should be reconsidered and allowed. For analogous reasons, independent Claims 20 and 28, as well as the claims which depend from Claim 20, should be reconsidered and allowed.

No Waiver

Applicants' arguments are made without prejudice or disclaimer. By not responding to additional statements made by the Examiner, Applicants do not acquiesce to the additional statements. The distinctions between the applied references and the claims are provided as examples only and are sufficient to overcome the rejections. Applicants reserve the right to discuss additional or other distinctions in a later response or on appeal, if appropriate.

Request for Evidentiary Support

If the Examiner is relying upon "common knowledge" or "well known" principles or "Official Notice" or other information within the Examiner's personal knowledge to establish a rejection, Applicants respectfully request that the Examiner cite a reference or provide an affidavit in support of the position in accordance with M.P.E.P. § 2144.03 and 37 C.F.R. 1.104(d)(2).

CONCLUSION

Applicants have made an earnest attempt to place this case in condition for allowance. For at least the foregoing reasons, Applicants respectfully request full allowance of all the pending claims.

If the Examiner believes a telephone conference would advance prosecution of this case in any way, the Examiner is invited to contact Roshan S. Mansinghani, the Attorney for Applicants, at the Examiner's convenience at (214) 953-6737.

As indicated on the accompanying RCE Transmittal form, the Commissioner is authorized to charge the amount of $1,700.00 (for the RCE fee) to Deposit Account No. 02-0384 of BAKER BOTTS L.L.P. Although Applicants believe no other fees are due, the Commissioner is authorized to charge any necessary additional fees and credit any overpayments to Deposit Account No. 02-0384 of BAKER BOTTS L.L.P.

Respectfully submitted,



Roshan S. Mansinghani
Attorney for Applicants
Reg. No. 62,429

Date: 07/31/13

Customer No: 106095

This paper or fee is being filed using the USPTO's electronic filing system EFS-Web, and is addressed to the Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Electronic Patent Application Fee Transmittal

Application Number:	11694483
Filing Date:	30-Mar-2007
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Filer:	Roshan Suresh Mansinghani/Laurie Scott
Attorney Docket Number:	063170.9310

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
RCE - 2nd and Subsequent Request	1820	1	1700	1700
		Total in USD ($)		**1700**

Electronic Acknowledgement Receipt

EFS ID:	16463752
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	106095
Filer:	Roshan Suresh Mansinghani/Laurie Scott
Filer Authorized By:	Roshan Suresh Mansinghani
Attorney Docket Number:	063170.9310
Receipt Date:	31-JUL-2013
Filing Date:	30-MAR-2007
Time Stamp:	12:57:41
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$1700
RAM confirmation Number	11121
Deposit Account	020384
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Continued Examination (RCE)	0631709310RCETrans073113.PDF	98075 cbf1b0241fce5693ffda1454476ef0e1b9dd66b9	no	1
Warnings:					
This is not a USPTO supplied RCE SB30 form.					
Information:					
2		0631709310RCEAmend073113.PDF	552669 a26ff567fab7107ed92c68e2770ef9bc4a62aeea	yes	12

Multipart Description/PDF files in .zip description

Document Description	Start	End
Amendment Submitted/Entered with Filing of CPA/RCE	1	1
Claims	2	8
Applicant Arguments/Remarks Made in an Amendment	9	12

Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	30377 e964772dd18796de13cd63e280c7f4c49e6054d3	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	681121		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (09-11)
Approved for use through 1/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 11/694,483	Filing Date 03/30/2007	☐ To be Mailed

ENTITY: ☐ LARGE ☒ SMALL ☐ MICRO

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 = *		X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 = *		X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).			
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))				
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 07/31/2013	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 26	Minus	** 28	= 0	X $40 =	0
Independent (37 CFR 1.16(h))	* 3	Minus	***3	= 0	X $210 =	0
☐ Application Size Fee (37 CFR 1.16(s))						
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
					TOTAL ADD'L FEE	0

AMENDMENT	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))						
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
					TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

LIE
/PAMELA YOUNG/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**
If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

106095 7590 07/19/2013

Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

NOTIFICATION DATE	DELIVERY MODE
07/19/2013	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOmail1@bakerbotts.com

Netflix, Inc. - Ex. 1002, Page 000050
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Applicant-Initiated Interview Summary	**Application No.** 11/694,483	**Applicant(s)** SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	**Art Unit** 2158	

All participants (applicant, applicant's representative, PTO personnel):

(1) *ANGELICA RUIZ*. (3)_____.

(2) *ROSHAN S. MANSINGHANI (Reg. No. 62,429)*. (4)_____.

Date of Interview: *09 July 2013*.

Type: ☒ Telephonic ☐ Video Conference
☐ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☒ No.
If Yes, brief description: _____.

Issues Discussed ☐101 ☐112 ☐102 ☒103 ☐Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: *1*.

Identification of prior art discussed: *Matena et al (US 2005/0005200) and Mrsland et al (US 20030110173)*.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant's representative explained the invention extensively. The Examiner explained her interpretation of the claim language with respect to the prior art. The Examiner suggested further amending the "image instance" in order to reflect what Applicant's representative explained as the invention. The Examiner also suggested further amending, as discussed, the functionality of the mentioned execution in order to differentiate from the prior art of record. No agreement was reached. Arguments have been taken into consideration. .

Applicant recordation instructions: The formal written reply to the last Office action must include the substance of the interview. (See MPEP section 713.04). If a reply to the last Office action has already been filed, applicant is given a non-extendable period of the longer of one month or thirty days from this interview date, or the mailing date of this interview summary form, whichever is later, to file a statement of the substance of the interview

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☐ Attachment

/Angelica Ruiz/ Primary Examiner, Art Unit 2158	

Netflix, Inc. - Ex. 1002, Page 000051
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Summary of Record of Interview Requirements

Manual of Patent Examining Procedure (MPEP), Section 713.04, Substance of Interview Must be Made of Record
A complete written statement as to the substance of any face-to-face, video conference, or telephone interview with regard to an application must be made of record in the application whether or not an agreement with the examiner was reached at the interview.

Title 37 Code of Federal Regulations (CFR) § 1.133 Interviews
Paragraph (b)

In every instance where reconsideration is requested in view of an interview with an examiner, a complete written statement of the reasons presented at the interview as warranting favorable action must be filed by the applicant. An interview does not remove the necessity for reply to Office action as specified in §§ 1.111, 1.135. (35 U.S.C. 132)

37 CFR §1.2 Business to be transacted in writing.
All business with the Patent or Trademark Office should be transacted in writing. The personal attendance of applicants or their attorneys or agents at the Patent and Trademark Office is unnecessary. The action of the Patent and Trademark Office will be based exclusively on the written record in the Office. No attention will be paid to any alleged oral promise, stipulation, or understanding in relation to which there is disagreement or doubt.

———

The action of the Patent and Trademark Office cannot be based exclusively on the written record in the Office if that record is itself incomplete through the failure to record the substance of interviews.

It is the responsibility of the applicant or the attorney or agent to make the substance of an interview of record in the application file, unless the examiner indicates he or she will do so. It is the examiner's responsibility to see that such a record is made and to correct material inaccuracies which bear directly on the question of patentability.

Examiners must complete an Interview Summary Form for each interview held where a matter of substance has been discussed during the interview by checking the appropriate boxes and filling in the blanks. Discussions regarding only procedural matters, directed solely to restriction requirements for which interview recordation is otherwise provided for in Section 812.01 of the Manual of Patent Examining Procedure, or pointing out typographical errors or unreadable script in Office actions or the like, are excluded from the interview recordation procedures below. Where the substance of an interview is completely recorded in an Examiners Amendment, no separate Interview Summary Record is required.

The Interview Summary Form shall be given an appropriate Paper No., placed in the right hand portion of the file, and listed on the "Contents" section of the file wrapper. In a personal interview, a duplicate of the Form is given to the applicant (or attorney or agent) at the conclusion of the interview. In the case of a telephone or video-conference interview, the copy is mailed to the applicant's correspondence address either with or prior to the next official communication. If additional correspondence from the examiner is not likely before an allowance or if other circumstances dictate, the Form should be mailed promptly after the interview rather than with the next official communication.

The Form provides for recordation of the following information:
- Application Number (Series Code and Serial Number)
- Name of applicant
- Name of examiner
- Date of interview
- Type of interview (telephonic, video-conference, or personal)
- Name of participant(s) (applicant, attorney or agent, examiner, other PTO personnel, etc.)
- An indication whether or not an exhibit was shown or a demonstration conducted
- An identification of the specific prior art discussed
- An indication whether an agreement was reached and if so, a description of the general nature of the agreement (may be by attachment of a copy of amendments or claims agreed as being allowable). Note: Agreement as to allowability is tentative and does not restrict further action by the examiner to the contrary.
- The signature of the examiner who conducted the interview (if Form is not an attachment to a signed Office action)

It is desirable that the examiner orally remind the applicant of his or her obligation to record the substance of the interview of each case. It should be noted, however, that the Interview Summary Form will not normally be considered a complete and proper recordation of the interview unless it includes, or is supplemented by the applicant or the examiner to include, all of the applicable items required below concerning the substance of the interview.

A complete and proper recordation of the substance of any interview should include at least the following applicable items:
1) A brief description of the nature of any exhibit shown or any demonstration conducted,
2) an identification of the claims discussed,
3) an identification of the specific prior art discussed,
4) an identification of the principal proposed amendments of a substantive nature discussed, unless these are already described on the Interview Summary Form completed by the Examiner,
5) a brief identification of the general thrust of the principal arguments presented to the examiner,
   (The identification of arguments need not be lengthy or elaborate. A verbatim or highly detailed description of the arguments is not required. The identification of the arguments is sufficient if the general nature or thrust of the principal arguments made to the examiner can be understood in the context of the application file. Of course, the applicant may desire to emphasize and fully describe those arguments which he or she feels were or might be persuasive to the examiner.)
6) a general indication of any other pertinent matters discussed, and
7) if appropriate, the general results or outcome of the interview unless already described in the Interview Summary Form completed by the examiner.

Examiners are expected to carefully review the applicant's record of the substance of an interview. If the record is not complete and accurate, the examiner will give the applicant an extendable one month time period to correct the record.

Examiner to Check for Accuracy

If the claims are allowable for other reasons of record, the examiner should send a letter setting forth the examiner's version of the statement attributed to him or her. If the record is complete and accurate, the examiner should place the indication, "Interview Record OK" on the paper recording the substance of the interview along with the date and the examiner's initials.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

106095 7590 05/10/2013

Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

NOTIFICATION DATE	DELIVERY MODE
05/10/2013	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOmail1@bakerbotts.com

Netflix, Inc. - Ex. 1002, Page 000053
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Office Action Summary	Application No.	Applicant(s)	
	11/694,483	SUNDAR ET AL.	
	Examiner	Art Unit	
	ANGELICA RUIZ	2158	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 January 2013*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

5)☒ Claim(s) *1-28* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-28* is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

3) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

4) ☐ Other: _____.

DETAILED ACTION

1. The Action is responsive to Applicant's amendment, filed on January 17, 2013.

2. It is acknowledged that as a result of the amendment, Claims 1-5, 7-9, 11-12, and 15-28 have been amended.

3. Claims 1-28 are pending.

Response to Arguments

4. Applicant's arguments with respect to claims 1-28 have been considered but are moot in view of the new grounds of rejection necessitated by Applicant's amendment of the claims.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims **1 – 16, 18 –24 and 26 –28** are rejected under 35 U.S.C. 103(a) as being unpatentable over Matena et al (US 2005/0005200 A1), in view of Marsland et al (US 2003/0110173), hereinafter "Matena" and "Marsland" respectively.

As per Claim 1, Matena discloses:

A distributed computing system comprising: (Title, "Method and apparatus for executing applications on a distributed computer system") and (Abstract, "multiple nodes interconnected through a network…") ***a software image repository comprising non-transitory, computer-readable media operable to store:*** (Par [0015] and (Par

[0081], "…creation of execution modules to store new server objects.") ***(i) a plurality image instances of a virtual machine manager that is executable on a plurality of application nodes,*** (Par [0090], "Java classes" being the "virtual machine" executing on the application nodes, as claimed), ***wherein when executed on the applications nodes, the image instances of the virtual machine manager provide a plurality virtual machines,*** (Par [0193], includes the instance of the "virtual machine" as claimed***) and (ii) a plurality of image instances of a plurality of software applications that are executable on the virtual machines;*** (Par [0015], "The application controllers utilize the execution controller. To its users, the execution *control system provides a single-system image of the distributed computer system and applications running on it…*"; Par [0195] and see also Figures 21-22) ***and a control node*** (Par [0015], "one or more node controllers, and one or more application controllers….") ***that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes*** (Par [0085], "The node controllers 103, execution controller 102 and application controllers are distributed over the nodes of a distributed computer system."; see figure 2) ***and to provide autonomic deployment of the image instances of the software applications on the virtual machines.*** (Par [0089]; par [0112], "can automatically distribute application modules of multiple applications to the node…"; par [0113], "… *same application could be deployed on several distributed computer systems, each having a different configuration, and each possibly requiring different*

assignment of application module replicas to nodes.", the "replicas" being the "images" as claimed).

Matena does not specifically discloses the "plurality" and the ***"each virtual machine operable to provide an environment that emulates a computer platform"***

Marsland discloses the mentioned limitations as follows:

(Par [0032], "virtualized platforms, the virtual machine executive 205 also includes a computer system interface 209. The computer system interface 209 allows the virtual machine executive to communicate with the actual physical hardware. In one implementation, the computer system interface 209 can include device drivers for communicating with various peripherals in the computer system 130. In another implementation, the computer system interface 209 may not include device drivers and may merely map instructions issued by a virtual machine device driver to a computer system device. One tool for mapping instructions is the memory mapper 213. Another implementation translates instructions issued on a mapped virtual device into a different set of instructions issued on a physical device to enable complete emulation of physical device…").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Marsland into the method of Matena to take advantage of using a plurality of images and a virtual machine to emulate a computer platform. The modification would have been obvious because one of the ordinary skills in the art would implement emulating the platform to allocate system resources (See Marsland par [0030])

As per Claim 2, the rejection of Claim 1 is incorporated and further Marsland discloses: ***wherein the image instances of the plurality software applications comprise virtual disk image files.*** (Par [0033], "…*The virtual bus 215 can be used to allow each virtual machine to act as a diskful system*.").

As per Claim 3, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein when the automation infrastructure deploys a first image instance of the plurality of image instances of the plurality of software applications, the automation infrastructure copies the first image instance to a first one of the application nodes.*** (Par [0251] and Par [0289], "…A purpose of the replication constraint is *to specify that if one replica of an execution module is assigned to a node that is included in the first node group, then the other replica of the same execution module should be assigned to a node included in the second node group*.").
Matena does not specifically disclose the "plurality" *Marsland discloses the mentioned limitations as follows:*
(Par [0032]).
Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Marsland into the method of Matena to take advantage of using a plurality of images and a virtual machine to emulate a computer platform. The modification would have been obvious because one

of the ordinary skills in the art would implement emulating the platform to allocate system resources (See Marsland par [0030])

As per Claim 4, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein a first image instance of the plurality of image instances of the plurality of software applications comprises an operating system.*** (Par [0084], "…including operating-system level processes; a Java application controller 105, which is suitable for controlling Java applications 108 including containers and execution modules."). Matena does not specifically disclose the "plurality" *Marsland discloses the mentioned limitations as follows:*

(Par [0032]).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Marsland into the method of Matena to take advantage of using a plurality of images and a virtual machine to emulate a computer platform. The modification would have been obvious because one of the ordinary skills in the art would implement emulating the platform to allocate system resources (See Marsland par [0030])

As per Claim 5, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein a first image instance of the plurality of image instances of the plurality***

of software applications_comprises a server application. (Par [0081], "The disclosure further describes how an *application controller controls remote invocation and creation of server objects located in the execution modules*. The advantages include location transparency of server objects; ability to relocate server objects; transparent failure recovery; partitioning of server objects; and on-demand creation of execution modules to store new server objects.").

Matena does not specifically disclose the "plurality" *Marsland discloses the mentioned limitations as follows:*

(Par [0032]).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Marsland into the method of Matena to take advantage of using a plurality of images and a virtual machine to emulate a computer platform. The modification would have been obvious because one of the ordinary skills in the art would implement emulating the platform to allocate system resources (See Marsland par [0030])

As per Claim 6, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein a first one of the virtual machines provides an environment that emulates an x86 platform.*** (Par [0412], "… Each node includes an image of a general-purpose operating system (e.g. Linux, Solaris, or Windows). Node controllers are realized as a

Java 2 Platform, Standard Edition ("J2SE") applications…" the x86 being an instance of the Windows operating system, as claimed).

As per Claim 7, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the automation infrastructure captures a first image instance of the image instances of the virtual machine manager from a first application node of the plurality of application nodes.*** (Par [0193], "JAC manages how the containers are distributed over the nodes of the distributed system and how the application execution modules are distributed over the containers. FIG. 21 also depicts the parts of the container used in an embodiment of the invention. *The container 2102 is a process that includes a Java Virtual Machine ("JVM") 2104 and application server classes 2105.* The application server classes 2105 include Java classes that implement the operations that *JAC uses to manage the lifecycle of the execution modules (such as the classes that implement the creation of execution modules)* and Java classes that are provided as runtime library classes for the execution modules (such as classes that implement the Enterprise JavaBeans container functions).").

Matena does not specifically disclose the "plurality" *Marsland discloses the mentioned limitations as follows:*

(Par [0032]).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Marsland into the method of Matena to take advantage of using a plurality of images and a virtual machine to

emulate a computer platform. The modification would have been obvious because one of the ordinary skills in the art would implement emulating the platform to allocate system resources (See Marsland par [0030])

As per Claim 8, the rejection of Claim 7 is incorporated and further Marsland discloses: ***wherein the first application node executes the virtual machine manager.*** (Par [0049], *"For example, a network file server and may be receiving 50 percent of its requests from a first virtual machine and 25 percent of its requests from a second and a third virtual machine. The virtual machine executive can then dynamically allocate a large portion of the resources to the first virtual machine.").*

As per Claim 9, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node maintains the plurality of image instances of the virtual machine manager and the plurality of image instances of the plurality of software applications as persistent or non-persistent based on a configuration.***
(Par [0105], "…When a node controller detects a process failure, it sends an event notification to all the subscribers for the event. For example, the execution controller subscribes to the process failure events so that it can maintain an up-to-date state model of the processes in the distributed system." The "event notification" when "up-to-date" being the "persistent" as claimed).

Matena does not specifically disclose the "plurality" *Marsland disclose the mentioned limitations as follows:*

(Par [0032]).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Marsland into the method of Matena to take advantage of using a plurality of images and a virtual machine to emulate a computer platform. The modification would have been obvious because one of the ordinary skills in the art would implement emulating the platform to allocate system resources (See Marsland par [0030]).

As per Claim 10, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.*** (Par [0267], " In the embodiment of the invention depicted in FIG. 36, JAC uses the Distribution Manager objects, Replication Constraint policy objects, the Container Distribution Policy objects, and EM Distribution Policy objects when making decisions to which nodes and to which containers to assign execution module replicas. The objective of this flexible, policy-based distribution of replicas…" which shows replication and distribution policy object to determine whether to deploy an application).

As per Claim 11, the rejection of Claim 10 is incorporated and further Matena discloses: ***wherein the one or more rule engines are forward- chaining rule engines.*** (Par [0271], "… The replication constraint further specifies that there should be two replicas of the execution module and that the replication type is Hot-Standby. For example, since one replica of the execution module Auction-1 is on node A, the second replica can be only on one of the nodes D, E, or F (in the example, it is on Node E)" being the "forward-chaining rule" as claimed).

As per Claim 12, the rejection of Claim 10 is incorporated and further Matena discloses: ***wherein the automation infrastructure automatically updates the one or more rules engines to autonomically control the deployment of the software applications to the application nodes in accordance with a current state of an application matrix.*** (Par [0121], "The *state model* includes objects that present nodes 909, node groups 910, processes 911, and application controllers 912 in the distributed computer system. The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system.", that being the "current state" as claimed; the "State model" being the "application matrix" as claimed; Par [0195], "…In some embodiments of the invention, the information in the deployment descriptor is represented using Extensible Markup Language (XML) technologies.", the information described as an "XML" being the application matrix, as claimed; par [0153]

As the Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…").

As per Claim 13, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.*** (Par [0121], the "State model" being the "application matrix" as claimed; Par [0195]; par [0153] As the Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…" Par [0142], "The Distribution Policy objects 1014 *provide parameters to the algorithm of the Distribution Manager objects* 1010…" Par [0201] Application definition 2 2302 includes a deployment descriptor 2304….").

As per Claim 14, the rejection of Claim 1 is incorporated and further Matena discloses: ***further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.*** (Par [0084], "For example, the exemplary ECS 101 in FIG. 1 includes three application controllers: a service application controller 104, which is suitable for controlling applications that are services 107 including operating-system level processes; a Java application controller 105, which is suitable for controlling Java

applications 108 including containers and execution modules…" the "application controller" being the "governor" as claimed).

As per Claim 15, the rejection of Claim 10 is incorporated and further Matena discloses: ***wherein the automation infrastructure automatically updates the one or more rules engines to monitor the execution of the applications when deployed to the software application nodes in accordance with a current state of the application matrix.*** (Par [0121], "The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system."; Par [0170], "In response to the "Process Started" message, the execution controller updates the state of Process 1 object in its state model to indicate that the process has successfully started 1310...").

As per Claim 16, the rejection of Claim 11 is incorporated and further Matena discloses: ***wherein the automation subsystem includes a rule compiler that compiles rules into one or more discrimination networks.***
(Par [0271], "In some embodiments of the invention, the application programmer or system administrator defines the node groups, replication groups, and replication constraints. The distributed computer system then observes the above rules when assigning execution module replicas to the nodes of the distributed computer system.").

As per Claim 18, the rejection of Claim 11 is incorporated and further Matena discloses: ***wherein the application nodes are organized into tiers, and wherein the status data is collected based on the tiers.***

(See Figures 1 and 2, that shows the organization into “tiers” as claimed; see Par [0117], “The "node group management" 905 operations include the operations to create and remove a node group; operations to add and remove a node from a node group; and operations to obtain status information about the node groups.”).

As per Claim 19, the rejection of Claim 1 is incorporated and further Matena discloses: ***further comprising: a database storing data that defines a model for a hierarchical organization of the distributed computing system,*** (Par [0097], “FIG. 6 illustrates a representational node 601 suitable for being used in a method and system of the invention. A node is a computer system including one or more central-processing units (CPU) 602, main memory 603, optional secondary storage 604, and one or more communication interfaces 605…”)

wherein the model specifies a fabric having one or more domains, (See Figure 8)

and wherein each domain has at least one tier that includes at least one of the application nodes, (Par [0099], “Each node of a distributed computer system includes a node controller 701. FIG. 7 depicts the functionality provided by a node controller and its main components. A node controller 701 implements the following functions: node membership 702, master election 703, start process 704, stop process 705, and detect process failure 706.”).

wherein one or more rule engines automatically produce action requests to control the deployment of the software applications in accordance with the hierarchical organization of the distributed computing system defined by the model. (Par [0112], "A node group is a named subset of the nodes in the distributed computer system. A node can be a member of multiple node groups. Using node groups in a distributed computer system has several advantages. First, node groups simplify many management tasks because instead of listing a large number of individual nodes when performing system a administration task, a node group name can be given instead. Second, it is easy to adjust the processing capacity or distribution of multiple applications by simply adding or removing nodes from a node group associated with the applications. And (Par [0113])

As per Claims 20-24 and 26-27 being the method claims corresponding to the system claims 1-4, 13, 12, 11 respectively and rejected under the same reason set forth in connection of the rejections of Claims 1 -4, 13, 12, 11 and further Matena discloses: (Title, "Method and apparatus for executing applications on a distributed computer system").

As per Claim 28 being the computer readable medium claim corresponding to the system claims 1 respectively and rejected under the same reason set forth in connection of the rejections of Claim 1 and further Matena discloses: (Abstract and

Claim 114, "A computer-readable medium including instructions for performing a method when executed by a processor").

7. **Claims 17 and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over Matena et al, in view of Marsland et al, and further in view of Coleman et al (US 2004/0181794 A1).

As per Claim 17, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises:*** (Par [0015], "one or more node controllers, and one or more application controllers….").

Matena do not disclose the following features:

Coleman discloses the claimed features as follows:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes (Par [0023] The deployment manager 130 interfaces with relevant resources of the application level 140 to monitor measurement/configuration data…"; see Figures 1- 2, feature 202, the "forecaster" being the "subsystem" as claimed that monitors the data and provides the status information) ***and a business logic tier that provides expected state data to the automation subsystem***, (Par [0019], "The solution manager 120, itself, comprises a control logic engine 122, a forecaster module 124, a performance estimator module 126, an admission control module 128, and a tuning module 129. The deployment

manager 130, itself, comprises a monitoring module 132…" par [0024], "e-commerce site...").

wherein the expected state data represents an expected state for the application nodes, (Par [0034], "Based on monitored data such as the transaction rate history, the solution manager 120 *uses the forecaster module 124 to project into the future the expected workload* (step 204)…").

wherein one or more rule engines analyze the status data from the monitoring subsystem (Par [0034] "…The forecast can be long-term (e.g., hours/days) or short-term (e.g., seconds/minutes). To deal with unanticipated changes in the workload (e.g., outside of normal daily/weekly, etc.) variations, the forecaster *is preferably adaptive and rapidly learns from recent history.* The forecast horizon does not need to be long, e.g., the horizon may be approximately the amount of time needed for the resource action (application provisioning in this situation) to take effect.").

and apply a set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state. (Par [0036], "*The solution manager 120 then uses the performance estimator 126 to check if the current resource deployment is insufficient, sufficient (i.e., adequate) or overly sufficient (i.e., excessive) for maintaining the service objective threshold* (e.g., response time) based on the recent and forecasted workload traffic (step 206)…").

Matena and Coleman are analogous art because both are related to dynamic deploying of data.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Coleman and Marsland into the method of Matena to use a monitoring subsystem. The modification would have been obvious because one of the ordinary skills in the art would implement using the resources to rapidly and automatically adding resources when needed, for example, in response to an unexpected workload surge, but also removing resources when no longer needed, minimizing the cost of additional resources (Coleman, par [0023]).

As per Claim 25 being the method claims corresponding to the system claim 17 respectively and rejected under the same reason set forth in connection of the rejections of Claim 17 and further Matena discloses: (Title, "Method and apparatus for executing applications on a distributed computer system").

Conclusion

8. THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to ANGELICA RUIZ whose telephone number is (571)270-3158. The examiner can normally be reached on 9:00 a.m. to 5:30 p.m., ET.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jacob Betit can be reached on (571) 272-4105. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Angelica Ruiz/
Primary Examiner, Art Unit 2158

Notice of References Cited	Application/Control No. 11/694,483	Applicant(s)/Patent Under Reexamination SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	Art Unit 2158	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2003/0110173 A1	06-2003	Marsland, Timothy	707/10
*	B	US-2003/0192035 A1	10-2003	Duesterwald ald, Evelyn	717/138
*	C	US-2004/0230948 A1	11-2004	Talwar et al.	717/114
*	D	US-7,093,239 B1	08-2006	van der Made, Peter A. J	717/135
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Netflix, Inc. - Ex. 1002, Page 000073
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	Angelica Ruiz	2158

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	08/26/2010	02/09/2011	09/21/2012	05/06/2013					
ALLO	1	✓	✓	✓	✓					
	2	✓	✓	✓	✓					
	3	✓	✓	✓	✓					
	4	✓	✓	✓	✓					
	5	✓	✓	✓	✓					
	6	✓	✓	✓	✓					
	7	✓	✓	✓	✓					
	8	✓	✓	✓	✓					
	9	✓	✓	✓	✓					
	10	✓	✓	✓	✓					
	11	✓	✓	✓	✓					
	12	✓	✓	✓	✓					
	13	✓	✓	✓	✓					
	14	✓	✓	✓	✓					
	15	✓	✓	✓	✓					
	16	✓	✓	✓	✓					
	17	✓	✓	✓	✓					
	18	✓	✓	✓	✓					
	19	✓	✓	✓	✓					
	20	✓	✓	✓	✓					
	21	✓	✓	✓	✓					
	22	✓	✓	✓	✓					
	23	✓	✓	✓	✓					
	24	✓	✓	✓	✓					
	25	✓	✓	✓	✓					
	26	✓	✓	✓	✓					
	27	✓	✓	✓	✓					
	28	✓	✓	✓	✓					

Netflix, Inc. - Ex. 1002, Page 000074
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	Art Unit
	ANGELICA RUIZ	2158

CPC- SEARCHED		
Symbol	Date	Examiner

CPC COMBINATION SETS - SEARCHED		
Symbol	Date	Examiner

US CLASSIFICATION SEARCHED			
Class	Subclass	Date	Examiner
707	600-831, 899, and 999.001-999.206, by keyword	8/26/2010	A. Ruiz
709	223-227, by keyword	8/26/2010	A. Ruiz
711	6, by keyword	8/26/2010	A. Ruiz
719	315 -324, by keyword	8/26/2010	A. Ruiz

SEARCH NOTES		
Search Notes	Date	Examiner
Consulted Rehana Perveen for (QAS/WQAS) for 101 issues	8/26/2010	A. Ruiz
See ACM Portal Search strategy, attached	8/26/2010	A. Ruiz
See EAST Search history, including inventor	8/26/2010	A. Ruiz
See Google scholar including patents strategy, attached	8/26/2010	A. Ruiz
See EAST Search history updated, attached	2/9/2011	A. Ruiz
See EAST Search history updated, attached	9/21/2012	A. Ruiz
See Google scholar including patents strategy, attached	9/21/2012	A. Ruiz
See EAST Search history updated, attached	5/6/2013	A. Ruiz

INTERFERENCE SEARCH			
US Class/ CPC Symbol	US Subclass / CPC Group	Date	Examiner

	/ANGELICA RUIZ/ Primary Examiner.Art Unit 2158

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	2	"20050005200"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/05/06 11:18
L2	14252	(server same software same application) and operating near4 system and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/05/06 11:18
L3	3216	L2 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/05/06 11:18
L4	271	L3 and (control$4 near4 node)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/05/06 11:18
L5	201	L4 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/05/06 11:18
L6	131	L5 and (server near4 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/05/06 11:18
L7	16	(emulat$4 with platform) same (virtual near4 machine).ab.	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:19
L8	2	L7 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:19
L9	2	L8 and (emulat$4 with platform) same (virtual near4 machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:19
L10	118	"707"/$.ccls. and (enterprise near4 comput$5) same (tier hierarchical)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2013/05/06 11:19
L11	4723	(virtual same machine) same emulat$5	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:20
L12	1298	L11 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:20
L13	247	L12 and (plural$5 same virtual same machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:20
L14	40	L13 and (emulat$5 same computer same platform)	US-PGPUB; USPAT; DERWENT;	OR	ON	2013/05/06 11:20

Netflix, Inc. - Ex. 1002, Page 000076
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

			IBM_TDB			
L15	30	L14 and (automatic$5 dynamic$6)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:20
L17	7	L15 and (emulat$4 with platform) same (virtual near4 machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2013/05/06 11:20

EAST Search History (Interference)

< This search history is empty>

5/6/2013 11:27:18 AM
C:\Users\aruiz1\Documents\EAST\Workspaces\11694483_Final2.wsp

Netflix, Inc. - Ex. 1002, Page 000077
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

106095 7590 01/23/2013

Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

NOTIFICATION DATE	DELIVERY MODE
01/23/2013	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOmail1@bakerbotts.com

Netflix, Inc. - Ex. 1002, Page 000078
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Applicant-Initiated Interview Summary	**Application No.** 11/694,483	**Applicant(s)** SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	**Art Unit** 2158	

All participants (applicant, applicant's representative, PTO personnel):

(1) *ANGELICA RUIZ*. (3)_____.

(2) *ROSHAN MANSINGHANI (Reg. 62429)*. (4)_____.

Date of Interview: *16 January 2013*.

Type: ☒ Telephonic ☐ Video Conference
☐ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☒ No.
If Yes, brief description: _____.

Issues Discussed ☐101 ☐112 ☐102 ☒103 ☐Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: *1*.

Identification of prior art discussed: *Matena et al (US 2005/0138370)*.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant's representative explained the invention extensively and discussed the proposed amendments. The Examiner explained her interpretation of the claim language with respect to the prior art. Also, The Examiner explained that when the proposed amendments are officially submitted further search will be needed. The amendment of "plurality" instead of the "one or more" is supposedly not recited into the primary art of record as to Matena is focused with just a virtual machine. No agreement was reached. Arguments have been taken into consideration. .

Applicant recordation instructions: The formal written reply to the last Office action must include the substance of the interview. (See MPEP section 713.04). If a reply to the last Office action has already been filed, applicant is given a non-extendable period of the longer of one month or thirty days from this interview date, or the mailing date of this interview summary form, whichever is later, to file a statement of the substance of the interview

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☐ Attachment

/Angelica Ruiz/ Primary Examiner, Art Unit 2158	

Netflix, Inc. - Ex. 1002, Page 000079
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Summary of Record of Interview Requirements

Manual of Patent Examining Procedure (MPEP), Section 713.04, Substance of Interview Must be Made of Record
A complete written statement as to the substance of any face-to-face, video conference, or telephone interview with regard to an application must be made of record in the application whether or not an agreement with the examiner was reached at the interview.

Title 37 Code of Federal Regulations (CFR) § 1.133 Interviews
Paragraph (b)

In every instance where reconsideration is requested in view of an interview with an examiner, a complete written statement of the reasons presented at the interview as warranting favorable action must be filed by the applicant. An interview does not remove the necessity for reply to Office action as specified in §§ 1.111, 1.135. (35 U.S.C. 132)

37 CFR §1.2 Business to be transacted in writing.
All business with the Patent or Trademark Office should be transacted in writing. The personal attendance of applicants or their attorneys or agents at the Patent and Trademark Office is unnecessary. The action of the Patent and Trademark Office will be based exclusively on the written record in the Office. No attention will be paid to any alleged oral promise, stipulation, or understanding in relation to which there is disagreement or doubt.

———

The action of the Patent and Trademark Office cannot be based exclusively on the written record in the Office if that record is itself incomplete through the failure to record the substance of interviews.

It is the responsibility of the applicant or the attorney or agent to make the substance of an interview of record in the application file, unless the examiner indicates he or she will do so. It is the examiner's responsibility to see that such a record is made and to correct material inaccuracies which bear directly on the question of patentability.

Examiners must complete an Interview Summary Form for each interview held where a matter of substance has been discussed during the interview by checking the appropriate boxes and filling in the blanks. Discussions regarding only procedural matters, directed solely to restriction requirements for which interview recordation is otherwise provided for in Section 812.01 of the Manual of Patent Examining Procedure, or pointing out typographical errors or unreadable script in Office actions or the like, are excluded from the interview recordation procedures below. Where the substance of an interview is completely recorded in an Examiners Amendment, no separate Interview Summary Record is required.

The Interview Summary Form shall be given an appropriate Paper No., placed in the right hand portion of the file, and listed on the "Contents" section of the file wrapper. In a personal interview, a duplicate of the Form is given to the applicant (or attorney or agent) at the conclusion of the interview. In the case of a telephone or video-conference interview, the copy is mailed to the applicant's correspondence address either with or prior to the next official communication. If additional correspondence from the examiner is not likely before an allowance or if other circumstances dictate, the Form should be mailed promptly after the interview rather than with the next official communication.

The Form provides for recordation of the following information:
- Application Number (Series Code and Serial Number)
- Name of applicant
- Name of examiner
- Date of interview
- Type of interview (telephonic, video-conference, or personal)
- Name of participant(s) (applicant, attorney or agent, examiner, other PTO personnel, etc.)
- An indication whether or not an exhibit was shown or a demonstration conducted
- An identification of the specific prior art discussed
- An indication whether an agreement was reached and if so, a description of the general nature of the agreement (may be by attachment of a copy of amendments or claims agreed as being allowable). Note: Agreement as to allowability is tentative and does not restrict further action by the examiner to the contrary.
- The signature of the examiner who conducted the interview (if Form is not an attachment to a signed Office action)

It is desirable that the examiner orally remind the applicant of his or her obligation to record the substance of the interview of each case. It should be noted, however, that the Interview Summary Form will not normally be considered a complete and proper recordation of the interview unless it includes, or is supplemented by the applicant or the examiner to include, all of the applicable items required below concerning the substance of the interview.

A complete and proper recordation of the substance of any interview should include at least the following applicable items:
1) A brief description of the nature of any exhibit shown or any demonstration conducted,
2) an identification of the claims discussed,
3) an identification of the specific prior art discussed,
4) an identification of the principal proposed amendments of a substantive nature discussed, unless these are already described on the Interview Summary Form completed by the Examiner,
5) a brief identification of the general thrust of the principal arguments presented to the examiner,
   (The identification of arguments need not be lengthy or elaborate. A verbatim or highly detailed description of the arguments is not required. The identification of the arguments is sufficient if the general nature or thrust of the principal arguments made to the examiner can be understood in the context of the application file. Of course, the applicant may desire to emphasize and fully describe those arguments which he or she feels were or might be persuasive to the examiner.)
6) a general indication of any other pertinent matters discussed, and
7) if appropriate, the general results or outcome of the interview unless already described in the Interview Summary Form completed by the examiner.

Examiners are expected to carefully review the applicant's record of the substance of an interview. If the record is not complete and accurate, the examiner will give the applicant an extendable one month time period to correct the record.

Examiner to Check for Accuracy

If the claims are allowable for other reasons of record, the examiner should send a letter setting forth the examiner's version of the statement attributed to him or her. If the record is complete and accurate, the examiner should place the indication, "Interview Record OK" on the paper recording the substance of the interview along with the date and the examiner's initials.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.:	11/694,483	Confirmation No.	1040
Applicant:	Jagane Sundar et al.		
Filed:	March 30, 2007		
Art Unit:	2158		
Examiner:	Angelica Ruiz		
Docket No.:	CAS-013		
Customer No.	106095		

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

AMENDMENT AND RESPONSE PURSUANT TO 37 C.F.R. § 1.111

This paper is being filed in response to the Office Action mailed September 27, 2012. Reconsideration and reexamination are respectfully requested in light of the amendments and remarks below.

Amendments to the claims are reflected in the listing of claims which begins on page 2 of this paper.

Remarks/Arguments begin on page 9 of this paper.

In the Claims

1. (Currently Amended) A distributed computing system comprising:

a software image repository comprising non-transitory, computer-readable media operable to store: (i) ~~one or more~~ a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide ~~one or more~~ a plurality of virtual machines, each of the plurality of virtual machine operable to provide an environment that emulates a computer platform, and (ii) ~~one or more~~ a plurality of image instances of ~~one or more~~ a plurality of software applications that are executable on the plurality of virtual machines; and

a control node that comprises an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the plurality of image instances of the software applications on the virtual machines.

2. (Currently Amended) The distributed computing system of claim 1, wherein the image instances of the ~~one or more~~ plurality of software applications comprise virtual disk image files.

3. (Currently Amended) The distributed computing system of claim 1, wherein when the automation infrastructure deploys a first image instance of the plurality of image instances of the ~~one or more~~ plurality of software applications, the automation infrastructure copies ~~a first one of~~ the first image instance ~~instances~~ to a first one of the application nodes.

4. (Currently Amended) The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications ~~an application image in the set of application images~~ comprises an operating system.

5. (Currently Amended) The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications ~~an application image in the set of application images~~ comprises a server application.

6. (Original) The distributed computing system of claim 1, wherein a first one of the virtual machines provides an environment that emulates an x86 platform.

7. (Currently Amended) The distributed computing system of claim 1, wherein the automation infrastructure captures a first image instance of the image instances of the virtual machine manager from a first application node of the plurality of ~~an~~ application nodes. ~~node.~~

8. (Currently Amended) The distributed computing system of claim 7, wherein the first application node executes ~~automation infrastructure captures the image of the application from an application node executing~~ the virtual machine manager.

9. (Currently Amended) The distributed computing system of claim 1, wherein the control node maintains the plurality of image instances of the virtual machine manager and the plurality of image instances of the plurality of software applications as persistent or non-persistent based on a configuration.

10. (Original) The distributed computing system of claim 1, wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.

11. (Currently Amended) The distributed computing system of claim 10, wherein the one or more rule engines are forward-chaining rule engines.

12. (Currently Amended) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the one or more rules engines to autonomically control the deployment of the software applications to the application nodes in accordance with a current state of an application matrix.

13. (Original) The distributed computing system of claim 1, wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.

14. (Original) The distributed computing system of claim 13, further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.

15. (Currently Amended) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the one or more rules engines to monitor the execution of the software applications when deployed to the application nodes in accordance with a current state of the application matrix.

16. (Currently Amended) The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles ~~the~~ rules into one or more discrimination networks.

17. (Currently Amended) The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects ~~the~~ status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein ~~the~~ one or more rule engines analyze the status data from the monitoring subsystem and apply ~~the~~ a set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

18. (Currently Amended) The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein ~~each of the information collectors collects the~~ status data is collected based on the ~~from the application nodes of different~~ tiers.

19. (Currently Amended) The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having [[a]] one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein ~~the~~ one or more rule engines automatically produce action requests to control the deployment of the software applications in accordance with [[a]] the hierarchical organization of the distributed computing system defined by the model.

20. (Currently Amended) A method comprising:

storing ~~one or more~~ a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide ~~one or more~~ a plurality of virtual machines, each virtual machine operable to provide an environment that emulates a computer platform;

storing ~~one or more~~ a plurality of image instances of ~~one or more~~ a plurality of software applications that are executable on the plurality of virtual machines;

executing a rules engine to perform operations to autonomically deploy the image instances of the virtual machine manager on the application nodes; ~~within a distributed computing system;~~ and

executing the rules engine to perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

21. (Currently Amended) The method of claim 20, wherein storing the image instances of the ~~one or more~~ plurality of software applications ~~comprise~~ comprises storing virtual disk image files.

22. (Currently Amended) The method of claim 20, wherein performing operations to autonomically deploy the image instances of the virtual machine manager comprises copying a first one of the image instances of the virtual machine manager to a first one of the application nodes.

23. (Currently Amended) The method of claim 20, wherein a first image instance of the plurality of image instances of the plurality of software applications ~~the software applications~~ comprises ~~comprise~~ an operating system.

24. (Currently Amended) The method of claim 20,

further comprising generating an application matrix that specifies data for controlling the deployment of the software applications within [[a]] the distributed computing system; and

wherein performing operations to provide autonomic control comprises providing ~~provide~~ autonomic control over the deployment, undeployment, and execution of the software applications on the virtual machines within the distributed computing system in accordance with parameters of the application matrix.

25. (Currently Amended) The method of claim 20, wherein performing operations to autonomically deploy the image instances of the software applications on the virtual machines comprises:

receiving status data for the distributed computing system, ~~having the plurality of application nodes interconnected via a communications network,~~ wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the application nodes ~~distributed computing system~~; and

performing the operations to provide autonomic control over the deployment of the software applications on the virtual machines within the distributed computing system in accordance with the rules.

26. (Currently Amended) The method of claim 20, further comprising accessing parameters of ~~the~~ an application matrix and ~~generating~~ determining one or more rules for use in autonomically controlling the deployment of the software applications on the virtual machines.

27. (Currently Amended) The method of claim 26, further comprising processing the one or more rules with a plurality of forward-chaining rule engines that compile the one or more rules into one or more discrimination networks.

28. (Currently Amended) A non-transitory, computer-readable medium comprising instructions stored thereon, that when executed by a programmable processor:

store ~~one or more~~ a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide ~~one or more~~ a plurality of virtual machines, each virtual machine operable to provide an environment that emulates a computer platform;

store ~~one or more~~ a plurality of image instances of ~~one or more~~ a plurality of software applications that are executable on the plurality of virtual machines;

perform operations to autonomically deploy the image instances of the virtual machine manager on the application nodes; ~~within a distributed computing system;~~ and

perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

REMARKS

This Application has been carefully reviewed in light of the Office Action. Claims 1-28 are pending and rejected. Applicants amend Claims 1-5, 7-9, 11, 12, and 15-28. Applicants respectfully request reconsideration and favorable action in this case.

Summary of Interview

Applicants thank the Examiner for the telephonic interview held with Applicants' attorney Roshan S. Mansinghani (Reg. No. 62,429) on January 16, 2013. During the interview, amended Claim 1 and the *Matena* reference were discussed. The Examiner agreed that the present amendments would overcome the Section 103 rejections.

Request for Interview

If the Examiner intends to issue a new Office Action in response to Applicants' present submission, Applicants respectfully request that the Examiner contact Applicants' attorney prior to issuing the new Office Action to discuss a possible resolution to any outstanding issues.

Rejections under 35 U.S.C. § 103

The Examiner rejects Claims 1-16, 18-24, and 26-28 under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent Publication No. 2005/0005200 by Matena et al. ("*Matena*") in view of U.S. Patent Publication No. 2005/0138370 by Goud et al. ("*Goud*"). Claims 17 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Matena* in view of *Goud,* and further in view of U.S. Patent Publication No. 2004/0181794 by Coleman et al. ("*Coleman*"). For at least the reasons provided below, Applicants respectfully traverse these rejections.

The Examiner has not established a prima facie rejection under Section 103 of Claims 1-16, 18-24, and 26-28 at least because the cited portions of the proposed *Matena-Goud* combination do not disclose, teach, or suggest all of the limitations of Claim 1. For example, amended Claim 1 recites:

> a software image repository comprising non-transitory, computer-readable media operable to store: (i) a plurality of image instances of a virtual

> machine manager that is executable on a plurality of application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide a plurality of virtual machines, each of the plurality of virtual machine operable to provide an environment that emulates a computer platform, and (ii) a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines[.]

In rejecting a previous version of Claim 1, the Examiner cited to, inter alia, ¶¶ 81, 193, 195 of *Matena*:

> The disclosure further describes how an application controller controls remote invocation and creation of server objects located in the execution models. [¶ 81]
>
> JAC manages how the containers are distributed over the nodes of the distributed system and how the application execution modules are distributed over the containers. . . . The container 2102 is a process that includes a Java Virtual Machine ("JVM") 2104 and application server classes 2105. The application server classes 2105 include Java classes that implement the operations that JAC uses to manage the lifecycle of the execution models (such as the classes that implement the creation of execution models) and Java classes that are provided as runtime library classes for the execution models (such as classes that implement the Enterprise JavaBeans container functions). [¶ 193]
>
> An application definition 2201 includes Java classes 2202 and deployment descriptors 2203. The Java classes 2202 include the program instructions of the application's execution modules. The deployment descriptors 2203 include the definitions of one or more execution module ("EM definitions") 2204. An EM definition 2204 is a template for one or more execution modules created from the EM definition 2204. In one embodiment of the invention, the distribution policies specify how many instances of the execution module should be created, the name of the container group in which the execution modules should be created, and how the execution modules should be distributed over the containers. [¶ 195]

Matena further discloses at ¶ 233 (not cited by the Examiner):

> A system management tool sends the "Start Application" message to JAC 2501. In response to the message, JAC reads the application definition files and creates the Application object 2601, EM definition object 2602, and EM Distribution Policy object 2603 in its state model 2502. The exemplary EM Distribution Policy is the EMContainerDistribution(2) distribution policy object, which indicates that there shall be two execution modules in each container. The EM definition included in the application definition may specify the name of the container group to associate with the execution modules.

That is, the cited portions of *Matena* disclose that a JAC manages the creation of execution modules (i.e., Java objects) from Java classes as well as the creation of containers in which the execution modules are to operate. Figure 26 of *Matena* further discloses that the JAC causes the operation of execution modules on nodes by sending commands to create execution modules to the nodes. The nodes send messages to the JAC indicating whether the attempt to create the execution modules was successful. *See Matena* at ¶ 239. As the Examiner agreed to during the interview, this does not disclose, teach, or suggest "a software image repository . . . operable to store: (i) a plurality of image instances of a virtual machine manager . . . and (ii) a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines," as recited in amended Claim 1 at least because *Matena* discloses storing information (such as Java classes) that are used to compile Java objects by the nodes at the direction of the JAC. The cited portions of *Goud* do not remedy these deficiencies as they merely disclose a virtual machine and a virtual machine manager.

For at least these reasons, independent Claim 1 and its dependent claims should be reconsidered and allowed. For analogous reasons, independent Claims 20 and 28, as well as the claims which depend from Claim 20, should be reconsidered and allowed.

No Waiver

Applicants' arguments are made without prejudice or disclaimer. By not responding to additional statements made by the Examiner, Applicants do not acquiesce to the additional statements. The distinctions between the applied references and the claims are provided as examples only and are sufficient to overcome the rejections. Applicants reserve the right to discuss additional or other distinctions in a later response or on appeal, if appropriate.

Request for Evidentiary Support

If the Examiner is relying upon "common knowledge" or "well known" principles or "Official Notice" or other information within the Examiner's personal knowledge to establish a rejection, Applicants respectfully request that the Examiner cite a reference or provide an affidavit in support of the position in accordance with M.P.E.P. § 2144.03 and 37 C.F.R. 1.104(d)(2).

CONCLUSION

Applicants have made an earnest attempt to place this case in condition for allowance. For at least the foregoing reasons, Applicants respectfully request full allowance of all the pending claims.

If the Examiner believes a telephone conference would advance prosecution of this case in any way, the Examiner is invited to contact Luke K. Pedersen, the Attorney for Applicants, at the Examiner's convenience at (202) 639-7730.

The Commissioner is authorized to charge a one-month extension-of-time fee in the amount of $150.00 to Deposit Account No. 02-0384 of Baker Botts L.L.P. Although Applicants believe no other fees are due, the Commissioner is authorized to charge any necessary additional fees and credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

Respectfully submitted,



Roshan S. Mansinghani
Attorney for Applicants
Reg. No. 62,429

Date: January 17, 2013

Customer No: 106095

This paper or fee is being filed using the USPTO's electronic filing system EFS-Web, and is addressed to the Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Electronic Patent Application Fee Transmittal

Application Number:	11694483
Filing Date:	30-Mar-2007
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Filer:	Roshan Suresh Mansinghani
Attorney Docket Number:	063170.9310

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 1 month with $0 paid	1251	1	150	150

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**150**

Electronic Acknowledgement Receipt	
EFS ID:	14726321
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	106095
Filer:	Roshan Suresh Mansinghani
Filer Authorized By:	
Attorney Docket Number:	063170.9310
Receipt Date:	17-JAN-2013
Filing Date:	30-MAR-2007
Time Stamp:	14:39:36
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$150
RAM confirmation Number	1070
Deposit Account	020384
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		0631709310RespOA011713.PDF	561731 ce6b032def46a2c4e6698db3247b87af76f3f467	yes	12
	Multipart Description/PDF files in .zip description				
	Document Description		**Start**	**End**	
	Amendment/Req. Reconsideration-After Non-Final Reject		1	1	
	Claims		2	8	
	Applicant Arguments/Remarks Made in an Amendment		9	12	
Warnings:					
Information:					
2	Fee Worksheet (SB06)	fee-info.pdf	30630 b242efba67a3877a514e1f4d3732e5ac5f7a220d	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	592361		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 11/694,483	Filing Date 03/30/2007	☐ To be Mailed

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☒		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☒ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	75		N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	75		TOTAL	

APPLICATION AS AMENDED – PART II

		(Column 1)		(Column 2)	(Column 3)	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
	01/17/2013	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
AMENDMENT	Total (37 CFR 1.16(i))	* 28	Minus	** 28	= 0	X $31 =	0	OR	X $ =	
	Independent (37 CFR 1.16(h))	* 3	Minus	***3	= 0	X $125 =	0	OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE	

		(Column 1)		(Column 2)	(Column 3)					
		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
AMENDMENT	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/TER JACKSON WOODRUFF/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

106095 7590 09/27/2012

Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

NOTIFICATION DATE	DELIVERY MODE
09/27/2012	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOmail1@bakerbotts.com

Netflix, Inc. - Ex. 1002, Page 000098
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Office Action Summary	Application No.	Applicant(s)	
	11/694,483	SUNDAR ET AL.	
	Examiner	Art Unit	
	ANGELICA RUIZ	2158	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *25 October 2011*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

5)☒ Claim(s) *1-28* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-28* is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

12)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. The Action is responsive to Applicant's request for continued examination, filed on July 20, 2011.

2. It is acknowledged that as a result of the amendment, Claims 1 and 20 have been amended.

3. Claims 1-28 are pending.

Response to Arguments

4. Applicant's arguments with respect to claims 1-28 have been considered but are moot in view of the new grounds of rejection necessitated by Applicant's amendment of the claims.

Applicant argues that the prior art of record fails to disclose, teach, or suggest "a software image repository comprising…one or more image instances of one or more software applications that are executable on the virtual machine" The Examiner respectfully submits that the claim language clearly states that "one or more" the Examiner relied in the claim language to provide the mentioned citations that refer to the *one or more image instances of a virtual machine manager that is executable on a plurality of application nodes,* (Par [0090], "Java classes" being the "virtual machine" executing on the application nodes, as claimed), *wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines,* (Par [0193], includes the instance of the "virtual machine" as claimed*) and (ii) one or more image instances of one or more software applications that are executable*

on the virtual machines; (Par [0015], "…execution *control system provides a single-system image of the distributed computer system and applications running on it…*"; Par [0195] and see also Figures 21-22, that shows the "java virtual Machine" and the "application server classes" the mentioned figure clearly shows the recited limitations of "one or more virtual machine" and the "one or more software application that are executable on the virtual machines" as claimed.) However Matena does not specifically disclose the newly added limitations: *"each virtual machine operable to provide an environment that emulates a computer platform" Goud discloses the mentioned limitations as follows:* (Par [0008] "…environment for emulating a trusted platform module ("TPM") using a virtual machine monitor ("VMM"), in accordance with an embodiment of the present invention." Par [0014] and see figure 2).

In view of the above, the Examiner contends that all limitations as recited in the claims have been addressed in this Action.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims **1 – 16, 18 –24 and 26 –28** are rejected under 35 U.S.C. 103(a) as being unpatentable over Matena et al (US 2005/0005200 A1), in view of Goud et al (US 2005/0138370), hereinafter "Matena" and "Goud" respectively.

As per Claim 1, Matena discloses:

A distributed computing system comprising: (Title, "Method and apparatus for executing applications on a distributed computer system") and (Abstract, "multiple nodes interconnected through a network…") ***a software image repository comprising non-transitory, computer-readable media operable to store:*** (Par [0015] and (Par [0081], "…creation of execution modules to store new server objects.") ***(i) one or more image instances of a virtual machine manager that is executable on a plurality of application nodes,*** (Par [0090], "Java classes" being the "virtual machine" executing on the application nodes, as claimed), ***wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines,*** (Par [0193], includes the instance of the "virtual machine" as claimed) ***and (ii) <u>one</u> or more image instances of one or more software applications that are executable on the virtual machines;*** (Par [0015], "The application controllers utilize the execution controller. To its users, the execution *control system provides a single-system image of the distributed computer system and applications running on it*…"; Par [0195] and see also Figures 21-22) ***and a control node*** (Par [0015], "one or more node controllers, and one or more application controllers….") ***that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual***

machine manager on the application nodes (Par [0085], “The node controllers 103, execution controller 102 and application controllers are distributed over the nodes of a distributed computer system.”; see figure 2) ***and to provide autonomic deployment of the image instances of the software applications on the virtual machines.*** (Par [0089]; par [0112], “can automatically distribute application modules of multiple applications to the node…”; par [0113], "… *same application could be deployed on several distributed computer systems, each having a different configuration, and each possibly requiring different assignment of application module replicas* to nodes.”, the “replicas” being the “images” as claimed).

Matena does not specifically discloses the ***“each virtual machine operable to provide an environment that emulates a computer platform”***

Goud discloses the mentioned limitations as follows:

(Par [0008] "…environment for emulating a trusted platform module ("TPM") using a virtual machine monitor ("VMM"), in accordance with an embodiment of the present invention.”, Par [0014] and see figure 2).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Goud into the method of Matena to take advantage of using a virtual machine to emulate a computer platform. The modification would have been obvious because one of the ordinary skills in the art would implement emulating the platform to allocate system resources (See Goud par [0017])

As per Claim 2, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the image instances of the one or more software applications comprise virtual disk image files.*** (Par [0108], "… *with different operating systems.* ...To shield the other components of the execution control system from the differences in the command line format, the *NC API 707 uses a virtual, operating-system* neutral format of the command line. The node controllers 701 have the logic to translate the virtual command line to the format understood by the local operating system.").

As per Claim 3, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein when the automation infrastructure deploys the image of the one or more applications, the automation infrastructure copies a first one of the image instances to a first one of the application nodes.*** (Par [0251] and Par [0289], "…A purpose of the replication constraint is *to specify that if one replica of an execution module is assigned to a node that is included in the first node group, then the other replica of the same execution module should be assigned to a node included in the second node group*.").

As per Claim 4, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein an application image in the set of application images comprises an operating system.***

(Par [0084], "…including operating-system level processes; a Java application controller 105, which is suitable for controlling Java applications 108 including containers and execution modules.").

As per Claim 5, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein an application image in the set of application images comprises a server application.***

(Par [0081], "The disclosure further describes how an *application controller controls remote invocation and creation of server objects located in the execution modules*. The advantages include location transparency of server objects; ability to relocate server objects; transparent failure recovery; partitioning of server objects; and on-demand creation of execution modules to store new server objects.").

As per Claim 6, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein a first one of the virtual machines provides an environment that emulates an x86 platform.***

(Par [0412], "… Each node includes an image of a general-purpose operating system (e.g. Linux, Solaris, or Windows). Node controllers are realized as a Java 2 Platform, Standard Edition ("J2SE") applications…" the x86 being an instance of the Windows operating system, as claimed).

As per Claim 7, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the automation infrastructure captures the image of the virtual machine manager from an application node.***

(Par [0193], “JAC manages how the containers are distributed over the nodes of the distributed system and how the application execution modules are distributed over the containers. FIG. 21 also depicts the parts of the container used in an embodiment of the invention. *The container 2102 is a process that includes a Java Virtual Machine ("JVM") 2104 and application server classes 2105.* The application server classes 2105 include Java classes that implement the operations that *JAC uses to manage the lifecycle of the execution modules (such as the classes that implement the creation of execution modules)* and Java classes that are provided as runtime library classes for the execution modules (such as classes that implement the Enterprise JavaBeans container functions).”).

As per Claim 8, the rejection of Claim 7 is incorporated and further Matena discloses: ***wherein the automation infrastructure captures the image of the application from an application node executing the virtual machine manager.*** (Par [0193], *which is captured from the “application node" as claimed).*

As per Claim 9, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node maintains the image instances as persistent or non-persistent based on a configuration.***

(Par [0105], "…When a node controller detects a process failure, it sends an event notification to all the subscribers for the event. For example, the execution controller subscribes to the process failure events so that it can maintain an up-to-date state model of the processes in the distributed system." The "event notification" when "up-to-date" being the "persistent" as claimed).

As per Claim 10, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.***
(Par [0267], " In the embodiment of the invention depicted in FIG. 36, JAC uses the Distribution Manager objects, Replication Constraint policy objects, the Container Distribution Policy objects, and EM Distribution Policy objects when making decisions to which nodes and to which containers to assign execution module replicas. The objective of this flexible, policy-based distribution of replicas…" which shows replication and distribution policy object to determine whether to deploy an application).

As per Claim 11, the rejection of Claim 10 is incorporated and further Matena discloses: ***wherein the rule engines are forward- chaining rule engines.***
(Par [0271], "… The replication constraint further specifies that there should be two replicas of the execution module and that the replication type is Hot-Standby. For example, since one replica of the execution module Auction-1 is on node A,

the second replica can be only on one of the nodes D, E, or F (in the example, it is on Node E)" being the "forward-chaining rule" as claimed).

As per Claim 12, the rejection of Claim 10 is incorporated and further Matena discloses: ***wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of an application matrix.***

(Par [0121], "The *state model* includes objects that present nodes 909, node groups 910, processes 911, and application controllers 912 in the distributed computer system. The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system.", that being the "current state" as claimed; the "State model" being the "application matrix" as claimed; Par [0195], "…In some embodiments of the invention, the information in the deployment descriptor is represented using Extensible Markup Language (XML) technologies.", the information described as an "XML" being the application matrix, as claimed; par [0153] As the Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…").

As per Claim 13, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises an application matrix***

that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.

(Par [0121], the "State model" being the "application matrix" as claimed; Par [0195]; par [0153] As the Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…" Par [0142], "The Distribution Policy objects 1014 *provide parameters to the algorithm of the Distribution Manager objects* 1010…" Par [0201] Application definition 2 2302 includes a deployment descriptor 2304….").

As per Claim 14, the rejection of Claim 1 is incorporated and further Matena discloses: ***further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.***

(Par [0084], "For example, the exemplary ECS 101 in FIG. 1 includes three application controllers: a service application controller 104, which is suitable for controlling applications that are services 107 including operating-system level processes; a Java application controller 105, which is suitable for controlling Java applications 108 including containers and execution modules…" the "application controller" being the "governor" as claimed).

As per Claim 15, the rejection of Claim 10 is incorporated and further Matena discloses: ***wherein the automation infrastructure automatically updates the***

rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

(Par [0121], "The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system."; Par [0170], "In response to the "Process Started" message, the execution controller updates the state of Process 1 object in its state model to indicate that the process has successfully started 1310...").

As per Claim 16, the rejection of Claim 11 is incorporated and further Matena discloses: ***wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.***

(Par [0271], "In some embodiments of the invention, the application programmer or system administrator defines the node groups, replication groups, and replication constraints. The distributed computer system then observes the above rules when assigning execution module replicas to the nodes of the distributed computer system.").

As per Claim 18, the rejection of Claim 11 is incorporated and further Matena discloses: ***wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.***

(See Figures 1 and 2, that shows the organization into “tiers” as claimed; see Par [0117], “The "node group management" 905 operations include the operations to create and remove a node group; operations to add and remove a node from a node group; and operations to obtain status information about the node groups.”).

As per Claim 19, the rejection of Claim 1 is incorporated and further Matena discloses: ***further comprising: a database storing data that defines a model for a hierarchical organization of the distributed computing system,*** (Par [0097], “FIG. 6 illustrates a representational node 601 suitable for being used in a method and system of the
invention. A node is a computer system including one or more central-processing units (CPU) 602, main memory 603, optional secondary storage 604, and one or more communication interfaces 605…”)
wherein the model specifies a fabric having a one or more domains, (See Figure 8)
and wherein each domain has at least one tier that includes at least one of the application nodes, (Par [0099], “Each node of a distributed computer system includes a node controller 701. FIG. 7 depicts the functionality provided by a node controller and its main components. A node controller 701 implements the following functions: node membership 702, master election 703, start process 704, stop process 705, and detect process failure 706.”).

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model.

(Par [0112], "A node group is a named subset of the nodes in the distributed computer system. A node can be a member of multiple node groups. Using node groups in a distributed computer system has several advantages. First, node groups simplify many management tasks because instead of listing a large number of individual nodes when performing system a administration task, a node group name can be given instead. Second, it is easy to adjust the processing capacity or distribution of multiple applications by simply adding or removing nodes from a node group associated with the applications. And (Par [0113])

As per Claims 20-24 and 26-27 being the method claims corresponding to the system claims 1-4, 13, 12, 11 respectively and rejected under the same reason set forth in connection of the rejections of Claims 1 -4, 13, 12, 11 and further Matena discloses: (Title, “Method and apparatus for executing applications on a distributed computer system”).

As per Claim 28 being the computer readable medium claim corresponding to the system claims 1 respectively and rejected under the same reason set forth in connection of the rejections of Claim 1 and further Matena discloses: (Abstract

and Claim 114, “A computer-readable medium including instructions for performing a method when executed by a processor”).

7. **Claims 17 and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over Matena et al (US 2005/0005200 A1), in view of Goud et al (US 2005/0138370), and further in view of Coleman et al (US 2004/0181794 A1).

As per Claim 17, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises:*** (Par [0015], “one or more node controllers, and one or more application controllers….”).

Matena do not disclose the following features:

Coleman discloses the claimed features as follows:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes (Par [0023] The deployment manager 130 interfaces with relevant resources of the application level 140 to monitor measurement/configuration data…”; see Figures 1- 2, feature 202, the “forecaster” being the “subsystem” as claimed that monitors the data and provides the status information) ***and a business logic tier that provides expected state data to the automation subsystem***, (Par [0019], “The solution manager 120, itself, comprises a control logic engine 122, a forecaster module 124, a performance estimator module 126, an admission control module 128, and a tuning module 129. The deployment manager 130, itself, comprises a monitoring module 132…” par [0024], "e-commerce site...").

wherein the expected state data represents an expected state for the application nodes, (Par [0034], "Based on monitored data such as the transaction rate history, the solution manager 120 *uses the forecaster module 124 to project into the future the expected workload* (step 204)…").

wherein the rule engines analyze the status data from the monitoring subsystem (Par [0034] "…The forecast can be long-term (e.g., hours/days) or short-term (e.g., seconds/minutes). To deal with unanticipated changes in the workload (e.g., outside of normal daily/weekly, etc.) variations, the forecaster *is preferably adaptive and rapidly learns from recent history.* The forecast horizon does not need to be long, e.g., the horizon may be approximately the amount of time needed for the resource action (application provisioning in this situation) to take effect.").

and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state. (Par [0036], "*The solution manager 120 then uses the performance estimator 126 to check if the current resource deployment is insufficient, sufficient (i.e., adequate) or overly sufficient (i.e., excessive) for maintaining the service objective threshold* (e.g., response time) based on the recent and forecasted workload traffic (step 206)…").

Matena and Coleman are analogous art because both are related to dynamic deploying of data.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Coleman and

Goud into the method of Matena to use a monitoring subsystem. The modification would have been obvious because one of the ordinary skills in the art would implement using the resources to rapidly and automatically adding resources when needed, for example, in response to an unexpected workload surge, but also removing resources when no longer needed, minimizing the cost of additional resources (Coleman, par [0023]).

As per Claim 25 being the method claims corresponding to the system claim 17 respectively and rejected under the same reason set forth in connection of the rejections of Claim 17 and further Matena discloses: (Title, "Method and apparatus for executing applications on a distributed computer system").

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to ANGELICA RUIZ whose telephone number is (571)270-3158. The examiner can normally be reached on 9:00 a.m. to 5:30 p.m., ET.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jacob Betit can be reached on (571) 272-4105. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/**Angelica Ruiz**/
Examiner, Art Unit 2158

Notice of References Cited	Application/Control No. 11/694,483	Applicant(s)/Patent Under Reexamination SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	Art Unit 2158	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2003/0135658 A1	07-2003	Haggar et al.	709/312
*	B	US-2005/0138370 A1	06-2005	Goud et al.	713/164
*	C	US-2006/0259292 A1	11-2006	Solomon et al.	703/027
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Netflix, Inc. - Ex. 1002, Page 000117
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	56	(virtual near2 machine) same application same node same (control$5) and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L2	24	L1 and (software near3 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L3	1	L2 and (node same x86)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L4	11974	(server same software same application) and operating near4 system and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L5	3216	L4 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L6	271	L5 and (control$4 near4 node)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L7	201	L6 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L8	131	L7 and (server near4 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L9	2	"20050005200"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L10	11974	(server same software same application) and operating near4 system and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L11	3216	L10 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L12	271	L11 and (control$4 near4 node)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L13	201	L12 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L14	131	L13 and (server near4 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L15	30	L14 and "707"/$.ccls.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L16	111	"707"/$.ccls. and (enterprise near4 comput$5) same (tier hierarchical)	US-PGPUB; USPAT;	OR	ON	2012/09/22 00:20

Netflix, Inc. - Ex. 1002, Page 000118
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

			IBM_TDB			
L17	35	L16 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L18	8	L17 and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L19	8	L18 and (not sundar)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2012/09/22 00:20
L20	418	(emulat$4 with platform) same (virtual near4 machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2012/09/22 00:21
L21	100	20 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2012/09/22 00:21
L22	45	21 and (replica image instance) same virtual with machine	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2012/09/22 00:22
L25	12	22 and (node)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2012/09/22 00:22
L26	12	(emulat$4 with platform) same (virtual near4 machine).ab.	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2012/09/22 00:34
L27	2	26 and @py<="2006"	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2012/09/22 00:34
L28	2	27 and (emulat$4 with platform) same (virtual near4 machine)	US-PGPUB; USPAT; DERWENT; IBM_TDB	OR	ON	2012/09/22 00:36

EAST Search History (Interference)

< This search history is empty>

9/ 22/ 2012 12:53:57 AM
C:\ Users\ aruiz1\ Documents\ EAST\ Workspaces\ 11694483_RCE.wsp

Netflix, Inc. - Ex. 1002, Page 000119
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	ANGELICA RUIZ	2158

SEARCHED			
Class	**Subclass**	**Date**	**Examiner**
707	600-831, 899, and 999.001-999.206, by keyword	8/26/2010	A. Ruiz
709	223-227, by keyword	8/26/2010	A. Ruiz
711	6, by keyword	8/26/2010	A. Ruiz
719	315 -324, by keyword	8/26/2010	A. Ruiz

SEARCH NOTES		
Search Notes	**Date**	**Examiner**
Consulted Rehana Perveen for (QAS/WQAS) for 101 issues	8/26/2010	A. Ruiz
See ACM Portal Search strategy, attached	8/26/2010	A. Ruiz
See EAST Search history, including inventor	8/26/2010	A. Ruiz
See Google scholar including patents strategy, attached	8/26/2010	A. Ruiz
See EAST Search history updated, attached	2/9/2011	A. Ruiz
See EAST Search history updated, attached	9/21/2012	A. Ruiz
See Google scholar including patents strategy, attached	9/21/2012	A. Ruiz

INTERFERENCE SEARCH			
Class	**Subclass**	**Date**	**Examiner**

/ANGELICA RUIZ/ Examiner.Art Unit 2158	

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	Art Unit
	Angelica Ruiz	2158

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	08/26/2010	02/09/2011	09/21/2012						
ALLO	1	✓	✓	✓						
	2	✓	✓	✓						
	3	✓	✓	✓						
	4	✓	✓	✓						
	5	✓	✓	✓						
	6	✓	✓	✓						
	7	✓	✓	✓						
	8	✓	✓	✓						
	9	✓	✓	✓						
	10	✓	✓	✓						
	11	✓	✓	✓						
	12	✓	✓	✓						
	13	✓	✓	✓						
	14	✓	✓	✓						
	15	✓	✓	✓						
	16	✓	✓	✓						
	17	✓	✓	✓						
	18	✓	✓	✓						
	19	✓	✓	✓						
	20	✓	✓	✓						
	21	✓	✓	✓						
	22	✓	✓	✓						
	23	✓	✓	✓						
	24	✓	✓	✓						
	25	✓	✓	✓						
	26	✓	✓	✓						
	27	✓	✓	✓						
	28	✓	✓	✓						

Netflix, Inc. - Ex. 1002, Page 000121
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Web Images More... Sign in
My Citations
Use our modern look Learn more
virtual machine emulators a computer platform "java"
Search
Advanced Scholar Search

Scholar Articles and patents - 2006 include citations Create email alert

Results **1** - **10** of about **2,420**. (**0.05** sec)

The architecture of **virtual** machines [PDF] from ku.edu
JE Smith, R Nair - **Computer**, 2005 - ieeexplore.ieee.org
... Emulators and dynamic binary translators A more challenging problem for process-level VMs is that of supporting program binaries com- piled to an **...** Page 5. 36 **Computer ...** System VMs emerged during the 1960s and early 1970s5 and were the origin of the term **virtual machine**. **...**
Cited by 208 Related articles All 21 versions

Server and terminal **emulator** for persistent connection to a legacy host system with printer emulation
TH Butts, SH Burris Jr, PC Zuefeldt - US Patent 6,233,543, 2001 - Google Patents
... EMULATION) 44 36 38 40 CLIENT SYSTEM WEB BROWSER **VIRTUAL MACHINE** APPLET PROCESS **...** particularly to a server and a web browser terminal **emulator** for persistent **...** BACKGROUND OF THE INVENTION 25 Many organizations operate **computer** network environ **...**
Cited by 43 Related articles All 2 versions

Development of a Universal **Virtual Computer** (UVC) for long-term preservation of digital objects [PDF] from psu.edu
JR Van Der Hoeven, RJ Van Diessen... - Journal of Information ..., 2005 - jis.sagepub.com
... key to run **computer** games written for an obsolete **platform** on a current **machine**. **...** These are: • Universal **Virtual Computer**; • UVC program (format decoder); • Logical Data Schema (LDS) with **...** It defines four primary aspects of the **emulator**: • The architecture • The loading/state **...**
Cited by 60 Related articles All 10 versions

Java™ on the bare metal of wireless sensor devices: the squawk **Java virtual machine** [PDF] from usenix.org
D Simon, C Cifuentes, D Cleal, J Daniels... - ... conference on **Virtual** ..., 2006 - dl.acm.org
... One medium for enabling this vision is the tiny, wireless **computer** that connects to the world with sensors and actuators. **...** This paper concentrates on the **Java virtual machine** that is available on the Sun SPOT **platform**: the Squawk **virtual machine**. **...**
Cited by 154 Related articles All 18 versions

Welcome to the opportunities of binary translation [PDF] from psu.edu
ER Altman, D Kaeli, Y Sheffer - **Computer**, 2000 - ieeexplore.ieee.org
... 40 **Computer ... Emulators** are relatively easy to develop and with only modest effort can be made highly compatible with the legacy architecture. **...** into bytecode, so that binary distributions of **Java** applications can be ported to any **platform** for which a **Java virtual machine** (VM) is **...**
Cited by 108 Related articles BL Direct All 25 versions

Internet access to real equipment at **computer** architecture laboratories using the **Java**/CORBA paradigm
FJ González-Castaño, L Anido-Rifón... - Computers & ..., 2001 - Elsevier
... The system also includes several communication tools, both synchronous (chat, talk, **virtual** whiteboard) and asynchronous (mail, bulletin board, notebooks), and a tutoring and monitoring tool. **...** Students interact with the board using a PC terminal **emulator**, and several **...**
Cited by 28 Related articles BL Direct All 5 versions

Stacking them up: a Comparison of **Virtual** Machines [PDF] from npi-tu.ru
KJ Gough - Australian **Computer** Science Communications, 2001 - dl.acm.org
... 3]. As a measure of the com- plexity of the **virtual machine emulator**, the interpreter was about lk lines of assembly language, with the floating point **emulator** a further lk **...** Australian **Computer** Society. **...** [5] T. Lindholm and E Yellin, The **Java Virtual Machine** Specification, Addison **...**
Cited by 61 Related articles BL Direct All 53 versions

[BOOK] Programming for the **Java virtual machine**
J Engel - 1999 - books.google.com
... paper) 1. **Java** (**Computer** program language) 2. **Java virtual machine ...** Introduction to the **Java Virtual Machine** 1 1.1 **Java Virtual Machine** and **Java** 2 1.2 What Is the **Java Virtual Machine**? **...** 1.2.1 Instruction Set 3 1.2.2 class File Format 5 1.2.3 Verification 5 1.3 **Java Platform** 6 1.4 **...**
Cited by 86 Related articles Library Search All 2 versions

Host information access via distributed programmed objects

Netflix, Inc. - Ex. 1002, Page 000122
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

TF King, GT Knowles, RS Rittenhouse... - US Patent 6,014,702, 2000 - Google Patents
... a server to a client in the form of byte codes for execution on a **virtual machine** was also **...** user desktops, there is no longer a need to maintain **emulator** code on each individual **machine** and client **...** 1 depicts a basic network environment with standa- lone client **emulators** (Prior Art **...**
Cited by 81 Related articles All 2 versions

Method of run-time tracking of object references in Java programs

AR Adl-Tabatabai, GY Lueh - US Patent 6,093,216, 2000 - Google Patents
... A **Java Virtual Machine** is not an actual hardware **platform**, but rather a low level software **emulator** that can 50 be implemented on many different **computer** processor architectures and under many different operating systems. **...**
Cited by 68 Related articles All 2 versions

Create email alert

Goooooooooogle
1 2 3 4 5 6 7 8 9 10 Next

virtual machine emulators a computer platform "java"

©2012 Google

Netflix, Inc. - Ex. 1002, Page 000123
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	PATENT NUMBER	GROUP ART UNIT	FILE WRAPPER LOCATION
11/694,483		2158	

OC000000054923488

Correspondence Address/Fee Address Change

The following fields have been set to Customer Number 106095 on 06/18/2012

- **Correspondence Address**
- **Maintenance Fee Address**

The address of record for Customer Number 106095 is:

106095
Baker Botts LLP
2001 Ross Avenue, 6th Floor
Dallas, TX 75201

Netflix, Inc. - Ex. 1002, Page 000124
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	Jagane Sundar et al.
Serial No.:	11/694,483
Filing Date:	March 30, 2007
Confirmation No.:	1040
Group Art Unit:	2158
Examiner:	Angelica Ruiz
Title:	***Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images***

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Preliminary Amendment

Prior to the initial review of this non-provisional utility patent application entitled ***Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images***, please amend the application as follows.

In the Claims

1. **(Currently Amended)** A distributed computing system comprising:

~~**a plurality of application nodes interconnected via a communications network;**~~

a software image repository comprising non-transitory, computer-readable media operable to store: (i) one or more image instances of a virtual machine manager that is executable on ~~**the**~~ **a plurality of** application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, each virtual machine operable to provide an environment that emulates a computer platform, and (ii) one or more image instances of one or more software applications that are executable on the virtual machines; and

a control node that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the image instances of the software applications on the virtual machines.

2. (Original) The distributed computing system of claim 1, wherein the image instances of the one or more software applications comprise virtual disk image files.

3. (Original) The distributed computing system of claim 1, wherein when the automation infrastructure deploys the image of the one or more applications, the automation infrastructure copies a first one of the image instances to a first one of the application nodes.

4. (Original) The distributed computing system of claim 1, wherein an application image in the set of application images comprises an operating system.

5. (Original) The distributed computing system of claim 1, wherein an application image in the set of application images comprises a server application.

6. (Original) The distributed computing system of claim 1, wherein a first one of the virtual machines provides an environment that emulates an x86 platform.

7. (Original) The distributed computing system of claim 1, wherein the automation infrastructure captures the image of the virtual machine manager from an application node.

8. (Original) The distributed computing system of claim 7, wherein the automation infrastructure captures the image of the application from an application node executing the virtual machine manager.

9. (Original) The distributed computing system of claim 1, wherein the control node maintains the image instances as persistent or non-persistent based on a configuration.

10. (Original) The distributed computing system of claim 1, wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.

11. (Original) The distributed computing system of claim 10, wherein the rule engines are forward-chaining rule engines.

12. (Previously presented) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of an application matrix.

13. (Original) The distributed computing system of claim 1, wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.

14. (Original) The distributed computing system of claim 13, further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.

15. (Original) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

16. (Original) The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

17. (Original) The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

18. (Original) The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

19. (Original) The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model.

20. (Previously presented) A method comprising:

storing one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines, each virtual machine operable to provide an environment that emulates a computer platform;

storing one or more image instances of one or more software applications that are executable on the virtual machines;

executing a rules engine to perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

executing the rules engine to perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

21. (Original) The method of claim 20, wherein storing the image instances of the one or more software applications comprise storing virtual disk image files.

22. (Original) The method of claim 20, wherein performing operations to autonomically deploy the image instances comprises copying a first one of the image instances to a first one of the application nodes.

23. (Original) The method of claim 20, wherein the software applications comprise an operating system.

24. (Original) The method of claim 20,

further comprising generating an application matrix that specifies data for controlling the deployment of the software applications within a distributed computing system; and

wherein performing operations to provide autonomic control comprises provide autonomic control over the deployment, undeployment, and execution of the software applications on the virtual machines within the distributed computing system in accordance with parameters of the application matrix.

25. (Original) The method of claim 20, wherein performing operations comprises:

receiving status data for the distributed computing system having the plurality of application nodes interconnected via a communications network, wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system; and

performing the operations to provide autonomic control over the deployment of the software applications on the virtual machines within the distributed computing system in accordance with the rules.

26. (Original) The method of claim 20, further comprising accessing parameters of the application matrix and generating one or more rules for use in autonomically controlling the deployment of the software applications on the virtual machines.

27. (Original) The method of claim 26, further comprising processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

28. (Previously presented) A non-transitory, computer-readable medium comprising instructions stored thereon, that when executed by a programmable processor:

store one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines, each virtual machine operable to provide an environment that emulates a computer platform;

store one or more image instances of one or more software applications that are executable on the virtual machines;

perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

REMARKS

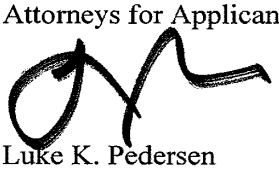
Early and favorable acceptance of this application is respectfully requested.

Although Applicants believe no fees are due at this time, the Commissioner is hereby authorized to charge any necessary fees and credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

If the Examiner believes a telephone conference would advance prosecution of this case in any way, the Examiner is invited to contact Luke K. Pedersen, Attorney for Applicants, at the Examiner's convenience at (214) 953-6655.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Applicants



Luke K. Pedersen
Reg. No. 45,003

Date: October 25, 2011

CORRESPONDENCE ADDRESS:

Customer Number: **05073**

Electronic Acknowledgement Receipt

EFS ID:	11261375
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	5073
Filer:	Bradley Jay Birchfield/Esmarie Garland
Filer Authorized By:	Bradley Jay Birchfield
Attorney Docket Number:	063170.9310
Receipt Date:	25-OCT-2011
Filing Date:	30-MAR-2007
Time Stamp:	15:43:13
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		0631709310PrelAmend.PDF	317960 4489dab524b2588aa14aea56c6d0e9c0ad438327	yes	8

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Preliminary Amendment	1	1
	Claims	2	7
	Applicant Arguments/Remarks Made in an Amendment	8	8
Warnings:			
Information:			
	Total Files Size (in bytes):	317960	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/694,483	03/30/2007	

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☒		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 10/25/2011

	(Column 1)		(Column 2)	(Column 3)	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 28	Minus	** 28	= 0	X $30 =	0	OR	X $ =	
Independent (37 CFR 1.16(h))	* 3	Minus	***3	= 0	X $125 =	0	OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE	

AMENDMENT

	(Column 1)		(Column 2)	(Column 3)					
	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/DIANE FLOYD/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/30 (10-01)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL

Address to:
Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	*11/694,483*
Filing Date	*March 30, 2007*
First Named Inventor	*Jagane Sundar*
Art Unit	*2158; Conf. #1040*
Examiner Name	*Angelica Ruiz*
Attorney Docket Number	*063170.9310*

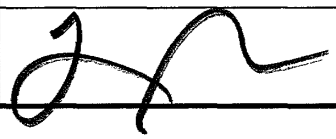
This is a Request for Continued Examination (RCE) under 37 C.F.R. 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. **Submission required under 37 C.F.R. § 1.114**
   - a. ☐ Previously submitted
     - i. ☐ Consider the amendment(s)/reply under 37 C.F.R. 1.116 filed.
       (Any unentered amendment(s) referred to above will be entered).
     - ii. ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____
     - iii. ☐ Other _____
   - b. ☒ **Enclosed**
     - i. ☒ **Amendment/Reply**
     - ii. ☐ Affidavit(s)/Declaration(s)
     - iii. ☐ Information Disclosure Statement (IDS)
     - iv. ☐ Other:

2. Miscellaneous
   - a. ☐ Suspension of action on the above-identified application is requested under 37 C.F.R. 1.103(c) for a period of _____months. (Period of suspension shall not exceed 3 months; Fee under 37 C.F.R. 1.117(i) **required**)
   - b. ☐ Other _____

3. **Fees** The RCE fee under 37 C.F.R. 1.17(e) is required by 37 C.F.R 1.114 when the RCE is filed.
   - a. ☒ **The Director is hereby authorized to charge any additional fee or credit any overpayment to Baker Botts L.L.P. Deposit Account No. 02-0384.**
     - i. ☒ **RCE fee required under 37 C.F.R. 1.17(e). The Director is hereby authorized to charge the fee of $810.00 to Baker Botts L.L.P. Deposit Account No. 02-0384.**
     - ii. ☐ A one-month extension of time and filing fee is being filed concurrently herewith.
     - iii. ☐ Other _____
   - b. ☐ Check in the amount of $_____ enclosed to cover the RCE fee
   - c. ☐ Payment by credit card (Form PTO-2038 enclosed)
     WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Name (Print /Type)	**Luke K. Pedersen**	*Registration No. (Attorney/Agent)*	**45,003**
Signature		*Date*	7-20-11

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND Fees and Completed Forms to the following address: Box RCE, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450.

DAL01:1172660

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	Jagane Sundar et al.
Serial No.:	11/694,483
Filing Date:	March 30, 2007
Confirmation No.:	1040
Group Art Unit:	2158
Examiner:	Angelica Ruiz
Title:	***Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images***

Mail Stop RCE

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

REQUEST FOR CONTINUED EXAMINATION

In response to the Final Office Action dated February 17, 2011, and the Notice of Panel Decision from Pre-Appeal Brief Review dated June 28, 2011, Applicants respectfully request continued examination to allow the Examiner to reconsider the rejection of the claims in view of the amendments and comments set forth below.

In the Claims

1. **(Currently Amended)** A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network;

a software image repository comprising non-transitory, computer-readable media operable to store: (i) one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines**, each virtual machine operable to provide an environment that emulates a computer platform**, and (ii) one or more image instances of one or more software applications that are executable on the virtual machines; and

a control node that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the image instances of the software applications on the virtual machines.

2. (Original) The distributed computing system of claim 1, wherein the image instances of the one or more software applications comprise virtual disk image files.

3. (Original) The distributed computing system of claim 1, wherein when the automation infrastructure deploys the image of the one or more applications, the automation infrastructure copies a first one of the image instances to a first one of the application nodes.

4. (Original) The distributed computing system of claim 1, wherein an application image in the set of application images comprises an operating system.

5. (Original) The distributed computing system of claim 1, wherein an application image in the set of application images comprises a server application.

6. (Original) The distributed computing system of claim 1, wherein a first one of the virtual machines provides an environment that emulates an x86 platform.

7. (Original) The distributed computing system of claim 1, wherein the automation infrastructure captures the image of the virtual machine manager from an application node.

8. (Original) The distributed computing system of claim 7, wherein the automation infrastructure captures the image of the application from an application node executing the virtual machine manager.

9. (Original) The distributed computing system of claim 1, wherein the control node maintains the image instances as persistent or non-persistent based on a configuration.

10. (Original) The distributed computing system of claim 1, wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.

11. (Original) The distributed computing system of claim 10, wherein the rule engines are forward-chaining rule engines.

12. (Previously presented) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of an application matrix.

13. (Original) The distributed computing system of claim 1, wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.

14. (Original) The distributed computing system of claim 13, further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.

15. (Original) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

16. (Original) The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

17. (Original) The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

18. (Original) The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

19. (Original) The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model.

20. **(Currently Amended)** A method comprising:

storing one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines**, each virtual machine operable to provide an environment that emulates a computer platform**;

storing one or more image instances of one or more software applications that are executable on the virtual machines;

executing a rules engine to perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

executing the rules engine to perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

21. (Original) The method of claim 20, wherein storing the image instances of the one or more software applications comprise storing virtual disk image files.

22. (Original) The method of claim 20, wherein performing operations to autonomically deploy the image instances comprises copying a first one of the image instances to a first one of the application nodes.

23. (Original) The method of claim 20, wherein the software applications comprise an operating system.

24. (Original) The method of claim 20,

further comprising generating an application matrix that specifies data for controlling the deployment of the software applications within a distributed computing system; and

wherein performing operations to provide autonomic control comprises provide autonomic control over the deployment, undeployment, and execution of the software applications on the virtual machines within the distributed computing system in accordance with parameters of the application matrix.

25. (Original) The method of claim 20, wherein performing operations comprises:

receiving status data for the distributed computing system having the plurality of application nodes interconnected via a communications network, wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system; and

performing the operations to provide autonomic control over the deployment of the software applications on the virtual machines within the distributed computing system in accordance with the rules.

26. (Original) The method of claim 20, further comprising accessing parameters of the application matrix and generating one or more rules for use in autonomically controlling the deployment of the software applications on the virtual machines.

27. (Original) The method of claim 26, further comprising processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

28. **(Currently Amended)** A non-transitory, computer-readable medium comprising instructions stored thereon, that when executed by a programmable processor:

store one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines**, each virtual machine operable to provide an environment that emulates a computer platform**;

store one or more image instances of one or more software applications that are executable on the virtual machines;

perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

REMARKS

The Application has been reviewed in light of the Final Office Action dated February 17, 2011 ("*Office Action*") and the Notice of Panel Decision from Pre-Appeal Brief Review dated June 28, 2011. Claims 1-28 are pending and stand rejected. To advance prosecution of the Application, Applicants have responded to each notation by the Examiner. Applicants respectfully request reconsideration and favorable action in this case.

Section 102 Rejections

The Examiner rejects Claims 1-16, 18-24, and 26-28 under 35 U.S.C. § 102(b) as being anticipated by U.S. Publication No. 2005/0005200 filed by Matena et al. ("*Matena*"). Applicants respectfully traverse these claim rejections for at least the reasons below.

At the outset, Applicants note that the Examiner relies on the Java classes in *Matena* to teach both the "virtual machine" *and* the "instances of the software applications on the virtual machine," as required by Claim 1. For example, the Examiner relies on *Matena* at [0112]–[0113] to teach "autonomic deployment of the image instances of the software applications on the virtual machine." *Office Action* pg. 6. In those passages, *Matena* discloses an application controller that can automatically distribute application modules to nodes. *See Matena* at [0112]–[0113]. According to an earlier passage in *Matena*, "application modules are, for example, operating-system level processes . . . ***such as execution modules of Java applications***." *Matena* at [0089] (emphasis added). Such execution modules are later described as "***includ[ing] the application's Java classes*** loaded into the container." *Matena* at [0090] (emphasis added). That is, the Java classes and application modules of *Matena* are one in the same; *Matena* discloses Java classes as an example of application modules.

Therefore, when the Examiner relies on the application modules in *Matena* to teach the "image instances of the software applications on the virtual machine" as recited in Claim 1, the Examiner also relies on Java classes in *Matena* to teach "image instances of the software applications on the virtual machine" as well. Earlier, however, the Examiner contends that the Java classes in *Matena* disclose a "virtual machine executing on the application nodes." *Office Action* pg. 5 ("'Java classes' being the 'virtual machine'"). That is, the Examiner relies upon the Java classes in *Matena* to teach *both* the "virtual machine"

and the "instances of the software applications on the virtual machine" recited in Claim 1. In doing so, the Examiner improperly relies on the Java classes in *Matena* to teach two distinct and separate limitations of Claim 1.

Nevertheless, *Matena* fails to disclose, teach, or suggest every limitation of Claim 1. For example, *Matena* fails to disclose, teach, or suggest "a software image repository comprising non-transitory, computer-readable media operable to store . . . one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines." The Examiner contends that the Java classes disclosed in *Matena* teach a "'virtual machine' executing on the application nodes." *Office Action* pg. 5. First, Applicants note that Claim 1 does not recite a "virtual machine executing on the application nodes" as the Examiner indicates. Rather, Claim 1 recites "one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines." Second, even assuming the Java classes in *Matena* may be construed as virtual machines executing on application nodes as the Examiner indicates (which Applicants do not believe or concede), the Examiner fails to point to, and *Matena* does not disclose, a "virtual machine ***manager***," much less "one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, ***the image instances of the virtual machine manager provide one or more virtual machines***" as Claim 1 requires.

Moreover, *Matena* fails to disclose "wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, each virtual machine operable to provide an environment that emulates a computer platform." The Examiner points to *Matena* at [0193] as teaching a previous version of this limitation, stating that the cited passage "includes the instance of the 'virtual machine' as claimed." *Office Action*, pg. 5. However, the cited passage merely discloses a "container" that is a "process that includes a Java Virtual Machine ('JVM') and application server classes." *Matena* at [0193]. The cited passage does not disclose "image instances of [a] virtual machine manager provid[ing] one or more virtual machines," much less "***virtual machine[s] operable to provide an environment that emulates a computer platform***" as Claim 1 requires.

For at least these reasons, independent Claim 1 and its dependent claims are allowable under 35 U.S.C. § 102. For analogous reasons, independent Claims 20 and 28 and their respective dependent claims are also allowable under 35 U.S.C. § 102. Accordingly, Applicants respectfully request reconsideration and allowance of Claims 1-16, 18-24, and 26-28.

Section 103 Rejections

The Examiner rejects Claims 17 and 25 under 35 U.S.C. § 103(a) as being unpatentable over *Matena* in view of U.S. Publication No. 2004/0181794 filed by Coleman et al. ("*Coleman*"). Applicants respectfully traverse these rejections. In the rejections of Claims 17 and 25, the Examiner relies upon *Matena* to teach every limitation of Claims 1 and 20, respectively. However, for at least the reasons discussed above, *Matena* fails to disclose, teach or suggest every limitation of Claims 1 and 20. Therefore, Applicants contend that Claims 17 and 25 are allowable for at least the same reasons discussed above with regard to their respective base claims. Accordingly, Applicants respectfully request reconsideration and allowance of Claims 17 and 25.

No Waiver

All of Applicants' arguments and amendments are without prejudice or disclaimer. Additionally, Applicants have merely discussed example distinctions from the *Matena* reference. Other distinctions may exist, and Applicants reserve the right to discuss these additional distinctions in a later Response or on Appeal, if appropriate. By not responding to additional statements made by the Examiner, Applicants do not acquiesce to the Examiner's additional statements. The example distinctions discussed by Applicants are sufficient to overcome the anticipation and obviousness rejections.

CONCLUSION

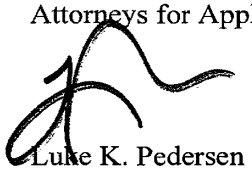
Applicants have made an earnest attempt to place this case in condition for allowance. For the foregoing reasons, and for other apparent reasons, Applicants respectfully request full allowance of all pending claims.

The Examiner is hereby authorized to charge $810.00 for this continued examination, and to the extent necessary, charge any additional fees or to credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

If the Examiner believes a telephone conference would advance prosecution of this case in any way, the Examiner is invited to contact Luke K. Pedersen, Attorney for Applicants, at the Examiner's convenience at (214) 953-6655.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Applicants



Luke K. Pedersen
Reg. No. 45,003

Date: July 20, 2011

CORRESPONDENCE ADDRESS:

Customer Number: **05073**

Electronic Patent Application Fee Transmittal

Application Number:	11694483
Filing Date:	30-Mar-2007
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Filer:	Bradley Jay Birchfield/Esmarie Garland
Attorney Docket Number:	063170.9310

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Request for continued examination	1801	1	810	810
		Total in USD ($)		**810**

Electronic Acknowledgement Receipt

EFS ID:	10562523
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	05073
Filer:	Bradley Jay Birchfield/Esmarie Garland
Filer Authorized By:	Bradley Jay Birchfield
Attorney Docket Number:	063170.9310
Receipt Date:	20-JUL-2011
Filing Date:	30-MAR-2007
Time Stamp:	16:38:10
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$810
RAM confirmation Number	4807
Deposit Account	020384
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees) Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)	

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)	
Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)	
Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		0631709310RCEwResp.PDF	587917 3e32e1981000a18bd9b9171a39c861424dbc6021	yes	12

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Request for Continued Examination (RCE)	1	1
Amendment Submitted/Entered with Filing of CPA/RCE	2	2
Claims	3	8
Applicant Arguments/Remarks Made in an Amendment	9	12

Warnings:					
Information:					
2	Fee Worksheet (SB06)	fee-info.pdf	30625 85579a6fe8db81d632932a21318fd580bdc4ab7a	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	618542		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/694,483	03/30/2007	

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☒		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 07/20/2011	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 28	Minus	** 28	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	* 3	Minus	***3	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

AMENDMENT	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/ELMIRA HALL/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

5073 7590 06/28/2011

BAKER BOTTS L.L.P.
2001 ROSS AVENUE
SUITE 600
DALLAS, TX 75201-2980

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

MAIL DATE	DELIVERY MODE
06/28/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of Panel Decision from Pre-Appeal Brief Review	Application/Control No.	Applicant(s)/Patent under Reexamination
	11/694,483	SUNDAR ET AL.
	ANGELICA RUIZ	Art Unit: 2158

This is in response to the Pre-Appeal Brief Request for Review filed 16 May 2011.

1. ☐ **Improper Request** – The Request is improper and a conference will not be held for the following reason(s):

☐ The Notice of Appeal has not been filed concurrent with the Pre-Appeal Brief Request.
☐ The request does not include reasons why a review is appropriate.
☐ A proposed amendment is included with the Pre-Appeal Brief request.
☐ Other: .

The time period for filing a response continues to run from the receipt date of the Notice of Appeal or from the mail date of the last Office communication, if no Notice of Appeal has been received.

2. ☒ **Proceed to Board of Patent Appeals and Interferences** – A Pre-Appeal Brief conference has been held. The application remains under appeal because there is at least one actual issue for appeal. Applicant is required to submit an appeal brief in accordance with 37 CFR 41.37. The time period for filing an appeal brief will be reset to be one month from mailing this decision, or the balance of the two-month time period running from the receipt of the notice of appeal, whichever is greater. Further, the time period for filing of the appeal brief is extendible under 37 CFR 1.136 based upon the mail date of this decision or the receipt date of the notice of appeal, as applicable.

☒ The panel has determined the status of the claim(s) is as follows:
Claim(s) allowed: _____.
Claim(s) objected to: _____.
Claim(s) rejected: *1-28*.
Claim(s) withdrawn from consideration: _____.

3. ☐ **Allowable application** – A conference has been held. The rejection is withdrawn and a Notice of Allowance will be mailed. Prosecution on the merits remains closed. No further action is required by applicant at this time.

4. ☐ **Reopen Prosecution** – A conference has been held. The rejection is withdrawn and a new Office action will be mailed. No further action is required by applicant at this time.

All participants:

(1) *ANGELICA RUIZ*.

(2) *MOHAMMAD ALI*.

(3) *LESLIE WONG*.

(4) _____.

/Mohammad Ali/
Supervisory Patent Examiner, Art Unit 2158

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	Jagane Sundar et al.
Serial No.:	11/694,483
Filing Date:	March 30, 2007
Confirmation No.:	1040
Group Art Unit:	2158
Examiner:	Angelica Ruiz
Title:	***Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images***

Mail Stop AF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

PRE-APPEAL BRIEF REQUEST FOR REVIEW

The following Pre-Appeal Brief Request for Review ("*Request*") is being filed in accordance with the provisions set forth in the Official Gazette Notice of July 12, 2005 ("*OG Notice*"). Pursuant to the *OG Notice*, this Request is being filed concurrently with a Notice of Appeal. Applicants respectfully request reconsideration of the Application in light of the remarks set forth below.

Netflix, Inc. - Ex. 1002, Page 000155
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

REMARKS

Previously, Applicants received a Final Office Action dated February 17, 2011 ("*Office Action*"). At the time of the *Office Action*, Claims 1-28 were pending and rejected. For brevity, the Request only seeks review of the rejections of Claims 4, 6, and 23. Applicants do not seek review of Claims 1-3, 5, 7-22 and 24-28. As the rejections of these Claims contain clear legal and factual deficiencies, Applicants request the Panel to find the rejections improper and withdraw them.

i. Claim 4, 23

The Examiner rejects Claims 4 and 23 under 35 U.S.C. § 102(b) as being anticipated by U.S. Publication No. 2005/0005200 filed by Matena et al. ("*Matena*"). Applicants respectfully traverse these rejections.

At the outset, Applicants note that the Examiner relies on the Java classes in *Matena* to teach both the "virtual machine" *and* the "instances of the software applications on the virtual machine," as recited in Claim 1 and incorporated by Claim 4. For example, the Examiner relies on *Matena* at [0112]–[0113] to teach "autonomic deployment of the image instances of the software applications on the virtual machine." *Office Action* pg. 6. In those passages, *Matena* discloses an application controller that can automatically distribute application modules to nodes. *See Matena* at [0112]–[0113]. According to an earlier passage in *Matena*, "application modules are, for example, operating-system level processes . . . such as execution modules of Java applications." *Matena* at [0089]. Such execution modules are later described as "includ[ing] the application's Java classes loaded into the container." *Matena* at [0090]. Thus, the Java classes and application modules of *Matena* are one in the same. *Matena* merely discloses Java classes as a subset of application modules.

Therefore, when the Examiner relies on application modules in *Matena* to teach the "image instances of the software applications on the virtual machine" required by Claim 4, the Examiner also relies on Java classes in *Matena* to teach "image instances of the software applications on the virtual machine" as well. Earlier, however, the Examiner contends that the Java classes in *Matena* disclose a "virtual machine executing on the application nodes." *Office Action* pg. 5. That is, the Examiner relies upon the Java classes in *Matena* to teach *both* the "virtual machine" *and* the "instances of the software applications on the virtual

Netflix, Inc. - Ex. 1002, Page 000156
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

machine" required by Claim 4. In doing so, the Examiner improperly relies on the Java classes in *Matena* to teach two distinct and separate limitations of Claim 4.

Even assuming the Examiner's mapping in the *Office Action* is correct (which Applicants do not concede), *Matena* fails to disclose, teach, or suggest "one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines" as recited in Claim 1 and incorporated by Claim 4. The Examiner contends that the Java classes in *Matena* disclose a "'virtual machine' executing on the application nodes." *Office Action* pg. 5. However, the Examiner fails to point to "one or more image instances of a virtual machine manager," much less "wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines" as Claim 4 requires.

In addition, *Matena* fails to disclose, teach, or suggest "one or more image instances of one or more software applications that are executable on the virtual machines" as recited in Claim 1 and incorporated by Claim 4. The Examiner first contends that the Java classes in *Matena* disclose a "'virtual machine' executing on the application nodes." *Office Action* pg. 5. The Examiner then cites to *Matena* at [0015], which merely discloses "the execution controls system provides a single-system image of the distributed computer system and applications running on it." *Office Action* pg. 5. The Examiner, however, does not point to "one or more image instances of one or more software applications that are ***executable on the virtual machines***." (emphasis added). Even assuming that Java classes can be construed as being virtual machines (which Applicants do not concede), the Examiner fails to establish how the Java classes in *Matena* acting as virtual machines can execute applications as Claim 4 requires.

Furthermore, *Matena* fails to disclose, teach, or suggest "wherein an application image in the set of application images comprises an operating system" as Claim 4 further requires. The Examiner relies on *Matena* at [0084] as allegedly teaching this element of Claim 4. *Office Action*, pg. 7. However, that passage merely discloses an "execution control system" that "includes three application controllers: a service application controller, which is suitable for controlling applications that are services including operating-system level processes . . . ." *Matena* at [0084]. That is, *Matena* discloses an application controller that can control operating-system level processes. *Matena* fails to disclose "one or more image

Netflix, Inc. - Ex. 1002, Page 000157
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

instances of one or more software applications that are executable on the virtual machines," much less "wherein an application image in the set of application images comprises an operating system" as Claim 4 requires. Accordingly, *Matena* fails to disclose, teach, or suggest every element of Claim 4.

For at least these reasons, Claim 4 is allowable over *Matena*. Claim 23 contains limitations that are similar to those discussed above with regard to Claim 4. Accordingly, Applicants respectfully request reconsideration and allowance of Claims 4 and 23.

ii. Claim 6

The Examiner also rejects Claim 6 under 35 U.S.C. § 102(b) as being anticipated by *Matena*. Applicants respectfully traverse this rejection.

In rejecting dependent Claim 6, the Examiner relies on *Matena* to teach every element of Claim 1. However, for at least the reasons given above, *Matena* fails to disclose, teach, or suggest every element of Claim 1. Because Claim 6 incorporates every element of Claim 1, *Matena* therefore also fails to disclose, teach, or suggest every element of Claim 6.

In addition, *Matena* fails to disclose, teach, or suggest, "wherein a first one of the virtual machines provides an environment that emulates an x86 platform" as Claim 6 further requires. The Examiner relies on *Matena* at [0412] as allegedly teaching this element. *Office Action*, pg. 8. That passage merely discloses "nodes [that] are realized as general-purpose server computers or server blades," where "each node includes an image of a general purpose operating system . . . ." *Matena* at [0412]. The cited passage of *Matena* does not disclose, however, "one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, wherein a first one of the virtual machines provides an environment that emulates an x86 platform." Moreover, even assuming that Java classes can be construed as being virtual machines as the Examiner contends (*see Office Action* pg. 5; Applicants do not concede this point), the Examiner fails to show how Java classes acting as "virtual machines" may "provide[] an environment that emulates an x86 platform" as Claim 6 requires. Accordingly, *Matena* fails to disclose, teach, or suggest every element of Claim 6.

For at least these reasons, Claim 6 is allowable over *Matena*. Accordingly, Applicants respectfully request reconsideration and allowance of Claim 6.

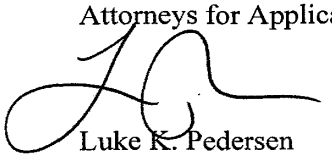
Netflix, Inc. - Ex. 1002, Page 000158
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CONCLUSION

As the rejections of Claims 4, 6, and 23 contain clear legal and factual deficiencies, Applicants respectfully request a finding of allowance of Claims 4, 6, and 23. If the PTO determines that an interview is appropriate, Applicants would appreciate the opportunity to participate in such an interview. To the extent necessary, the Commissioner is hereby authorized to charge any required fees or credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Applicant



Luke K. Pedersen
Reg. No. 45,003
Tel. 214.953.6655

Date: May 16, 2011

Customer No. **05073**

Netflix, Inc. - Ex. 1002, Page 000159
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Electronic Patent Application Fee Transmittal

Application Number:	11694483
Filing Date:	30-Mar-2007
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Filer:	Bradley Jay Birchfield/Esmarie Garland
Attorney Docket Number:	063170.9310

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Notice of appeal	1401	1	540	540
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**540**

Electronic Acknowledgement Receipt

EFS ID:	10097726
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	05073
Filer:	Bradley Jay Birchfield/Esmarie Garland
Filer Authorized By:	Bradley Jay Birchfield
Attorney Docket Number:	063170.9310
Receipt Date:	16-MAY-2011
Filing Date:	30-MAR-2007
Time Stamp:	15:37:24
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$540
RAM confirmation Number	2075
Deposit Account	020384
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees) Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)	

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Notice of Appeal Filed	0631709310NoticeofAppeal.PDF	52881 95931bc3f09bc4c3ca414d97a88144c6726dbbc9	no	2
Warnings:					
Information:					
2	Pre-Brief Conference request	0631709310PreAppBrf.PDF	254586 fd9a49922da251c5a937ed55ad6f9d18db88d54d	no	5
Warnings:					
Information:					
3	Fee Worksheet (PTO-875)	fee-info.pdf	30561 f6cd12189a2e3183917a9c6e0030001a924b3488	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	338028		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	Jagane Sundar et al.
Serial No.:	11/694,483
Filing Date:	March 30, 2007
Confirmation No.:	1040
Group Art Unit:	2158
Examiner:	Angelica Ruiz
Title:	***Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images***

Mail Stop AF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Notice of Appeal

In response to the Final Office Action dated February 17, 2011, Applicants respectfully initiates an appeal and respectfully requests the Board of Patent Appeals and Interferences to reconsider the rejections to the claims and objections to the Application.

Netflix, Inc. - Ex. 1002, Page 000164
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

REMARKS

Applicants hereby appeal to the Board of Patent Appeals and Interferences from the decisions of the Examiner in the Final Office Action dated February 17, 2011 rejecting the pending claims.

Attached herewith is a Pre-Appeal Brief Request for Review for consideration with this Notice of Appeal.

The Commissioner is hereby authorized to charge the $540.00 fee for this Notice Appeal, and to the extent necessary, charge any additional fees or credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

If the Examiner feels that a telephone conference would advance prosecution of this application in any manner, the Examiner is invited to contact Luke K. Pedersen, Attorney for Applicants, at the Examiner's convenience at (214) 953-6655.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Applicants

Luke K. Pedersen
Reg. No. 45,003

Date: May 16, 2011

Customer Number: **05073**

Netflix, Inc. - Ex. 1002, Page 000165
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

5073 7590 02/17/2011

BAKER BOTTS L.L.P.
2001 ROSS AVENUE
SUITE 600
DALLAS, TX 75201-2980

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

NOTIFICATION DATE	DELIVERY MODE
02/17/2011	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

ptomail1@bakerbotts.com
glenda.orrantia@bakerbotts.com

Netflix, Inc. - Ex. 1002, Page 000166
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Office Action Summary	Application No.	Applicant(s)	
	11/694,483	SUNDAR ET AL.	
	Examiner	Art Unit	
	ANGELICA RUIZ	2158	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 December 2010*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. The Action is responsive to Applicant's amendment, filed on December 3, 2010.

2. It is acknowledged that as a result of the amendment, Claims 1, 12, and 28.

3. Claims 1-28 are pending.

Response to Arguments

4. Applicant's arguments with respect to claims 1-28 have been considered but are not persuasive at least for the following reason:

Applicant argues that the prior art of record fails to disclose, teach, or suggest "a software image repository comprising…one or more image instances of one or more software applications that are executable on the virtual machine" The Examiner respectfully submits that the claim language clearly states that "one or more" the Examiner relied in the claim language to provide the mentioned citations that refer to the *one or more image instances of a virtual machine manager that is executable on the application nodes,* (Par [0090], "Java classes" being the "virtual machine" executing on the application nodes, as claimed), *wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines,* (Par [0193], includes the instance of the "virtual machine" as claimed*) and (ii) one or more image instances of one or more software applications that are executable on the virtual machines;* (Par [0015], "The application controllers utilize the execution

controller. To its users, the execution *control system provides a single-system image of the distributed computer system and applications running on it*…"; Par [0195] and see also Figures 21-22, that shows the "java virtual Machine" and the "application server classes" the mentioned figure clearly shows the recited limitations of "one or more virtual machine" and the "one or more software application that are executable on the virtual machines" as claimed.

The claims and only the claims form the metes and bounds of the invention. "Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023, 1027-28 (Fed. Cir. 1997). *Limitations appearing in the specification but not recited in the claim are not read into the claim*. In re Prater, 415 F.2d 1393, 1404-05, 162 USPQ 541,550-551 (CCPA 1969)" (MPEP p 2100-8, c 2, l 45-48; p 2100-9, c 1, l 1-4). The Examiner has full latitude to interpret each claim in the broadest reasonable sense. The Examiner will reference prior art using terminology familiar to one of ordinary skill in the art. Such an approach is broad in concept and can be either explicit or implicit in meaning.

In view of the above, the Examiner contends that all limitations as recited in the claims have been addressed in this Action.

For the above reasons, the Examiner believes that the rejections of the last Office action were proper.

Specification

5. In view of amendment to the Abstract, the Examiner withdraws the pending objection.

Claim Rejections - 35 USC § 112

6. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior office action.

7. In view of the amendments to the claims, the Examiner withdraws all pending rejections under 35 U.S.C. 112.

Claim Rejections - 35 USC § 101

8. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior office action.

9. In view of the amendments to the claims, the Examiner withdraws all pending rejections under 35 U.S.C. 101.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. **Claims 1 – 16, 18 –24 and 26 –28** are rejected under 35 U.S.C. 102(b) as being anticipated by Matena et al (US 2005/0005200 A1), hereinafter "Matena".

As per Claim 1, Matena discloses:

A distributed computing system comprising: a plurality of application nodes interconnected via a communications network; (Title, "Method and apparatus for executing applications on a distributed computer system") and (Abstract, "multiple nodes interconnected through a network…") ***a software image repository comprising non-transitory, computer-readable media operable to store:*** (Par [0015] and (Par [0081], "…creation of execution modules to store new server objects.") ***(i) one or more image instances of a virtual machine manager that is executable on the application nodes,*** (Par [0090], "Java classes" being the "virtual machine" executing on the application nodes, as claimed), ***wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines,*** (Par [0193], includes the instance of the "virtual machine" as claimed) ***and (ii) one or more image instances of one or more software applications that are executable on the virtual machines;*** (Par [0015], "The application controllers utilize the execution controller. To its users, the execution *control system provides a single-system image of the distributed computer system and applications running on it…*"; Par [0195] and see also Figures 21-22) ***and a control node*** (Par [0015], "one or more node controllers, and one or more application controllers….") ***that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes*** (Par [0085], "The node controllers 103, execution controller 102 and application controllers are distributed over the nodes of a distributed computer system."; see figure 2) ***and to provide autonomic deployment of the image instances of the***

software applications on the virtual machines. (Par [0089]; par [0112], "can automatically distribute application modules of multiple applications to the node…"; par [0113], "… *same application could be deployed on several distributed computer systems, each having a different configuration, and each possibly requiring different assignment of application module replicas* to nodes.", the "replicas" being the "images" as claimed).

As per Claim 2, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the image instances of the one or more software applications comprise virtual disk image files.***

(Par [0108], "In some embodiments of the invention, the distributed computer system may include *nodes with different operating systems*. ...To shield the other components of the execution control system from the differences in the command line format, the *NC API 707 uses a virtual, operating-system* neutral format of the command line. The node controllers 701 have the logic to translate the virtual command line to the format understood by the local operating system.").

As per Claim 3, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein when the automation infrastructure deploys the image of the one or more applications, the automation infrastructure copies a first one of the image instances to a first one of the application nodes.***

(Par [0251], "…Also, embodiments replicate the state of each application execution module on multiple nodes. Each copy of the application execution module is called a

replica. If one replica of an execution module fails, other replicas of the execution module remain available, thereby allowing the application to provide service to its users.") and (Par [0289], "…A purpose of the replication constraint is *to specify that if one replica of an execution module is assigned to a node that is included in the first node group, then the other replica of the same execution module should be assigned to a node included in the second node group*.").

As per Claim 4, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein an application image in the set of application images comprises an operating system.***

(Par [0084], " For example, the exemplary ECS 101 in FIG. 1 includes three application controllers: a service application controller 104, which is suitable for controlling applications that are services 107 including operating-system level processes; a Java application controller 105, which is suitable for controlling Java applications 108 including containers and execution modules (container and execution modules are explained later in this disclosure); and custom application controller 106, which is suitable for controlling applications 109 that have a custom structure and use a custom application execution model.").

As per Claim 5, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein an application image in the set of application images comprises a server application.***

(Par [0081], "The disclosure further describes how an *application controller controls*

remote invocation and creation of server objects located in the execution modules. The advantages include location transparency of server objects; ability to relocate server objects; transparent failure recovery; partitioning of server objects; and on-demand creation of execution modules to store new server objects.").

As per Claim 6, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein a first one of the virtual machines provides an environment that emulates an x86 platform.***

(Par [0412], "In one embodiment of the invention, nodes are realized as general-purpose server computers or server blades. Each node includes an image of a general-purpose operating system (e.g. Linux, Solaris, or Windows). Node controllers are realized as a Java 2 Platform, Standard Edition ("J2SE") applications…" the x86 being an instance of the Windows operating system, as claimed).

As per Claim 7, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the automation infrastructure captures the image of the virtual machine manager from an application node.***

(Par [0193], "JAC manages how the containers are distributed over the nodes of the distributed system and how the application execution modules are distributed over the containers. FIG. 21 also depicts the parts of the container used in an embodiment of the invention. *The container 2102 is a process that includes a Java Virtual Machine ("JVM") 2104 and application server classes 2105.* The application server classes 2105 include Java classes that implement the operations that *JAC uses to manage the*

lifecycle of the execution modules (such as the classes that implement the creation of execution modules) and Java classes that are provided as runtime library classes for the execution modules (such as classes that implement the Enterprise JavaBeans container functions).").

As per Claim 8, the rejection of Claim 7 is incorporated and further Matena discloses: ***wherein the automation infrastructure captures the image of the application from an application node executing the virtual machine manager.***

(Par [0193], "JAC manages how the *containers are distributed over the nodes of the distributed system* and how the application execution modules are distributed over the containers. FIG. 21 also depicts the parts of the container used in an embodiment of the invention. *The container 2102 is a process that includes a Java Virtual Machine ("JVM") 2104 and application server classes 2105…." which is captured from the "application node" as claimed).*

As per Claim 9, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node maintains the image instances as persistent or non-persistent based on a configuration.***

(Par [0105], "…When a node controller detects a process failure, it sends an event notification to all the subscribers for the event. For example, the execution controller subscribes to the process failure events so that it can maintain an up-to-date state model of the processes in the distributed system." The "event notification" when "up-to-date" being the "persistent" as claimed).

As per Claim 10, the rejection of Claim 1 is incorporated and further Matena discloses:

wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.

(Par [0267], " In the embodiment of the invention depicted in FIG. 36, JAC uses the Distribution Manager objects, Replication Constraint policy objects, the Container Distribution Policy objects, and EM Distribution Policy objects when making decisions to which nodes and to which containers to assign execution module replicas. The objective of this flexible, policy-based distribution of replicas…" which shows replication and distribution policy object to determine whether to deploy an application).

As per Claim 11, the rejection of Claim 10 is incorporated and further Matena discloses:

wherein the rule engines are forward- chaining rule engines.

(Par [0271], "The distributed computer system then observes the
above rules when assigning execution module replicas to the nodes of the distributed computer system. For example, the replication constraint depicted in FIG. 37 constrains the assignment of replicas such that if the first replica of an execution module is assigned to one of the nodes A, B, or C, then the second replica of the same execution module is assigned to one of the nodes D, E, or F. The replication constraint further specifies that there should be two replicas of the execution module and that the replication type is Hot-Standby. For example, since one replica of the execution module

Auction-1 is on node A, the second replica can be only on one of the nodes D, E, or F (in the example, it is on Node E)" being the "forward-chaining rule" as claimed).

As per Claim 12, the rejection of Claim 10 is incorporated and further Matena discloses:

wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of an application matrix.

(Par [0121], "The *state model* includes objects that present nodes 909, node groups 910, processes 911, and application controllers 912 in the distributed computer system. The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system.", that being the "current state" as claimed; the "State model" being the "application matrix" as claimed; Par [0195], "…In some embodiments of the invention, the information in the deployment descriptor is represented using Extensible Markup Language (XML) technologies.", the information described as an "XML" being the application matrix, as claimed; par [0153] As the Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…").

As per Claim 13, the rejection of Claim 1 is incorporated and further Matena discloses:

wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the

software applications to the virtual machines within the distributed computing system.

(Par [0121], "The *state model* includes objects that present nodes 909, node groups 910, processes 911, and application controllers 912 in the distributed computer system. The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system.", the "State model" being the "application matrix" as claimed; Par [0195], "…In some embodiments of the invention, the information in the *deployment descriptor* is represented using Extensible Markup Language (XML) technologies.", par [0153] As the Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…" Par [0142], "The Distribution Policy objects 1014 *provide parameters to the algorithm of the Distribution Manager objects* 1010…" Par [0201] Application definition 2 2302 includes a deployment descriptor 2304….").

As per Claim 14, the rejection of Claim 1 is incorporated and further Matena discloses: ***further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.***

(Par [0084], "For example, the exemplary ECS 101 in FIG. 1 includes three application controllers: a service application controller 104, which is suitable for controlling applications that are services 107 including operating-system level processes; a Java application controller 105, which is suitable for controlling Java applications 108

including containers and execution modules…" the "application controller" being the "governor" as claimed).

As per Claim 15, the rejection of Claim 10 is incorporated and further Matena discloses:

wherein the automation infrastructure automatically updates the rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

(Par [0121], "The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system."; Par [0170], "In response to the "Process Started" message, the execution controller updates the state of Process 1 object in its state model to indicate that the process has successfully started 1310. Then the execution controller sends the "Process Started" message to SAC 1311. SAC updates its internal Process 1 object in its state model to reflect that process 1 has successfully started 1312. This completes the steps of starting a process. SAC checks if there are more service processes that should be started 1317. If yes, the steps of starting a process are repeated. If no, the start service operation has completed.").

As per Claim 16, the rejection of Claim 11 is incorporated and further Matena discloses:

wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

(Par [0271], “In some embodiments of the invention, the application programmer or system administrator defines the node groups, replication groups, and replication constraints. The distributed computer system then observes the above rules when assigning execution module replicas to the nodes of the distributed computer system.”).

As per Claim 18, the rejection of Claim 11 is incorporated and further Matena discloses:

wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

(See Figures 1 and 2, that shows the organization into “tiers” as claimed; see Par [0117], “The "node group management" 905 operations include the operations to create and remove a node group; operations to add and remove a node from a node group; and operations to obtain status information about the node groups.”).

As per Claim 19, the rejection of Claim 1 is incorporated and further Matena discloses:

further comprising: a database storing data that defines a model for a hierarchical organization of the distributed computing system, (Par [0097], “FIG. 6 illustrates a representational node 601 suitable for being used in a method and system of the

invention. A node is a computer system including one or more central-processing units (CPU) 602, main memory 603, optional secondary storage 604, and one or more communication interfaces 605…")

wherein the model specifies a fabric having a one or more domains, (See Figure 8)

and wherein each domain has at least one tier that includes at least one of the application nodes, (Par [0099], "Each node of a distributed computer system includes a node controller 701. FIG. 7 depicts the functionality provided by a node controller and its main components. A node controller 701 implements the following functions: node membership 702, master election 703, start process 704, stop process 705, and detect process failure 706.").

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model. (Par [0112], "A node group is a named subset of the nodes in the distributed computer system. A node can be a member of multiple node groups. Using node groups in a distributed computer system has several advantages. First, node groups simplify many management tasks because instead of listing a large number of individual nodes when performing system a administration task, a node group name can be given instead. Second, it is easy to adjust the processing capacity or distribution of multiple applications by simply adding or removing nodes from a node group associated with the applications. And (Par [0113])

12. **As per Claims 20-24 and 26-27** being the method claims corresponding to the system claims 1-4, 13, 12, 11 respectively and rejected under the same reason set forth

in connection of the rejections of Claims 1 -4, 13, 12, 11 and further Matena discloses: (Title, "Method and apparatus for executing applications on a distributed computer system").

13. **As per Claim 28** being the computer readable medium claim corresponding to the system claims 1 respectively and rejected under the same reason set forth in connection of the rejections of Claim 1 and further Matena discloses: (Abstract and Claim 114, "A computer-readable medium including instructions for performing a method when executed by a processor").

Claim Rejections - 35 USC § 103

14. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

15. **Claims 17 and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over Matena et al (US 2005/0005200 A1), in view of Coleman et al (US 2004/0181794 A1).

As per Claim 17, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises:*** (Par [0015], "one or more node controllers, and one or more application controllers….").

Matena do not disclose the following features:

Coleman discloses the claimed features as follows:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes (Par [0023] The deployment manager 130 interfaces with relevant resources of the application level 140 to monitor measurement/configuration data…"; see Figures 1- 2, feature 202, the "forecaster" being the "subsystem" as claimed that monitors the data and provides the status information) ***and a business logic tier that provides expected state data to the automation subsystem***, (Par [0019], "The solution manager 120, itself, comprises a control logic engine 122, a forecaster module 124, a performance estimator module 126, an admission control module 128, and a tuning module 129. The deployment manager 130, itself, comprises a monitoring module 132…" par [0024], "e-commerce site...").

wherein the expected state data represents an expected state for the application nodes, (Par [0034], "Based on monitored data such as the transaction rate history, the solution manager 120 *uses the forecaster module 124 to project into the future the expected workload* (step 204)…").

wherein the rule engines analyze the status data from the monitoring subsystem (Par [0034] "…The forecast can be long-term (e.g., hours/days) or short-term (e.g., seconds/minutes). To deal with unanticipated changes in the workload (e.g., outside of normal daily/weekly, etc.) variations, the forecaster *is preferably adaptive and rapidly learns from recent history.* The forecast horizon does not need to be long, e.g., the horizon may be approximately the amount of time needed for the resource action (application provisioning in this situation) to take effect.").

and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state. (Par [0036], "*The solution manager 120 then uses the performance estimator 126 to check if the current resource deployment is insufficient, sufficient (i.e., adequate) or overly sufficient (i.e., excessive) for maintaining the service objective threshold* (e.g., response time) based on the recent and forecasted workload traffic (step 206)…").

Matena and Coleman are analogous art because both are related to dynamic deploying of data.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Coleman into the method of Matena to use a monitoring subsystem. The modification would have been obvious because one of the ordinary skills in the art would implement using the resources to rapidly and automatically adding resources when needed, for example, in response to an unexpected workload surge, but also removing resources when no longer needed, minimizing the cost of additional resources (Coleman, par [0023]).

16. **As per Claim 25** being the method claims corresponding to the system claim 17 respectively and rejected under the same reason set forth in connection of the rejections of Claim 17 and further Matena discloses: (Title, "Method and apparatus for executing applications on a distributed computer system").

Conclusion

17. THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

18. Any inquiry concerning this communication or earlier communications from the examiner should be directed to ANGELICA RUIZ whose telephone number is (571)270-3158. The examiner can normally be reached on 8:00 a.m. to 4:30 p.m., ET.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Mohammad Ali can be reached on (571) 272-4105. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/**Angelica Ruiz**/
Examiner, Art Unit 2158

/**Mohammad Ali**/
Supervisory Patent Examiner, Art Unit 2158

Notice of References Cited	Application/Control No. 11/694,483	Applicant(s)/Patent Under Reexamination SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	Art Unit 2158	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-6,256,637 B1	07-2001	Venkatesh et al.	1/1
*	B	US-6,513,059 B1	01-2003	Gupta et al.	709/202
*	C	US-2003/0078958 A1	04-2003	Pace et al.	709/201
*	D	US-2003/0074360 A1	04-2003	Chen et al.	707/100
*	E	US-2003/0135509 A1	07-2003	Davis et al.	707/100
*	F	US-2005/0039180 A1	02-2005	Fultheim et al.	718/001
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Netflix, Inc. - Ex. 1002, Page 000187
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	Art Unit
	Angelica Ruiz	2158

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	08/26/2010	02/09/2011							
	1	✓	✓							
	2	✓	✓							
	3	✓	✓							
	4	✓	✓							
	5	✓	✓							
	6	✓	✓							
	7	✓	✓							
	8	✓	✓							
	9	✓	✓							
	10	✓	✓							
	11	✓	✓							
	12	✓	✓							
	13	✓	✓							
	14	✓	✓							
	15	✓	✓							
	16	✓	✓							
	17	✓	✓							
	18	✓	✓							
	19	✓	✓							
	20	✓	✓							
	21	✓	✓							
	22	✓	✓							
	23	✓	✓							
	24	✓	✓							
	25	✓	✓							
	26	✓	✓							
	27	✓	✓							
	28	✓	✓							

Netflix, Inc. - Ex. 1002, Page 000188
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	ANGELICA RUIZ	2158

SEARCHED			
Class	**Subclass**	**Date**	**Examiner**
707	600-831, 899, and 999.001-999.206, by keyword	8/26/2010	A. Ruiz
709	223-227, by keyword	8/26/2010	A. Ruiz
711	6, by keyword	8/26/2010	A. Ruiz
719	315 -324, by keyword	8/26/2010	A. Ruiz

SEARCH NOTES		
Search Notes	**Date**	**Examiner**
Consulted Rehana Perveen for (QAS/WQAS) for 101 issues	8/26/2010	A. Ruiz
See ACM Portal Search strategy, attached	8/26/2010	A. Ruiz
See EAST Search history, including inventor	8/26/2010	A. Ruiz
See Google scholar including patents strategy, attached	8/26/2010	A. Ruiz
See EAST Search history updated, attached	2/9/2011	A. Ruiz

INTERFERENCE SEARCH			
Class	**Subclass**	**Date**	**Examiner**

/ANGELICA RUIZ/ Examiner.Art Unit 2158	

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	2	"20050005200"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:09
L2	7973	(server same software same application) and operating near4 system and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:18
L3	3216	L2 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:18
L4	271	L3 and (control$4 near4 node)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:18
L5	201	L4 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:18
L6	131	L5 and (server near4 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:18
L7	30	L6 and "707"/$.ccls.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:19
L8	83	"707"/$.ccls. and (enterprise near4 comput$5) same (tier hierarchical)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:22
L9	35	L8 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:22
L10	8	L9 and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:23
L11	8	L10 and (not sundar)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 23:23

Netflix, Inc. - Ex. 1002, Page 000190
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

S1	56	(virtual near2 machine) same application same node same (control$5) and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:01
S2	24	S1 and (software near3 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:03
S3	1	S2 and (node same x86)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:03
S4	7973	(server same software same application) and operating near4 system and (virtual near4 machine)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:04
S5	3216	S4 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:04
S6	271	S5 and (control$4 near4 node)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:04
S7	201	S6 and (automat$4)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:05
S8	131	S7 and (server near4 application)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2011/02/09 20:05

EAST Search History (Interference)

<This search history is empty>

2/9/2011 11:26:02 PM
C:\Documents and Settings\aruiz1\My Documents\EAST\Workspaces\11694483.wsp

Netflix, Inc. - Ex. 1002, Page 000191
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	Jagane Sundar et al.
Serial No.:	11/694,483
Filing Date:	March 30, 2007
Confirmation No.:	1040
Group Art Unit:	2158
Examiner:	Angelica Ruiz
Title:	***Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images***

Mail Stop: Amendment

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Response Pursuant to 37 C.F.R. § 1.111

In response to the Office Action dated September 3, 2010, Applicants respectfully request the Examiner to reconsider the rejections of the claims in view of the following amendments and remarks. Please amend the Application as follows.

In the Abstract

Please amend the Abstract to read as follows:

~~**A distributed computing system conforms to a multi-level, hierarchical organizational model.**~~ One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system. ~~**in accordance with the organization model. For instance, a distributed computing system comprises a plurality of application nodes interconnected via a communications network. In addition, the**~~ **The** distributed computing system includes a software image repository storing: (i) one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, and (ii) one or more image instances of one or more software applications that are executable on the virtual machines. The distributed computing system also includes a control node that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the image instances of the software applications on the virtual machines.

In the Claims

1. **(Currently Amended)** A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network;

a software image repository ~~**storing**~~ **comprising non-transitory, computer-readable media operable to store**: (i) one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, and (ii) one or more image instances of one or more software applications that are executable on the virtual machines; and

a control node that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the image instances of the software applications on the virtual machines.

2. (Original) The distributed computing system of claim 1, wherein the image instances of the one or more software applications comprise virtual disk image files.

3. (Original) The distributed computing system of claim 1, wherein when the automation infrastructure deploys the image of the one or more applications, the automation infrastructure copies a first one of the image instances to a first one of the application nodes.

4. (Original) The distributed computing system of claim 1, wherein an application image in the set of application images comprises an operating system.

5. (Original) The distributed computing system of claim 1, wherein an application image in the set of application images comprises a server application.

6. (Original) The distributed computing system of claim 1, wherein a first one of the virtual machines provides an environment that emulates an x86 platform.

7. (Original) The distributed computing system of claim 1, wherein the automation infrastructure captures the image of the virtual machine manager from an application node.

8. (Original) The distributed computing system of claim 7, wherein the automation infrastructure captures the image of the application from an application node executing the virtual machine manager.

9. (Original) The distributed computing system of claim 1, wherein the control node maintains the image instances as persistent or non-persistent based on a configuration.

10. (Original) The distributed computing system of claim 1, wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.

11. (Original) The distributed computing system of claim 10, wherein the rule engines are forward-chaining rule engines.

12. **(Currently Amended)** The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of ~~**the**~~ **an** application matrix.

13. (Original) The distributed computing system of claim 1, wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.

14. (Original) The distributed computing system of claim 13, further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.

15. (Original) The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

16. (Original) The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

17. (Original) The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

18. (Original) The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

19. (Original) The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model.

20. (Original) A method comprising:

storing one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines;

storing one or more image instances of one or more software applications that are executable on the virtual machines;

executing a rules engine to perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

executing the rules engine to perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

21. (Original) The method of claim 20, wherein storing the image instances of the one or more software applications comprise storing virtual disk image files.

22. (Original) The method of claim 20, wherein performing operations to autonomically deploy the image instances comprises copying a first one of the image instances to a first one of the application nodes.

23. (Original) The method of claim 20, wherein the software applications comprise an operating system.

24. (Original) The method of claim 20,

further comprising generating an application matrix that specifies data for controlling the deployment of the software applications within a distributed computing system; and

wherein performing operations to provide autonomic control comprises provide autonomic control over the deployment, undeployment, and execution of the software applications on the virtual machines within the distributed computing system in accordance with parameters of the application matrix.

25. (Original) The method of claim 20, wherein performing operations comprises:

receiving status data for the distributed computing system having the plurality of application nodes interconnected via a communications network, wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system; and

performing the operations to provide autonomic control over the deployment of the software applications on the virtual machines within the distributed computing system in accordance with the rules.

26. (Original) The method of claim 20, further comprising accessing parameters of the application matrix and generating one or more rules for use in autonomically controlling the deployment of the software applications on the virtual machines.

27. (Original) The method of claim 26, further comprising processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

28. **(Currently Amended)** A **non-transitory,** computer-readable medium comprising instructions **stored thereon,** that ~~cause a~~ **when executed by a** programmable processor ~~to~~:

store one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines;

store one or more image instances of one or more software applications that are executable on the virtual machines;

perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

REMARKS

The Application has been reviewed in light of the Office Action dated September 3, 2009 ("*Office Action*"). Claims 1-28 are pending and rejected in the Application. To advance prosecution of the Application, Applicants have responded to each notation by the Examiner. Applicants respectfully request reconsideration, further examination, and favorable action in this case.

In the Abstract

The Examiner objects to the Abstract. Applicants have amended the Abstract as requested by the Examiner.

Section 112 Rejections

The Examiner rejects Claim 12 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and claim the subject matter which Applicants regard as the invention. Specifically, the Examiner asserts that there is insufficient antecedent basis for "a current state of the application matrix." *Office Action*, pg. 3. Applicants have amended Claim 12 in a manner that renders such rejection moot. For at least these reasons, Applicant respectfully requests that the rejection of Claim 12 under 35 U.S.C. § 112, second paragraph be withdrawn.

Section 101 Rejections

The Examiner rejects Claims 1–19 and 28 under 35 U.S.C. § 101 because the claimed invention is allegedly directed to non-statutory subject matter for various reasons.

First, the Examiner rejects Claims 1–19 as being "software per se." *Office Action*, pg. 3–4. Although Applicants do not necessarily agree with the Examiner's characterization of these claims as "software per se," in order to advance prosecution, Applicants have amended Claim 1 in a manner that renders such rejections moot. Accordingly, Applicants respectfully request reconsideration and allowance of Claims 1–19.

Second, the Examiner rejects Claim 28 because it covers both statutory and non-statutory embodiments. *Office Action*, pg. 4–5. Although Applicants do not necessarily agree with the Examiner's rationale for rejecting Claim 28, Applicants have amended Claim 28 to recite "A non-transitory, computer-readable medium comprising instructions stored

thereon, that when executed by a programmable processor [perform limitations recited in the claim]." Applicants submit that these amendments render the rejection of Claim 28 moot, and Applicants respectfully request that this rejection be withdrawn.

Section 102 Rejections

The Examiner rejects Claims 1-16, 18-24, and 26-28 under 35 U.S.C. § 102(b) as being anticipated by U.S. Publication No. 2005/0005200 filed by Matena et al. ("*Matena*"). Applicants respectfully traverse these claim rejections at least because *Matena* fails to disclose, or even teach or suggest, the elements specifically recited in Applicants' claims.

At the outset, Applicants note that the Examiner relies on the Java classes in *Matena* to teach both the "virtual machine" *and* the "instances of the software applications on the virtual machine," as required by Claim 1. For example, the Examiner relies on *Matena* at [0112]–[0113] to teach "autonomic deployment of the image instances of the software applications on the virtual machine." *Office Action* pg. 6–7. In those passages, *Matena* discloses an application controller that can automatically distribute application modules to nodes. *See Matena* at [0112]–[0113]. According to an earlier passage in *Matena*, "application modules are, for example, operating-system level processes . . . such as execution modules of Java applications." *Matena* at [0089]. Such execution modules are later described as "includ[ing] the application's Java classes loaded into the container." *Matena* at [0090]. Thus, the Java classes and application modules of *Matena* are one in the same. *Matena* merely discloses Java classes as a subset of application modules.

Therefore, when the Examiner relies on application modules in *Matena* to teach the "image instances of the software applications on the virtual machine" recited in Claim 1, the Examiner also relies on Java classes in *Matena* to teach "image instances of the software applications on the virtual machine" as well. Earlier, however, the Examiner contends that the Java classes in *Matena* disclose a "virtual machine executing on the application nodes." *Office Action* pg. 6. That is, the Examiner relies upon the Java classes in *Matena* to teach *both* the "virtual machine" *and* the "instances of the software applications on the virtual machine" recited in Claim 1. In doing so, the Examiner improperly relies on the Java classes in *Matena* to teach two distinct and separate limitations of Claim 1.

Moreover, *Matena* fails to disclose, teach, or suggest "a software image repository comprising non-transitory, computer-readable media operable to store . . . one or more image

instances of one or more software applications that are executable on the virtual machines" as required by Claim 1. The Examiner contends that the Java classes in *Matena* disclose a "'virtual machine' executing on the application nodes." *Office Action* pg. 6. The Examiner, however, does not point to "a software image repository comprising non-transitory, computer-readable media operable to store . . . one or more image instances of one or more ***software applications that are executable on the virtual machines***." Even assuming that Java classes can be construed as being virtual machines (which Applicants do not concede), the Examiner fails to establish how a Java class acting as a virtual machine can execute applications as required by Claim 1. Accordingly, *Matena* fails to disclose, teach, or suggest every element of Claim 1.

For at least these reasons, independent Claim 1 and its dependent claims are allowable under 35 U.S.C. § 102. For analogous reasons, independent Claims 20 and 28 and their respective dependent claims are also allowable under 35 U.S.C. § 102. Accordingly, Applicants respectfully request reconsideration and allowance of Claims 1-16, 18-24, and 26-28.

Section 103 Rejections

The Examiner rejects Claims 17 and 25 under 35 U.S.C. § 103(a) as being unpatentable over *Matena* in view of U.S. Publication No. 2004/0181794 filed by Coleman et al. ("*Coleman*"). Applicants respectfully traverse these claim rejections. In the rejections of Claims 17 and 25, the Examiner relies upon *Matena* to teach "a software image repository comprising non-transitory, computer-readable media operable to store . . . one or more image instances of one or more software applications that are executable on the virtual machines." However, for at least the reasons discussed above, *Matena* fails to disclose, teach or suggest every limitation of Claims 17 and 25. Therefore, Applicants contend that Claims 17 and 25 are allowable. Accordingly, Applicants respectfully request reconsideration and allowance of Claims 17 and 25.

No Waiver

All of Applicants' arguments and amendments are without prejudice or disclaimer. Additionally, Applicants have merely discussed example distinctions from the *Matena* reference. Other distinctions may exist, and Applicants reserve the right to discuss these additional distinctions in a later Response or on Appeal, if appropriate. By not responding to additional statements made by the Examiner, Applicants do not acquiesce to the Examiner's additional statements. The example distinctions discussed by Applicants are sufficient to overcome the anticipation and obviousness rejections.

CONCLUSION

Applicants have made an earnest attempt to place this case in condition for allowance. For the foregoing reasons, and for other apparent reasons, Applicants respectfully request full allowance of all pending Claims.

If the Examiner feels that a conference would advance prosecution of the Application in any manner, Luke K. Pedersen stands willing to conduct such a telephone interview at the convenience of the Examiner.

No additional fee is believed to be due. However, the Commissioner is hereby authorized to charge any additional fees or credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Applicant

Luke K. Pedersen
Reg. No. 45,003
Tel. 214.953.6655

Date: 12-3-10

Customer No. **05073**

Electronic Acknowledgement Receipt

EFS ID:	8965154
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	05073
Filer:	Bradley Jay Birchfield/Esmarie Garland
Filer Authorized By:	Bradley Jay Birchfield
Attorney Docket Number:	063170.9310
Receipt Date:	03-DEC-2010
Filing Date:	30-MAR-2007
Time Stamp:	16:30:55
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		0631709310Response.PDF	546487 fb0bc0074829041b58aa4601275bb3cb03d8b1ff	yes	13

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Amendment/Req. Reconsideration-After Non-Final Reject	1	1
	Abstract	2	2
	Claims	3	8
	Applicant Arguments/Remarks Made in an Amendment	9	13
Warnings:			
Information:			
	Total Files Size (in bytes):	546487	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/694,483	03/30/2007	

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☒		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT	12/03/2010	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 28	Minus	** 28	= 0	X $26 =	0	OR	X $ =	
	Independent (37 CFR 1.16(h))	* 3	Minus	***3	= 0	X $110 =	0	OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE	

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)	OR	RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/PAUL M. STANBACK/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,483	03/30/2007	Jagane Sundar	063170.9310	1040

5073 7590 09/03/2010

BAKER BOTTS L.L.P.
2001 ROSS AVENUE
SUITE 600
DALLAS, TX 75201-2980

EXAMINER	
RUIZ, ANGELICA	
ART UNIT	PAPER NUMBER
2158	

NOTIFICATION DATE	DELIVERY MODE
09/03/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

ptomail1@bakerbotts.com
glenda.orrantia@bakerbotts.com

Netflix, Inc. - Ex. 1002, Page 000208
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Office Action Summary	Application No.	Applicant(s)	
	11/694,483	SUNDAR ET AL.	
	Examiner	Art Unit	
	ANGELICA RUIZ	2158	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 March 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 March 2007* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *7/9/2007, 6/20/2008, and 8/17/2009*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Netflix, Inc. - Ex. 1002, Page 000209
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

DETAILED ACTION

1. Claims 1-28 are pending.

Information Disclosure Statement

2. The information disclosure statements (IDS) submitted on 7/9/2007, 6/20/2008, and 8/17/2009 are in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statements are being considered by the examiner.

Specification

3. Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The

disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

The abstract of the disclosure is objected to because of length in excess of 150 words. Appropriate correction is required. See MPEP § 608.01 (b).

Claim Rejections - 35 USC § 112

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claim 12 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

6. Claim 12 recites the limitation "a current state of the application matrix" in lines 3-4. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 101

7. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1 – 19 and 28 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1 – 19 are rejected as being "software per se". The claims are directed towards a system. However, it is noted that the use of the word “system” does not

inherently mean that the claims are directed towards a machine or article of manufacture (e.g., a "system" can be a computer software system). The claimed invention is also addressed towards a software image, all of which can be interpreted as comprising entirely of software per se according to one of ordinary skill in the art *(see paragraph [0055] which clearly recites that the "golden image may be viewed as a "master copy" of the software image…") which as mentioned in the same paragraph the system recites as making copy of installed software and the software image for a particular computer function.* Therefore, the claim language fails to provide the necessary hardware required for the claim to fall within the statutory category of a machine or article of manufacture (see MPEP 2106).

Accordingly, the claim becomes nothing more than sets of software instructions which are "software per se".

"Software per se" is non-statutory under 35 USC 101 because it is merely a set of instructions without any defined tangible output or tangible result being produced. The requirement for tangible result under 35 USC 101 is defined in State Street Bank & Trust Co. v. Signature Financial Group Inc., 149 F.3d 1368, 47USPQ2d 1596 (Fed. Cir. 1998).

Claim 28 is directed towards a computer-readable medium. In view of Applicant's disclosure, paragraph [0044], "for executing software instructions stored on one or more computer-readable media." the medium is not limited to statutory embodiments, instead being defined as including both statutory embodiments (e.g., random access memory, computer diskette) and non-statutory embodiments (e.g., propagation

medium). In particular, the disclosure refers to "communications media" which improperly include network transmission lines (interpreted as wired and wireless transmission), wireless transmission media, signals propagating through space, radio waves, infrared signals, etc.

See, e.g., In re Nuitjen, Docket no. 2006-1371 (Fed. Cir. Sept. 20, 2007)(slip. op. at 18) "A transitory, propagating signal like Nuitjen's is not a process, machine, manufacture, or composition of matter.' … Thus, such a signal cannot be patentable subject matter."

Therefore, the claimed subject matter fails to fall within on of the four statutory classes *(see MPEP 2106)*.

The Examiner suggests amending the preamble of the claims to include language of "a non transitory computer-readable medium comprising instructions stored thereon, that when executed by a processor", provided such language is supported by the specification.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1 – 16, 18 –24 and 26 –28 are rejected under 35 U.S.C. 102(b) as being anticipated by Matena et al (US 2005/0005200 A1), hereinafter "Matena".

As per Claim 1, Matena discloses:

A distributed computing system comprising: a plurality of application nodes interconnected via a communications network; (Title, "Method and apparatus for executing applications on a distributed computer system") and (Abstract, "multiple nodes interconnected through a network…") ***a software image repository storing:*** (Par [0015] and (Par [0081], "…creation of execution modules to store new server objects.") ***(i) one or more image instances of a virtual machine manager that is executable on the application nodes,*** (Par [0090], "Java classes" being the "virtual machine" executing on the application nodes, as claimed), ***wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines,*** (Par [0193], includes the instance of the "virtual machine" as claimed*)* ***and (ii) one or more image instances of one or more software applications that are executable on the virtual machines;*** (Par [0015], "The application controllers utilize the execution controller. To its users, the execution *control system provides a single-system image of the distributed computer system and applications running on it*…") ***and a control node*** (Par [0015], "one or more node controllers, and one or more application controllers….") ***that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes*** (Par [0085], "The node controllers 103, execution controller 102 and application controllers are distributed over the nodes of a distributed computer system."; see figure 2) ***and to provide autonomic deployment of the image instances of the software***

applications on the virtual machines. (Par [0089]; par [0112], "can automatically distribute application modules of multiple applications to the node…"; par [0113], "… *same application could be deployed on several distributed computer systems, each having a different configuration, and each possibly requiring different assignment of application module replicas* to nodes.", the "replicas" being the "images" as claimed).

As per Claim 2, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the image instances of the one or more software applications comprise virtual disk image files.***

(Par [0108], "In some embodiments of the invention, the distributed computer system may include *nodes with different operating systems*. ...To shield the other components of the execution control system from the differences in the command line format, the *NC API 707 uses a virtual, operating-system* neutral format of the command line. The node controllers 701 have the logic to translate the virtual command line to the format understood by the local operating system.").

As per Claim 3, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein when the automation infrastructure deploys the image of the one or more applications, the automation infrastructure copies a first one of the image instances to a first one of the application nodes.***

(Par [0251], "…Also, embodiments replicate the state of each application execution module on multiple nodes. Each copy of the application execution module is called a

replica. If one replica of an execution module fails, other replicas of the execution module remain available, thereby allowing the application to provide service to its users.") and (Par [0289], "…A purpose of the replication constraint is *to specify that if one replica of an execution module is assigned to a node that is included in the first node group, then the other replica of the same execution module should be assigned to a node included in the second node group*.").

As per Claim 4, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein an application image in the set of application images comprises an operating system.***

(Par [0084], " For example, the exemplary ECS 101 in FIG. 1 includes three application controllers: a service application controller 104, which is suitable for controlling applications that are services 107 including operating-system level processes; a Java application controller 105, which is suitable for controlling Java applications 108 including containers and execution modules (container and execution modules are explained later in this disclosure); and custom application controller 106, which is suitable for controlling applications 109 that have a custom structure and use a custom application execution model.").

As per Claim 5, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein an application image in the set of application images comprises a server application.***

(Par [0081], "The disclosure further describes how an *application controller controls remote invocation and creation of server objects located in the execution modules*. The advantages include location transparency of server objects; ability to relocate server objects; transparent failure recovery; partitioning of server objects; and on-demand creation of execution modules to store new server objects.").

As per Claim 6, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein a first one of the virtual machines provides an environment that emulates an x86 platform.***

(Par [0412], "In one embodiment of the invention, nodes are realized as general-purpose server computers or server blades. Each node includes an image of a general-purpose operating system (e.g. Linux, Solaris, or Windows). Node controllers are realized as a Java 2 Platform, Standard Edition ("J2SE") applications…" the x86 being an instance of the Windows operating system, as claimed).

As per Claim 7, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the automation infrastructure captures the image of the virtual machine manager from an application node.***

(Par [0193], "JAC manages how the containers are distributed over the nodes of the distributed system and how the application execution modules are distributed over the containers. FIG. 21 also depicts the parts of the container used in an embodiment of the invention. *The container 2102 is a process that includes a Java Virtual Machine*

("JVM") 2104 and application server classes 2105. The application server classes 2105 include Java classes that implement the operations that *JAC uses to manage the lifecycle of the execution modules (such as the classes that implement the creation of execution modules)* and Java classes that are provided as runtime library classes for the execution modules (such as classes that implement the Enterprise JavaBeans container functions).").

As per Claim 8, the rejection of Claim 7 is incorporated and further Matena discloses: ***wherein the automation infrastructure captures the image of the application from an application node executing the virtual machine manager.***

(Par [0193], "JAC manages how the *containers are distributed over the nodes of the distributed system* and how the application execution modules are distributed over the containers. FIG. 21 also depicts the parts of the container used in an embodiment of the invention. *The container 2102 is a process that includes a Java Virtual Machine ("JVM") 2104 and application server classes 2105…." which is captured from the "application node" as claimed).*

As per Claim 9, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node maintains the image instances as persistent or non-persistent based on a configuration.***

(Par [0105], "…When a node controller detects a process failure, it sends an

event notification to all the subscribers for the event. For example, the execution controller subscribes to the process failure events so that it can maintain an up-to-date state model of the processes in the distributed system." The "event notification" when "up-to-date" being the "persistent" as claimed).

As per Claim 10, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.***

(Par [0267], " In the embodiment of the invention depicted in FIG. 36, JAC uses the Distribution Manager objects, Replication Constraint policy objects, the Container Distribution Policy objects, and EM Distribution Policy objects when making decisions to which nodes and to which containers to assign execution module replicas. The objective of this flexible, policy-based distribution of replicas…" which shows replication and distribution policy object to determine whether to deploy an application).

As per Claim 11, the rejection of Claim 10 is incorporated and further Matena discloses:

wherein the rule engines are forward- chaining rule engines.

(Par [0271], "The distributed computer system then observes the above rules when assigning execution module replicas to the nodes of the distributed computer system. For example, the replication constraint depicted in FIG. 37 constrains

the assignment of replicas such that if the first replica of an execution module is assigned to one of the nodes A, B, or C, then the second replica of the same execution module is assigned to one of the nodes D, E, or F. The replication constraint further specifies that there should be two replicas of the execution module and that the replication type is Hot-Standby. For example, since one replica of the execution module Auction-1 is on node A, the second replica can be only on one of the nodes D, E, or F (in the example, it is on Node E)" being the "forward-chaining rule" as claimed).

As per Claim 12, the rejection of Claim 10 is incorporated and further Matena discloses:

wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of the application matrix.

(Par [0121], "The *state model* includes objects that present nodes 909, node groups 910, processes 911, and application controllers 912 in the distributed computer system. The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system.", that being the "current state" as claimed; the "State model" being the "application matrix" as claimed; Par [0195], "…In some embodiments of the invention, the information in the deployment descriptor is represented using Extensible Markup Language (XML) technologies.", the information described as an "XML" being the application matrix, as claimed; par [0153] As the

Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…").

As per Claim 13, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.***

(Par [0121], "The *state model* includes objects that present nodes 909, node groups 910, processes 911, and application controllers 912 in the distributed computer system. The state model also includes relationships among the objects. The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system.", the "State model" being the "application matrix" as claimed; Par [0195], "…In some embodiments of the invention, the information in the *deployment descriptor* is represented using Extensible Markup Language (XML) technologies.", par [0153] As the Distribution Policy object 1102 for the service in FIG. 11…Distribution Manager object 1102 will automatically create two processes on the added node…" Par [0142], "The Distribution Policy objects 1014 *provide parameters to the algorithm of the Distribution Manager objects* 1010…" Par [0201] Application definition 2 2302 includes a deployment descriptor 2304….").

As per Claim 14, the rejection of Claim 1 is incorporated and further Matena discloses:

further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.

(Par [0084], "For example, the exemplary ECS 101 in FIG. 1 includes three application controllers: a service application controller 104, which is suitable for controlling applications that are services 107 including operating-system level processes; a Java application controller 105, which is suitable for controlling Java applications 108 including containers and execution modules…" the "application controller" being the "governor" as claimed).

As per Claim 15, the rejection of Claim 10 is incorporated and further Matena discloses:

wherein the automation infrastructure automatically updates the rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

(Par [0121], "The state model also includes relationships among the objects.
The execution controller maintains its state model objects up to date so that they reflect accurately the current state of the distributed computer system."; Par [0170], "In response to the "Process Started" message, the execution controller updates the state of Process 1 object in its state model to indicate that the process has successfully started 1310. Then the execution controller sends the "Process Started" message to SAC 1311. SAC updates its internal Process 1 object in its state model to reflect that

process 1 has successfully started 1312. This completes the steps of starting a process. SAC checks if there are more service processes that should be started 1317. If yes, the steps of starting a process are repeated. If no, the start service operation has completed.").

As per Claim 16, the rejection of Claim 11 is incorporated and further Matena discloses:

wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

(Par [0271], "In some embodiments of the invention, the application programmer or system administrator defines the node groups, replication groups, and replication constraints. The distributed computer system then observes the above rules when assigning execution module replicas to the nodes of the distributed computer system.").

As per Claim 18, the rejection of Claim 11 is incorporated and further Matena discloses:

wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

(See Figures 1 and 2, that shows the organization into "tiers" as claimed; see Par [0117], "The "node group management" 905 operations include the operations to create

and remove a node group; operations to add and remove a node from a node group; and operations to obtain status information about the node groups.").

As per Claim 19, the rejection of Claim 1 is incorporated and further Matena discloses: ***further comprising: a database storing data that defines a model for a hierarchical organization of the distributed computing system,*** (Par [0097], "FIG. 6 illustrates a representational node 601 suitable for being used in a method and system of the invention. A node is a computer system including one or more central-processing units (CPU) 602, main memory 603, optional secondary storage 604, and one or more communication interfaces 605…")

wherein the model specifies a fabric having a one or more domains, (See Figure 8)

and wherein each domain has at least one tier that includes at least one of the application nodes, (Par [0099], "Each node of a distributed computer system includes a node controller 701. FIG. 7 depicts the functionality provided by a node controller and its main components. A node controller 701 implements the following functions: node membership 702, master election 703, start process 704, stop process 705, and detect process failure 706.").

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model. (Par [0112], "A node group is a named subset of the nodes in the distributed computer system. A node can be a member of multiple node groups. Using node groups in a distributed computer system

has several advantages. First, node groups simplify many management tasks because instead of listing a large number of individual nodes when performing system a administration task, a node group name can be given instead. Second, it is easy to adjust the processing capacity or distribution of multiple applications by simply adding or removing nodes from a node group associated with the applications. And (Par [0113])

9. **As per Claims 20-24 and 26-27** being the method claims corresponding to the system claims 1-4, 13, 12, 11 respectively and rejected under the same reason set forth in connection of the rejections of Claims 1 -4, 13, 12, 11 and further Matena discloses: (Title, "Method and apparatus for executing applications on a distributed computer system").

10. **As per Claim 28** being the computer readable medium claim corresponding to the system claims 1 respectively and rejected under the same reason set forth in connection of the rejections of Claim 1 and further Matena discloses: (Abstract and Claim 114, "A computer-readable medium including instructions for performing a method when executed by a processor").

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the

invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. **Claims 17 and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over Matena et al (US 2005/0005200 A1), in view of Coleman et al (US 2004/0181794 A1).

As per Claim 17, the rejection of Claim 1 is incorporated and further Matena discloses: ***wherein the control node further comprises:*** (Par [0015], "one or more node controllers, and one or more application controllers….").

Matena do not disclose the following features:

Coleman discloses the claimed features as follows:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes (Par [0023] The deployment manager 130 interfaces with relevant resources of the application level 140 to monitor measurement/configuration data…"; see Figures 1- 2, feature 202, the "forecaster" being the "subsystem" as claimed that monitors the data and provides the status information) ***and a business logic tier that provides expected state data to the automation subsystem***, (Par [0019], "The solution manager 120, itself, comprises a control logic engine 122, a forecaster module 124, a performance estimator module 126, an admission control module 128, and a tuning module 129. The deployment manager 130, itself, comprises a monitoring module 132…" par [0024], "e-commerce site...").

wherein the expected state data represents an expected state for the application nodes, (Par [0034], "Based on monitored data such as the transaction rate history, the solution manager 120 *uses the forecaster module 124 to project into the future the expected workload* (step 204)…").

wherein the rule engines analyze the status data from the monitoring subsystem (Par [0034] "…The forecast can be long-term (e.g., hours/days) or short-term (e.g., seconds/minutes). To deal with unanticipated changes in the workload (e.g., outside of normal daily/weekly, etc.) variations, the forecaster *is preferably adaptive and rapidly learns from recent history.* The forecast horizon does not need to be long, e.g., the horizon may be approximately the amount of time needed for the resource action (application provisioning in this situation) to take effect.").

and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state. (Par [0036], "*The solution manager 120 then uses the performance estimator 126 to check if the current resource deployment is insufficient, sufficient (i.e., adequate) or overly sufficient (i.e., excessive) for maintaining the service objective threshold* (e.g., response time) based on the recent and forecasted workload traffic (step 206)…").

Matena and Coleman are analogous art because both are related to dynamic deploying of data.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time of invention was made to incorporate the teachings of Coleman into the method of

Matena to use a monitoring subsystem. The modification would have been obvious because one of the ordinary skills in the art would implement using the resources to rapidly and automatically adding resources when needed, for example, in response to an unexpected workload surge, but also removing resources when no longer needed, minimizing the cost of additional resources (Coleman, par [0023]).

13. **As per Claim 25** being the method claims corresponding to the system claim 17 respectively and rejected under the same reason set forth in connection of the rejections of Claim 17 and further Matena discloses: (Title, "Method and apparatus for executing applications on a distributed computer system").

Conclusion

14. Any inquiry concerning this communication or earlier communications from the examiner should be directed to ANGELICA RUIZ whose telephone number is (571)270-3158. The examiner can normally be reached on 8:00 a.m. to 4:30 p.m., ET.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Mohammad Ali can be reached on (571) 272-4105. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Angelica Ruiz/
Examiner, Art Unit 2158

/Mohammad Ali/
Supervisory Patent Examiner, Art Unit 2158

Notice of References Cited	Application/Control No. 11/694,483	Applicant(s)/Patent Under Reexamination SUNDAR ET AL.	
	Examiner ANGELICA RUIZ	Art Unit 2158	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2002/0129129 A1	09-2002	Bloch et al.	709/220
*	B	US-2007/0169049 A1	07-2007	Gingell et al.	717/151
*	C	US-2007/0168919 A1	07-2007	Henseler et al.	717/101
*	D	US-7,246,351 B2	07-2007	Bloch et al.	717/175
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

RECEIVED
CENTRAL FAX CENTER
JUN 20 2008

Form 1449*	Docket Number:	Application Number:
INFORMATION DISCLOSURE STATEMENT	1072-021US01	11/694,483
IN AN APPLICATION	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
(Use several sheets if necessary)	Filing Date: March 30, 2007	Group Art Unit: ~~2161~~ 2158
	Examiner Name: ~~Unknown~~ A. Ruiz	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

	U.S. Patent Application No. 11/607,819 entitled, "Automated Deployment and Configuration of Applications in an Autonomically Controlled Distributed Computing System", filed December 1, 2006, Docket No. 1072-015US01
	U.S. Patent Application No. 11/607,820 entitled, "Automated Deployment and Configuration of Applications in an Autonomically Controlled Distributed Computing System", filed December 1, 2006, Docket No. 1072-016US01

EXAMINER /Angelica Ruiz/	**Date Considered** 08/26/2010

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /A.R./

Form 1449*	Docket Number:	Application Number:
INFORMATION DISCLOSURE STATEMENT	1072-021US01	11/694,483
IN AN APPLICATION	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
(Use several sheets if necessary)	Filing Date: March 30, 2007	Group Art Unit: ~~Unknown~~ 2158
	Examiner Name: ~~Unknown~~ A. Ruiz	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate
	2003/0177176 A1	09/18/2003	Hirschfeld et al.	
	2003/0126265 A1	07/03/2003	Aziz et al.	
	2003/0140282 A1	07/24/2003	Kaler et al.	
	2006/0173895 A1	08/03/2006	Engquist et al.	
	2006/0173994 A1	08/03/2006	Emeis et al.	
	2006/0173984 A1	08/03/2006	Emeis et al.	
	2006/0173857 A1	08/03/2006	Jackson	
	2006/0173856 A1	08/03/2006	Jackson et al.	
	2002/0156877 A1	10/24/2002	Lu et al.	
	2004/0187104 A1	09/23/2004	Sardesai et al.	
	2005/0193265 A1	09/01/2005	Lin et al.	
	2004/0260734 A1	12/23/2004	Ren et al.	
	6,775,829 B1	08/10/2004	Kroening	
	2004/0088694 A1	05/06/2004	Ho	
	2006/0174238 A1	08/03/2006	Henseler et al.	
	2006/0173993 A1	08/03/2006	Henseler et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No
	WO/2006/083893 A1	08/10/2006	PCT		
	WO/2006/083894 A1	08/10/2006	PCT		
	WO/2006/083901 A1	08/10/2006	PCT		
	WO/2006/083895 A1	08/10/2006	PCT		
	WO/2006/083727 A1	08/10/2006	PCT		
	WO/2006/081503 A1	08/03/2006	PCT		

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

EXAMINER	**Date Considered**
/Angelica Ruiz/	08/26/2010

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /A.R./

Form 1449*	Docket Number: 1072-021US01	Application Number: 11/694,483
INFORMATION DISCLOSURE STATEMENT	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
IN AN APPLICATION	Filing Date: March 30, 2007	Group Art Unit: ~~Unknown~~ 2158
(Use several sheets if necessary)	Examiner Name: ~~Unknown~~ A. Ruiz	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

	E.N. Herness et al., "WebSphere Application Server: A Foundation for on Demand Computing," IBM Systems Journal IBM, Vol. 43, No. 2, pgs. 213-237, XP-002383791, 2004.
	B. Urgaonkar et al., "Resource Overbooking and Application Profiling in Shared Hosting Platforms," Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 17 pgs., XP-002387427, 2002.
	G. Lodi et al., "QoS-aware Clustering of Application Servers," Proceedings of the 1st IEEE Workshop on Quality of Service for Application Servers, In Conjunction With the 23rd International Symposium on Reliable Distributed Systems, 6 pages, XP-002383792, October 17, 2004.
	Preinstalling Microsoft Windows XP by Using the OEM Preinstallation Kit, Part 1, XP-002301441, April 4, 2003, 24 pages.
	R. Mark Koan et al., *It Takes a Village to Build an Image*, XP-002384269, 2003, pages 200-207.

EXAMINER	**Date Considered**
/Angelica Ruiz/	08/26/2010

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820 (Also form PTO-1449) Patent and Trademark Office, U.S. Department of Commerce

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /A.R./

Form 1449*	Docket Number:	Application Number:
INFORMATION DISCLOSURE STATEMENT	1072-021US01	11/694,483
IN AN APPLICATION	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
(Use several sheets if necessary)	Filing Date: March 30, 2007	Group Art Unit: ~~2161~~ 2158
	Examiner Name: ~~Unknown~~ A. Ruiz	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate
	2004/0181794 A1	09/16/2004	Coleman et al.	
	6,865,737 B1	03/08/2005	Lucas et al.	
	2004/0162741 A1	08/19/2004	Flaxer et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No
	WO/2003/085526 A1	10/16/2003	PCT		

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

EXAMINER	**Date Considered**
/Angelica Ruiz/	08/26/2010

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /A.R./

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	ANGELICA RUIZ	2158

SEARCHED			
Class	**Subclass**	**Date**	**Examiner**
707	600-831, 899, and 999.001-999.206, by keyword	8/26/2010	A. Ruiz
709	223-227, by keyword	8/26/2010	A. Ruiz
711	6, by keyword	8/26/2010	A. Ruiz
719	315 -324, by keyword	8/26/2010	A. Ruiz

SEARCH NOTES		
Search Notes	**Date**	**Examiner**
Consulted Rehana Perveen for (QAS/WQAS) for 101 issues	8/26/2010	A. Ruiz
See ACM Portal Search strategy, attached	8/26/2010	A. Ruiz
See EAST Search history, including inventor	8/26/2010	A. Ruiz
See Google scholar including patents strategy, attached	8/26/2010	A. Ruiz

INTERFERENCE SEARCH			
Class	**Subclass**	**Date**	**Examiner**

/ANGELICA RUIZ/ Examiner.Art Unit 2158	

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694483	SUNDAR ET AL.
	Examiner	**Art Unit**
	Angelica Ruiz	2158

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	08/26/2010								
	1	✓								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								
	19	✓								
	20	✓								
	21	✓								
	22	✓								
	23	✓								
	24	✓								
	25	✓								
	26	✓								
	27	✓								
	28	✓								

Netflix, Inc. - Ex. 1002, Page 000236
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Form 1449*	Docket Number:	Application Number:
INFORMATION DISCLOSURE STATEMENT	1072-021US01	11/694,483
IN AN APPLICATION	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
(Use several sheets if necessary)	Filing Date: March 30, 2007	Group Art Unit: ~~2161~~ 2158
	Examiner Name: ~~Unknown~~	A.Ruiz

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate
	2005/0005200 A1	01/06/2005	Matena et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

EXAMINER	**Date Considered**
/Angelica Ruiz/	08/26/2010

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /A.R./

ACM PORTAL USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

SEARCH

Searching within **The ACM Digital Library** with **Advanced Search**: (application node software copy or image automatic platform operating system server virtual OR disk "virtual machine") (start a new search)

Found **9,874** of **286,598**

REFINE YOUR SEARCH

Refine by Keywords

SEARCH

Discovered Terms

Refine by People

Names
Institutions
Authors
Editors
Reviewers

Refine by Publications

Publication Year
Publication Names
ACM Publications
All Publications
Content Formats
Publishers

Refine by Conferences

Sponsors
Events
Proceeding Series

ADVANCED SEARCH

Advanced Search

FEEDBACK

Please provide us

Search Results • Related Journals • Related Magazines • Related SIGs • Related Conferences

Results 1 - 50 of 9,874

Sort by relevance in condensed form

Save results to a Binder

Result page: **1** 2 3 4 5 6 7 8 9 10 next >>

1 On the design of the local variable cache in a hardware translation-based java virtual machine

Hitoshi Oi

July 2005 **LCTES '05:** Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems

2 Multiview access protocols for large-scale replication

Xiangning Liu, Abdelsalam Helal, Weimin Du

June 1998 **Transactions on Database Systems (TODS)** , Volume 23 Issue 2

3 Design and implementation of a software bridge with packet filtering and statistics collection functions

A. Khan, N. Al-Darwish, M. Guizani, M. Benten, H. Youssef

September 1997 **International Journal of Network Management** , Volume 7 Issue 5

4 A comprehensive approach for the development of modular software architecture description languages

Eric M. Dashofy, André van der Hoek, Richard N. Taylor

April 2005 **Transactions on Software Engineering and Methodology (TOSEM)** , Volume 14 Issue 2

5 State-slice: new paradigm of multi-query optimization of window-based stream queries

Song Wang, Elke Rundensteiner, Samrat Ganguly, Sudeept Bhatnagar

September 2006 **VLDB '06:** Proceedings of the 32nd international conference on Very large data bases

6 Lightweight lexical source model extraction

Gail C. Murphy, David Notkin

July 1996 **Transactions on Software Engineering and Methodology (TOSEM)** , Volume 5 Issue 3

Netflix, Inc. - Ex. 1002, Page 000238
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

with feedback

Found **9,874** of **286,598**

7 Functional partitioning improvements over structural partitioning for packaging constraints and synthesis: tool performance

Frank Vahid, Thuy Dm Le, Yu-Chin Hsu

April 1998 **Transactions on Design Automation of Electronic Systems (TODAES)** , Volume 3 Issue 2

8 Surface reconstruction using shadow profilometry

Christopher P Walker, Norbert Maerz, Michael G Hilgers

March 2005 **SAC '05:** Proceedings of the 2005 ACM symposium on Applied computing

9 Evaluating lateration-based positioning algorithms for fine-grained tracking

Andrew Rice, Robert Harle

September 2005 **DIALM-POMC '05:** Proceedings of the 2005 joint workshop on Foundations of mobile computing

10 Order preserving encryption for numeric data

Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, Yirong Xu

June 2004 **SIGMOD '04:** Proceedings of the 2004 ACM SIGMOD international conference on Management of data

11 Getting a cybersecurity program started on low budget

Harpal Dhillon, Mariana Hentea

March 2005 **ACM-SE 43: Proceedings of the 43rd annual Southeast regional conference - Volume 1** , Volume 1

12 R-SOX: runtime semantic query optimization over XML streams

Song Wang, Hong Su, Ming Li, Mingzhu Wei, Shoushen Yang, Drew Ditto, Elke A. Rundensteiner, Murali Mani

September 2006 **VLDB '06:** Proceedings of the 32nd international conference on Very large data bases

13 Frontmatter (TOC, Miscellaneous material)

ACM SIGSOFT Software Engineering Notes staff

November 2006 **SIGSOFT Software Engineering Notes** , Volume 31 Issue 6

14 Intrusion detection systems and multisensor data fusion

Tim Bass

April 2000 **Communications of the ACM** , Volume 43 Issue 4

15 Design concepts for learning spatial relationships

Giuliano Benelli, Maurizio Caporali, Antonio Rizzo, Elisa Rubegni

October 2001 **SIGDOC '01:** Proceedings of the 19th annual international conference on Computer documentation

16 Constructing collaborative desktop storage caches for large scientific datasets

Sudharshan S. Vazhkudai, Xiaosong Ma, Vincent W. Freeh, Jonathan W. Strickland, Nandan Tammineedi, Tyler Simon, Stephen L. Scott

August 2006 **Transactions on Storage (TOS)** , Volume 2 Issue 3

Netflix, Inc. - Ex. 1002, Page 000239
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

17 Improving the reliability of commodity operating systems



Michael M. Swift, Brian N. Bershad, Henry M. Levy

February 2005 **Transactions on Computer Systems (TOCS)** , Volume 23 Issue 1

18 Clustering pair-wise dissimilarity data into partially ordered sets



Jinze Liu, Qi Zhang, Wei Wang, Leonard McMillan, Jan Prins

August 2006 **KDD '06:** Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining

19 An application-driven perspective on wireless sensor network security



Eric Sabbah, Adnan Majeed, Kyoung-Don Kang, Ke Liu, Nael Abu-Ghazaleh

October 2006 **Q2SWinet '06:** Proceedings of the 2nd ACM international workshop on Quality of service & security for wireless and mobile networks

20 Distributed meetings: a meeting capture and broadcasting system



Ross Cutler, Yong Rui, Anoop Gupta, JJ Cadiz, Ivan Tashev, Li-wei He, Alex Colburn, Zhengyou Zhang, Zicheng Liu, Steve Silverberg

December 2002 **MULTIMEDIA '02:** Proceedings of the tenth ACM international conference on Multimedia

21 A DRM security architecture for home networks



Bogdan C. Popescu, Bruno Crispo, Andrew S. Tanenbaum, Frank L.A.J. Kamperman

October 2004 **DRM '04:** Proceedings of the 4th ACM workshop on Digital rights management

22 Compus: visualization and analysis of structured documents for understanding social life in the 16th century



Jean-Daniel Fekete, Nicole Dufournaud

June 2000 **DL '00:** Proceedings of the fifth ACM conference on Digital libraries

23 A mobile computing middleware for location- and context-aware internet data services



Paolo Bellavista, Antonio Corradi, Rebecca Montanari, Cesare Stefanelli

November 2006 **Transactions on Internet Technology (TOIT)** , Volume 6 Issue 4

24 Disk cache replacement algorithm for storage resource managers in data grids

Ekow Otoo, Frank Olken, Arie Shoshani

November 2002 **Supercomputing '02:** Proceedings of the 2002 ACM/IEEE conference on Supercomputing

25 A small-world model of the human mind



José Luís Malaquias, Agostinho Cláudio da Rosa, Carlos Manuel B. A. Correia

April 2006 **SAC '06:** Proceedings of the 2006 ACM symposium on Applied computing

26 A model of CPS translation and interpretation



Ramon Zatarain, R. Stansifer

April 2004 **ACM-SE 42:** Proceedings of the 42nd annual Southeast regional conference

Netflix, Inc. - Ex. 1002, Page 000240
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

27 Designing intelligent sales-agent for online selling

Shiu-li Huang, Fu-ren Lin

August 2005 **ICEC '05:** Proceedings of the 7th international conference on Electronic commerce

28 Supporting objects in run-time bytecode specialization

Reynald Affeldt, Hidehiko Masuhara, Eijiro Sumii, Akinori Yonezawa

September 2002 **ASIA-PEPM '02:** Proceedings of the ASIAN symposium on Partial evaluation and semantics-based program manipulation

29 An administration concept for the enterprise role-based access control model

Axel Kern, Andreas Schaad, Jonathan Moffett

June 2003 **SACMAT '03:** Proceedings of the eighth ACM symposium on Access control models and technologies

30 Consistent and compact data management in distributed storage systems

Baruch Awerbuch, Christian Scheideler

June 2004 **SPAA '04:** Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures

31 Shape intrinsic fingerprints for free-form object matching

K. H. Ko, T. Maekawa, N. M. Patrikalakis, H. Masuda, F.-E. Wolter

June 2003 **SM '03:** Proceedings of the eighth ACM symposium on Solid modeling and applications

32 A GSA-based compiler infrastructure to extract parallelism from complex loops

Manuel Arenaz, Juan Touriño, Ramón Doallo

June 2003 **ICS '03:** Proceedings of the 17th annual international conference on Supercomputing

33 The Cutting Edge: LiS: Linux STREAMS

Fransisco Ballesteros, Dennis Froschauer, David Grothe

May 1999 **Linux Journal** , Volume 1999 Issue 61es

34 ATOP-space and time adaptation for parallel and grid applications via flexible data partitioning

Angela C. Sodan, Lin Han

October 2004 **ARM '04:** Proceedings of the 3rd workshop on Adaptive and reflective middleware

35 xTagger: a new approach to authoring document-centric XML

Ionut E. Iacob, Alex Dekhtyar

June 2005 **JCDL '05:** Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries

36 RINSE: The Real-Time Immersive Network Simulation Environment for Network Security Exercises

Michael Liljenstam, Jason Liu, David Nicol, Yougu Yuan, Guanhua Yan, Chris Grier

June 2005 **PADS '05:** Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation

Netflix, Inc. - Ex. 1002, Page 000241
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

37 Dynamic online reconfiguration for customizable and self-optimizing operating systems

Simon Oberthür, Carsten Böke, Björn Griese

September 2005 **EMSOFT '05:** Proceedings of the 5th ACM international conference on Embedded software

38 An intelligent distributed environment for active learning

August 2001 **Journal on Educational Resources in Computing (JERIC)** , Volume 1 Issue 2es

39 Perspective shadow maps

Marc Stamminger, George Drettakis

July 2002 **SIGGRAPH '02:** Proceedings of the 29th annual conference on Computer graphics and interactive techniques

40 Algorithm for Achieving Minimum Energy Consumption in CMOS Circuits Using Multiple Supply and Threshold Voltages at the Module Level

Yuvraj Singh Dhillon, Abdulkadir Utku Diril, Abhijit Chatterjee, Hsien-Hsin Sean Lee

November 2003 **ICCAD '03:** Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design

41 An orthogonal taxonomy for hyperlink anchor generation in video streams using OvalTine

Jason McC. Smith, David Stotts, Sang-Uok Kum

May 2000 **HYPERTEXT '00:** Proceedings of the eleventh ACM on Hypertext and hypermedia

42 From road maps to 3D-scenes: a reconstruction system

C. Glauser, H. Bieri

April 2003 **SCCG '03:** Proceedings of the 19th spring conference on Computer graphics

43 Are quorums an alternative for data replication?

Ricardo Jiménez-Peris, M. Patiño-Martínez, Gustavo Alonso, Bettina Kemme

September 2003 **Transactions on Database Systems (TODS)** , Volume 28 Issue 3

44 Interactive exploration of very large relational datasets through 3D dynamic projections

Li Yang

August 2000 **KDD '00:** Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining

45 A semantic model for safe protocol interaction

Sebastian Gutierrez-Nolasco, Nalini Venkatasubramanian, Carolyn Talcott

April 2006 **SAC '06:** Proceedings of the 2006 ACM symposium on Applied computing

46 Harp: a distributed query system for legacy public libraries and structured databases

Ee-Peng Lim, Ying Lu

July 1999 **Transactions on Information Systems (TOIS)** , Volume 17 Issue 3

Netflix, Inc. - Ex. 1002, Page 000242
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

47 Open data types and open functions

Andres Löh, Ralf Hinze

July 2006 **PPDP '06:** Proceedings of the 8th ACM SIGPLAN symposium on Principles and practice of declarative programming

48 The scientific workstation of the future may be a pile of PCs

Thomas Sterling

September 1996 **Communications of the ACM** , Volume 39 Issue 9

49 Reflective middleware for wireless sensor networks

Flávia C. Delicato, Paulo F. Pires, Luiz Rust, Luci Pirmez, José Ferreira de Rezende

March 2005 **SAC '05:** Proceedings of the 2005 ACM symposium on Applied computing

50 Attack-resilient hierarchical data aggregation in sensor networks

Sankardas Roy, Sanjeev Setia, Sushil Jajodia

October 2006 **SASN '06:** Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks

Result page: **1** 2 3 4 5 6 7 8 9 10 next >>

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2010 ACM, Inc.

Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

Netflix, Inc. - Ex. 1002, Page 000243
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 1040

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/694,483	03/30/2007 RULE	707	2158	063170.9310

APPLICANTS

Jagane Sundar, Saratoga, CA;
Sanjay Radia, Fremont, CA;
David A. Henseler, Maplewood, MN;

**** CONTINUING DATA ***************************

This appln claims benefit of 60/787,280 03/30/2006

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED **** ** SMALL ENTITY **
04/10/2007

Foreign Priority claimed ☐ Yes ☑ No 35 USC 119(a-d) conditions met ☐ Yes ☑ No Verified and Acknowledged /ANGELICA RUIZ/ Examiner's Signature	☐ Met after Allowance Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		CA	29	28	3

ADDRESS

BAKER BOTTS L.L.P.
2001 ROSS AVENUE
SUITE 600
DALLAS, TX 75201-2980
UNITED STATES

TITLE

DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

FILING FEE RECEIVED	FEES: Authority has been given in Paper	☐ All Fees
690	No._____ to charge/credit DEPOSIT ACCOUNT	☐ 1.16 Fees (Filing)
	No._____ for following:	☐ 1.17 Fees (Processing Ext. of time)
		☐ 1.18 Fees (Issue)
		☐ Other _____
		☐ Credit

BIB (Rev. 05/07).

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	20	("20020156877" | "20030126265" | "20030140282" | "20030177176" | "20040088694" | "20040162741" | "20040181794" | "20040187104" | "20040260734" | "20050005200" | "20050193265" | "20060173856" | "20060173857" | "20060173895" | "20060173984" | "20060173993" | "20060173994" | "20060174238" | "6775829" | "6865737").PN.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 11:12
L2	25	("6023755" | "6425017" | "6618737" | "6868442" | "5625811" | "5675791" | "7401238" | "7478097" | "7571154" | "5687363" | "5721909" | "5764977" | "5835757" | "5892828" | "5475858" | "5751962" | "5960421" | "5991821" | "5996087" | "6026447" | "4914619" | "4845744" | "5193202" | "5337311" | "5402394").PN.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 12:20
L3	2	((JAGANE) near2 (SUNDAR)).INV.	US-PGPUB; USPAT; USOCR	OR	ON	2010/08/26 13:10
L4	41	((SANJAY) near2 (RADIA)).INV.	US-PGPUB; USPAT; USOCR	OR	ON	2010/08/26 13:10

Netflix, Inc. - Ex. 1002, Page 000245
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

L5	16	((DAVID) near2 (HENSELER)).INV.	US-PGPUB; USPAT; USOCR	OR	ON	2010/08/26 13:10
L6	114	(application near10 node) same (network) and (image same (repository database)) and (virtual near4 machine) and (automat $4 (artificial near5 intelligence))	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 13:19
L7	22	(application near10 node) same (network) and (image same (repository database)) and (virtual near4 machine) and (automat $4 (artificial near5 intelligence)).clm.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 13:19
L8	1	L3 and (application near10 node) same (network) and (image same (repository database)) and (virtual near4 machine) and (automat$4 (artificial near5 intelligence)). clm.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 13:19
L9	3	L4 and (application near10 node) same (network) and (image same (repository database)) and (virtual near4 machine) and (automat$4 (artificial near5 intelligence)). clm.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 13:20
L10	3	L5 and (application near10 node) same (network) and (image same (repository database)) and (virtual near4 machine) and (automat$4 (artificial near5 intelligence)). clm.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 13:22
L16	19	L7 and (virtual near2 machine) same software	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 14:11

Netflix, Inc. - Ex. 1002, Page 000246
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

L17	19	L16 and (node or (comput$4))	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 14:11
L18	16	L17 and (server and platform)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 14:11
L19	1	L18 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 14:12
L20	124	(image copy) near10 (software near10 application) same node and rule	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 14:34
L21	46	L20 and (server and platform)	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 14:34
L22	7	L21 and ((server and platform) and (virtual near4 machine) and (automat$4 (artificial near5 intelligence)))	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 14:36
L23	1	(US-20050005200-$).did.	US-PGPUB	OR	ON	2010/08/26 14:44
L27	97353	(709/223-227 "707"/$ 711/6 719/315-324).ccls.	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 15:00
L28	1221	L27 and ((server and platform) and (virtual near4 machine) and (automat$4 (artificial near5 intelligence)))	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 15:00
L30	608	L28 and (virtual near2 machine) same software	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 15:00
L31	309	L30 and @py<="2006"	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 15:01
L34	27	L31 and (software) same node and rule	US-PGPUB; USPAT; IBM_TDB	OR	ON	2010/08/26 15:01

EAST Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp

Netflix, Inc. - Ex. 1002, Page 000247
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

L24	12	(application near10 node) same (network) and (image same (repository database)) and (virtual near4 machine) and (automat $4 (artificial near5 intelligence)).clm.	USPAT; UPAD	OR	ON	2010/08/26 14:49
L25	2	L24 and @py<="2006"	USPAT; UPAD	OR	ON	2010/08/26 14:51
L26	42161	(709/223-227 "707"/$ 711/6 719/315-324).ccls.	USPAT; UPAD	OR	ON	2010/08/26 14:59

8/26/2010 3:02:16 PM
C:\Documents and Settings\aruiz1\My Documents\EAST\Workspaces\11694483.wsp

Netflix, Inc. - Ex. 1002, Page 000248
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Web Images Videos Maps News Shopping Gmail more ▼

Sign in

Google scholar application node software copy or image autom Search Advanced Scholar Search Scholar Preferences

Scholar Articles and patents - 2006 include citations Create email alert

Results **1** - **10** of about **1,340**. (**0.23** sec)

Vmplants: Providing and managing **virtual machine** execution environments for grid computing — psu.edu [PDF]

I Krsul, A Ganguly, J Zhang, JAB Fortes, ... - Proceedings of the ..., 2004 - ieeexplore.ieee.org

... VM instance (eg installation of libraries, setup of user identities and machine credentials, **application** startup) and **...** DAG refers to a "blank" **virtual machine** with no **operating system** or **software** installed **...** Subsequent **nodes** define actions to be taken to bring the VM to a desired state **...**

Cited by 171 - Related articles - All 22 versions

[BOOK] Distributed **systems** — psu.edu [PDF]

AS Tanenbaum, M Van Steen - 2002 - Citeseer

... Distributed **Systems** – Networking and Internetworking Conceptual layering of protocol **software** Layer n **...** IP, ATM **virtual** circuits Data link Responsible for transmission of packets between **nodes** that are directly connected by a physical link. **...** Physical **Application** ATM layer **...**

Cited by 1727 - Related articles - View as HTML - Library Search - All 45 versions

An integrated experimental environment for distributed **systems** and networks — psu.edu [PDF]

B White, J Lepreau, L Stoller, R Ricci, S ... - ... **Operating Systems** ..., 2002 - portal.acm.org

... Local-Area Resources: Netbed **software** currently controls two clusters: one at the University of Utah com- prised of 168 PCs and another at the **...** A program object is associated with an ns **node** in the script and attaches arbitrary **applications** to the corresponding local **node**. **...**

Cited by 844 - Related articles - BL Direct - All 16 versions

SoftUDC: A **software**-based data center for utility computing

M Kallahalla, M Uysal, R Swaminathan, DE Lowell ... - Computer, 2004 - ieeexplore.ieee.org

... SoftUDC uses a **copy**-on-write clone of the first **operating system's** boot **image** to initiate a second OS instance on the **node**. **...** OS to the upgraded state by installing an upgrade to the OS, applying a patch, setting up a new version of the **application**, or reconfiguring other **software**. **...**

Cited by 73 - Related articles - BL Direct - All 5 versions

[CITATION] Single **system image**

R Buyya, T Cortes, H Jin - ... Journal of High Performance Computing **Applications**, 2001

Cited by 32 - Related articles - BL Direct - All 5 versions

GridBuilder: A tool for creating **virtual** grid testbeds — tcd.ie [PDF]

S Childs, B Coghlan, J ... - Second IEEE International ..., 2006 - ieeexplore.ieee.org

... The ability to track particular snapshots is also useful when comparing behaviour with different **software** versions **...** developers can quickly create fresh VMs based on the base filesystem im- age

Netflix, Inc. - Ex. 1002, Page 000249
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

for the **node** type **...** Another good **application** of the tool is the creation of tutorial testbeds **...**
Cited by 12 - Related articles - All 8 versions

[PDF] Managing mixed-use clusters with Cluster-on-Demand psu.edu [PDF]
J Moore, D Irwin, L Grit, S Sprenkle, J Chase - ... University Department of ..., 2002 - Citeseer
... Examples include Beowulf, load-leveling batch schedulers, grid managers, enterprise **application** monitors [25], cluster-based network **...** Because COD uses bit-**copy** partition **images** by default, priming is independent of the **operating system** and other **node software**. **...**
Cited by 31 - Related articles - View as HTML - All 25 versions

Scalability, fidelity, and containment in the potemkin **virtual** honeyfarm psu.edu [PDF]
M Vrable, J Ma, J Chen, D Moore, E ... - ... **Operating Systems** ..., 2005 - portal.acm.org
... new VMs quickly by copying and modifying a host reference **image**, thus avoiding **...** or **application** initialization, while the latter optimizes this **copy** operation using **copy**-on-write **...** a collec- tion of monitored Internet hosts running common **operating system** and **application software**. **...**
Cited by 176 - Related articles - BL Direct - All 39 versions

MANTIS OS: An embedded multithreaded **operating system** for wireless micro sensor platforms psu.edu [PDF]
S Bhatti, J Carlson, H Dai, J Deng, J Rose, A ... - ... and **Applications**, 2005 - portal.acm.org
... forward **application** thread, which is avail- able along with the complete open source MANTIS **software** release at **...** of the sleep, but differs in the behavior of the OS after all **application** threads have **...** A typical wireless **node** will last only a few days on two AA batteries when used for **...**
Cited by 222 - Related articles - All 25 versions

[PDF] The Jalapeno **virtual machine** ucsb.edu [PDF]
B Alpern, CR Attanasio, JJ Barton, MG Burke, P ... - IBM **Systems** ..., 2000 - cs.ucsb.edu
... The deliverException method is called with **software**-generated exceptions as well. **...** features of the Java language is its provision for loading classes during the execution of an **application**. **... system** routine is called (through the C library) with an address to which to **copy** its result. **...**
Cited by 538 - Related articles - View as HTML - BL Direct - All 20 versions

Create email alert

Gooooooooogle
Result Page: 1 2 3 4 5 6 7 8 9 10 Next

application node software copy or in Search

Go to Google Home - About Google - About Google Scholar

©2010 Google

Netflix, Inc. - Ex. 1002, Page 000250
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/694,483	03/30/2007	Jagane Sundar	1072-021US01

CONFIRMATION NO. 1040

5073
BAKER BOTTS L.L.P.
2001 ROSS AVENUE
SUITE 600
DALLAS, TX 75201-2980

POA ACCEPTANCE LETTER

OC00000041998701

Date Mailed: 06/11/2010

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 06/03/2010.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

/sibrahim/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Netflix, Inc. - Ex. 1002, Page 000251
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/694,483	03/30/2007	Jagane Sundar	1072-021US01

CONFIRMATION NO. 1040

28863
SHUMAKER & SIEFFERT, P. A.
1625 RADIO DRIVE
SUITE 300
WOODBURY, MN 55125

POWER OF ATTORNEY NOTICE

OC000000041998661

Date Mailed: 06/11/2010

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 06/03/2010.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

/sibrahim/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Netflix, Inc. - Ex. 1002, Page 000252
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

PTO/SB/80 (11-08)
Approved for use through 11/30/2011. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

POWER OF ATTORNEY TO PROSECUTE APPLICATIONS BEFORE THE USPTO

I hereby revoke all previous powers of attorney given in the application identified in the attached statement under 37 CFR 3.73(b).

I hereby appoint:

☑ Practitioners associated with the Customer Number: 05073

OR

☐ Practitioner(s) named below (if more than ten patent practitioners are to be named, then a customer number must be used):

Name	Registration Number	Name	Registration Number

as attorney(s) or agent(s) to represent the undersigned before the United States Patent and Trademark Office (USPTO) in connection with any and all patent applications assigned only to the undersigned according to the USPTO assignment records or assignment documents attached to this form in accordance with 37 CFR 3.73(b).

Please change the correspondence address for the application identified in the attached statement under 37 CFR 3.73(b) to:

☑ The address associated with Customer Number: 05073

OR

☐ Firm or Individual Name			
Address			
City		State	Zip
Country			
Telephone		Email	

Assignee Name and Address:

Computer Associates Think, Inc.
One CA Plaza
Islandia, New York 11749

A copy of this form, together with a statement under 37 CFR 3.73(b) (Form PTO/SB/96 or equivalent) is required to be filed in each application in which this form is used. The statement under 37 CFR 3.73(b) may be completed by one of the practitioners appointed in this form if the appointed practitioner is authorized to act on behalf of the assignee, and must identify the application in which this Power of Attorney is to be filed.

SIGNATURE of Assignee of Record

The individual whose signature and title is supplied below is authorized to act on behalf of the assignee

Signature		Date	May 26, 2010
Name	Steven P. Klocinski	Telephone	631.342.2653
Title	Secretary, Computer Associates Think, Inc.		

This collection of information is required by 37 CFR 1.31, 1.32 and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

9810

PTO/SB/96 (01-08)
Approved for use through 06/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(b)

Applicant/Patent Owner: Jagane Sundar et al.

Application No./Patent No.: 11/694,483 Filed/Issue Date: March 30, 2007

Entitled: Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images

Computer Associates Think, Inc., a Delaware corporation
(Name of Assignee) (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that it is:

1. ☑ the assignee of the entire right, title, and interest; or

2. ☐ an assignee of less than the entire right, title and interest
(The extent (by percentage) of its ownership interest is _____ %)

in the patent application/patent identified above by virtue of either:

A. ☐ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

OR

B. ☑ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

1. From: the Inventors To: Cassatt Corporation
The document was recorded in the United States Patent and Trademark Office at
Reel 019410, Frame 0430, or for which a copy thereof is attached.

2. From: Cassatt Corporation To: Computer Associates Think, Inc.
The document was recorded in the United States Patent and Trademark Office at
Reel 022793, Frame 0113, or for which a copy thereof is attached.

3. From: _____ To: _____
The document was recorded in the United States Patent and Trademark Office at
Reel _____, Frame _____, or for which a copy thereof is attached.

☐ Additional documents in the chain of title are listed on a supplemental sheet.

☐ As required by 37 CFR 3.73(b)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (*i.e.*, a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

Signature	May 26, 2010 Date
Steven P. Klocinski Printed or Typed Name	631.342.2653 Telephone Number
Secretary, Computer Associates Think, Inc. Title	

This collection of information is required by 37 CFR 3.73(b). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	7736424
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	28863
Filer:	Samir A Bhavsar/Christine Fortier
Filer Authorized By:	Samir A Bhavsar
Attorney Docket Number:	1072-021US01
Receipt Date:	03-JUN-2010
Filing Date:	30-MAR-2007
Time Stamp:	11:40:27
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	PowerOfAttorney.PDF	139394 e675d7b1c14cda09572ffd30a78a521b65796fde	no	2

Warnings:	
Information:	

Total Files Size (in bytes):	139394

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Jagane Sundar, Sanjay Radia and David A. Henseler	Confirmation No.	1040
Serial No.:	11/694,483	Customer No.:	28863
Filed:	March 30, 2007	Group Art Unit:	2161
Examiner:	Unknown	Docket No.:	1072-021US01
Title:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES		

CERTIFICATE UNDER 37 CFR 1.8 I hereby certify that this correspondence is being transmitted via the United States Patent and Trademark Office electronic filing system on August 17, 2009.

By: Nancy J. Burr
Name: Nancy J. Burr

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

Mail Stop Amendment
Commissioner for Patents
Alexandria, VA 22313-1450

Dear Sir:

Applicant submits the references listed on the attached form PTO-1449. This statement is being filed, to the best of Applicant's knowledge, before the receipt of a first Office Action on the merits.

The copy of the U.S. patent is not enclosed as this requirement has been waived by the U.S. Patent Office.

No fee is believed due. Please apply any deficiencies or credits to Deposit Account No. 50-1778.

Respectfully submitted,

Date: 08-17-2009

By: Gregory J. Albin
Name: Gregory J. Albin
Reg. No.: 56,292

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.286.8366
Facsimile: 651.735.1102

Form 1449*	Docket Number:	Application Number:
INFORMATION DISCLOSURE STATEMENT	1072-021US01	11/694,483
IN AN APPLICATION	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
(Use several sheets if necessary)	Filing Date: March 30, 2007	Group Art Unit: 2161
	Examiner Name: Unknown	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate
	2005/0005200 A1	01/06/2005	Matena et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

EXAMINER	**Date Considered**

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

Electronic Acknowledgement Receipt

EFS ID:	5903308
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	28863
Filer:	Gregory James Albin/Nancy Burr
Filer Authorized By:	Gregory James Albin
Attorney Docket Number:	1072-021US01
Receipt Date:	17-AUG-2009
Filing Date:	30-MAR-2007
Time Stamp:	17:46:07
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Information Disclosure Statement (IDS) Filed (SB/08)	IDS.pdf	70875 ff18e016e04428c423ab873cb022a387f645e6c9	no	2

Warnings:	
Information:	

This is not an USPTO supplied IDS fillable form	
Total Files Size (in bytes):	70875

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

SHUMAKER & SIEFFERT, P.A.
1625 RADIO DRIVE, SUITE 300
WOODBURY, MINNESOTA 55125
TEL 651.735-1100
FAX 651.735-1102
WWW.SSIPLAW.COM

RECEIVED
CENTRAL FAX CENTER
JUN 20 2008

FACSIMILE SUBMISSION UNDER 37 CFR 1.8

TO:	FROM:
Mail Stop Amendment	Gregory J. Albin
COMPANY: U.S. Patent & Trademark Office	DATE: JUNE 20, 2008
FAX NUMBER: 1-571-273-8300	TOTAL NO. OF PAGES INCLUDING COVER: 3
PHONE NUMBER:	SENDER'S REFERENCE NUMBER: 1072-021US01
RE: Supplemental Information Disclosure Statement	YOUR REFERENCE NUMBER: 11/694,483

This facsimile message is intended for the sole use of the intended recipient(s) and may contain information that is confidential, privileged and/or attorneys' work product. Any review or distribution by any other person is prohibited. If you are not an intended recipient, please immediately contact the sender and delete all copies.

RECEIVED
CENTRAL FAX CENTER
JUN 20 2008

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Jagane Sundar; Sanjay Radia; David A. Henseler	Confirmation No.	1040
Serial No.:	11/694,483	Customer No.:	28863
Filed:	March 30, 2007	Group Art Unit:	2161
Examiner:	Unknown	Docket No.:	1072-021US01
Title:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES		

CERTIFICATE UNDER 37 CFR 1.8 I hereby certify that this correspondence is being transmitted via facsimile to the United States Patent and Trademark Office on June 20, 2008.

By: Nancy J. Burr
Name: Nancy J. Burr

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

Mail Stop Amendment
Commissioner for Patents
Alexandria, VA 22313-1450

Dear Sir:

Applicant submits the references listed on the attached form PTO-1449. This statement is being filed, to the best of Applicant's knowledge, before the receipt of a first Office Action on the merits.

Copies of the U.S. applications are not enclosed as this requirement has been waived by the U.S. Patent Office.

No fee is believed due. Please apply any deficiencies or credits to Deposit Account No. 50-1778.

Respectfully submitted,

Date: 06-20-2008

Gregory J. Albin
By: Gregory J. Albin
Reg. No. 56,292

Shumaker & Sieffert, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Phone: (651) 735-1100
Fax: (651) 735-1102

Netflix, Inc. - Ex. 1002, Page 000262
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

RECEIVED
CENTRAL FAX CENTER
JUN 20 2008

Form 1449*	Docket Number: 1072-021US01	Application Number: 11/694,483
INFORMATION DISCLOSURE STATEMENT	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
IN AN APPLICATION	Filing Date: March 30, 2007	Group Art Unit: 2161
(Use several sheets if necessary)	Examiner Name: Unknown	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

	U.S. Patent Application No. 11/607,819 entitled, "Automated Deployment and Configuration of Applications in an Autonomically Controlled Distributed Computing System", filed December 1, 2006, Docket No. 1072-015US01
	U.S. Patent Application No. 11/607,820 entitled, "Automated Deployment and Configuration of Applications in an Autonomically Controlled Distributed Computing System", filed December 1, 2006, Docket No. 1072-016US01

EXAMINER	**Date Considered**

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/694,483	03/30/2007	Jagane Sundar	1072-021US01

CONFIRMATION NO. 1040

28863
SHUMAKER & SIEFFERT, P. A.
1625 RADIO DRIVE
SUITE 300
WOODBURY, MN55125

Title: DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

Publication No. US-2007-0233698-A1
Publication Date: 10/04/2007

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Pre-Grant Publication Division, 703-605-4283

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Jagane Sundar; Sanjay Radia; David A. Henseler	Confirmation No.	1040
Serial No.:	11/694,483		
Filed:	March 30, 2007	Customer No.:	28863
Examiner:	Unknown	Group Art Unit:	2161
Docket No.:	1072-021US01		
Title:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES		

INFORMATION DISCLOSURE STATEMENT

Mail Stop Amendment
Commissioner for Patents
Alexandria, VA 22313-1450

Dear Sir:

Applicant submits the references listed on the attached form PTO-1449. This statement is being filed, to the best of Applicant's knowledge, before the receipt of a first Office Action on the merits.

Applicant has enclosed a copy of each article cited and each foreign document cited. Copies of the U.S. patents and publications are not enclosed as this requirement has been waived by the U.S. Patent Office.

No fee is believed due. Please apply any deficiencies or credits to Deposit Account No. 50-1778.

Respectfully submitted,

Date: July 9, 2007

By: Kent J. Sieffert
Reg. No. 41,312

Shumaker & Sieffert, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Phone: (651) 735-1100
Fax: (651) 735-1102

Form 1449*	Docket Number:	Application Number:
INFORMATION DISCLOSURE STATEMENT	1072-021US01	11/694,483
IN AN APPLICATION	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
(Use several sheets if necessary)	Filing Date: March 30, 2007	Group Art Unit: Unknown
	Examiner Name: Unknown	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate
	2003/0177176 A1	09/18/2003	Hirschfeld et al.	
	2003/0126265 A1	07/03/2003	Aziz et al.	
	2003/0140282 A1	07/24/2003	Kaler et al.	
	2006/0173895 A1	08/03/2006	Engquist et al.	
	2006/0173994 A1	08/03/2006	Emeis et al.	
	2006/0173984 A1	08/03/2006	Emeis et al.	
	2006/0173857 A1	08/03/2006	Jackson	
	2006/0173856 A1	08/03/2006	Jackson et al.	
	2002/0156877 A1	10/24/2002	Lu et al.	
	2004/0187104 A1	09/23/2004	Sardesai et al.	
	2005/0193265 A1	09/01/2005	Lin et al.	
	2004/0260734 A1	12/23/2004	Ren et al.	
	6,775,829 B1	08/10/2004	Kroening	
	2004/0088694 A1	05/06/2004	Ho	
	2006/0174238 A1	08/03/2006	Henseler et al.	
	2006/0173993 A1	08/03/2006	Henseler et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No
	WO/2006/083893 A1	08/10/2006	PCT		
	WO/2006/083894 A1	08/10/2006	PCT		
	WO/2006/083901 A1	08/10/2006	PCT		
	WO/2006/083895 A1	08/10/2006	PCT		
	WO/2006/083727 A1	08/10/2006	PCT		
	WO/2006/081503 A1	08/03/2006	PCT		

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

EXAMINER	**Date Considered**

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

Form 1449*	Docket Number: 1072-021US01	Application Number: 11/694,483
INFORMATION DISCLOSURE STATEMENT	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
IN AN APPLICATION	Filing Date: March 30, 2007	Group Art Unit: Unknown
(Use several sheets if necessary)	Examiner Name: Unknown	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

	E.N. Herness et al., "WebSphere Application Server: A Foundation for on Demand Computing," IBM Systems Journal IBM, Vol. 43, No. 2, pgs. 213-237, XP-002383791, 2004.
	B. Urgaonkar et al., "Resource Overbooking and Application Profiling in Shared Hosting Platforms," Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 17 pgs., XP-002387427, 2002.
	G. Lodi et al., "QoS-aware Clustering of Application Servers," Proceedings of the 1st IEEE Workshop on Quality of Service for Application Servers, In Conjunction With the 23rd International Symposium on Reliable Distributed Systems, 6 pages, XP-002383792, October 17, 2004.
	Preinstalling Microsoft Windows XP by Using the OEM Preinstallation Kit, Part 1, XP-002301441, April 4, 2003, 24 pages.
	R. Mark Koan et al., *It Takes a Village to Build an Image*, XP-002384269, 2003, pages 200-207.

EXAMINER	**Date Considered**

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820 (Also form PTO-1449) — Patent and Trademark Office, U.S. Department of Commerce

Form 1449*	Docket Number: 1072-021US01	Application Number: 11/694,483
INFORMATION DISCLOSURE STATEMENT IN AN APPLICATION (Use several sheets if necessary)	Applicant: Jagane Sundar; Sanjay Radia; David A. Henseler	
	Filing Date: March 30, 2007	Group Art Unit: 2161
	Examiner Name: Unknown	

U.S. PATENT DOCUMENTS

Examiner Initial	Document Number	Issue/Document Publication Date	Name	Filing Date If Appropriate
	2004/0181794 A1	09/16/2004	Coleman et al.	
	6,865,737 B1	03/08/2005	Lucas et al.	
	2004/0162741 A1	08/19/2004	Flaxer et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document Number	Publication Date	Country	Translation Yes	Translation No
	WO/2003/085526 A1	10/16/2003	PCT		

OTHER DOCUMENTS (Including Authors, Title of Item, Page(s), Vol/Issue No., Publisher, Place of Publication)

EXAMINER	**Date Considered**

*Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Based on Form PTO-FB-A820
(Also form PTO-1449)

Patent and Trademark Office, U.S. Department of Commerce

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 August 2006 (10.08.2006)

PCT

(10) International Publication Number
WO 2006/083894 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2006/003437

(22) International Filing Date: 31 January 2006 (31.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

11/047,468	31 January 2005 (31.01.2005)	US
11/074,291	7 March 2005 (07.03.2005)	US
11/176,161	7 July 2005 (07.07.2005)	US
11/191,882	28 July 2005 (28.07.2005)	US

(71) Applicant *(for all designated States except US)*: **CASSATT CORPORATION** [US/US]; 1740 Technology Drive, San Jose, California 95110 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **EMEIS, Mark, W.** [US/US]; 20250 Silver Horn Lane, Monument, Colorado 80132 (US). **JACKSON, Jerry, R.** [US/US]; 11270 Wakely Road, Colorado Springs, Colorado 80908 (US).

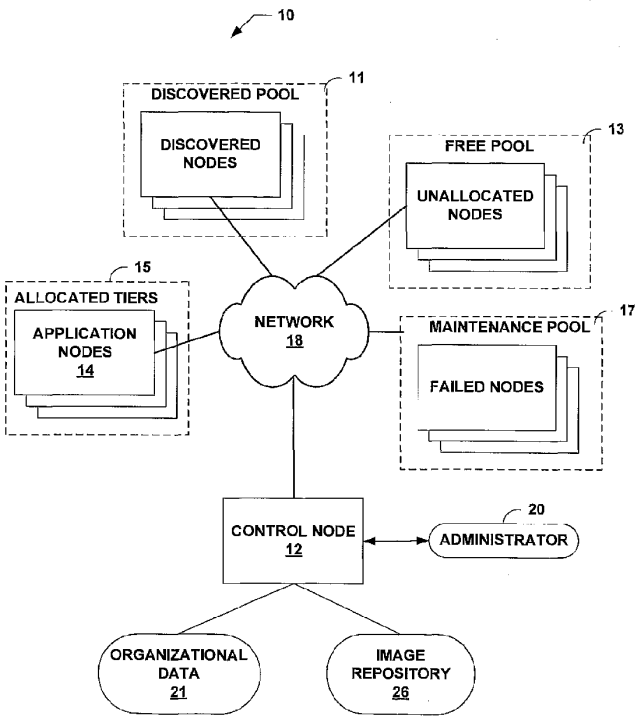
(74) Agent: **SIEFFERT, Kent, J.**; SHUMAKER & SIEFFERT, P.A., 8425 SEASONS PARKWAY, Suite 105, St. Paul, Minnesota 55125 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: AUTONOMIC CONTROL OF A DISTRIBUTED COMPUTING SYSTEM USING AN APPLICATION MATRIX TO CONTROL APPLICATION DEPLOYMENT



(57) Abstract: A distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of distributed computing system. The control node includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules. A pluggable, application-specific application matrix is selected and installed within the control node. The application matrix contains a logical definition of the applications, and parameters for controlling the deployment of the applications within the distributed computing system.

WO 2006/083894 A1

FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

AUTONOMIC CONTROL OF A DISTRIBUTED COMPUTING SYSTEM USING AN APPLICATION MATRIX TO CONTROL APPLICATION DEPLOYMENT

TECHNICAL FIELD

[0001] The invention relates to computing environments and, more specifically, to distributed computing systems.

BACKGROUND

[0002] Distributed computing systems are increasingly being utilized to support business as well as technical applications. Typically, distributed computing systems are constructed from a collection of computing nodes that combine to provide a set of processing services to implement the distributed computing applications. Each of the computing nodes in the distributed computing system is typically a separate, independent computing device interconnected with each of the other computing nodes via a communications medium, e.g., a network.

[0003] One challenge with distributed computing systems is the organization, deployment and administration of such a system within an enterprise environment. For example, it is often difficult to manage the allocation and deployment of enterprise computing functions within the distributed computing system. An enterprise, for example, often includes several business groups, and each group may have competing and variable computing requirements.

SUMMARY

[0004] In general, the invention is directed to a distributed computing system that conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.

[0005] As described herein, the model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of the distributed computing system. The control nodes are

Netflix, Inc. - Ex. 1002, Page 000271
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

[0006] In one embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network and a control node. The control node comprises a set of one or more applications to be executed on the application nodes, an application matrix that includes parameters for controlling the deployment of the applications within the distributed computing system, and an automation unit having one or more rule engines that provide autonomic control of the application nodes and the applications in accordance with a set of one or more rules and the application matrix.

[0007] In another embodiment, a method comprises generating an application matrix that specifies data for controlling the deployment of a set of applications within a distributed computing system, and performing operations to provide autonomic control over the deployment of the applications within the distributed computing system in accordance with parameters of the application matrix.

[0008] In another embodiment, a computer-readable medium comprises instructions. The instructions cause the processor to generate an application matrix that specifies data for controlling the deployment of a set of applications within a distributed computing system, and perform operations to provide autonomic control over the deployment of the applications within the distributed computing system in accordance with parameters of the application matrix.

[0009] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0010] FIG. 1 is a block diagram illustrating a distributed computing system constructed from a collection of computing nodes.

[0011] FIG. 2 is a schematic diagram illustrating an example of a model of an enterprise that logically defines an enterprise fabric.

[0012] FIG. 3 is a flow diagram that provides a high-level overview of the operation of a control node when configuring the distributed computing system.

Netflix, Inc. - Ex. 1002, Page 000272
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0013] FIG. 4 is a flow diagram illustrating exemplary operation of the control node when assigning computing nodes to node slots of tiers.

[0014] FIG. 5 is a flow diagram illustrating exemplary operation of a control node when adding an additional computing node to a tier to meet additional processing demands.

[0015] FIG. 6 is a flow diagram illustrating exemplary operation of a control node harvesting excess node capacity from one of the tiers and returning the harvested computing node to the free pool.

[0016] FIG. 7 is a screen illustration of an exemplary user interface for defining tiers in a particular domain.

[0017] FIG. 8 is a screen illustration of an exemplary user interface for defining properties of the tiers.

[0018] FIG. 9 is a screen illustration of an exemplary user interface for viewing and identify properties of a computing node.

[0019] FIG. 10 is a screen illustration of an exemplary user interface for viewing software images.

[0020] FIG. 11 is a screen illustration of an exemplary user interface for viewing a hardware inventory report.

[0021] FIG. 12 is a screen illustration of an exemplary user interface for viewing discovered nodes that are located in the free pool.

[0022] FIG. 13 is a screen illustration of an exemplary user interface for viewing users of a distributed computing system.

[0023] FIG. 14 is a screen illustration of an exemplary user interface for viewing alerts for the distributed computing system.

[0024] FIG. 15 is a block diagram illustrating one embodiment of control node that includes a monitoring subsystem, a service level automation infrastructure (SLAI), and a business logic tier (BLT).

[0025] FIG. 16 is a block diagram illustrating one embodiment of the monitoring subsystem.

[0026] FIG. 17 is a block diagram illustrating one embodiment of the SLAI in further detail.

[0027] FIG. 18 is a block diagram of an example working memory associated with rule engines of the SLAI.

Netflix, Inc. - Ex. 1002, Page 000273
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0028] FIG. 19 is a block diagram illustrating an example embodiment for the BLT of the control node.

[0029] FIG. 20 is a block diagram illustrating one embodiment of a rule engine in further detail.

[0030] FIG. 21 is a block diagram illustrating another example embodiment of the control node.

[0031] FIG. 22 is a flowchart illustrating an exemplary mode of initiating a control node utilizing an application matrix.

[0032] FIG. 23 is a block diagram illustrating an exemplary application matrix.

DETAILED DESCRIPTION

[0033] FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide distributed processing.

[0034] In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated tiers 15 and a maintenance pool 17. In addition, distributed computing system 10 includes at least one control node 12.

[0035] Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

[0036] Discovered pool 11 includes a set of discovered nodes that have been automatically "discovered" within distributed computing system 10 by control node 12. For example, control node 12 may monitor dynamic host communication protocol (DHCP) leases to discover the connection of a node to network 18. Once detected, control node 12 automatically inventories the attributes for the discovered node and reassigns the discovered node to free pool 13. The node attributes identified during the inventory process may include a CPU count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics or other computing resources. Control node 12 may also

Netflix, Inc. - Ex. 1002, Page 000274
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

receive input identifying node attributes not detectable via the automatic inventory, such as whether the node includes I/O, such as HBA. Further details with respect to the automated discovery and inventory processes are described in U.S. Patent Application No. 11/070,851, having attorney docket no. 1072-009US01, entitled "AUTOMATED DISCOVERY AND INVENTORY OF NODES WITHIN AN AUTONOMIC DISTRIBUTED COMPUTING SYSTEM," filed March 2, 2005.

[0037] Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated tiers 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to add an application node to allocated tiers 15 to increase processing capacity of distributed computing system 10.

[0038] In general, allocated tiers 15 include one or more tiers of application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory). Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated tiers 15.

[0039] Control node 12 provides the system support functions for managing distributed computing system 10. More specifically, control node 12 manages the roles of each computing node within distributed computing system 10 and the execution of software applications within the distributed computing system. In general, distributed computing system 10 includes at least one control node 12, but may utilize additional control nodes to assist with the management functions.

[0040] Other control nodes 12 (not shown in FIG. 1) are optional and may be associated with a different subset of the computing nodes within distributed computing system 10. Moreover, control node 12 may be replicated to provide primary and backup administration functions, thereby allowing for graceful handling a failover in the event control node 12 fails.

[0041] Network 18 provides a communications interconnect for control node 12 and application nodes 14, as well as discovered nodes, unallocated nodes and failed nodes. Communications network 18 permits internode communications among the computing

Netflix, Inc. - Ex. 1002, Page 000275
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

nodes as the nodes perform interrelated operations and functions. Communications network 18 may comprise, for example, direct connections between one or more of the computing nodes, one or more customer networks maintained by an enterprise, local area networks (LANs), wide area networks (WANs) or a combination thereof. Communications network 18 may include a number of switches, routers, firewalls, load balancers, and the like.

[0042] In one embodiment, each of the computing nodes within distributed computing system 10 executes a common general-purpose operating system. One example of a general-purpose operating system is the Windows™ operating system provided by Microsoft Corporation. In some embodiments, the general-purpose operating system such as the Linux kernel may be used.

[0043] In the example of FIG. 1, control node 12 is responsible for software image management. The term "software image" refers to a complete set of software loaded on an individual computing node to provide an execution environment for one or more applications. The software image including the operating system and all boot code, middleware files, and may include application files. As described below embodiments of the invention provide application-level autonomic control over the deployment, execution and monitoring of applications onto software images associated with application nodes 14.

[0044] System administrator 20 may interact with control node 12 and identify the particular types of software images to be associated with application nodes 14. Alternatively, administration software executing on control node 12 may automatically identify the appropriate software images to be deployed to application nodes 14 based on the input received from system administrator 20. For example, control node 12 may determine the type of software image to load onto an application node 14 based on the functions assigned to the node by system administrator 20. Application nodes14 may be divided into a number of groups based on their assigned functionality. As one example, application nodes 14 may be divided into a first group to provide web server functions, a second group to provide business application functions and a third group to provide database functions. The application nodes 14 of each group may be associated with different software images.

[0045] Control node 12 provides for the efficient allocation and management of the various software images within distributed computing system 10. In some embodiments, control node 12 generates a "golden image" for each type of software image that may be

Netflix, Inc. - Ex. 1002, Page 000276
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

deployed on one or more of application nodes 14. As described herein, the term "golden image" refers to a reference copy of a complete software stack for providing an execution environment for applications.

[0046] System administrator 20 may create a golden image by installing an operating system, middleware and software applications on a computing node and then making a complete copy of the installed software. In this manner, a golden image may be viewed as a "master copy" of the software image for a particular computing function. Control node 12 maintains a software image repository 26 that stores the golden images associated with distributed computing system 10.

[0047] Control node 12 may create a copy of a golden image, referred to as an "image instance," for each possible image instance that may be deployed within distributed computing system 10 for a similar computing function. In other words, control node 12 pre-generates a set of K image instances for a golden image, where K represents the maximum number of image instances for which distributed computing system 10 is configured for the particular type of computing function. For a given computing function, control node 12 may create the complete set of image instance even if not all of the image instances will be initially deployed. Control node 12 creates different sets of image instances for different computing functions, and each set may have a different number of image instances depending on the maximum number of image instances that may be deployed for each set. Control node 12 stores the image instances within software image repository 26. Each image instance represents a collection of bits that may be deployed on an application node.

[0048] Further details of software image management are described in co-pending U.S. patent application Ser. No. 11/046,133, entitled "MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005 and co-pending U.S. patent application Ser. No. 11/046,152, entitled "UPDATING SOFTWARE IMAGES ASSOCIATED WITH A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005.

[0049] In general, distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

[0050] As used herein, the "fabric" level generally refers to the logical constructs that allow for definition, deployment, partitioning and management of distinct enterprise

Netflix, Inc. - Ex. 1002, Page 000277
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

applications. In other words, fabric refers to the integrated set of hardware, system software and application software that can be "knitted" together to form a complete enterprise system. In general, the fabric level consists of two elements: fabric components or fabric payload. Control node 12 provides fabric management and fabric services as described herein.

[0051] In contrast, a "domain" is a logical abstraction for containment and management within the fabric. The domain provides a logical unit of fabric allocation that enables the fabric to be partitioned amongst multiple uses, e.g. different business services.

[0052] Domains are comprised of tiers, such as a 4-tier application model (web server, application server, business logic, persistence layer) or a single tier monolithic application. Fabric domains contain the free pool of devices available for assignment to tiers.

[0053] A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

[0054] The lowest level, the node level, includes the physical components of the fabric. This includes computing nodes that, as described above, provide operating environments for system applications and enterprise software applications. In addition, the node level may include network devices (e.g., Ethernet switches, load balancers and firewalls) used in creating the infrastructure of network 18. The node level may further include network storage nodes that are network connected to the fabric.

[0055] System administrator 20 accesses administration software executing on control node 12 to logically define the hierarchical organization of distributed computing system 10. For example, system administrator 20 may provide organizational data 21 to develop a model for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.

[0056] More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation

Netflix, Inc. - Ex. 1002, Page 000278
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

management (CRM) service. System administrator 20 further defines one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a web tier, an application tier and a database tier. In this manner, distributed computing system 10 may be configured to automatically provide web server functions, business application functions and database functions.

[0057] For each of the tiers, control node 12 creates a number of "node slots" equal to the maximum number of application nodes 14 that may be deployed. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot. The node slots may, for instance, identify a particular software image instance associated with an application node 14 as well as a network address associated with that particular image instance.

[0058] In this manner, each of the tiers include one or more node slots that reference particular software image instances to boot on the application nodes 14 to which each software image instance is assigned. The application nodes 14 to which control node 12 assigns the image instances temporarily inherit the network address assigned to the image instance for as long as the image instance is deployed on that particular application node. If for some reason the image instance is moved to a different application node 14, control node 12A moves the network address to that new application node.

[0059] System administrator 20 may further define specific node requirements for each tier of the fabric. For example, the node requirements specified by system administrator 20 may include a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics that may be detected on the individual computing nodes. System administrator 20 may also specify user-defined hardware attributes of the computing nodes, such as whether I/O (like HBA) is required. The user-defined hardware attributes are typically not capable of detection during an automatic inventory. In this manner, system administrator 20 creates a list of attributes that the tier requires of its candidate computing nodes. In addition, particular node requirements may be defined for software image instances.

[0060] In addition to the node requirements described above, system administrator 20 may further define policies that are used when re-provisioning computing nodes within the fabric. System administrator 20 may define policies regarding tier characteristics,

Netflix, Inc. - Ex. 1002, Page 000279
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

such as a minimum number of nodes a tier requires, an indication of whether or not a failed node is dynamically replaced by a node from free pool 13, a priority for each tier relative to other tiers, an indication of whether or not a tier allows nodes to be re-provisioned to other tiers to satisfy processing requirements by other tiers of a higher priority or other policies. Control node 12 uses the policy information input by system administrator 20 to re-provision computing nodes to meet tier processing capacity demands.

[0061] After receiving input from system administrator 20 defining the architecture and policy of the enterprise fabric, control node 12 identifies unallocated nodes within free pool 13 that satisfy required node attributes. Control node 12 automatically assigns unallocated nodes from free pool 13 to respective tier node slots of a tier. As will be described in detail herein, in one embodiment, control node 12 may assign computing nodes to the tiers in a "best fit" fashion. Particularly, control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20. The assignment of the computing nodes may occur on a tier-by-tier basis beginning with a tier with the highest priority and ending with a tier with the lowest priority. Alternatively, or in addition, assignment of computing nodes may be based on dependencies defined between tiers.

[0062] As will be described in detail below, control node 12 may automatically add unallocated nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to the free pool when the tier has excess capacity, transfer nodes from tier to tier to meet processing demands, or replace failed nodes with nodes from the free pool. Thus, computing resources, i.e., computing nodes, may be automatically shared between tiers and domains within the fabric based on user-defined policies to dynamically address high-processing demands, failures and other events.

[0063] FIG. 2 is a schematic diagram illustrating an example embodiment of organizational data 21 that defines a model logically representing an enterprise fabric in accordance with the invention. In the example illustrated in FIG. 2, control node 12 (FIG. 1) maintains organizational data 21 to define a simple e-commerce fabric 32.

[0064] In this example, e-commerce fabric 32 includes a storefront domain 34A and a financial planning domain 34B. Storefront domain 34A corresponds to the enterprise storefront domain and allows customers to find and purchase products over a network,

Netflix, Inc. - Ex. 1002, Page 000280
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

such as the Internet. Financial planning domain 34B allows one or more employees to perform financial planning tasks for the enterprise.

[0065] Tier level 31C includes one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. For example, storefront domain 34A includes a web server tier (labeled "web tier") 36A, a business application tier (labeled "app tier") 36B, and a database tier (labeled "DB tier") 36C. Web server tier 36A, business application tier 36B and database tier 36C interact with one another to present a customer with an online storefront application and services. For example, the customer may interact with web server tier 36A via a web browser. When the customer searches for a product, web server tier 36A may interacts with business application tier 36B, which may in turn access a database tier 36C. Similarly, financial planning domain 34B includes a financial planning tier 36D that provides subcomponents of applications and services of the financial planning domain 34B. Thus, in this example, a domain may include a single tier.

[0066] Tier level 31D includes one or more logical node slots 38A-38H ("node slots 38") within each of the tiers. Each of node slots 38 include node specific information, such as software resources for an application node 14 that is assigned to a respective one of the node slots 38. Node slots 38 may, for instance, identify particular software image instances within image repository 26 and map the identified software image instances to respective application nodes 14. As an example, node slots 38A and 38B belonging to web server tier 36A may reference particular software image instances used to boot two application nodes 14 to provide web server functions. Similarly, the other node slots 38 may reference software image instances to provide business application functions, database functions, or financial application functions depending upon the tier to which the node slots are logically associated.

[0067] Although in the example of FIG. 2, there are two node slots 38 corresponding to each tier, the tiers may include any number of node slots depending on the processing capacity needed on the tier. Furthermore, not all of node slots 38 may be currently assigned to an application node 14. For example, node slot 28B may be associated with an inactive software image instance and, when needed, may be assigned to an application node 14 for deployment of the software image instance.

[0068] In this example, organizational data 21 associates free node pool 13 with the highest-level of the model, i.e., e-commerce fabric 32. As described above, control node 12 may automatically assign unallocated nodes from free node pool 13 to at least a

Netflix, Inc. - Ex. 1002, Page 000281
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

portion of tier node slots 38 of tiers 36 as needed using the "best fit" algorithm described above or another algorithm. Additionally, control node 12 may also add nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.

[0069] Although not illustrated, the model for the enterprise fabric may include multiple free node pools. For example, the model may associate free node pools with individual domains at the domain level or with individual tier levels. In this manner, administrator 20 may define policies for the model such that unallocated computing nodes of free node pools associated with domains or tiers may only be used within the domain or tier to which they are assigned. In this manner, a portion of the computing nodes may be shared between domains of the entire fabric while other computing nodes may be restricted to particular domains or tiers.

[0070] FIG. 3 is a flow diagram that provides a high-level overview of the operation of control node 12 when configuring distributed computing system 10. Initially, control node 12 receives input from a system administrator defining the hierarchical organization of distributed computing system 10 (50). In one example, control node 12 receives input that defines a model that specifies a number of hierarchically arranged nodes as described in detail in FIG. 2. Particularly, the defined architecture of distributed computing system 10 includes an overall fabric having a number of hierarchically arranged domains, tiers and node slots.

[0071] During this process, control node 12 may receive input specifying node requirements of each of the tiers of the hierarchical model (52). As described above, administrator 20 may specify a list of attributes, e.g., a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), or local disk characteristics, that the tiers require of their candidate computing nodes. In addition, control node 12 may further receive user-defined custom attributes, such as requiring the node to have I/O, such as HBA connectivity. The node requirements or attributes defined by system administrator 20 may each include a name used to identify the characteristic, a data type (e.g., integer, long, float or string), and a weight to define the importance of the requirement.

[0072] Control node 12 identifies the attributes for all candidate computing nodes within free pool 13 or a lower priority tier (54). As described above, control node 12 may have already discovered the computing nodes and inventoried the candidate computing nodes

Netflix, Inc. - Ex. 1002, Page 000282
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

to identify hardware characteristics of all candidate computing nodes. Additionally, control node 12 may receive input from system administrator 20 identifying specialized capabilities of one or more computing nodes that are not detectable by the inventory process.

[0073] Control node 12 dynamically assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a "best fit" algorithm.

[0074] FIG. 4 is a flow diagram illustrating exemplary operation of control node 12 when assigning computing nodes to node slots of tiers. Initially, control node 12 selects a tier to enable (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

[0075] Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each of the tiers. Control node 12 retrieves the node requirements for the selected tier from the database.

[0076] In addition, control node 12 accesses the database and retrieves the computing node attributes of one of the unallocated computing nodes of free pool 13. Control node 12 compares the node requirements of the tier to the node attributes of the selected computing node (64).

[0077] Based on the comparison, control node 12 determines whether the node attributes of the computing node meets the minimum node requirements of the tier (66). If the node attributes of the selected computing node do not meet the minimum node requirements of the tier, then the computing node is removed from the list of candidate nodes for this particular tier (68). Control node 12 repeats the process by retrieving the node attributes of another of the computing nodes of the free pool and compares the node requirements of the tier to the node attributes of the computing node.

Netflix, Inc. - Ex. 1002, Page 000283
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0078] If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the node requirements of the tier (70). If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).

[0079] Control node 12 then determines whether the node count for the tier is met (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20. When the node count for the tier is met, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority, and repeats the process until all defined tiers are enabled, i.e., populated with application nodes (60).

[0080] If the node attributes of the selected computing node and the node requirements of the tier are not a perfect match control node 12 calculates and records a "processing energy" of the node (76). As used herein, the term "processing energy" refers to a numerical representation of the difference between the node attributes of a selected node and the node requirements of the tier. A positive processing energy indicates the node attributes more than satisfy the node requirements of the tier. The magnitude of the processing energy represents the degree to which the node requirements exceed the tier requirements.

[0081] After computing and recording the processing energy of the nodes, control node 12 determines whether there are more candidate nodes in free pool 13 (78). If there are additional candidate nodes, control node 12 repeats the process by retrieving the computing node attributes of another one of the computing nodes of the free pool of computing nodes and comparing the node requirements of the tier to the node attributes of the computing node (64).

[0082] When all of the candidate computing nodes in the free pool have been examined, control node 12 selects the candidate computing node having the minimum positive processing energy and assigns the selected computing node to a node slot of the tier (80). Control node 12 determines whether the minimum node count for the tier is met (82). If the minimum node count for the tier has not been met, control node 12 assigns the

Netflix, Inc. - Ex. 1002, Page 000284
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

computing node with the next lowest calculated processing energy to the tier (80). Control node 12 repeats this process until the node count is met. At this point, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority (60).

[0083] In the event there are an insufficient number of computing nodes in free pool 13, or an insufficient number of computing nodes that meet the tier requirements, control node 12 notifies system administrator 20. System administrator 20 may add more nodes to free pool 13, add more capable nodes to the free pool, reduce the node requirements of the tier so more of the unallocated nodes meet the requirements, or reduce the configured minimum node counts for the tiers.

[0084] FIG. 5 is a flow diagram illustrating exemplary operation of control node 12 when adding an additional computing node to a tier to meet increased processing demands. Initially, control node 12 or system administrator 20 identifies a need for additional processing capacity on one of the tiers (90). Control node 12 may, for example, identify a high processing load on the tier or receive input from a system administrator identifying the need for additional processing capacity on the tier.

[0085] Control node 12 then determines whether there are any computing nodes in the free pool of nodes that meet the minimum node requirements of the tier (92). When there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and assigns the node to the tier (95). As described in detail with respect to FIG. 4, control node 12 may determine whether there are any nodes that have node attributes that are an exact match to the node requirements of the tier. If an exact match is found, the corresponding computing node is assigned to a node slot of the tier. If no exact match is found, control node 12 computes the processing energy for each node and assigns the computing node with the minimum processing energy to the tier. Control node 12 remotely powers on the assigned node and remotely boots the node with the image instance associated with the node slot. Additionally, the booted computing node inherits the network address associated with the node slot.

[0086] If there are no adequate computing nodes in the free pool, i.e., no nodes at all or no nodes that match the minimal node requirements of the tier, control node 12 identifies the tiers with a lower priority than the tier needing more processing capacity (96).

[0087] Control node 12 determines which of the nodes of the lower priority tiers meet the minimum requirements of the tier in need of processing capacity (98). Control node 12 may, for example, compare the attributes of each of the nodes assigned to node slots of

Netflix, Inc. - Ex. 1002, Page 000285
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the lower priority tiers to the node requirements of the tier in need of processing capacity. Lower priority tiers that have the minimum number of computing nodes may be removed from possible tiers from which to harvest an application node. If, however, all the lower priority tiers have the minimum number of computing nodes defined for the respective tier, the lowest priority tier is selected from which to harvest the one or more nodes.

[0088] Control node 12 calculates the processing energy of each of the nodes of the lower priority tiers that meet the minimum requirements (100). The energies of the nodes are calculated using the differences between the node attributes and the node requirements of the tier needing additional capacity. Control node 12 selects the computing node with the lowest processing energy that meets the minimum requirements, and assigns the selected computing node to the tier in need of processing capacity (102, 95).

[0089] FIG. 6 is a flow diagram illustrating exemplary operation of control node 12 when harvesting excess node capacity from one of the tiers and returning the harvested computing node to free pool 13. Initially, control node 12 identifies a tier having excess node capacity (110). Control node 12 may, for example, periodically check the node capacity of the tiers to identify any tiers having excess node capacity. Performing a periodic check and removal of excess nodes increases the likelihood that a capable computing node will be in free pool 13 in the event one of the tiers needs additional node capacity.

[0090] When harvesting a node, control node 12 calculates the processing energy of all the nodes in the tier as described above with reference to FIG. 4 (112). Control node 12 identifies the node within the tier with the highest processing energy and returns the identified node to the free pool of nodes (114, 116). As described above, the node with the highest processing energy corresponds to the node whose node attributes are the most in excess of the node requirements of the tier.

[0091] Returning the node to the free pool may involve remotely powering off the computing node and updating the database to associate the harvested node with free pool 13. In addition, control node 12 updates the database to disassociate the returned node with the node slot to which it was assigned. At this point, the node no longer uses the network address associated with the image instance mapped to the node slot. Control node 12 may, therefore, assign a temporary network address to the node while the node is assigned to free pool 13.

[0092] FIG. 7 is a screen illustration of an exemplary user interface 120 presented by control node 12 with which administrator 20 interacts to define tiers for a particular

Netflix, Inc. - Ex. 1002, Page 000286
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

domain. In the example illustrated in FIG. 7, system administrator 20 has selected the "Collage Domain." User interface 120 presents the tiers that are currently in the selected domain. In the example illustrated, the Collage Domain includes three tiers, "test tier 1," "test tier 2," and "test tier 3." As shown in FIG. 7, in this example, each of the tiers includes two nodes. In addition, user interface 120 lists the type of software image currently deployed to application nodes for each of the tiers. In the example illustrated, image "applone (1.0.0)" is deployed to the nodes of test tier 1 and image "appltwo (1.0.0)" is deployed to the nodes of test tier 2. System administrator 20 may add one or more tiers to the domain by clicking on new tier button 122.

[0093] FIG. 8 is a screen illustration of an exemplary user interface 130 for defining properties of the tiers. In particular, user interface 130 allows system administrator 20 to input a name for the tier, a description of the tier, and an image associated with the tier. The image associated with the tier refers to a golden image from which image instances are generated and deployed to the nodes assigned to the tier.

[0094] When configuring a tier, system administrator 20 may elect to activate email alerts. For example, system administrator 20 may activate the email alerts feature in order to receive email alerts providing system administrator 20 with critical and/or non-critical tier information, such as a notification that a tier has been upgraded, a node of the tier has failed or the like. Furthermore, system administrator 20 may input various policies, such node failure rules. For example, system administrator 20 may identify whether control node 12 should reboot a node in case of failure or whether the failed node should automatically be moved to maintenance pool 17. Similarly, system administrator 20 may identify whether nodes assigned to the tier may be harvested by other tiers.

[0095] User interface 130 may also allow system administrator 20 to input node requirements of a tier. In order to input node requirements of a tier, system administrator 20 may click on the "Requirements" tab 132, causing user interface 130 to present an input area to particular node requirements of the tier.

[0096] FIG. 9 is a screen illustration of an exemplary user interface 140 for viewing and identifying properties of a computing node. User interface 140 allows system administrator 20 to define a name, description, and location (including a rack and slot) of a computing node. In addition user interface 140 may specify user-defined properties of a node, such as whether the computing node has I/O HBA capabilities.

[0097] User interface 140 also displays properties that control node 12 has identified during the computing node inventory process. In this example, user interface 140

Netflix, Inc. - Ex. 1002, Page 000287
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

presents system administrator 20 with the a CPU node count, a CPU speed, the amount of RAM, the disk size and other characteristics that are identifiable during the automated node inventory. User interface 140 additionally presents interface information to system administrator 20. Specifically, user interface 140 provides system administrator 20 with a list of components and their associated IP and MAC addresses.

[0098] User interface 140 also allows system administrator 20 to define other custom requirements. For example, system administrator 20 may define one or more attributes and add those attributes to the list of node attributes presented to system administrator 20.

[0099] FIG. 10 is a screen illustration of an exemplary user interface 150 for viewing software images. User interface 150 presents to a system administrator or another user a list of images maintained by control node 12 within image repository 26. The image list further includes the status of each image (i.e., either active or inactive), the version of the image, the operating system on which the image should be run, the operating system version on which the image should be run and a brief description of the image.

[0100] System administrator 20 or another user may select an image by clicking on the box in front of the image identifier/name and perform one or more actions on the image. Actions that system administrator 20 may perform on an image include deleting the image, updating the image, and the like. System administrator 20 may select one of the image actions via dropdown menu 152. In some embodiments, user interface 150 may further display other details about the images such as the node to which the images are assigned (if the node status is "active"), the network address associated with the images and the like.

[0101] FIG. 11 is a screen illustration of an exemplary user interface 160 for viewing a hardware inventory report. User interface 160 presents to system administrator 20 or another user a list of the nodes that are currently assigned to a domain. System administrator 20 may elect to view the nodes for the entire domain, for a single tier within the domain or for a single rack within a tier.

[0102] For each node, user interface 160 presents a node ID, a status of the node, the tier to which the node belongs, a hostname associated with the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0103] System administrator 20 or other user may select a node by clicking on the box in front of the node identifier/name and perform one or more actions on the node. Actions

Netflix, Inc. - Ex. 1002, Page 000288
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

that system administrator 20 may perform on the node include deleting the node, updating the node attributes or other properties of the node, and the like. System administrator 20 may select one of the node actions via dropdown menu 162.

[0104] FIG. 12 is a screen illustration of an exemplary user interface 170 for viewing discovered nodes that are located in discovered pool 11. For each node, user interface 170 presents a node ID, a state of the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0105] FIG. 13 is a screen illustration of an exemplary user interface 180 for viewing users of distributed computing system 10. User interface 180 presents a list of users as well as the role assigned to each of the users and the status of each of the users. Thus, system administrator 20 may define different roles to each of the users. For example, a user may be either an operator (i.e., general user) or an administrator. System administrator 20 may add a new user to the list of users by clicking on the "New User" button 182.

[0106] FIG. 14 is a screen illustration of an exemplary user interface 190 for viewing alerts for distributed computing system 10. For each of the alerts, user interface 190 identifies the severity of the alert, whether the alert has been acknowledged, an object associated with the alert, an event associated with the alert, a state of the alert, a user associated with the alert and a date associated with the alert.

[0107] System administrator 20 or other user may select an alert by clicking on the box in front of the logged alert and perform one or more actions on the logged alert. Actions that system administrator 20 may perform include deleting the alert, changing the status of the alert, or the like. System administrator 20 may specify the log actions via dropdown menu 192.

[0108] FIG. 15 is a block diagram illustrating one embodiment of control node 12 in further detail. In the illustrated example, control node 12 includes a monitoring subsystem 202, a service level automation infrastructure (SLAI) 204, and a business logic tier (BLT) 206.

[0109] Monitoring subsystem 202 provides real-time monitoring of the distributed computing system 10. In particular, monitoring subsystem 202 dynamically collects status data 203 from the hardware and software operating within distributed computing system 10, and feeds the status data in the form of monitor inputs 208 to SLAI 204.

Netflix, Inc. - Ex. 1002, Page 000289
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Monitoring inputs 208 may be viewed as representing the actual state of the fabric defined for the organizational model implemented by distributed computing system 10. Monitoring subsystem 202 may utilize well defined interfaces, e.g., the Simple Network Management Protocol (SNMP) and the Java Management Extensions (JMX), to collect and export real-time monitoring information to SLAI 204.

[0110] SLAI 204 may be viewed as an automation subsystem that provides support for autonomic computing and acts as a central nervous system for the controlled fabric. In general, SLAI 204 receives monitoring inputs 208 from monitoring subsystem 202, analyzes the inputs and outputs appropriate action requests 212 to BLT 206. In one embodiment, SLAI 204 is a cybernetic system that controls the defined fabric via feedback loops. More specifically, administrator 20 may interact with BLT 206 to define an expected state 210 for the fabric. BLT 206 communicates expected state 210 to SLAI 204. SLAI 204 receives the monitoring inputs from monitoring subsystem 202 and applies rules to determine the most effective way of reducing the differences between the expected and actual states for the fabric.

[0111] For example, SLAI 204 may apply a rule to determine that a node within a high priority tier has failed and that the node should be replaced by harvesting a node from a lower priority tier. In this example, SLAI 204 outputs an action request 212 to invoke BLT 206 to move a node from one tier to the other.

[0112] In general, BLT 206 implements high-level business operations on fabrics, domains and tiers. SLAI 204 invokes BLT 206 to bring the actual state of the fabric into accordance with the expected state. In particular, BLT 206 outputs fabric actions 207 to perform the physical fabric changes. In addition, BLT 206 outputs an initial expected state 210 to SLAI 204 and initial monitoring information 214 to SLAI 204 and monitoring subsystem 202, respectively. In addition, BLT 206 outputs notifications 211 to SLAI 204 and monitoring subsystem 202 to indicate the state and monitoring changes to distributed computing system 10. As one example, BLT 206 may provide control operations that can be used to replace failed nodes. For example, BLT 206 may output an action request indicating that a node having address 10.10.10.10 has been removed from tier ABC and a node having address 10.10.10.11 has been added to tier XYZ. In response, monitoring subsystem 202 stops attempting to collect status data 203 from node 10.10.10.10 and starts monitoring for status data from node 10.10.10.11. In addition, SLAI 204 updates an internal model to automatically associate monitoring inputs from node 10.10.10.11 with tier XYZ.

Netflix, Inc. - Ex. 1002, Page 000290
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0113] FIG. 16 is a block diagram illustrating one embodiment of monitoring subsystem 202. In general, monitoring subsystem 202 dynamically detects and monitors a variety of hardware and software components within the fabric. For example, monitoring subsystem 202 identifies, in a timely and efficient manner, any computing nodes that have failed, i.e., any node that does not respond to a request to a known service. More generally, monitoring subsystem 202 provides a concise, consistent and constantly updating view of the components of the fabric.

[0114] As described further below, monitoring subsystem 202 employs a modular architecture that allows new detection and monitoring collectors 224 to be "plugged-in" for existing and new protocols and for existing and new hardware and software. As illustrated in FIG. 16, monitoring subsystem 202 provides a plug-in architecture that allows different information collectors 224 to be installed. In general, collectors 224 are responsible for protocol-specific collection of monitoring information. The plug-in architecture allows for new protocols to be added by simply adhering to a collector plug-in signature. In this example, monitoring subsystem 202 includes collectors 224A and 224B for collecting information from operating systems and applications executing on nodes within tier A and tier B, respectively.

[0115] In one embodiment, collectors 224 are loaded at startup of control node 12 and are configured with information retrieved from BLT 206. Monitoring engine 222 receives collection requests from SLAI 204, sorts and prioritizes the requests, and invokes the appropriate one of collectors 224 based on the protocol specified in the collection requests. The invoked collector is responsible for collecting the required status data and returning the status data to monitoring engine 222. If the collector is unable to collect the requested status data, the collector returns an error code.

[0116] In one embodiment, collectors 224 are Java code compiled into a jar file and loaded with a class loader at run time. Each of collectors 224 has an associated configuration file written in a data description language, such as the extensible markup language (XML). In addition, a user may interact with BLT 206 to add run-time configuration to dynamically configure collectors 224 for specific computing environments. Each of collectors 224 expose an application programming interface (API) to monitoring engine 222 for communication and data exchange.

[0117] A user, such as a system administrator, specifies the protocol or protocols to be used for monitoring a software image when the image is created. In addition, the users may specify the protocols to be used for monitoring the nodes and each service executing

Netflix, Inc. - Ex. 1002, Page 000291
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

on the nodes. Example protocols supported by the collectors 224 include Secure Shell (SSH), Simple Network Management Protocol (SNMP), Internet Control Message Protocol (ICMP) ping, Java Management Extensions (JMX) and the Hypertext Transfer Protocol (HTTP).

[0118] Some protocols require special privileges, e.g., root privileges, to perform the required data collection. In this case, the corresponding collectors 224 communicate with a separate process that executes as the root. Moreover, some protocols may require deployment and/or configuration of data providers within the fabric. Software agents may, for example, be installed and configured on nodes and configured on other hardware. If needed, custom in-fabric components may be deployed.

[0119] In this example, the modular architecture of monitoring subsystem 202 also supports one or more plug-in interfaces 220 for data collection from a wide range of third-party monitoring systems 228. Third-party monitoring systems 228 monitor portions of the fabric and may be vendor-specific.

[0120] FIG. 17 is a block diagram illustrating one embodiment of SLAI 204 in further detail. In the illustrated embodiment, SLAI 204 is composed of three subsystems: a sensor subsystem 240, an analysis subsystem 244 and an effector subsystem 248.

[0121] In general, sensor subsystem 240 receives actual state data from monitoring subsystem 202 in the form of monitoring inputs 208 and supplies ongoing, dynamic input data to analysis subsystem 244. For example, sensor subsystem 240 is notified of physical changes to distributed computing system 10 by monitoring subsystem 202. Sensor subsystem 240 uses the state data received from monitoring subsystem 202 to maintain ongoing, calculated values that can be sent to analysis subsystem 244 in accordance with scheduler 242.

[0122] In one embodiment, sensor subsystem 240 performs time-based hierarchical data aggregation of the actual state data in accordance with the defined organization model. Sensor subsystem 240 maintains organizational data in a tree-like structure that reflects the current configuration of the hierarchical organization model. Sensor subsystem 240 uses the organizational data to perform the real-time data aggregation and map tiers and domains to specific nodes. Sensor subsystem 240 maintains the organizational data based on notifications 211 received from BLT 206.

[0123] Sensor subsystem 240 sends inputs to analysis subsystem 244 to communicate the aggregated data on a periodic or event-driven basis. Analysis subsystem 244 may register an interest in a particular aggregated data value with sensor subsystem 240 and request

Netflix, Inc. - Ex. 1002, Page 000292
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

updates at a specified frequency. In response, sensor subsystem 240 interacts with monitoring subsystem 202 and scheduler 242 to generate the aggregated data required by analysis subsystem 244.

[0124] Sensor subsystem 240 performs arbitrary data aggregations via instances of plug-in classes (referred to as "triggers") that define the aggregations. Each trigger is registered under a compound name based on the entity being monitored and the type of data being gathered. For example, a trigger may be defined to aggregate and compute an average computing load for a tier every five minutes. Analysis subsystem 244 requests the aggregated data based on the registered names. In some embodiments, analysis subsystem 244 may define calculations directly and pass them to sensor subsystem 240 dynamically.

[0125] Analysis subsystem 244 is composed of a plurality of forward chaining rule engines 246A-246N. In general, rule engines 246 match patterns in a combination of configuration data and monitoring data, which is presented by extraction agent 251 in the form of events. Events contain the aggregated data values that are sent to rule engines 246 in accordance with scheduler 242.

[0126] Sensor subsystem 240 may interact with analysis subsystem 244 via trigger listeners 247 that receives updates from a trigger within sensor subsystem 240 when specified events occur. An event may be based on system state (e.g., a node transitioning to an up or down state) or may be time based.

[0127] Analysis subsystem 244 allows rule sets to be loaded in source form and compiled at load time into discrimination networks. Each rule set specifies trigger-delivered attributes. Upon loading the rule sets, analysis subsystem 244 establishes trigger listeners 247 to receive sensor notifications and update respective working memories of rule engines 246. As illustrated in FIG. 17, each of rule engines 246 may serve a different tier defined within the fabric. Alternatively, multiple rule engines 246 may serve a single tier or a single rule engine may serve multiple tiers.

[0128] Rule engines 246 process the events and invoke action requests via calls to effector subsystem 248. In addition, rule engines 246 provide a call-back interface so that effector subsystem 248 can inform a rule engine when an action has completed. Rule engines 246 prevent a particular rule from re-firing as long as any action invoked by the rule has not finished. In general, rules contain notification calls and service invocations though either may be disabled by configuration of effector subsystem 248. BLT 206 supplies initial system configuration descriptions to seed each of rule engines 246.

Netflix, Inc. - Ex. 1002, Page 000293
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0129] In general, rule engines 246 analyze the events and discover discrepancies between an expected state of the fabric and an actual state. Each of rule engines 246 may be viewed as software that performs logical reasoning using knowledge encoded in high-level condition-action rules. Each of rule engines 246 applies automated reasoning that works forward from preconditions to goals defined by system administrator 20. For example, rule engines 246 may apply modus ponens inferences rules.

[0130] Rule engines 246 output requests to effector subsystem 248 which produce actions requests 212 for BLT 206 to resolve the discrepancies. Effector subsystem 248 performs all operations on behalf of analysis subsystem 244. For example, event generator 250, task invocation module 252 and logger 254 of effector subsystem 248 perform event generation, BLT action invocation and rule logging, respectively. More specifically, task invocation module 252 invokes asynchronous operations within BLT 206. In response, BLT 206 creates a new thread of control for each task which is tracked by a unique task identifier (task id). Rules engine 246 uses the task id to determine when a task completes and, if needed, to re-fire any rules that were pended until completion of the task. These tasks may take arbitrary amounts of time, and rules engine 246 tracks the progress of individual task via change notifications 211 produced by BLT 206.

[0131] Event generator 250 creates persistent event records of the state of processing of SLAI 204 and stores the event records within a database. Clients uses these event records to track progress and determine the current state of the SLAI 204.

[0132] Logger 254 generates detailed trace information about system activities for use in rule development and debugging. The logging level can be raised or lowered as needed without changing operation of SLAI 204.

[0133] FIG. 18 is a block diagram of an example working memory 270 associated with rule engines 246. In this example, working memory 270 includes a read-only first data region 272 that stores the expected state received from BLT 206. Data region 272 is read-only in the sense that it cannot be modified in response to a trigger from sensor subsystem 240 or by rule engines 246 without notification from BLT 206.

[0134] In addition, working memory 270 includes a second data region 274 that is modifiable (i.e., read/write) and may be updated by monitoring subsystem 202 or used internally by rule engines 246. In general, data region 274 stores aggregated data representing the actual state of the fabric and can be updated by sensor subsystem 240 or by rule engines 246. The actual state may consist of a set of property annotations that can

Netflix, Inc. - Ex. 1002, Page 000294
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

be attached to objects received from BLT 206 or to objects locally defined within a rule engine, such as local object 276.

[0135] FIG. 19 is a block diagram illustrating an example embodiment for BLT 206. In this example, BLT 206 includes a set of one or more web service definition language (WSDL) interfaces 300, a report generator 302, a fabric administration interface service 304, a fabric view service 306, a user administration service 308, a task interface 311, a task manager 312 and an event subsystem 315.

[0136] As described, BLT 206 provides the facilities necessary to create and administer the organizational model (e.g., fabric, domains, tiers and nodes) implemented by distributed computing system 10. In general, BLT 206 abstracts access to the persisted configuration state of the fabric, and controls the interactions with interfaces to fabric hardware services. As such, BLT 206 provides fabric management capabilities, such as the ability to create a tier and replace a failed node. WSDL interfaces 300 provide web service interfaces to the functionality of BLT 206 that may be invoked by web service clients 313. Many of WSDL interfaces 300 offered by BLT 206 allow administrator 20 to define goals, such as specifying a goal of the expected state of the fabric. As further described below, rule engines 246 within SLAI 204, in turn, invoke task manger 312 to initiate one or more BLT tasks to achieve the specified goal. In general, web service clients 313 may be presentation layer applications, command line applications, or other clients.

[0137] BLT 206 abstracts all interaction with physical hardware for web service clients 313. BLT 206 is an enabling component for autonomic management behavior, but does not respond to real-time events that either prevent a goal from being achieved or produce a set of deviations between the expected state and the actual state of the system. In contrast, BLT 206 originates goals for autonomic reactions to changing configuration and state. SLAI 204 analyzes and acts upon these goals along with real-time state changes. BLT 206 sets the goals to which SLAI 204 strives to achieve, and provides functionality used by the SLAI in order to achieve the goals.

[0138] In general, BLT 206 does not dictate the steps taken in pursuit of a goal since these are likely to change based on the current state of distributed computing system 10 and changes to configurable policy. SLAI 204 makes these decisions based on the configured rule sets for the fabric and by evaluating monitoring data received from monitoring subsystem 202.

Netflix, Inc. - Ex. 1002, Page 000295
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0139] Fabric administration service 304 implements a set of methods for managing all aspects of the fabric. Example methods include methods for adding, viewing, updating and removing domains, tiers, nodes, notifications, assets, applications, software images, connectors, and monitors. Other example methods include controlling power at a node, and cloning, capturing, importing, exporting or upgrading software images. Rule engines 246 of SLAI 204 may, for example, invoke these methods by issuing action requests 212.

[0140] Task manager 312 receives action requests 212 via task interface 311. In general, task interface 311 provides an interface for receiving action requests 212 from SLAI 204 or other internal subsystem. In response, task manager 312 manages asynchronous and long running actions that are invoked by SLAI 204 to satisfy a goal or perform an action requested by a client.

[0141] Task manager 312 generates task data 310 that represents identification and status for each task. Task manager 312 returns a task identifier to the calling web service clients 313 or the internal subsystem, e.g., SLAI 204, that initiated the task. Rule engines 246 and web service clients 313 use the task identifiers to track progress and retrieve output, results, and errors associated with achieving the goal.

[0142] In one embodiment, there are no WSDL interfaces 300 for initiating specific tasks. Rather, administrator 20 interacts with BLT 206 though goal interfaces presented by WSDL interfaces 300 to define the goals for the fabric. In contrast, the term task is used to refer to internal system constructs that require no user interaction. Tasks are distinct, low-level units of work that affect the state of the fabric. SLAI 204 may combine tasks to achieve or maintain a goal state.

[0143] For example, administrator 20 can request configuration changes by either adding new goals to an object or by modifying the attributes on existing goals. Scheduled goals apply a configuration at a designated time. For example, the goals for a particular tier may specify the minimum, maximum, and target node counts for that tier. As a result, the tier can increase or decrease current node capacity by scheduling goals with different configuration values.

[0144] This may be useful, for example, in scheduling a software image upgrade. As another example, entire domains may transition online and offline per a defined grid schedule. Administrator 20 may mix and match goals on a component to achieve configurations specific to the application and environment. For example, a tier that does not support autonomic node replacement would not be configured with a harvesting goal.

Netflix, Inc. - Ex. 1002, Page 000296
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0145] In some embodiments, goals are either "in force" or "out of force." SLAI 204 only works to achieve and maintain those goals that are currently in force. SLAI 204 may applies a concept of "gravity" as the goals transition from in force to out of force. For example, SLAI 204 may transition a tier offline when an online goal is marked out of force. Some goal types may have prerequisite goals. For example, an image upgrade goal may require as a prerequisite that a tier be transitioned to offline before the image upgrade can be performed. In other embodiments, goals are always in force until modified.

[0146] SLAI 204 may automatically formulate dependencies between goals or may allow a user to specify the dependencies. For example, a user may request that a newly created tier come online. As a result of this goal, SLAI 204 may automatically direct task manager 312 to generate a task of harvesting a target number of nodes to enable the tier. Generally, all goals remain in-force by SLAI 204 until modified by BLT 206. In one embodiment, each goal remains in-force in one of three states: Satisfied, Warning, or Critical depending on how successful SLAI 204 was in achieving the goal at the time the event record was generated and stored.

[0147] In this manner, SLAI 204 controls the life cycle of a goal (i.e., the creation, scheduling, update, deletion of the goal), and provides a common implementation of these and other services such as timeout, event writing, goal conflicts, management of intra-goal dependencies, and tracking tasks to achieving the goals.

[0148] Progress toward a goal is tracked though event subsystem 315. In particular, event subsystem 315 tracks the progress of each in force goal based on the goal identifiers. Tasks executed to achieve a particular goal produce events to communicate result or errors. The events provide a convenient time-based view of all actions and behaviors.

[0149] Examples of goal types that may be defined by administrator 20 include software image management goals, node allocation goals, harvest goals, tier capacity goals, asset requirement goals, tier online / offline goals, and data gathering goals.

[0150] In one embodiment, BLT 206 presents a task interface to SLAI 204 for the creation and management of specific tasks in order to achieve the currently in force goals. In particular, rule engines 246 invoke the task interface based on evaluation of the defined rule sets in view of the expected state and actual state for the fabric. Example task interfaces include interfaces to: reserve node resources; query resources for a node slot; associate or disassociate an image with a node in a tier node slot; allocate, de-allocate,

Netflix, Inc. - Ex. 1002, Page 000297
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

startup or shutdown a node; move a node to a tier; apply, remove or cycle power of a node; create a golden image; create or delete an image instance; and delete an activity, node or tier.

[0151] Report generator 302 provides an extensible mechanism for generating reports 314. Typical reports include image utilization reports that contain information with respect to the number of nodes running each software image, inventory reports detailing both the logical and physical aspects of the fabric, and system event reports showing all events that have occurred within the fabric. Report generator 302 gathers, localizes, formats and displays data into report form for presentation to the user. Report generator 302 may include one or more data gathering modules (not shown) that gather events in accordance with a schedule and update an events table to record the events. The data gathering modules may write the events in XML format.

[0152] FIG. 20 is a block diagram illustrating one embodiment of a rule engine 246 (FIG. 17). In the illustrated embodiment, rule engine 246 includes a rule compiler 344 and an execution engine 346. Each of rules 342 represents a unit of code that conforms to a rule language and expresses a set of triggering conditions and a set of implied actions. When the conditions are met, the actions are eligible to occur. The following is one example of a configuration rule:

```
rule checkTierLoad {
      Tier t where status != "overloaded";
      LoadParameter p where app == t.app && maxload < t.load;
} -> {
      modify t {
            status: "overloaded";
      };
}
```

When translated, this example rule marks a tier as overloaded if an application is implemented by the tier and the maximum specified load for the application has been exceeded. Another example rule for outputting a notification that a tier is overloaded and automatically invoking a task within BLT 206 to add a node is:

```
rule tierOverloadNotify {
      Tier t where status == "overloaded";
} -> {
      notify "Tier: " + t + "is overloaded.";
      BLT.addNode(f);
}
```


Netflix, Inc. - Ex. 1002, Page 000298
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0153] Rule compiler 344 compiles each of rules 344 and translates match conditions of the rules into a discrimination network that avoids redundant tests during rule execution. Execution engine 346 handles rule administration, object insertion and retrieval, rule invocation and execution of rule actions. In general, execution engine 346 first matches a current set of rules 342 against a current state of working memory 348 and local objects 350. Execution engine 346 then collects all rules that match as well as the matched objects and selects a particular rule instantiation to fire. Next, execution engine 346 fires (executes) the instantiated rule and propagates any changes to working memory 348. Execution engine 346 repeats the process until no more matching rule instantiations can be found.

[0154] Firing of a rule typically produces a very small number of changes to working memory 348. This allows sophisticated rule engines to scale by retaining match state between cycles. Only the rules and rule instantiations affected by changes are updated, thereby avoiding the bulk of the matching process. One exemplary algorithm that may be used by execution engine 346 to handle the matching process includes the RETE algorithm that creates a decision tree that combines the patterns in all the rules and is intended to improve the speed of forward-chained rule system by limiting the effort required to re-compute a conflict set after a rule is fired. One example of a RETE algorithm is described in Forgy, C.L.: 1982, 'RETE: a fast algorithm for the many pattern/many object pattern match problem', Artificial Intelligence 19, 1737. Other alternatives include the TREAT algorithms, and LEAPS algorithm, as described by Miranker, D.P.: 'TREAT: A New and Efficient Match Algorithm for AI Production Systems'. ISBN 0934613710 Daniel P. Miranker, David A. Brant, Bernie Lofaso, David Gadbois: On the Performance of Lazy Matching in Production Systems. AAAI 1990: 685692.

[0155] FIG. 21 is a block diagram illustrating an alternative embodiment of control unit 12 (FIG. 15). In this embodiment, control unit 12 operates substantially as described above, but includes an application matrix 350, an application governor 352, a configuration processor 354, and an application service level automation infrastructure ("application SLAI") 358. As described below, application matrix 350, application governor 352, configuration processor 354, and application SLAI 358 provide a framework that allows control unit 12 to autonomically control the deployment, execution and monitoring of applications across application nodes 14 of distributed computing system 10.

Netflix, Inc. - Ex. 1002, Page 000299
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0156] Application matrix 350 contains all the information needed by control unit 12 to interact with one or more applications or application servers and provide autonomic control over a set of applications. Specifically, application matrix 350 provides a logical definition for deploying and controlling the set of applications to one or more tiers within distributed computing system 10. In one embodiment, application matrix 350 is an electronic document that conforms to a data description language, e.g., the extensible markup language (XML). Application SLAI 358 includes an application rules engine 355 dedicated to processing application-level rules, i.e., forward-chaining rules to provide autonomic control over the applications defined within application matrix 350. Like rules engine 246, application rules engine 355 contains a rule compiler, an execution engine, and a working memory. In order to give effect to changes in application matrix 350, application SLAI 358 automatically updates application rules engine 355 and monitoring subsystem 202. In particular, application matrix 350 sends an alert whenever application matrix 350 changes. In response to this alert, application SLAI 358 captures application-specific attributes from application matrix 350. Specifically, application SLAI 358 captures configuration attributes and rule attributes contained in application matrix 350. Application SLAI 358 transfers any new rule attributes to the working memory of application rules engine 355 to provide autonomic control over the deployment, monitoring and the execution of the applications defined within application matrix 350. In addition, application SLAI 358 updates monitoring subsystem 202 to collect information required to control the applications In this manner, administrator 20 may continue to add new application definitions and configurations to application matrix 350 after distributed control node 12 has started.

[0157] As described in further detail below, configuration processor 354 is a software module that generates an application matrix entry based on an application definition and application configuration properties of a "staged" application. A staged application is an application that has been deployed in a staging environment and customized for subsequent deployment within distributed computing system 10. After creating the application matrix entry, administrator 20 may insert the application matrix entry into application matrix 350.

[0158] Configuration processor 354 is "pluggable." That is, administrator 20 can "plug in" different implementations of configuration processor 354 as needed. For example, administrator 20 may need to "plug in" a different implementation of configuration processor 354 to handle applications that do not use an application server.

Netflix, Inc. - Ex. 1002, Page 000300
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0159] Application governor 352 is a software engine that performs application-level actions based on requests received from application rules engine 355. In this manner, BLT 206 effects fabric-level actions (e.g., deployment and monitoring of nodes and images) based on request from fabric-level rules engines 246 (FIG. 17), while matrix governor 352 performs application-level actions (e.g., deployment and monitoring of applications) based on requests from application rules engine 355.

[0160] Application governor 352 uses application matrix 350 as a source of parameters when carrying out the application-level operations requested by application rules engine 355. For example, application rules engine 355 may detect that a node to which the application is not deployed is ready for use by the application. As a result, application rules engine 355 directs application governor 352 to handle the details of deploying the application to the node. In turn, application governor 352 accesses application matrix 350 to retrieve application-specific parameters necessary to deploy the application. Storing application-specific parameters in application matrix 350 allows the application-specific parameters to change without having to recompile the rules within working memory of application rules engine 355.

[0161] Application governor 352 performs a similar procedure to undeploy an application. That is, application rules engine 355 may detect that a second application needs to use a node more than a first application that is currently deployed to the node. In this situation, application rules engine 355 sends an instruction to application governor 352 to undeploy the first application and deploy the second application. To carry out this instruction, application governor 352 accesses application matrix 350 to discover configuration parameters of both applications. Application governor 352 then uses the discovered configuration parameters to communicate with the applications.

[0162] Like configuration processor 354, application governor 352 is also "pluggable." That is, administrator 20 can easily install or remove implementations of application governor 352 depending on the circumstances. Because application governor 352, and configuration processor 354 represent interchangeable, plug-in modules, the other parts of control unit 12 and system 10, including application rules engine 355, can remain generic and application neutral while providing autonomic control over distributed computing system 10.

[0163] FIG. 22 provides a conceptual view of an exemplary application matrix 350. Although application matrix 350 is typically represented in an encoded, electronic

Netflix, Inc. - Ex. 1002, Page 000301
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

document (e.g., an XML document), FIG. 22 provides a conceptual view for ease of illustration.

[0164] In this example, application matrix 350 contains seven columns and two rows. Each row represents a different application entry for deployment within distributed computing system 10. In FIG. 22, only the first entry is shown in detail.

[0165] Each of the columns represents a different category of elements that make up an application's logical definition. In this example, the columns include:

(1) an application column 360 that includes elements generally related to the deployment of the application,

(2) an application nodes column 362 that contains elements related to the tier node slots to which the application may be assigned,

(3) a services column 364 that contains elements related to the executable services launched when the application is deployed,

(4) a node monitor values column 366 that contains elements related to attributes of the nodes that are to be monitored after the application is deployed to that node,

(5) a service monitored attributes column 368 that contains elements related to attributes of the services that are to be monitored after the application is deployed,

(6) a service levels column 370 that contains elements related to attributes for use when constructing rules to monitor execution of the services, and

(7) a deployment constraints column 372 that contains elements related to attributes for use when constructing rules to control deployment of the application. Different types of applications may have different elements in each column, and different numbers of columns.

[0166] In the example of FIG. 22, application matrix 350 contains two applications 360 "DataDomain" 374 and "PortalDomain" 376. DataDomain application 374 has eleven attributes that define how control node 12 accesses and launches the application. For instance, the "adminIP" and "adminPort" attributes instruct governor 352 as to which the server and port hosts the administrative part of the logically defined application. Other attributes like "maxNodes" and "minNodes" instruct application rules engine 355 to run the application no less than minNodes and no more than maxNodes. Applications other than application 374 may have different numbers or types of attributes. In XML format, the attributes of application 374 may appear as follows:

Netflix, Inc. - Ex. 1002, Page 000302
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
        <WebServerDomain
        name="DataDomain"
        adminIP="172.31.64.201"
        adminPort="1100"
        adminTier="Web Admin"
        clusterName="PathCluster"
        expectedStartupDelay="120"
        loadDelay="120"
        maxNodes="2"
        minNodes="1"
        nodeManagerPort="5811"
        nodeTier="Web App"
>
```

[0167] In addition to these attributes, application 374 contains a series of elements (columns 362 – 372). In general, application nodes 362 contain a list of all of the available tier-node slots to which control node 12 may deploy application 374. In this example, two tier-node slots are specified. In XML, managed servers 362 appears as:

```
<ManagedServers IP="172.31.64.201" name="managedServer_0"
state="STOPPED" />
<ManagedServers IP="172.31.64.201" name="managedServer_1"
state="STOPPED" />
```

[0168] Each services 364 element identifies a service that control node 12 launches when deploying application 374 on a given application node. A service element comprises a service name and a path to a file containing the executable service. Governor 352 uses the path to locate and launch the service. For example, the following XML code indicates that governor 352 must access the file at "/lib/worklistApp/worklistApp.ear".

```
<services name="Worklist User Interface"
path="/lib/worklistApp/worklistApp.ear" />
```

[0169] Node Monitored Values 366 elements represent characteristics for use in constructing rules for monitoring nodes to which the applications are deployed. Similarly, Service Monitored Values 368 elements represent characteristics for use in constructing rules for monitoring services that are launched once the application is deployed. In this example, a "nodeMonitoredValues" element defines characteristics of a particular node that are to be monitored. For instance, the amount of free memory in a node is one example of a characteristic listed as a "nodeMonitoredValues" element. On the other hand, a "serviceMonitoredValues" element is specific attribute of a service that is to be monitored. For example, the number of pending operating system-level requests for the service, the number of idle threads, etc., could be service monitored values. In a XML rendition of application matrix 350, node monitored values and service monitored values could appear as follows:

Netflix, Inc. - Ex. 1002, Page 000303
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
<nodeMonitoredValues name="Load5Average" />
<nodeMonitoredValues name="PercentMemoryFree" />
<serviceMonitoredValues name="PendingRequests" />
<serviceMonitoredValues name="ExecuteThreadIdleCount" />
<serviceMonitoredValues name="ExecuteThreadTotalCount" />
```

[0170] Deployment constraint elements 372 specify characteristics of a node under which application rules engine 355 should (or should not) deploy a service to the node. In this example, a deployment constraint element has five attributes: "attribute", "expression", "frequency", "maxThreshold", "minThreshold", and "period." The "attribute" attribute names the deployment constraint. The "expression" attribute specifies an arithmetic expression manipulating a monitored value. For example, the expression could be "PercentMemoryFree * 100", meaning monitor the value of the "PercentMemoryFree" node monitored value multiplied by 100. The "expression" attribute may specify one or more node monitored values. The "frequency" attribute informs application rules engine 355 how frequently to check the monitored value. The "maxThreshold" attribute tells application rules engine 355 to invoke a rule when the value of the expression exceeds the value specified by the "maxThreshold" attribute. Similarly, the "minThreshold" attribute tells application rules engine 355 to invoke the rule when the value of the expression drops below the value specified by the "minThreshold" attribute. Finally, the "period" attribute informs application rules engine 355 of the period over which to collect monitored value. For example, a deployment constraint element may specify that application rules engine 355 should monitor the PercentMemoryFree attribute of a node every 15 seconds for 60 seconds. If the value of PercentMemoryFree * 100 should drop below 1.0 (i.e. 1% of memory free) for 60 seconds, then application rules engine 355 should not deploy the application to that node. In XML, this rule would be represented as:

```
<deploymentConstraints attribute="FreeMemory"
expression="PercentMemoryFree*100" frequency="15" maxThreshold="-
1.0" minThreshold="1.0" period="60" />
```

[0171] Service level elements 370 have the same five attributes as deployment constraints elements: "attribute", "expression", "frequency", "maxThreshold", "minThreshold", and "period". However, the "expression" attribute deals with service monitored attributes rather than node monitored attributes. For example, a service level element may specify that application rules engine 355 check the "pendingRequest" service-monitored attribute every 15 seconds for 30 seconds. Then, if there are more than 20 pending requests for more than 30 seconds, application rules engine 355 should take the action of starting a

Netflix, Inc. - Ex. 1002, Page 000304
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

new application. On the other hand, if there are fewer than 5 pending requests for 30 seconds, application rules engine 355 enables the action to remove the application to free up space on a node. Such a service level element could be represented as:

```
<serviceLevels attribute="PendingRequests"
expression="PendingRequests" frequency="15" maxThreshold="20.0"
minThreshold="5.0" period="30" />
```

[0172] Put together, an XML representation of an application matrix logically defining a single application for deployment within autonomically controlled distributed computing system 10 may appear as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<appMatrixRegister>
<WebServerDomain adminIP="172.31.64.201" adminPort="1100" adminTier="Web
Admin" clusterName="PathCluster" expectedStartupDelay="120"
loadDelay="120" maxNodes="2" minNodes="1" name="DataDomain"
nodeManagerPort="5811" nodeTier="Web App">
    <ManagedServers IP="172.31.64.201" name="managedServer_0"
    state="STOPPED" />

    <ManagedServers IP="172.31.64.201" name="managedServer_1"
    state="STOPPED" />
    <applications name="AI Design-time" path="/lib/ai-designtime.ear" />

    <applications name="System EJBs" path="/lib/ejbs.ear" />

    <applications name="Worklist Worker User Interface"
    path="/lib/worklist/worklist.ear" />

    <applications name="DBMS_ADK" path="/lib/DBMS_ADK.ear" />

    <applications name="UserApp"
    path="/user_projects/domains/userApp.ear" />

    <nodeMonitoredValues name="Load5Average" />

    <nodeMonitoredValues name="PercentMemoryFree" />

    <serviceMonitoredValues name="PendingRequests" />

    <serviceMonitoredValues name="ExecuteThreadIdleCount" />
```


Netflix, Inc. - Ex. 1002, Page 000305
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
    <serviceMonitoredValues name="ExecuteThreadTotalCount" />

    <deploymentConstraints attribute="LoadAverage"
    expression="Load5Average" frequency="15" maxThreshold="4.0"
    minThreshold="-1.0" period="15" />

    <deploymentConstraints attribute="FreeMemory"
    expression="PercentMemoryFree*100" frequency="15" maxThreshold="-
    1.0" minThreshold="1.0" period="60" />

    <serviceLevels attribute="BusyThreadPercentage"
    expression="(ExecuteThreadTotalCount-
    ExecuteThreadIdleCount)*100/ExecuteThreadTotalCount" frequency="15"
    maxThreshold="20.0" minThreshold="5.0" period="30" />

    <serviceLevels attribute="PendingRequests"
    expression="PendingRequests" frequency="15" maxThreshold="20.0"
    minThreshold="5.0" period="30" />
</WebserverDomain>
</appMatrixRegister>
```

[0173] FIG. 23 is a flowchart illustrating an exemplary series of general steps that are performed to add an application to application matrix 350 in accordance with the principles of this invention. Administrator 20 first stages the application within a staging environment including deploying the application on a desired image, creating a domain, specifying any services and external system connectivity (e.g., connections to databases, queues) (380).

[0174] During this process, administrator 20 directs the application to generate an application definition file that represents an application-specific configuration based on the environment specified by the administrator. The application definition file contains information specific to an installation instance of the application. For example, the application definition file typically specifies the locations (paths) for all software components of the application as staged within the staging environment, a list of files to execute, connectivity information for the application, and other information that may be recorded by the configured application.

[0175] After staging the application, administrator 20 defines a set of application configuration properties that specify desired behavior of the application within distributed computing system 10 (382). Examples of such application configuration properties

Netflix, Inc. - Ex. 1002, Page 000306
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

include a minimum number of nodes, minimum resource requirements for a node, deployment timing information, network addresses of tier node slots that are able to execute the application and other properties.

[0176] Next, administrator 20 directs configuration processor 354 to generate an application entry for the application using the application definition and the application configuration properties (384). The application entry contains configuration attributes used by application governor 352 to interact with the application. In addition, the application entry contains rule attributes used by application rules engine 355 that define how control node 12 monitors the deployment and execution of the application within distributed computing environment 10.

[0177] After configuration processor 354 creates the application entry, administrator 20 may modify the application entry (386). In particular, administrator 20 may update start scripts or shell scripts for the application due to path changes between the staging environment and distributed computing system 10.

[0178] Once administrator 20 has finished modifying the application entry, administrator 20 inserts the application entry into application matrix 350 (388). Because application matrix 350 detects that it has changed, application matrix 350 sends an alert to application SLAI 358 (390).

[0179] In response to the alert, application SLAI 358 may update application rules engine 355 and monitoring subsystem 202 (392). In particular, application SLAI 358 automatically scans application matrix 350. If application SLAI 358 detects new rule attributes, application SLAI 358 creates local objects reflecting the new rule attributes in the working memory of application rules engine 355. In addition, if application SLAI 358 detects new monitored values, application SLAI 358 updates monitoring subsystem 202 to add new monitoring collectors 224 (FIG. 16).

[0180] After application SLAI 358 updates application rules engine 355 and monitoring subsystem 202, control node 12 has autonomic control over the deployment of applications to tiers based on the configuration of application matrix 350 (394). For example, control node 12 deploys images, deploys applications, monitors the state of nodes and the execution of the applications, and applies tier-level rules as well as application-specific rules. Application SLAI 358 continues to listen for alerts from application matrix 350.

[0181] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.

Netflix, Inc. - Ex. 1002, Page 000307
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS:

1. A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network; and

a control node that comprises:

a set of one or more applications to be executed on the application nodes;

an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the applications within the distributed computing system; and

an automation infrastructure having one or more rule engines that provide autonomic control of the application nodes and the applications in accordance with a set of one or more rules and the application matrix.

2. The distributed computing system of claim 1, further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the applications.

3. The distributed computing system of claim 1, wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of the application matrix.

4. The distributed computing system of claim 1, wherein the automation infrastructure automatically updates the rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

5. The distributed computing system of claim 1, further comprising:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes;

wherein the rule engines analyze the status data from the monitoring subsystem and apply a set of rules derived from the application matrix to control the applications.

Netflix, Inc. - Ex. 1002, Page 000308
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

6. The distributed computing system of claim 1, wherein the application matrix is described in a data description language.

7. The distributed computing system of claim 3, wherein the data description language is extensible markup language (XML).

8. The distributed computing system of claim 1, wherein the application matrix comprises a set of application descriptors that define a set of rules and parameters for each of the applications.

9. The distributed computing system of claim 1, wherein the application matrix includes deployment constraints that define criteria for use by the rules engine when selecting a set of the application nodes to which to deploy the applications.

10. The distributed computing system of claim 1, wherein the application matrix includes elements that set a desired minimum or maximum numbers of the application nodes onto which the control node may deploy each of the applications.

11. The distributed computing system of claim 1, wherein the rule engines are forward-chaining rule engines.

12. The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

Netflix, Inc. - Ex. 1002, Page 000309
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

13. The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

14. The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

15. The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model.

16. The distributed computing system of claim 15, wherein each tier represents a different computing function.

17. The distributed computing system of claim 1, wherein the rules are units of code that conform with a rule language.

18. The distributed computing system of claim 1, wherein the new applications may be added to the application matrix while the distributed computing system is operating.

Netflix, Inc. - Ex. 1002, Page 000310
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

19. The distributed computing system of claim 1,

wherein the application matrix is a pluggable module that is specific to the set of applications, and

wherein the distributed computing system includes a control node providing a framework to receive any of a plurality of pluggable application matrices that can be installed and removed from the control node to provide autonomic control over different groups of applications within the distributed computing system.

20. A method comprising:

generating an application matrix that specifies data for controlling the deployment of a set of applications within a distributed computing system; and

performing operations to provide autonomic control over the deployment, undeployment, and execution of the applications within the distributed computing system in accordance with parameters of the application matrix.

21. The method of claim 20, wherein performing operations comprises:

receiving status data for a distributed computing system having a plurality of application nodes interconnected via a communications network, wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system; and

performing the operations to provide autonomic control over the deployment of a set of applications within the distributed computing system in accordance with the rules.

22. The method of claim 20, further comprising accessing parameters of the application matrix and generating one or more rules for use in autonomically controlling the deployment of the applications.

23. The method of claim 22, further comprising processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

Netflix, Inc. - Ex. 1002, Page 000311
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

24. The method of claim 20, further comprising selecting the application matrix from a plurality of pluggable application matrices that can be installed and removed from a control node of the distributed computing system to provide autonomic control over different groups of applications.

25. The method of claim 20, further comprising capturing attributes from the application matrix to autonomically control the deployment of the applications based on the parameters of the application matrix.

26. The method of claim 25, wherein generating one or more attributes comprises capturing one or more of the attributes for use in monitoring the execution of the applications when deployed.

27. The method of claim 20, wherein generating an application matrix comprises generating the application matrix to conform to a data description language.

28. The method of claim 20, wherein the application matrix comprises a set of application descriptors that define a set of rules and parameters for each of the applications.

29. The method of claim 20, wherein the application matrix includes deployment constraints that define criteria for use by the rules engine when selecting a set of the application nodes to which to deploy the applications.

30. The method of claim 20, wherein the application matrix includes elements that set a desired minimum or maximum numbers of the application nodes onto which the control node may deploy each of the applications.

Netflix, Inc. - Ex. 1002, Page 000312
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

31. The method of claim 20,

wherein generating the application matrix comprises adding applications to the application matrix after the distributed computing system has begun operation; and

wherein performing operations to provide autonomic control over the deployment of the applications comprises performing operations to provide autonomic control over the deployment, undeployment, and execution of the application added to the application matrix after the distributed computing system has begun operation.

32. A computer-readable medium comprising instructions that cause a programmable processor to:

generate an application matrix that specifies data for controlling the deployment of a set of applications within a distributed computing system; and

perform operations to provide autonomic control over the deployment of the applications within the distributed computing system in accordance with parameters of the application matrix.

Netflix, Inc. - Ex. 1002, Page 000313
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

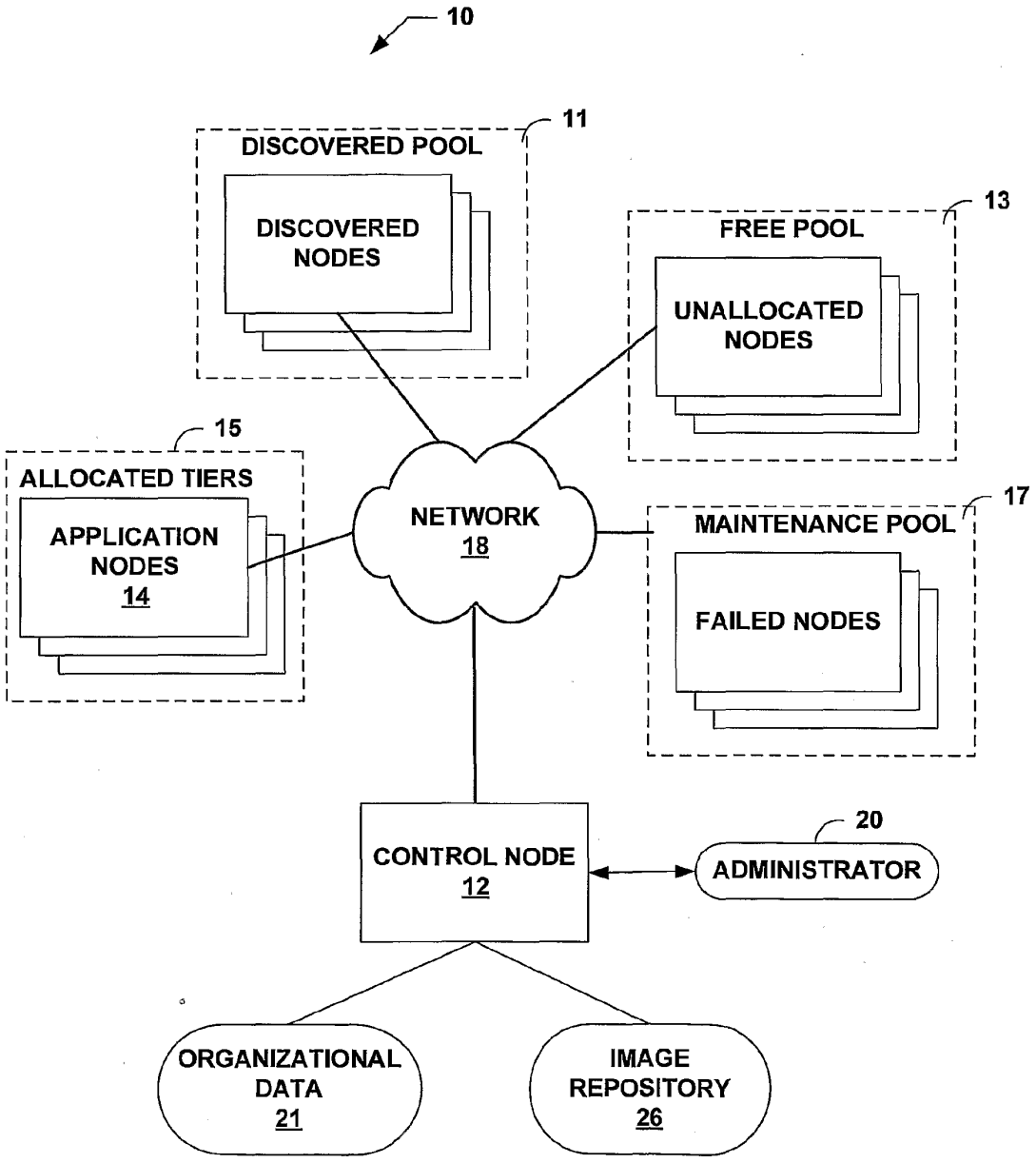

10
11
DISCOVERED POOL
DISCOVERED NODES
13
FREE POOL
UNALLOCATED NODES
15
ALLOCATED TIERS
APPLICATION NODES 14
NETWORK 18
17
MAINTENANCE POOL
FAILED NODES
CONTROL NODE 12
20
ADMINISTRATOR
ORGANIZATIONAL DATA 21
IMAGE REPOSITORY 26

FIG. 1

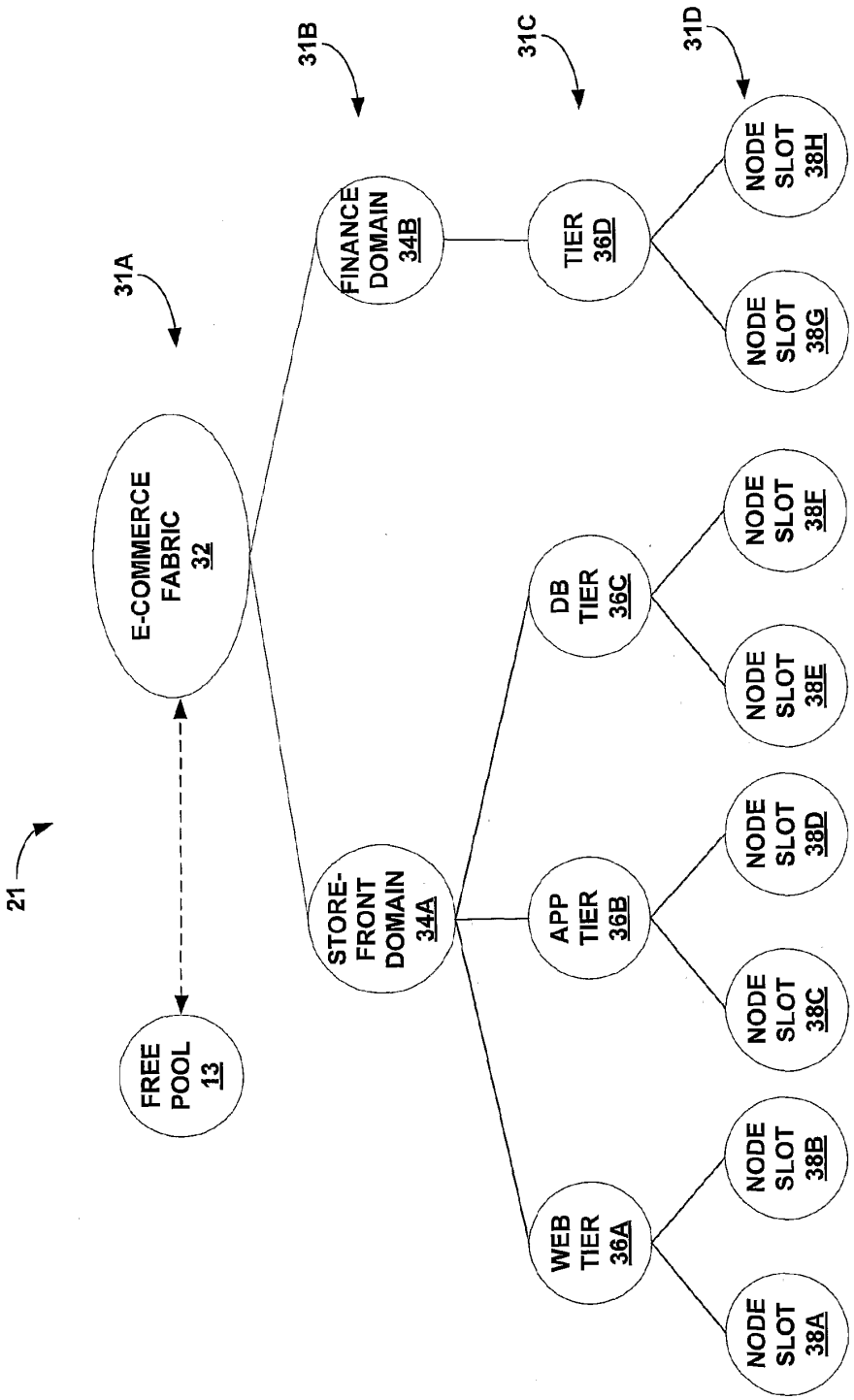

21
E-COMMERCE FABRIC 32
FREE POOL 13
31A
31B
31C
31D
FINANCE DOMAIN 34B
STORE-FRONT DOMAIN 34A
TIER 36D
DB TIER 36C
APP TIER 36B
WEB TIER 36A
NODE SLOT 38H
NODE SLOT 38G
NODE SLOT 38F
NODE SLOT 38E
NODE SLOT 38D
NODE SLOT 38C
NODE SLOT 38B
NODE SLOT 38A

FIG. 2

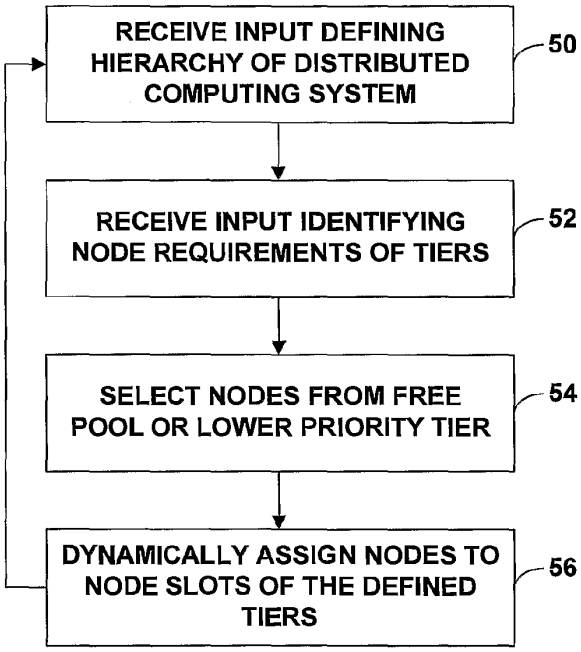

RECEIVE INPUT DEFINING HIERARCHY OF DISTRIBUTED COMPUTING SYSTEM
50
RECEIVE INPUT IDENTIFYING NODE REQUIREMENTS OF TIERS
52
SELECT NODES FROM FREE POOL OR LOWER PRIORITY TIER
54
DYNAMICALLY ASSIGN NODES TO NODE SLOTS OF THE DEFINED TIERS
56

FIG. 3

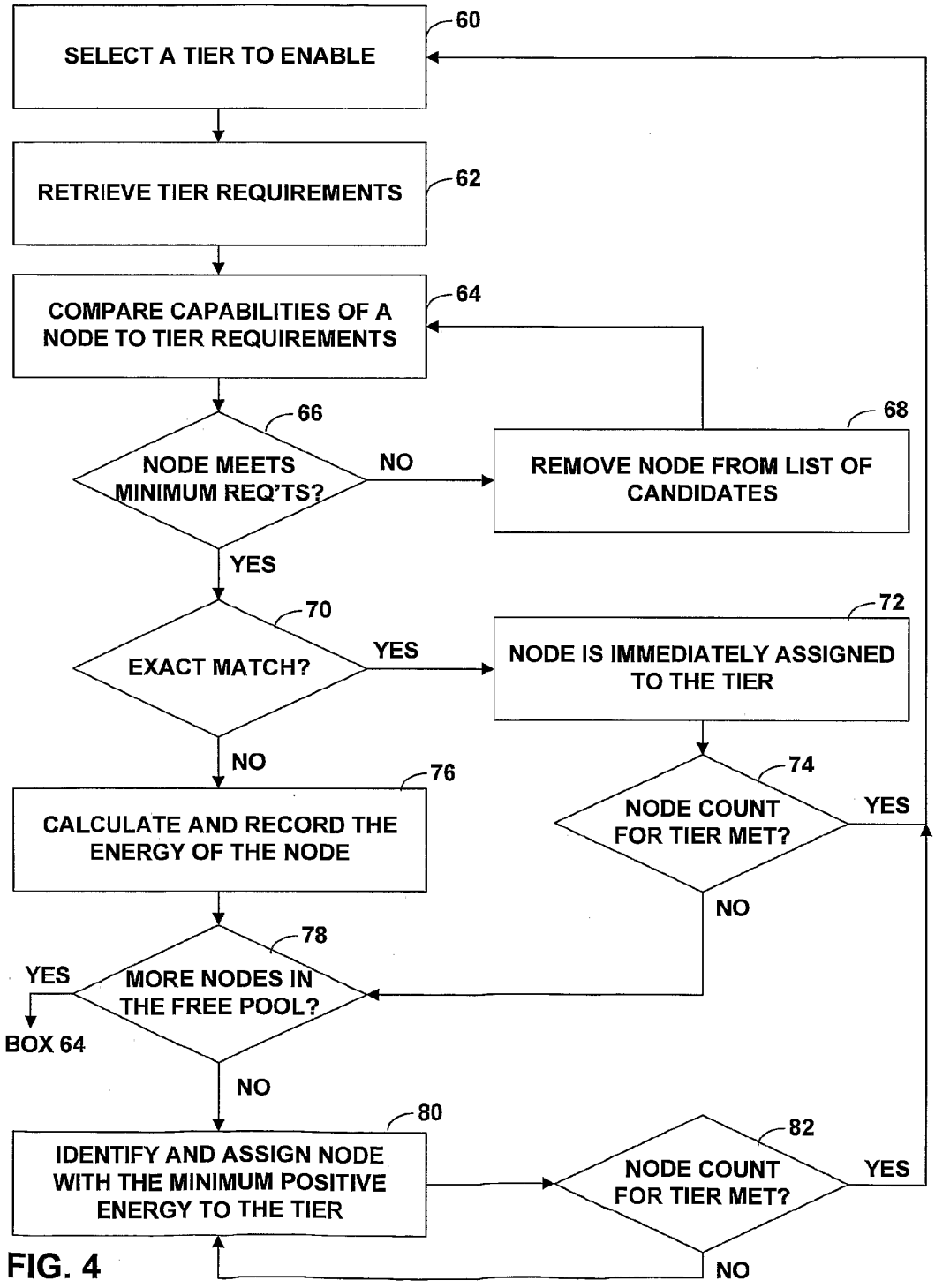

SELECT A TIER TO ENABLE
60
RETRIEVE TIER REQUIREMENTS
62
COMPARE CAPABILITIES OF A NODE TO TIER REQUIREMENTS
64
66
NODE MEETS MINIMUM REQ'TS?
NO
68
REMOVE NODE FROM LIST OF CANDIDATES
YES
70
EXACT MATCH?
YES
72
NODE IS IMMEDIATELY ASSIGNED TO THE TIER
NO
76
CALCULATE AND RECORD THE ENERGY OF THE NODE
74
NODE COUNT FOR TIER MET?
YES
NO
78
MORE NODES IN THE FREE POOL?
YES
BOX 64
NO
80
IDENTIFY AND ASSIGN NODE WITH THE MINIMUM POSITIVE ENERGY TO THE TIER
82
NODE COUNT FOR TIER MET?
YES
NO

FIG. 4

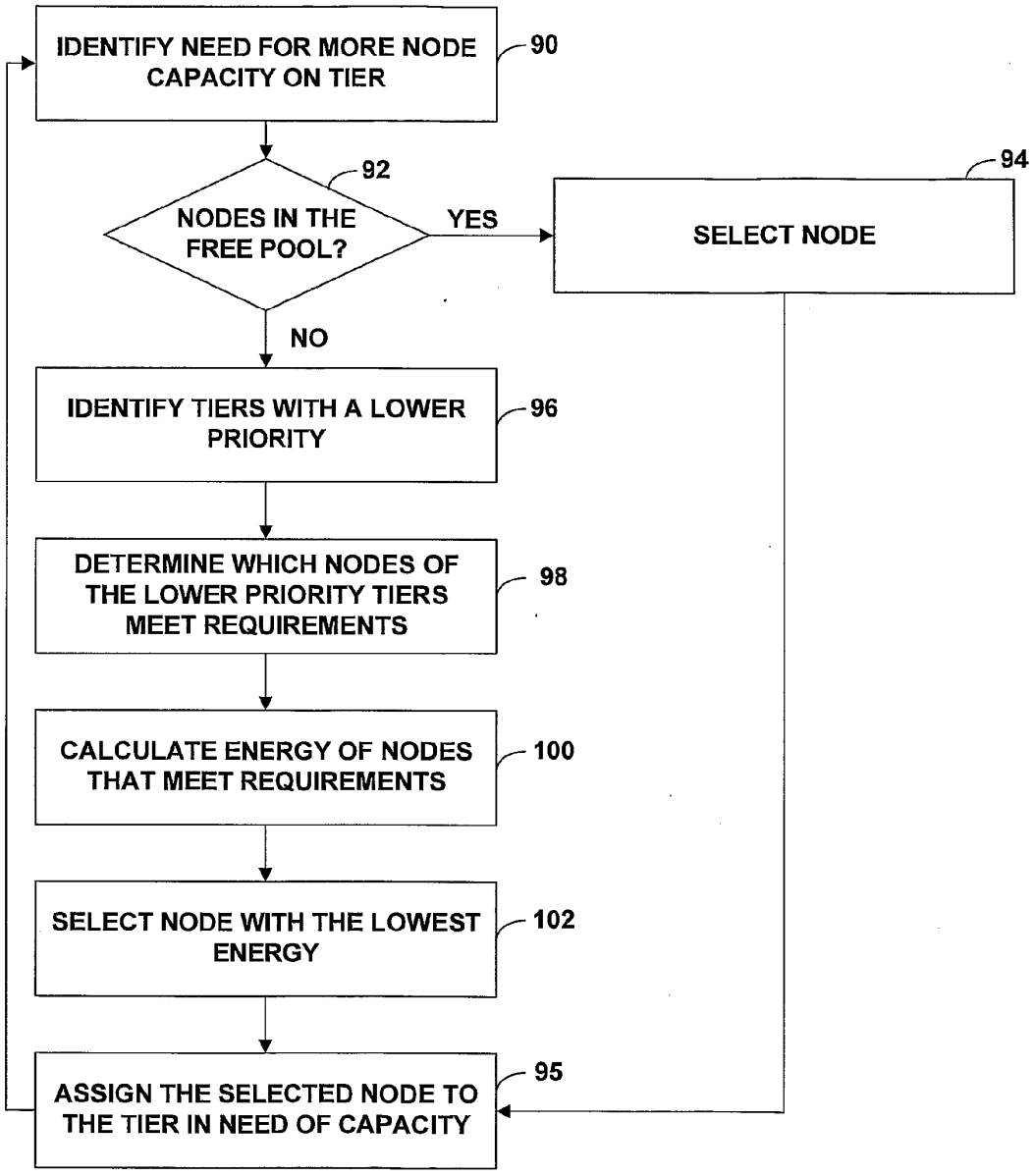

IDENTIFY NEED FOR MORE NODE CAPACITY ON TIER
90
92
NODES IN THE FREE POOL?
YES
94
SELECT NODE
NO
IDENTIFY TIERS WITH A LOWER PRIORITY
96
DETERMINE WHICH NODES OF THE LOWER PRIORITY TIERS MEET REQUIREMENTS
98
CALCULATE ENERGY OF NODES THAT MEET REQUIREMENTS
100
SELECT NODE WITH THE LOWEST ENERGY
102
ASSIGN THE SELECTED NODE TO THE TIER IN NEED OF CAPACITY
95

FIG. 5

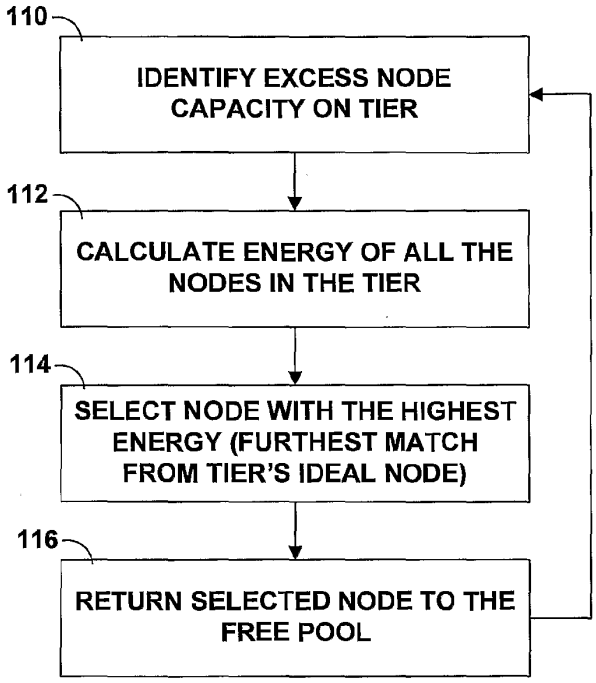

110
IDENTIFY EXCESS NODE CAPACITY ON TIER
112
CALCULATE ENERGY OF ALL THE NODES IN THE TIER
114
SELECT NODE WITH THE HIGHEST ENERGY (FURTHEST MATCH FROM TIER'S IDEAL NODE)
116
RETURN SELECTED NODE TO THE FREE POOL

FIG. 6

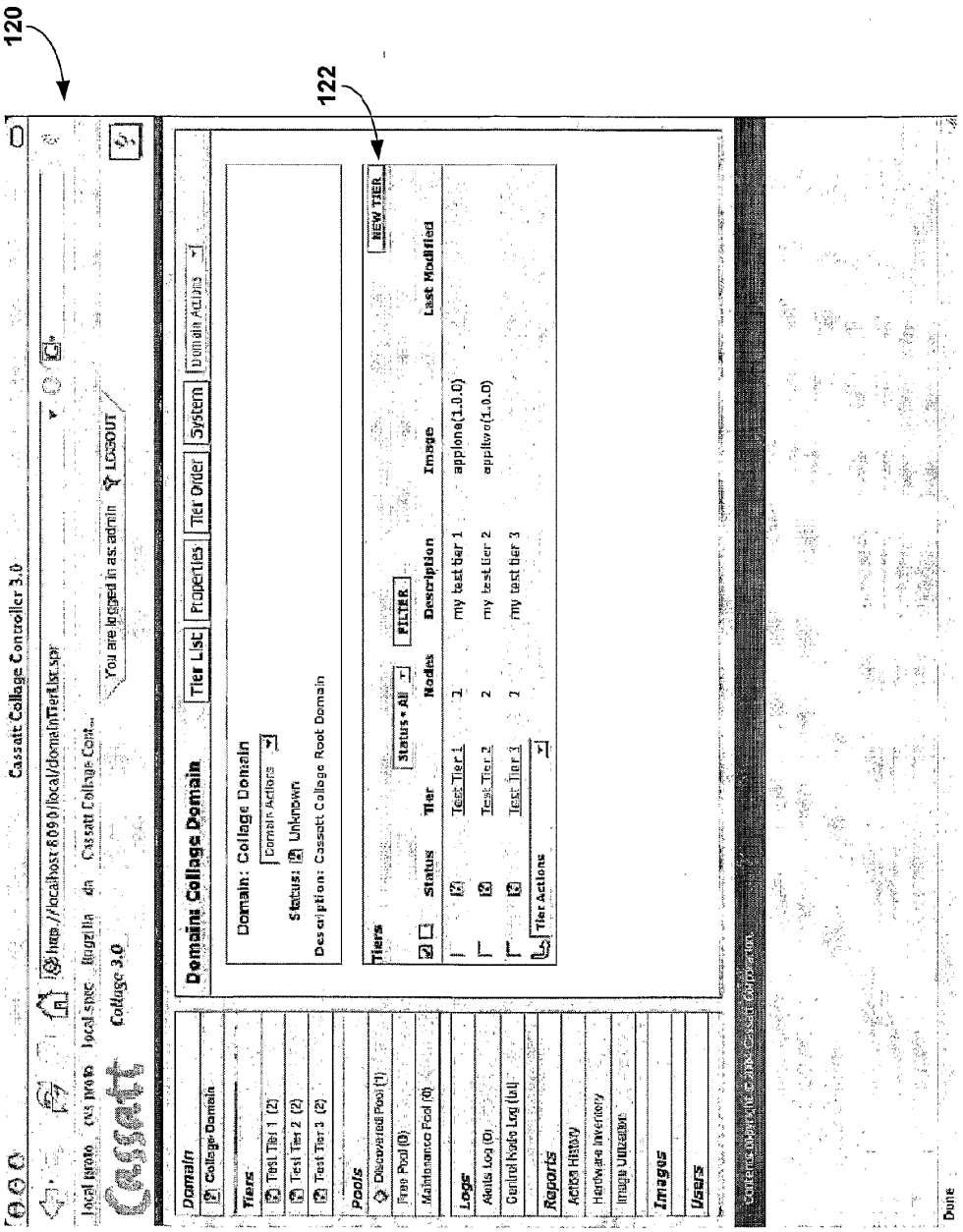

120
122
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Audit Log (0)
Control Node Log (full)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Domain: College Domain
Tier List
Properties
Tier Order
System
Domain Actions
Domain: College Domain
Domain Actions
Status: Unknown
Description: Cassatt College Root Domain
Tiers
Status: All
FILTER
NEW TIER
Status
Tier
Nodes
Description
Image
Last Modified
Test Tier 1
2
my test tier 1
appone(1.0.0)
Test Tier 2
2
my test tier 2
apptwo(1.0.0)
Test Tier 3
2
my test tier 3
Tier Actions
Done

FIG. 7

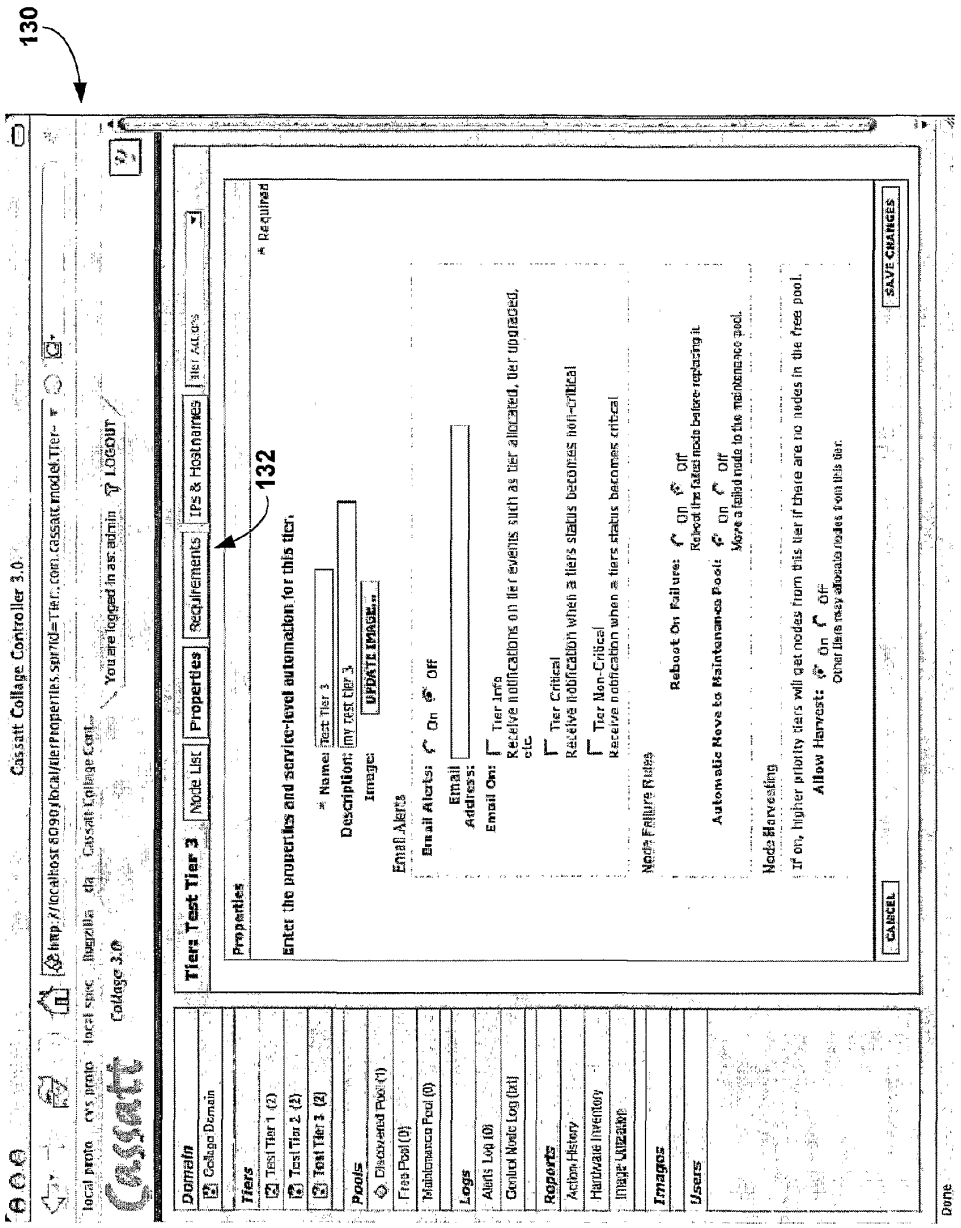

130
132
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (all)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Tiers Test Tier 3
Node List
Properties
Requirements
IPs & Hostnames
Tier Actions
Properties
Enter the properties and service-level automation for this tier.
* Required
* Name: Test Tier 3
Description: my test tier 3
Image:
UPDATE IMAGE...
Email Alerts
Email Alerts: On Off
Email Address:
Email On:
Tier Info
Receive notifications on tier events such as tier allocated, tier upgraded, etc
Tier Critical
Receive notification when a tiers status becomes non-critical
Tier Non-Critical
Receive notification when a tiers status becomes critical
Node Failure Rules
Reboot On Failures: On Off
Reboot the failed node before replacing it.
Automatic Move to Maintenance Pool: On Off
Move a failed node to the maintenance pool.
Node Harvesting
If on, higher priority tiers will get nodes from this tier if there are no nodes in the free pool.
Allow Harvest: On Off
Other tiers may allocate nodes from this tier.
CANCEL
SAVE CHANGES
Done

FIG. 8

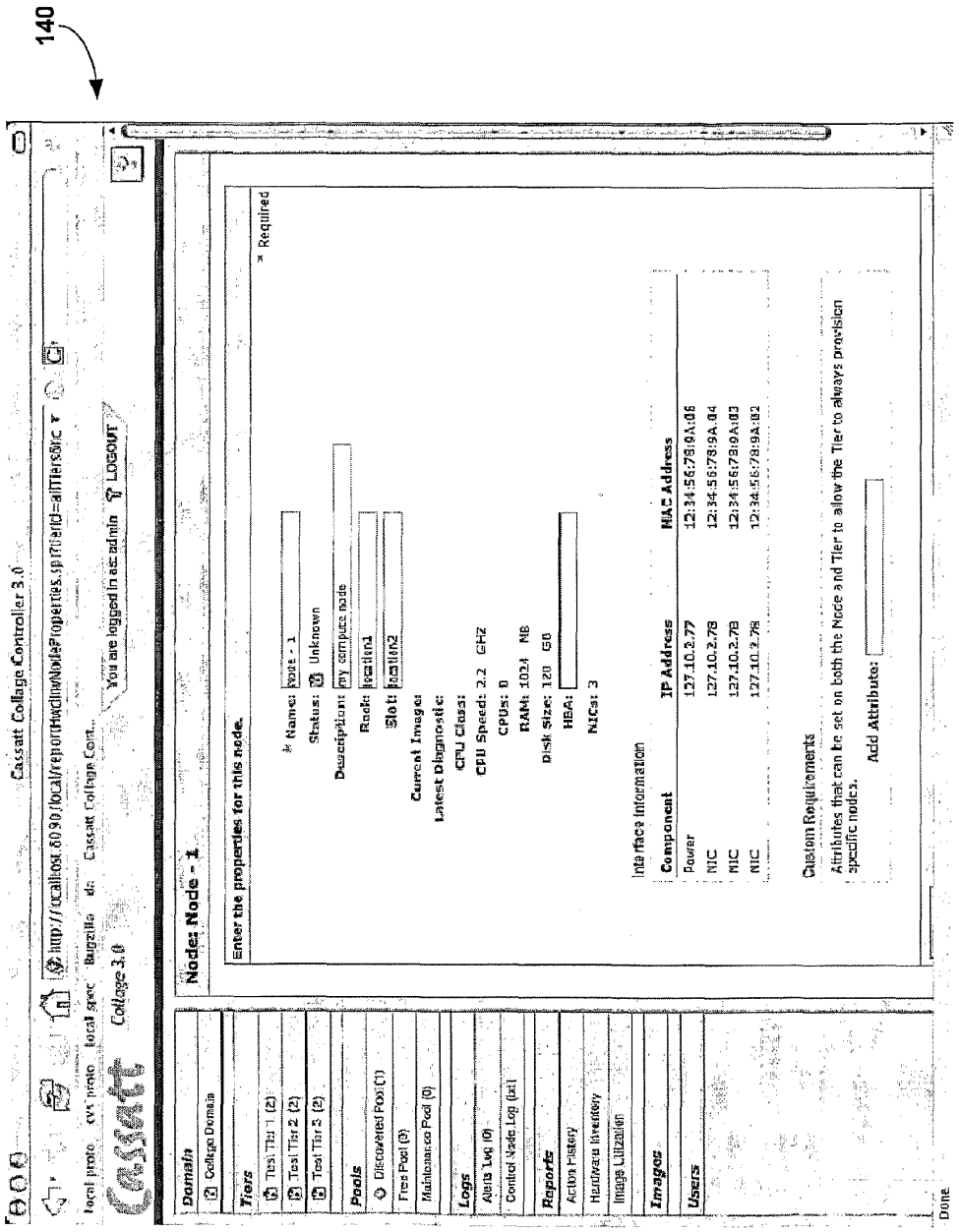

140
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (3)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (all
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Nodes: Node - 1
Enter the properties for this node.
* Required
* Name: Node - 1
Status: Unknown
Description: my compute node
Rack: location1
Slot: location2
Current Image:
Latest Diagnostic:
CPU Class:
CPU Speed: 2.2 GHz
CPUs: 8
RAM: 1024 MB
Disk Size: 120 GB
HBAs:
NICs: 3
Interface Information
Component
IP Address
MAC Address
Power
127.10.2.77
12:34:56:78:9A:08
NIC
127.10.2.78
12:34:56:78:9A:04
NIC
127.10.2.79
12:34:56:78:9A:03
NIC
127.10.2.76
12:34:56:78:9A:02
Custom Requirements
Attributes that can be set on both the Node and Tier to always provision specific nodes.
Add Attribute:
Done

FIG. 9

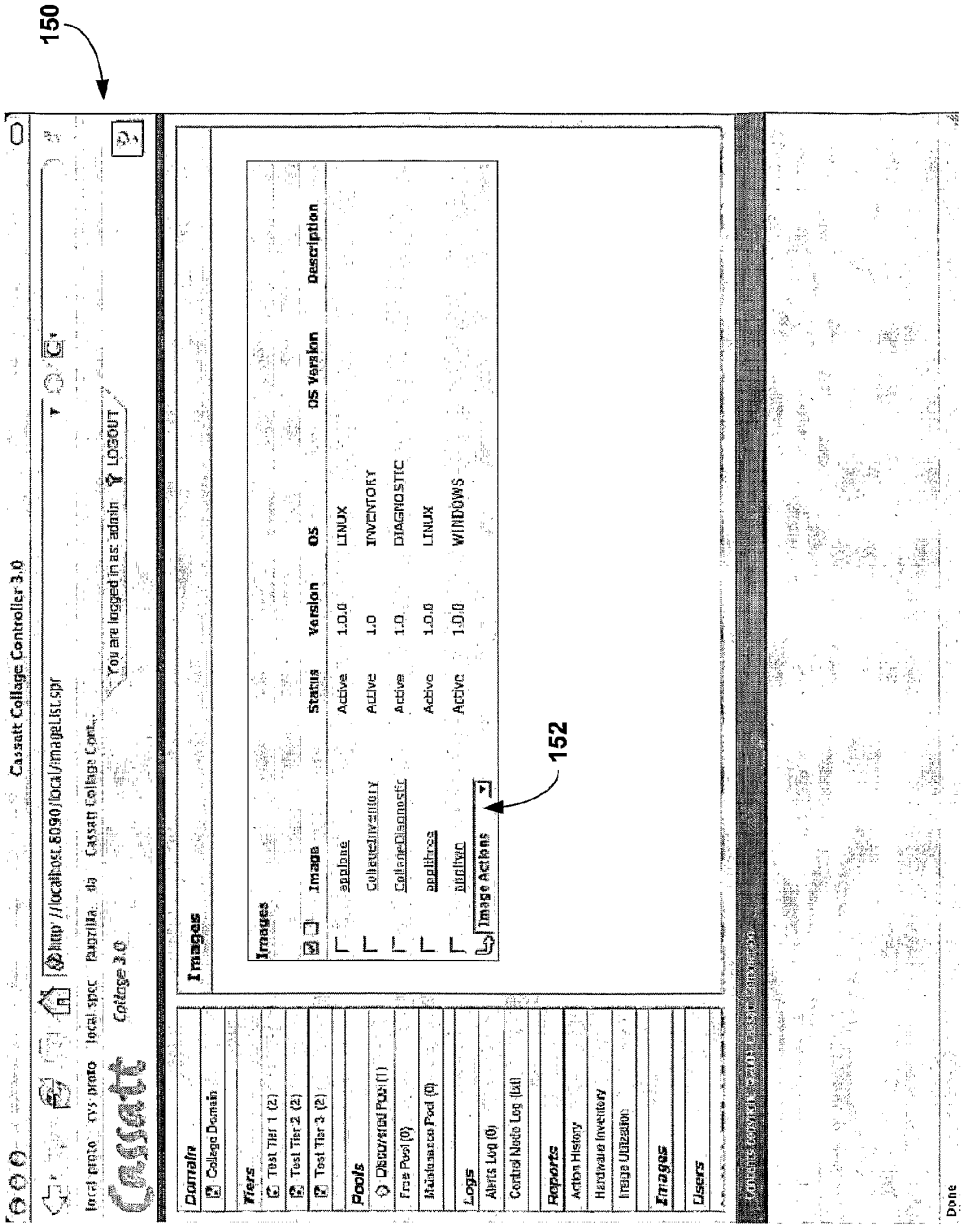

150
Cassatt College Controller 3.0
http://localhost:8080/local/imagelist.cpr
local proto
cvs proto
local spec
Bugzilla
Cassatt College Cont...
cassatt
College 3.0
You are logged in as: admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (full)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Images
Images
Image
Status
Version
OS
OS Version
Description
applione
Active
1.0.0
LINUX
CollegeInventory
Active
1.0
INVENTORY
CollegeDiagnostic
Active
1.0
DIAGNOSTIC
appltwo
Active
1.0.0
LINUX
applthree
Active
1.0.0
WINDOWS
Image Actions
152
Done

FIG. 10

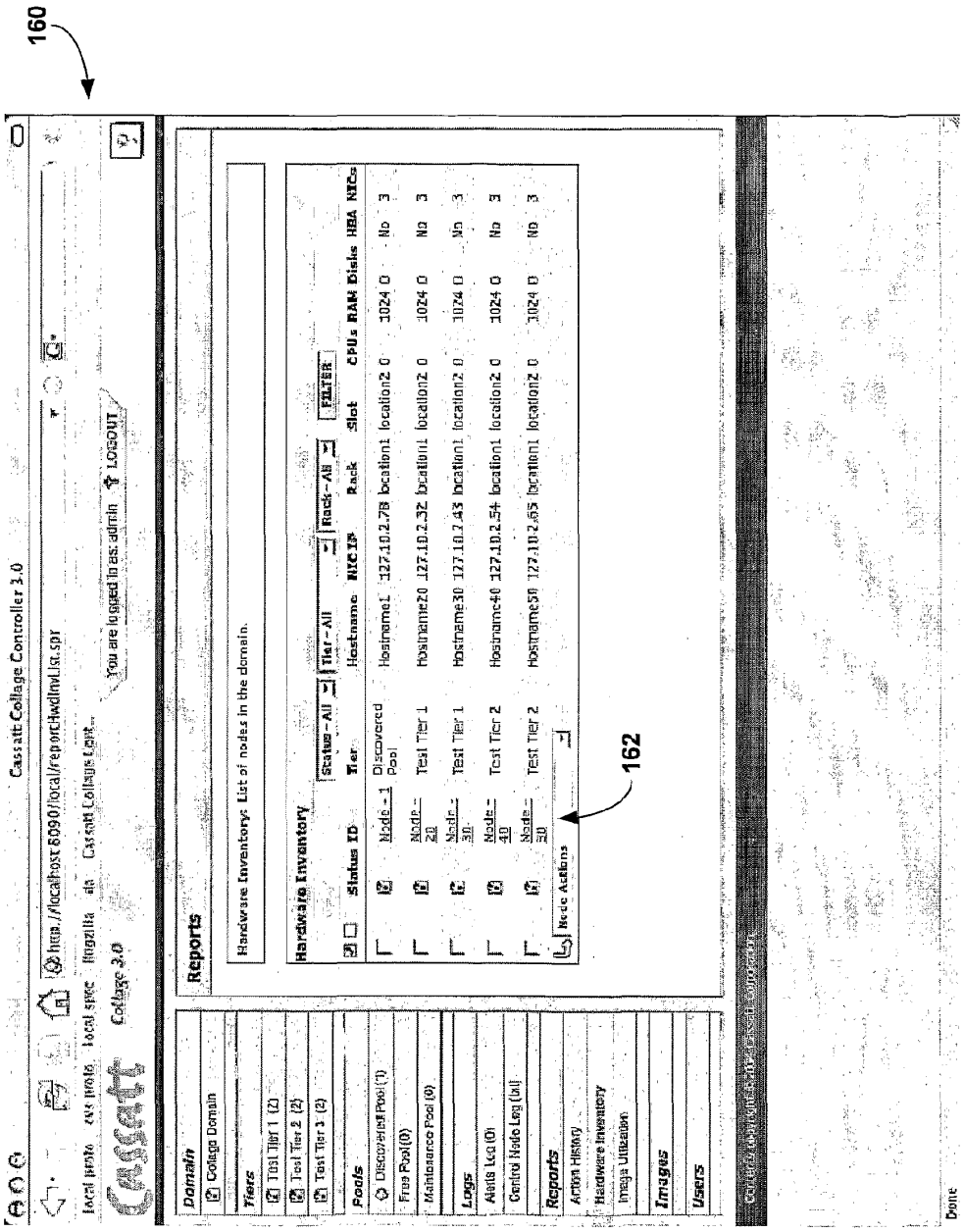

160
Cassatt Collage Controller 3.0
http://localhost:8090/local/reportHwdInvList.spr
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (3)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (all)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Reports
Hardware Inventory: List of nodes in the domain.
Hardware Inventory
Status - All
Tier - All
Rack - All
FILTER
Status ID
Tier
Hostname
NIC IP
Rack
Slot
CPUs
RAM
Disks
HBA
NICs
Node - 1
Discovered Pool
Hostname1
127.10.2.78
location1
location2
0
1024
0
No
3
Node - 20
Test Tier 1
Hostname20
127.10.2.32
location1
location2
0
1024
0
No
3
Node - 30
Test Tier 1
Hostname30
127.10.2.43
location1
location2
0
1024
0
No
3
Node - 40
Test Tier 2
Hostname40
127.10.2.54
location1
location2
0
1024
0
No
3
Node - 50
Test Tier 2
Hostname50
127.10.2.65
location1
location2
0
1024
0
No
3
Node Actions
162
Done

FIG. 11

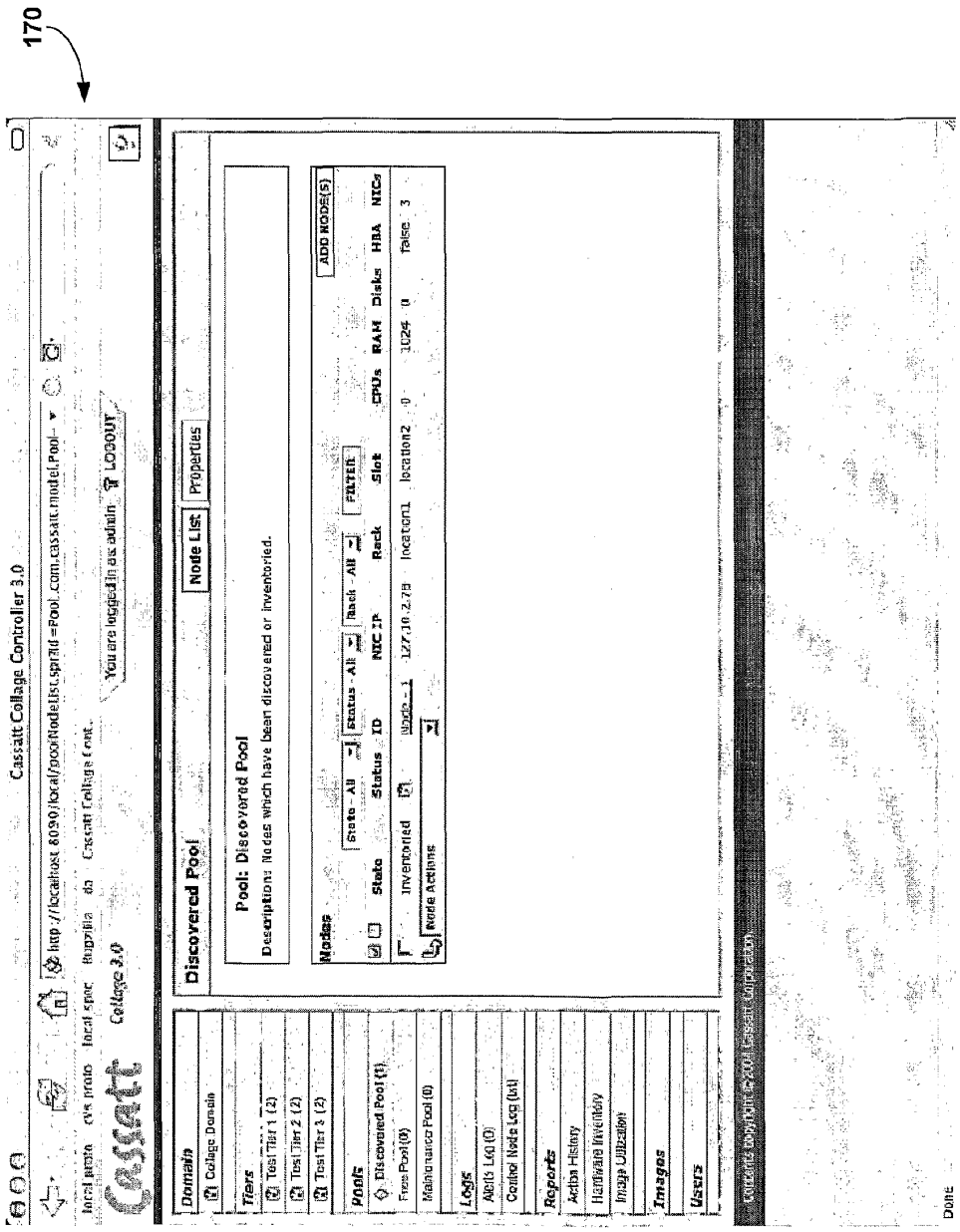

170
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (14)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Discovered Pool
Node List
Properties
Pool: Discovered Pool
Description: Nodes which have been discovered or inventoried.
Nodes
State - All
Status - All
Rack - All
FILTER
ADD NODE(S)
State
Status
ID
NIC IP
Rack
Slot
CPUs
RAM
Disks
HBA
NICs
Inventoried
node-1
127.10.2.79
location1
location2
0
1024
0
false
3
Node Actions
Done

FIG. 12

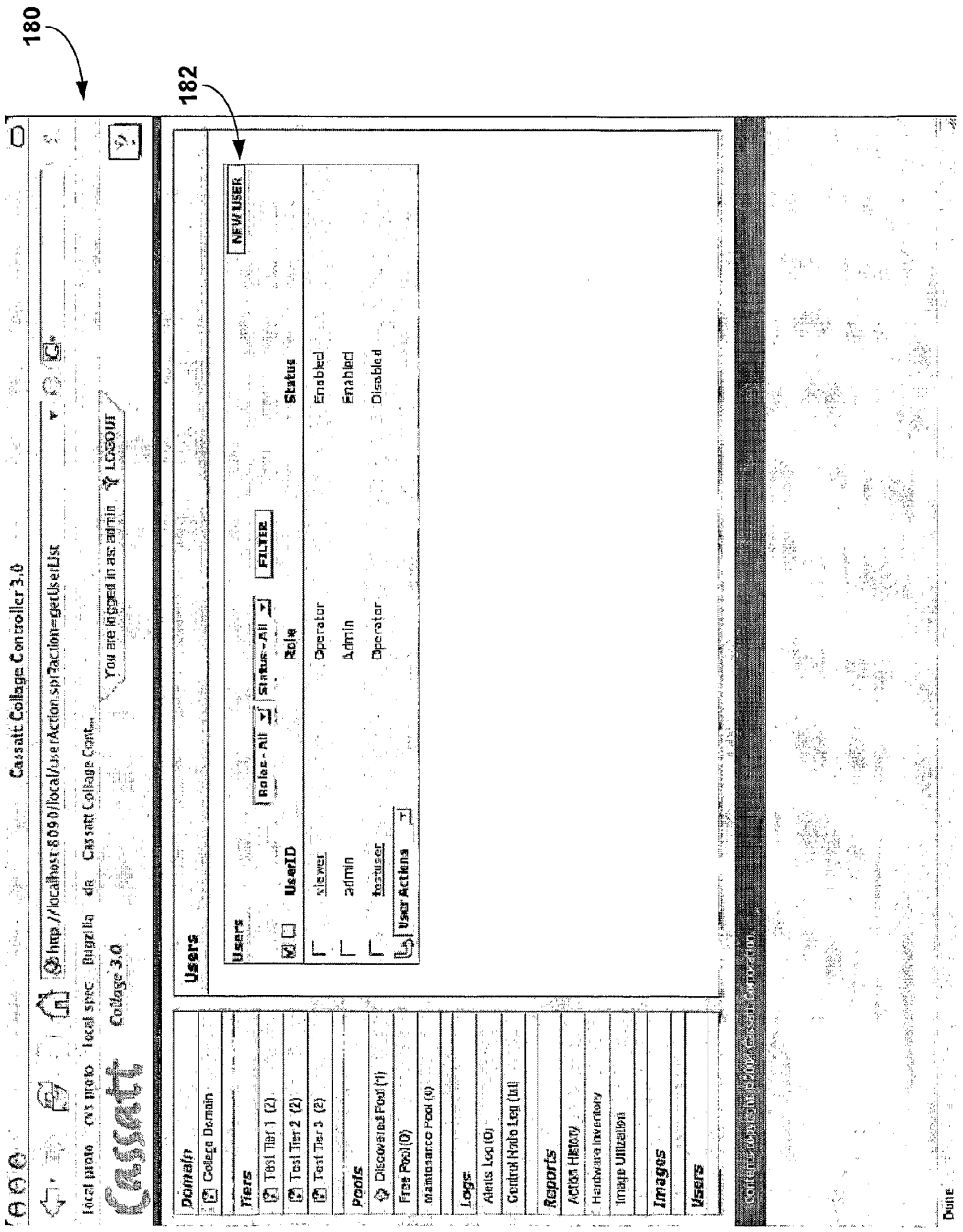

180
182
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (0)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Users
NEW USER
Status - All
FILTER
UserID
Role
Status
viewer
Operator
Enabled
admin
Admin
Enabled
testuser
Operator
Disabled
User Actions
Done

FIG. 13

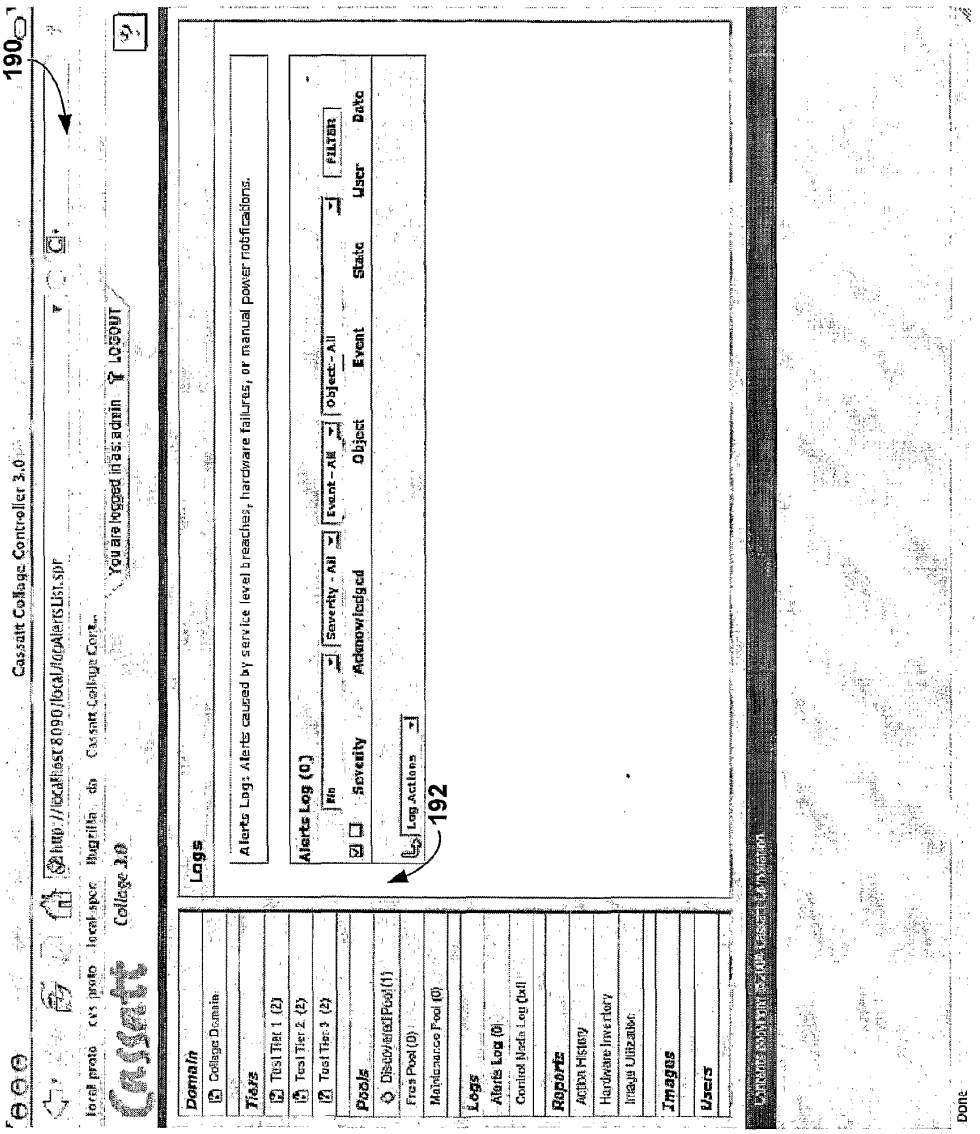

190
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
DiscoveredPool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (0)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Logs
Alerts Logs: Alerts caused by service level breaches, hardware failures, or manual power notifications.
Alerts Log (0)
Log Actions
192
Severity
Acknowledged
Object
Event
State
User
Date
FILTER
Done

FIG. 14

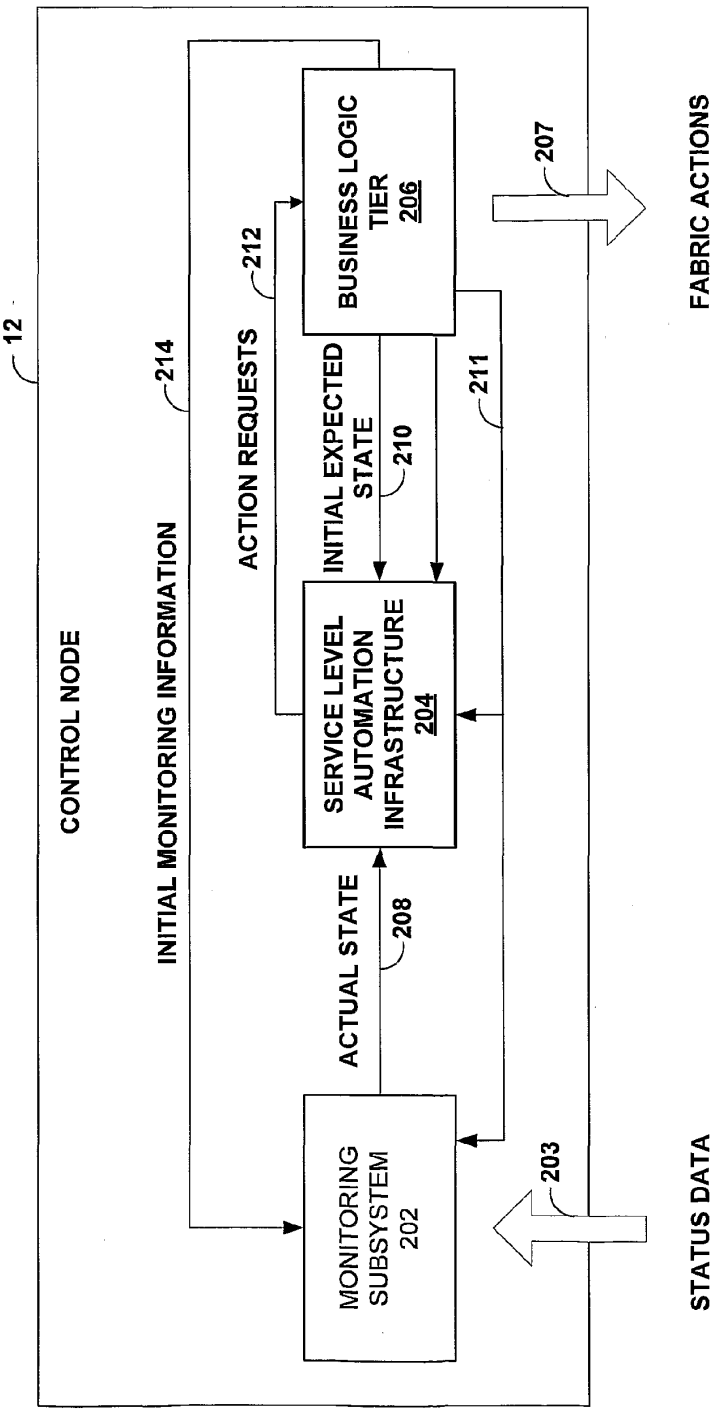

12
CONTROL NODE
214
INITIAL MONITORING INFORMATION
212
ACTION REQUESTS
BUSINESS LOGIC TIER 206
207
FABRIC ACTIONS
INITIAL EXPECTED STATE
210
211
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
ACTUAL STATE
208
MONITORING SUBSYSTEM 202
203
STATUS DATA

FIG. 15

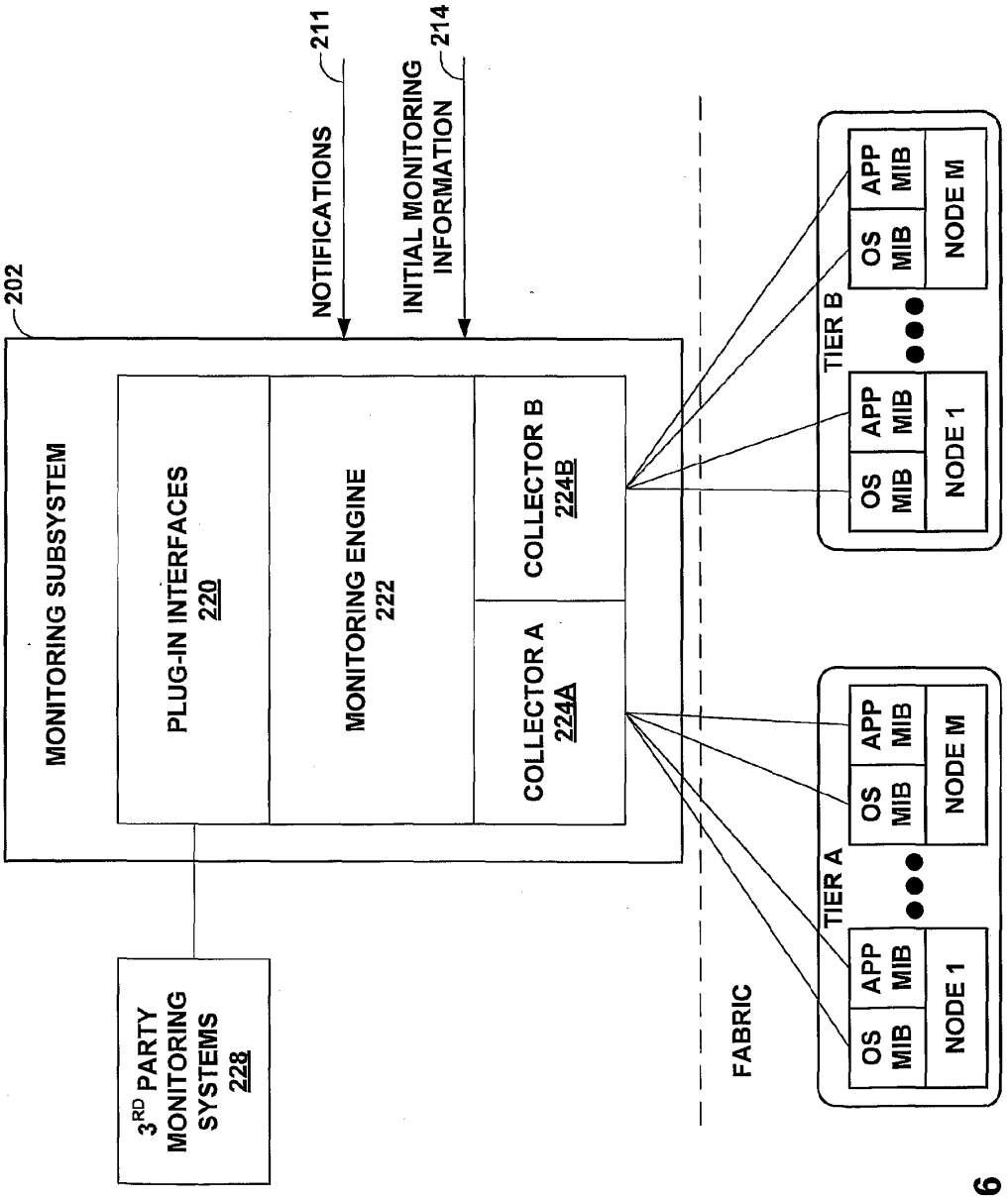

202
MONITORING SUBSYSTEM
PLUG-IN INTERFACES 220
MONITORING ENGINE 222
COLLECTOR A 224A
COLLECTOR B 224B
NOTIFICATIONS 211
INITIAL MONITORING INFORMATION 214
3RD PARTY MONITORING SYSTEMS 228
FABRIC
TIER A
TIER B
OS MIB
APP MIB
NODE 1
NODE M

FIG. 16

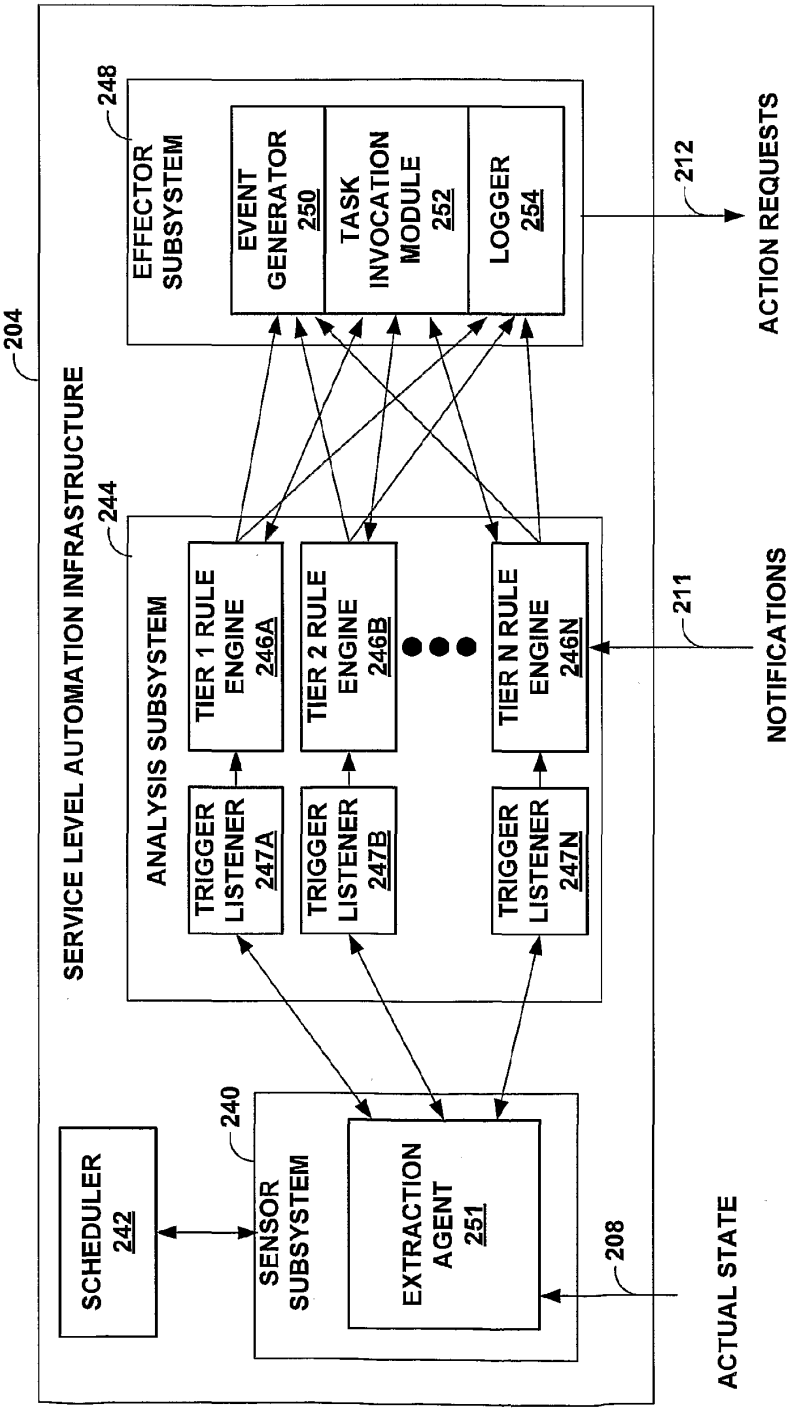

204
SERVICE LEVEL AUTOMATION INFRASTRUCTURE
248
EFFECTOR SUBSYSTEM
EVENT GENERATOR 250
TASK INVOCATION MODULE 252
LOGGER 254
212
ACTION REQUESTS
244
ANALYSIS SUBSYSTEM
TIER 1 RULE ENGINE 246A
TIER 2 RULE ENGINE 246B
TIER N RULE ENGINE 246N
TRIGGER LISTENER 247A
TRIGGER LISTENER 247B
TRIGGER LISTENER 247N
211
NOTIFICATIONS
240
SENSOR SUBSYSTEM
EXTRACTION AGENT 251
SCHEDULER 242
208
ACTUAL STATE

FIG. 17

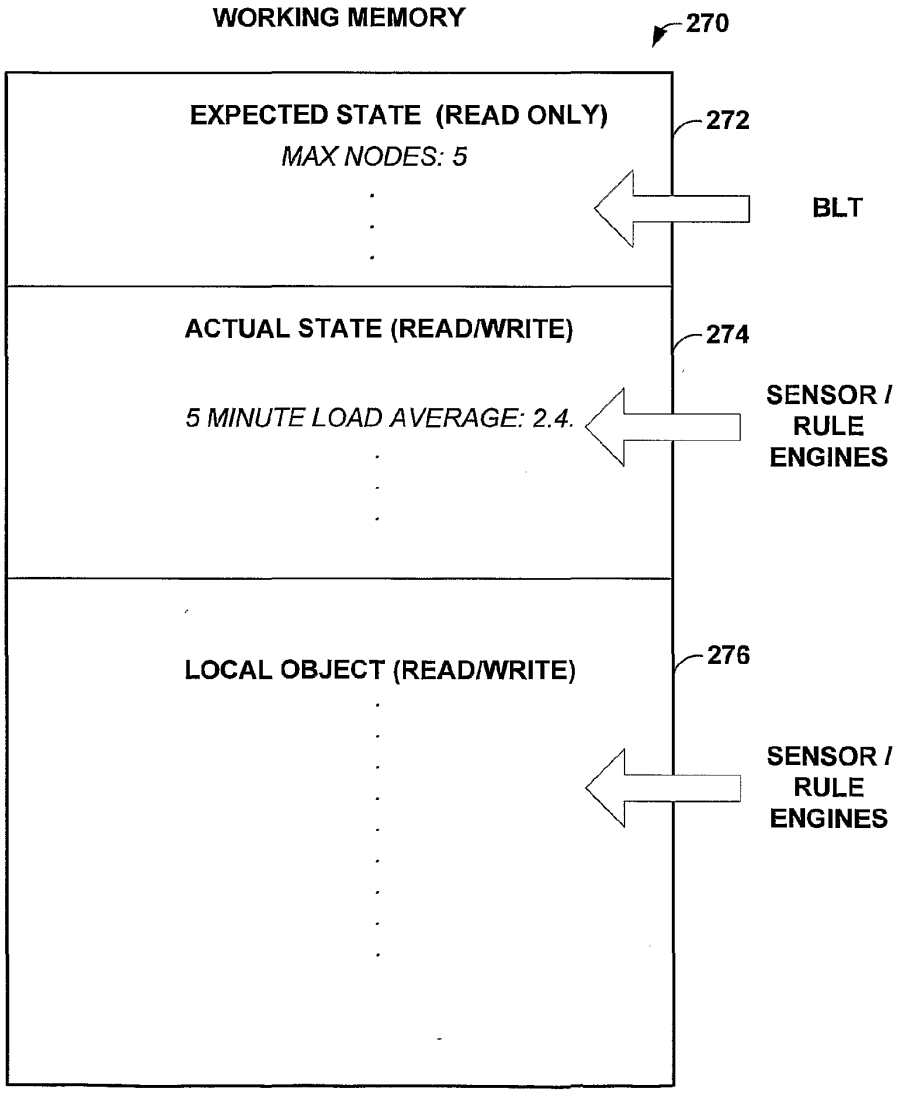

WORKING MEMORY
270
EXPECTED STATE (READ ONLY)
MAX NODES: 5
272
BLT
ACTUAL STATE (READ/WRITE)
274
5 MINUTE LOAD AVERAGE: 2.4.
SENSOR / RULE ENGINES
LOCAL OBJECT (READ/WRITE)
276
SENSOR / RULE ENGINES

FIG. 18

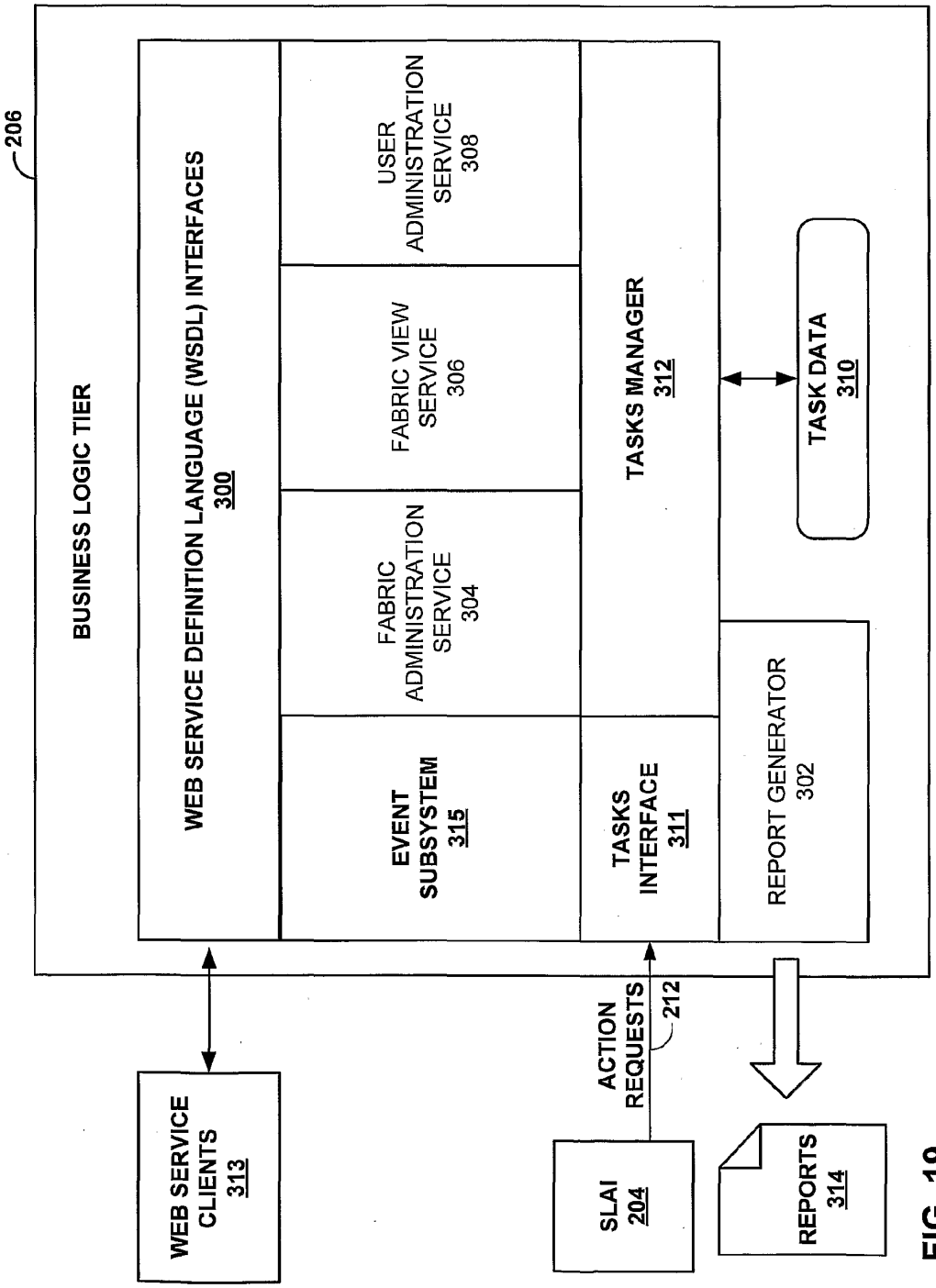

206
BUSINESS LOGIC TIER
WEB SERVICE DEFINITION LANGUAGE (WSDL) INTERFACES 300
USER ADMINISTRATION SERVICE 308
FABRIC VIEW SERVICE 306
FABRIC ADMINISTRATION SERVICE 304
EVENT SUBSYSTEM 315
TASKS MANAGER 312
TASKS INTERFACE 311
REPORT GENERATOR 302
TASK DATA 310
WEB SERVICE CLIENTS 313
ACTION REQUESTS 212
SLAI 204
REPORTS 314

FIG. 19

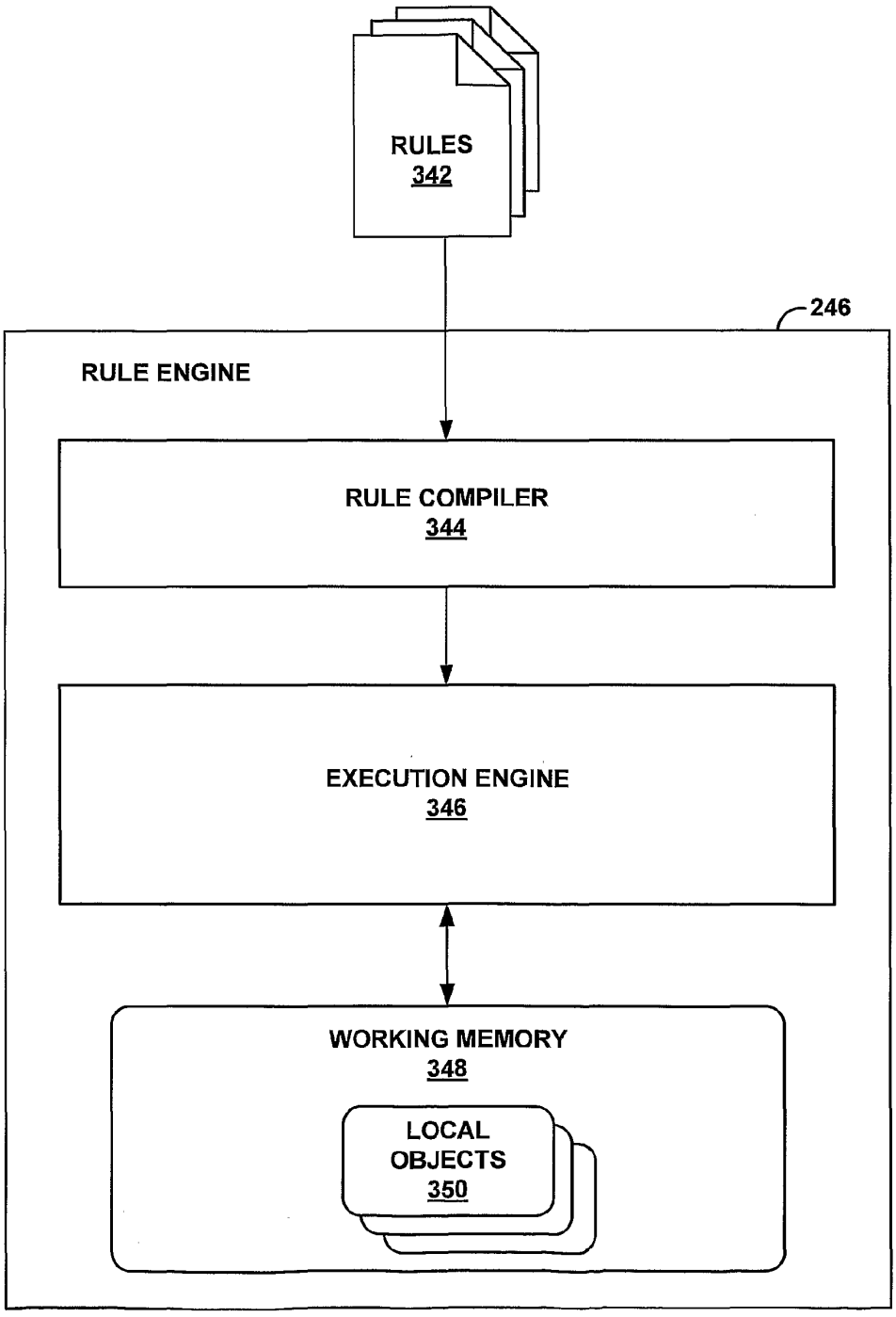

RULES
342
246
RULE ENGINE
RULE COMPILER
344
EXECUTION ENGINE
346
WORKING MEMORY
348
LOCAL
OBJECTS
350

FIG. 20

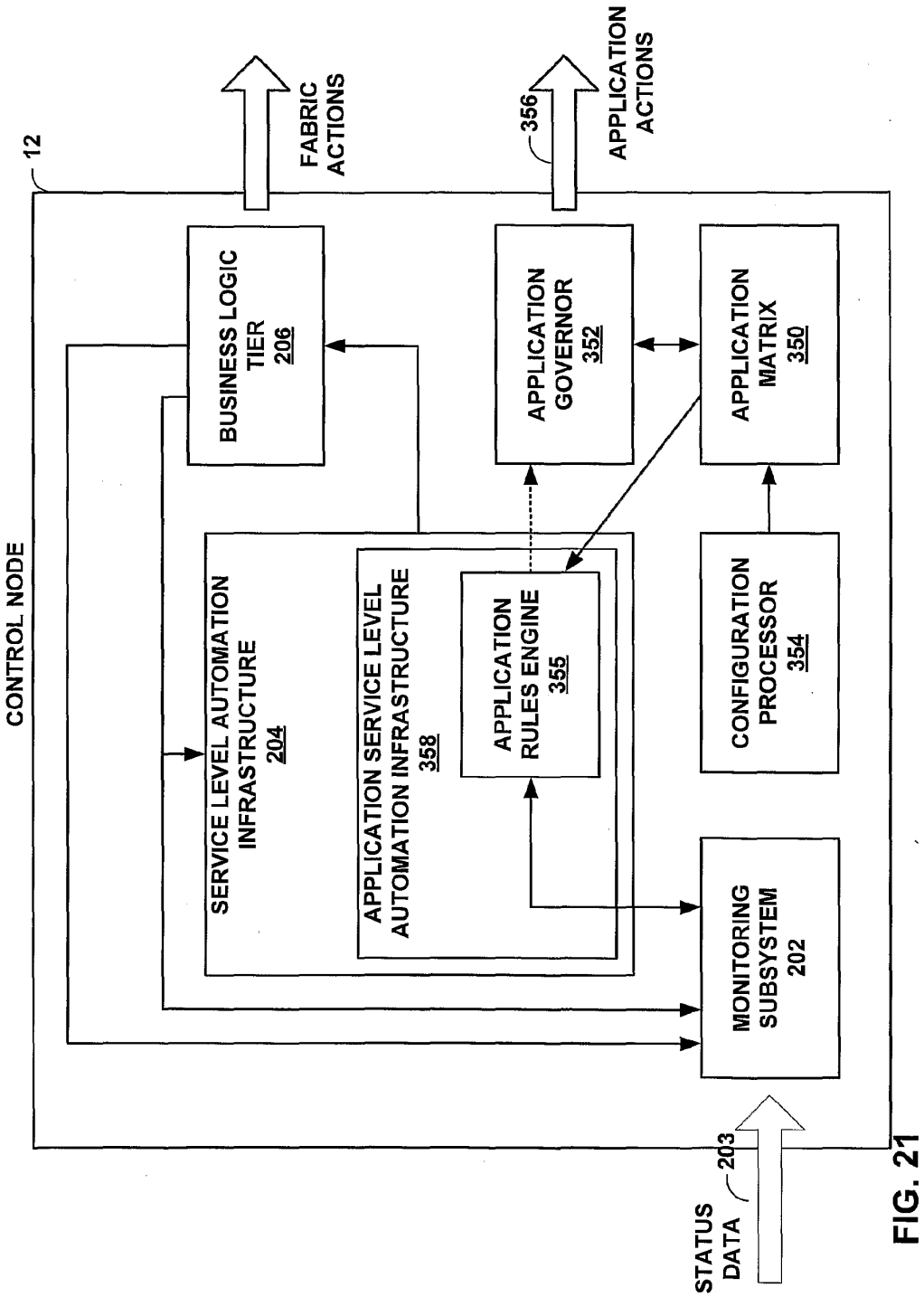

CONTROL NODE
12
FABRIC ACTIONS
356
APPLICATION ACTIONS
BUSINESS LOGIC TIER 206
APPLICATION GOVERNOR 352
APPLICATION MATRIX 350
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
APPLICATION SERVICE LEVEL AUTOMATION INFRASTRUCTURE 358
APPLICATION RULES ENGINE 355
CONFIGURATION PROCESSOR 354
MONITORING SUBSYSTEM 202
STATUS DATA
203

FIG. 21

360 APPLICATION	362 APPLICATION NODES	364 SERVICES	366 NODE MONITORED VALUES	368 SERVICE MONITORED ATTRIBUTES	370 SERVICE LEVELS	372 DEPLOYMENT CONSTRAINTS
Name= DataDomain **adminIP=** "172.31.64.201" **adminPort=** "1100" **adminTier=** "Web Admin" **clusterName=** "PathCluster" **expectedStartupDelay=**120 **loadDelay=**120 **maxNodes=**2 **minNodes=**1	**IP=** 172.31.64.201 **Name=** managedServer_0 **IP=** 172.31.64.202 **Name=** managedServer_1	**name=**"AI Design-time" **path=**"/lib/ai-designtime.ear" **name=**"System EJBs" **path=**"/lib/ejbs.ear" **name=**"Worklist Worker User Interface" **path=**"/lib/worklist/worklist.ear" **name=**"DBMS_ADK" **path=**"/lib/DBMS_ADK.ear" **name=**"App" **path=**"/user_projects/domains/data.ear"	**name=** Load5Average **name=** PercentFree-Memory	**name=** PendRequests **name=** ExecuteThread-IdleCount **name=** ExecuteThread-TotalCount	**attribute=** "BusyThread-Percentage" **expression=** "(ExecuteThreadIdleCount-ExecuteThreadIdleCount)*100/ExecuteThreadTotalCount" **frequency=**"15" **maxThreshold=**"20.0" **minThreshold=**"5.0" **period=**"30" **attribute=** "PendingRequests" **expression=** "PendingRequests" **frequency=**"15" **maxThreshold=**"20" **minThreshold=**"5" **period=**"30"	**attribute=** "Load5Average" **expression=** "Load5Average" **frequency=**"15" **maxThreshold=**"4" **minThreshold=**"-1" **period=**"15" **attribute=** "FreeMemory" **expression=** "PercentMemoryFree*100" **frequency=**"15" **maxThreshold=**"-1" **minThreshold=**"1" **period=**"60"
name= Portal-Domain						

376 374

FIG. 22

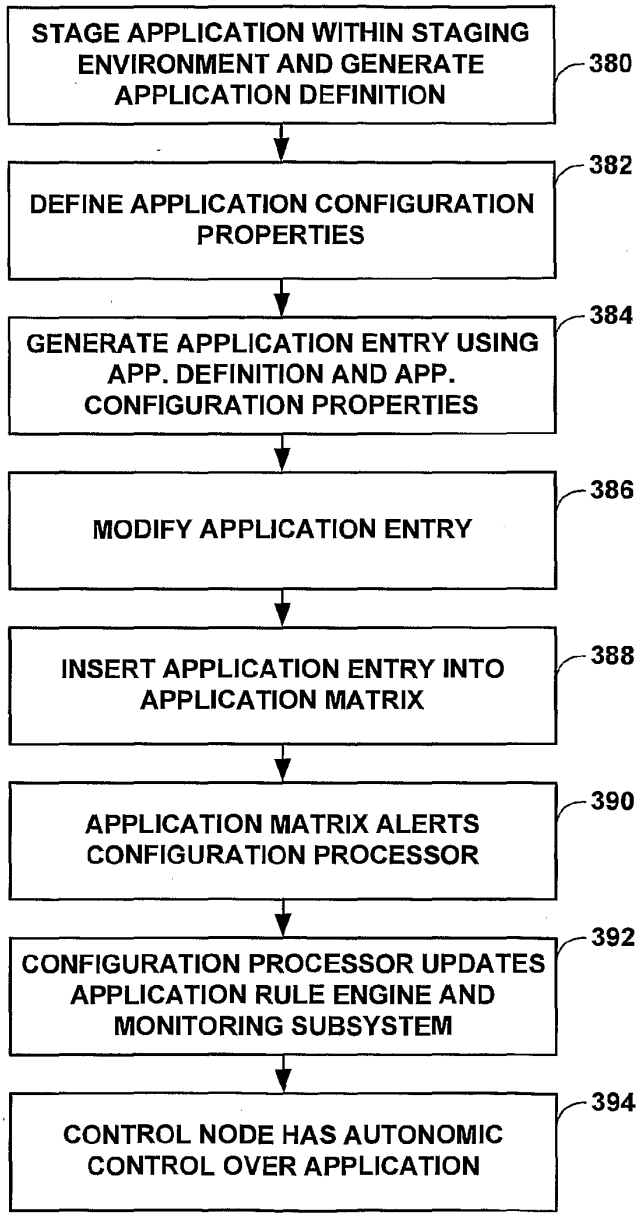

STAGE APPLICATION WITHIN STAGING ENVIRONMENT AND GENERATE APPLICATION DEFINITION
380
DEFINE APPLICATION CONFIGURATION PROPERTIES
382
GENERATE APPLICATION ENTRY USING APP. DEFINITION AND APP. CONFIGURATION PROPERTIES
384
MODIFY APPLICATION ENTRY
386
INSERT APPLICATION ENTRY INTO APPLICATION MATRIX
388
APPLICATION MATRIX ALERTS CONFIGURATION PROCESSOR
390
CONFIGURATION PROCESSOR UPDATES APPLICATION RULE ENGINE AND MONITORING SUBSYSTEM
392
CONTROL NODE HAS AUTONOMIC CONTROL OVER APPLICATION
394

FIG. 23

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003437

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an invantive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 July 2006	20/07/2006

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ross, C

Form PCT/ISA/210 (second sheet) (April 2005)

Netflix, Inc. - Ex. 1002, Page 000337
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003437

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevent to claim No.
X	HERNESS E N ET AL: "WebSphere Application Server: a foundation for on demand computing" IBM SYSTEMS JOURNAL IBM USA, vol. 43, no. 2, 2004, pages 213-237, XP002383791 ISSN: 0018-8670 Paragraph starting with "Multitiered distributed business computing": page 215, left-hand column page 220, left-hand column, line 26 - page 221, right-hand column, line 3 Paragraph starting with "Deployment automation": page 222, left-hand column - right-hand column, line 11 page 223, left-hand column, line 5 - line 23 page 224, right-hand column, line 11 - right-hand column, line 3 page 226, left-hand column, line 9 - page 229, left-hand column, line 2 page 229, right-hand column, line 1 - line 13 Paragraph starting with "Monitoring and application response measurement...": page 230, left-hand column	1-32
X	G LODI, F PANZIERI: "QoS-aware Clustering of Application Servers" PROCEEDINGS OF THE 1ST IEEE WORKSHOP ON QUALITY OF SERVICE FOR APPLICATION SERVERS, IN CONJUNCTION WITH THE 23RD INT'L SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS 2004), 17 October 2004 (2004-10-17), pages 1-6, XP002383792 Florianopolis, Brazil page 1, line 1 - line 19 page 2, left-hand column, line 10 - right-hand column, line 18 page 3, right-hand column, line 27 - line 45	1-32
X	US 2003/126265 A1 (AZIZ ASHAR ET AL) 3 July 2003 (2003-07-03) abstract; figure 1c paragraph [0049] paragraph [0052] paragraph [0088] - paragraph [0095] paragraph [0206]	1-32

-/--

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	URGAONKAR B ET AL: "Resource overbooking and application profiling in shared hosting platforms" PROCEEDINGS OF THE FIFTH SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION (OSDI'02) USENIX ASSOC BERKELEY, CA, USA, 2002, pages 239-254, XP002387427 ISBN: 1-931971-06-4 page 1, right-hand column, line 1 - line 19 Section 1.2 "System Model and Topology": page 3; figure 1 page 4, line 6 - line 20 -----	1-32

Netflix, Inc. - Ex. 1002, Page 000339
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/003437

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003126265 A1	03-07-2003	NONE	

Form PCT/ISA/210 (patent family annex) (April 2005)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 August 2006 (10.08.2006)

PCT

(10) International Publication Number
WO 2006/083901 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2006/003448

(22) International Filing Date: 31 January 2006 (31.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/047,468 31 January 2005 (31.01.2005) US
11/074,291 7 March 2005 (07.03.2005) US

(71) Applicant *(for all designated States except US)*: **CASSATT CORPORATION** [US/US]; 1740 Technology Drive, San Jose, California 95110 (US).

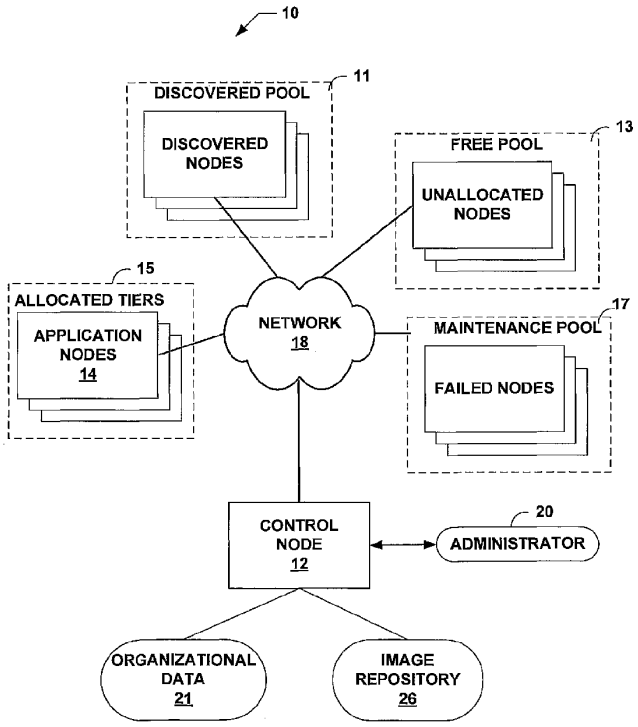
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **JACKSON, Jerry R.** [US/US]; 11270 Wakely Road, Colorado Springs, Colorado 80908 (US). **COLLINS, Doreen E.** [US/US]; 19175 Royal Troon, Monument, Colorado 80132 (US). **MURRISH, Randall L.** [US/US]; 1535 S. Clayton Street, Denver, Colorado 80210 (US). **EMEIS, Mark W.** [US/US]; 20250 Silver Horn Lane, Monument, Colorado 80132 (US). **FORNEY, Brian C.** [US/US]; 3852 Aldrich Ave. S., Minneapolis, Minnesota 55409 (US). **ENGQUIST, James D.** [US/US]; 7140 Oak Hills Lane, Colorado Springs, Colorado 80919 (US).

(74) Agent: **SIEFFERT, Kent J.**; SHUMAKER & SIEFFERT, P.A., 8425 SEASONS PARKWAY, Suite 105, St. Paul, Minnesota 55125 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI,

[Continued on next page]

(54) Title: AUTONOMIC CONTROL OF A DISTRIBUTED COMPUTING SYSTEM IN ACCORDANCE WITH A HIERACHICAL MODEL



(57) Abstract: A distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model. The model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of distributed computing system. The control node includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules.

WO 2006/083901 A1

NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

AUTONOMIC CONTROL OF A DISTRIBUTED COMPUTING SYSTEM IN ACCORDANCE WITH A HIERACHICAL MODEL

TECHNICAL FIELD

[0001] The invention relates to computing environments and, more specifically, to distributed computing systems.

BACKGROUND

[0002] Distributed computing systems are increasingly being utilized to support business as well as technical applications. Typically, distributed computing systems are constructed from a collection of computing nodes that combine to provide a set of processing services to implement the distributed computing applications. Each of the computing nodes in the distributed computing system is typically a separate, independent computing device interconnected with each of the other computing nodes via a communications medium, e.g., a network.

[0003] One challenge with distributed computing systems is the organization, deployment and administration of such a system within an enterprise environment. For example, it is often difficult to manage the allocation and deployment of enterprise computing functions within the distributed computing system. An enterprise, for example, often includes several business groups, and each group may have competing and variable computing requirements.

SUMMARY

[0004] In general, the invention is directed to a distributed computing system that conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.

[0005] As described herein, the model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes

Netflix, Inc. - Ex. 1002, Page 000343
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

to logically define the hierarchical organization of the distributed computing system. The control nodes are responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

[0006] In one embodiment, a method comprises receiving input that defines a model for a hierarchical organization of a distributed computing system having a plurality of computing nodes. The model specifies a fabric having one or more domains, and wherein each domain has at least one tier that includes at least one node slot. The method further comprises automatically configuring the distributed computing system in accordance with the hierarchical organization defined by the model.

[0007] In one embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network, and a control node. The control node includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules.

[0008] In another embodiment, a method comprises receiving input that defines an expected state for a distributed computing system having a plurality of application nodes interconnected via a communications network, and receiving status data that represents an actual state for the distributed computing system. The method further comprises processing rules with a set of rule engines to determine operations for reducing any difference between the actual state and the expected state, and applying the operations to the distributed computing system to control the application nodes in accordance with the rules.

[0009] In another embodiment, a computer-readable medium comprises instructions that cause a processor to process rules with a set of forward-chaining rule engines to determine operations for reducing differences between an actual state of a distributed computing system and an expected state. The instructions further cause the computer to apply the operations to the distributed computing system to provide autonomic control of the application nodes in accordance with the rules.

Netflix, Inc. - Ex. 1002, Page 000344
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0010] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0011] FIG. 1 is a block diagram illustrating a distributed computing system constructed from a collection of computing nodes.

[0012] FIG. 2 is a schematic diagram illustrating an example of a model of an enterprise that logically defines an enterprise fabric.

[0013] FIG. 3 is a flow diagram that provides a high-level overview of the operation of a control node when configuring the distributed computing system.

[0014] FIG. 4 is a flow diagram illustrating exemplary operation of the control node when assigning computing nodes to node slots of tiers.

[0015] FIG. 5 is a flow diagram illustrating exemplary operation of a control node when adding an additional computing node to a tier to meet additional processing demands.

[0016] FIG. 6 is a flow diagram illustrating exemplary operation of a control node harvesting excess node capacity from one of the tiers and returning the harvested computing node to the free pool.

[0017] FIG. 7 is a screen illustration of an exemplary user interface for defining tiers in a particular domain.

[0018] FIG. 8 is a screen illustration of an exemplary user interface for defining properties of the tiers.

[0019] FIG. 9 is a screen illustration of an exemplary user interface for viewing and identify properties of a computing node.

[0020] FIG. 10 is a screen illustration of an exemplary user interface for viewing software images.

[0021] FIG. 11 is a screen illustration of an exemplary user interface for viewing a hardware inventory report.

[0022] FIG. 12 is a screen illustration of an exemplary user interface for viewing discovered nodes that are located in the free pool.

Netflix, Inc. - Ex. 1002, Page 000345
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0023] FIG. 13 is a screen illustration of an exemplary user interface for viewing users of a distributed computing system.

[0024] FIG. 14 is a screen illustration of an exemplary user interface for viewing alerts for the distributed computing system.

[0025] FIG. 15 is a block diagram illustrating one embodiment of control node that includes a monitoring subsystem, a service level automation infrastructure (SLAI), and a business logic tier (BLT).

[0026] FIG. 16 is a block diagram illustrating one embodiment of the monitoring subsystem.

[0027] FIG. 17 is a block diagram illustrating one embodiment of the SLAI in further detail.

[0028] FIG. 18 is a block diagram of an example working memory associated with rule engines of the SLAI.

[0029] FIG. 19 is a block diagram illustrating an example embodiment for the BLT of the control node.

[0030] FIG. 20 is a block diagram illustrating one embodiment of a rule engine in further detail.

DETAILED DESCRIPTION

[0031] FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide distributed processing.

[0032] In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated tiers 15 and a maintenance pool 17. In addition, distributed computing system 10 includes at least one control node 12.

[0033] Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes

Netflix, Inc. - Ex. 1002, Page 000346
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

[0034] Discovered pool 11 includes a set of discovered nodes that have been automatically "discovered" within distributed computing system 10 by control node 12. For example, control node 12 may monitor dynamic host communication protocol (DHCP) leases to discover the connection of a node to network 18. Once detected, control node 12 automatically inventories the attributes for the discovered node and reassigns the discovered node to free pool 13. The node attributes identified during the inventory process may include a CPU count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics or other computing resources. Control node 12 may also receive input identifying node attributes not detectable via the automatic inventory, such as whether the node includes I/O, such as HBA. Further details with respect to the automated discovery and inventory processes are described in U.S. Patent Application No. 11/070,851, having attorney docket no. 1072-009US01, entitled "AUTOMATED DISCOVERY AND INVENTORY OF NODES WITHIN AN AUTONOMIC DISTRIBUTED COMPUTING SYSTEM," filed March 2, 2005.

[0035] Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated tiers 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to add an application node to allocated tiers 15 to increase processing capacity of distributed computing system 10.

[0036] In general, allocated tiers 15 include one or more tiers of application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory). Maintenance pool 17 includes a set of nodes that either could not be

Netflix, Inc. - Ex. 1002, Page 000347
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

inventoried or that failed and have been taken out of service from allocated tiers 15.

[0037] Control node 12 provides the system support functions for managing distributed computing system 10. More specifically, control node 12 manages the roles of each computing node within distributed computing system 10 and the execution of software applications within the distributed computing system. In general, distributed computing system 10 includes at least one control node 12, but may utilize additional control nodes to assist with the management functions.

[0038] Other control nodes 12 (not shown in FIG. 1) are optional and may be associated with a different subset of the computing nodes within distributed computing system 10. Moreover, control node 12 may be replicated to provide primary and backup administration functions, thereby allowing for graceful handling a failover in the event control node 12 fails.

[0039] Network 18 provides a communications interconnect for control node 12 and application nodes 14, as well as discovered nodes, unallocated nodes and failed nodes. Communications network 18 permits internode communications among the computing nodes as the nodes perform interrelated operations and functions. Communications network 18 may comprise, for example, direct connections between one or more of the computing nodes, one or more customer networks maintained by an enterprise, local area networks (LANs), wide area networks (WANs) or a combination thereof. Communications network 18 may include a number of switches, routers, firewalls, load balancers, and the like.

[0040] In one embodiment, each of the computing nodes within distributed computing system 10 executes a common general-purpose operating system. One example of a general-purpose operating system is the Windows™ operating system provided by Microsoft Corporation. In some embodiments, the general-purpose operating system such as the Linux kernel.

[0041] In the example of FIG. 1, control node 12 is responsible for software image management. The term "software image" refers to a complete set of software loaded on an individual computing node, including the operating system and all boot code, middleware and application files. System administrator 20 may interact with control node 12 and identify the particular types of software images to be

Netflix, Inc. - Ex. 1002, Page 000348
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

associated with application nodes 14. Alternatively, administration software executing on control node 12 may automatically identify the appropriate software images to be deployed to application nodes 14 based on the input received from system administrator 20. For example, control node 12 may determine the type of software image to load onto an application node 14 based on the functions assigned to the node by system administrator 20. Application nodes14 may be divided into a number of groups based on their assigned functionality. As one example, application nodes 14 may be divided into a first group to provide web server functions, a second group to provide business application functions and a third group to provide database functions. The application nodes 14 of each group may be associated with different software images.

[0042] Control node 12 provides for the efficient allocation and management of the various software images within distributed computing system 10. In some embodiments, control node 12 generates a "golden image" for each type of software image that may be deployed on one or more of application nodes 14. As described herein, the term "golden image" refers to a reference copy of a complete software stack.

[0043] System administrator 20 may create a golden image by installing an operating system, middleware and software applications on a computing node and then making a complete copy of the installed software. In this manner, a golden image may be viewed as a "master copy" of the software image for a particular computing function. Control node 12 maintains a software image repository 26 that stores the golden images associated with distributed computing system 10.

[0044] Control node 12 may create a copy of a golden image, referred to as an "image instance," for each possible image instance that may be deployed within distributed computing system 10 for a similar computing function. In other words, control node 12 pre-generates a set of K image instances for a golden image, where K represents the maximum number of image instances for which distributed computing system 10 is configured for the particular type of computing function. For a given computing function, control node 12 may create the complete set of image instance even if not all of the image instances will be initially deployed. Control node 12 creates different sets of image instances for different computing

Netflix, Inc. - Ex. 1002, Page 000349
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

functions, and each set may have a different number of image instances depending on the maximum number of image instances that may be deployed for each set. Control node 12 stores the image instances within software image repository 26. Each image instance represents a collection of bits that may be deployed on an application node.

[0045] Further details of software image management are described in co-pending U.S. patent application Ser. No. 11/046,133, entitled "MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005 and co-pending U.S. patent application Ser. No. 11/046,152, entitled "UPDATING SOFTWARE IMAGES ASSOCIATED WITH A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005.

[0046] In general, distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

[0047] As used herein, the "fabric" level generally refers to the logical constructs that allow for definition, deployment, partitioning and management of distinct enterprise applications. In other words, fabric refers to the integrated set of hardware, system software and application software that can be "knitted" together to form a complete enterprise system. In general, the fabric level consists of two elements: fabric components or fabric payload. Control node 12 provides fabric management and fabric services as described herein.

[0048] In contrast, a "domain" is a logical abstraction for containment and management within the fabric. The domain provides a logical unit of fabric allocation that enables the fabric to be partitioned amongst multiple uses, e.g. different business services.

[0049] Domains are comprised of tiers, such as a 4-tier application model (web server, application server, business logic, persistence layer) or a single tier monolithic application. Fabric domains contain the free pool of devices available for assignment to tiers.

Netflix, Inc. - Ex. 1002, Page 000350
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0050] A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

[0051] The lowest level, the node level, includes the physical components of the fabric. This includes computing nodes that, as described above, provide operating environments for system applications and enterprise software applications. In addition, the node level may include network devices (e.g., Ethernet switches, load balancers and firewalls) used in creating the infrastructure of network 18. The node level may further include network storage nodes that are network connected to the fabric.

[0052] System administrator 20 accesses administration software executing on control node 12 to logically define the hierarchical organization of distributed computing system 10. For example, system administrator 20 may provide organizational data 21 to develop a model for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.

[0053] More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation management (CRM) service. System administrator 20 further defines one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a web tier, an application tier and a database tier. In this manner, distributed computing system 10 may be configured to automatically provide web server functions, business application functions and database functions.

Netflix, Inc. - Ex. 1002, Page 000351
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0054] For each of the tiers, control node 12 creates a number of "node slots" equal to the maximum number of application nodes 14 that may be deployed. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot. The node slots may, for instance, identify a particular software image instance associated with an application node 14 as well as a network address associated with that particular image instance.

[0055] In this manner, each of the tiers include one or more node slots that reference particular software image instances to boot on the application nodes 14 to which each software image instance is assigned. The application nodes 14 to which control node 12A assigns the image instances temporarily inherit the network address assigned to the image instance for as long as the image instance is deployed on that particular application node. If for some reason the image instance is moved to a different application node 14, control node 12A moves the network address to that new application node.

[0056] System administrator 20 may further define specific node requirements for each tier of the fabric. For example, the node requirements specified by system administrator 20 may include a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics that may be detected on the individual computing nodes. System administrator 20 may also specify user-defined hardware attributes of the computing nodes, such as whether I/O (like HBA) is required. The user-defined hardware attributes are typically not capable of detection during an automatic inventory. In this manner, system administrator 20 creates a list of attributes that the tier requires of its candidate computing nodes. In addition, particular node requirements may be defined for software image instances.

[0057] In addition to the node requirements described above, system administrator 20 may further define policies that are used when re-provisioning computing nodes within the fabric. System administrator 20 may define policies regarding tier characteristics, such as a minimum number of nodes a tier requires, an indication of whether or not a failed node is dynamically replaced by a node from free pool 13, a priority for each tier relative to other tiers, an indication of whether or not a

Netflix, Inc. - Ex. 1002, Page 000352
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

tier allows nodes to be re-provisioned to other tiers to satisfy processing requirements by other tiers of a higher priority or other policies. Control node 12 uses the policy information input by system administrator 20 to re-provision computing nodes to meet tier processing capacity demands.

[0058] After receiving input from system administrator 20 defining the architecture and policy of the enterprise fabric, control node 12 identifies unallocated nodes within free pool 13 that satisfy required node attributes. Control node 12 automatically assigns unallocated nodes from free pool 13 to respective tier node slots of a tier. As will be described in detail herein, in one embodiment, control node 12 may assign computing nodes to the tiers in a "best fit" fashion. Particularly, control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20. The assignment of the computing nodes may occur on a tier-by-tier basis beginning with a tier with the highest priority and ending with a tier with the lowest priority. Alternatively, or in addition, assignment of computing nodes may be based on dependencies defined between tiers.

[0059] As will be described in detail below, control node 12 may automatically add unallocated nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to the free pool when the tier has excess capacity, transfer nodes from tier to tier to meet processing demands, or replace failed nodes with nodes from the free pool. Thus, computing resources, i.e., computing nodes, may be automatically shared between tiers and domains within the fabric based on user-defined policies to dynamically address high-processing demands, failures and other events.

[0060] FIG. 2 is a schematic diagram illustrating an example embodiment of organizational data 21 that defines a model logically representing an enterprise fabric in accordance with the invention. In the example illustrated in FIG. 2, control node 12 (FIG. 1) maintains organizational data 21 to define a simple e-commerce fabric 32.

[0061] In this example, e-commerce fabric 32 includes a storefront domain 34A and a financial planning domain 34B. Storefront domain 34A corresponds to the enterprise storefront domain and allows customers to find and purchase products

Netflix, Inc. - Ex. 1002, Page 000353
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

over a network, such as the Internet. Financial planning domain 34B allows one or more employees to perform financial planning tasks for the enterprise.

[0062] Tier level 31C includes one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. For example, storefront domain 34A includes a web server tier (labeled "web tier") 36A, a business application tier (labeled "app tier") 36B, and a database tier (labeled "DB tier") 36C. Web server tier 36A, business application tier 36B and database tier 36C interact with one another to present a customer with an online storefront application and services. For example, the customer may interact with web server tier 36A via a web browser. When the customer searches for a product, web server tier 36A may interacts with business application tier 36B, which may in turn access a database tier 36C. Similarly, financial planning domain 34B includes a financial planning tier 36D that provides subcomponents of applications and services of the financial planning domain 34B. Thus, in this example, a domain may include a single tier.

[0063] Tier level 31D includes one or more logical node slots 38A-38H ("node slots 38") within each of the tiers. Each of node slots 38 include node specific information, such as software resources for an application node 14 that is assigned to a respective one of the node slots 38. Node slots 38 may, for instance, identify particular software image instances within image repository 26 and map the identified software image instances to respective application nodes 14. As an example, node slots 38A and 38B belonging to web server tier 36A may reference particular software image instances used to boot two application nodes 14 to provide web server functions. Similarly, the other node slots 38 may reference software image instances to provide business application functions, database functions, or financial application functions depending upon the tier to which the node slots are logically associated.

[0064] Although in the example of FIG. 2, there are two node slots 38 corresponding to each tier, the tiers may include any number of node slots depending on the processing capacity needed on the tier. Furthermore, not all of node slots 38 may be currently assigned to an application node 14. For example, node slot 28B may be associated with an inactive software image instance and,

Netflix, Inc. - Ex. 1002, Page 000354
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

when needed, may be assigned to an application node 14 for deployment of the software image instance.

[0065] In this example, organizational data 21 associates free node pool 13 with the highest-level of the model, i.e., e-commerce fabric 32. As described above, control node 12 may automatically assign unallocated nodes from free node pool 13 to at least a portion of tier node slots 38 of tiers 36 as needed using the "best fit" algorithm described above or another algorithm. Additionally, control node 12 may also add nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.

[0066] Although not illustrated, the model for the enterprise fabric may include multiple free node pools. For example, the model may associate free node pools with individual domains at the domain level or with individual tier levels. In this manner, administrator 20 may define policies for the model such that unallocated computing nodes of free node pools associated with domains or tiers may only be used within the domain or tier to which they are assigned. In this manner, a portion of the computing nodes may be shared between domains of the entire fabric while other computing nodes may be restricted to particular domains or tiers.

[0067] FIG. 3 is a flow diagram that provides a high-level overview of the operation of control node 12 when configuring distributed computing system 10. Initially, control node 12 receives input from a system administrator defining the hierarchical organization of distributed computing system 10 (50). In one example, control node 12 receives input that defines a model that specifies a number of hierarchically arranged nodes as described in detail in FIG. 2. Particularly, the defined architecture of distributed computing system 10 includes an overall fabric having a number of hierarchically arranged domains, tiers and node slots.

[0068] During this process, control node 12 may receive input specifying node requirements of each of the tiers of the hierarchical model (52). As described above, administrator 20 may specify a list of attributes, e.g., a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), or local disk

Netflix, Inc. - Ex. 1002, Page 000355
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

characteristics, that the tiers require of their candidate computing nodes. In addition, control node 12 may further receive user-defined custom attributes, such as requiring the node to have I/O, such as HBA connectivity. The node requirements or attributes defined by system administrator 20 may each include a name used to identify the characteristic, a data type (e.g., integer, long, float or string), and a weight to define the importance of the requirement.

[0069] Control node 12 identifies the attributes for all candidate computing nodes within free pool 13 or a lower priority tier (54). As described above, control node 12 may have already discovered the computing nodes and inventoried the candidate computing nodes to identify hardware characteristics of all candidate computing nodes. Additionally, control node 12 may receive input from system administrator 20 identifying specialized capabilities of one or more computing nodes that are not detectable by the inventory process.

[0070] Control node 12 dynamically assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a "best fit" algorithm.

[0071] FIG. 4 is a flow diagram illustrating exemplary operation of control node 12 when assigning computing nodes to node slots of tiers. Initially, control node 12 selects a tier to enable (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

[0072] Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each

Netflix, Inc. - Ex. 1002, Page 000356
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

of the tiers. Control node 12 retrieves the node requirements for the selected tier from the database.

[0073] In addition, control node 12 accesses the database and retrieves the computing node attributes of one of the unallocated computing nodes of free pool 13. Control node 12 compares the node requirements of the tier to the node attributes of the selected computing node (64).

[0074] Based on the comparison, control node 12 determines whether the node attributes of the computing node meets the minimum node requirements of the tier (66). If the node attributes of the selected computing node do not meet the minimum node requirements of the tier, then the computing node is removed from the list of candidate nodes for this particular tier (68). Control node 12 repeats the process by retrieving the node attributes of another of the computing nodes of the free pool and compares the node requirements of the tier to the node attributes of the computing node.

[0075] If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the node requirements of the tier (70). If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).

[0076] Control node 12 then determines whether the node count for the tier is met (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20. When the node count for the tier is met, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority, and repeats the process until all defined tiers are enabled, i.e., populated with application nodes (60).

[0077] If the node attributes of the selected computing node and the node requirements of the tier are not a perfect match control node 12 calculates and records a "processing energy" of the node (76). As used herein, the term

Netflix, Inc. - Ex. 1002, Page 000357
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

"processing energy" refers to a numerical representation of the difference between the node attributes of a selected node and the node requirements of the tier. A positive processing energy indicates the node attributes more than satisfy the node requirements of the tier. The magnitude of the processing energy represents the degree to which the node requirements exceed the tier requirements.

[0078] After computing and recording the processing energy of the nodes, control node 12 determines whether there are more candidate nodes in free pool 13 (78). If there are additional candidate nodes, control node 12 repeats the process by retrieving the computing node attributes of another one of the computing nodes of the free pool of computing nodes and comparing the node requirements of the tier to the node attributes of the computing node (64).

[0079] When all of the candidate computing nodes in the free pool have been examined, control node 12 selects the candidate computing node having the minimum positive processing energy and assigns the selected computing node to a node slot of the tier (80). Control node 12 determines whether the minimum node count for the tier is met (82). If the minimum node count for the tier has not been met, control node 12 assigns the computing node with the next lowest calculated processing energy to the tier (80). Control node 12 repeats this process until the node count is met. At this point, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority (60).

[0080] In the event there are an insufficient number of computing nodes in free pool 13, or an insufficient number of computing nodes that meet the tier requirements, control node 12 notifies system administrator 20. System administrator 20 may add more nodes to free pool 13, add more capable nodes to the free pool, reduce the node requirements of the tier so more of the unallocated nodes meet the requirements, or reduce the configured minimum node counts for the tiers.

[0081] FIG. 5 is a flow diagram illustrating exemplary operation of control node 12 when adding an additional computing node to a tier to meet increased processing demands. Initially, control node 12 or system administrator 20 identifies a need for additional processing capacity on one of the tiers (90). Control node 12 may, for example, identify a high processing load on the tier or

Netflix, Inc. - Ex. 1002, Page 000358
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

receive input from a system administrator identifying the need for additional processing capacity on the tier.

[0082] Control node 12 then determines whether there are any computing nodes in the free pool of nodes that meet the minimum node requirements of the tier (92). When there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and assigns the node to the tier (95). As described in detail with respect to FIG. 4, control node 12 may determine whether there are any nodes that have node attributes that are an exact match to the node requirements of the tier. If an exact match is found, the corresponding computing node is assigned to a node slot of the tier. If no exact match is found, control node 12 computes the processing energy for each node and assigns the computing node with the minimum processing energy to the tier. Control node 12 remotely powers on the assigned node and remotely boots the node with the image instance associated with the node slot. Additionally, the booted computing node inherits the network address associated with the node slot.

[0083] If there are no adequate computing nodes in the free pool, i.e., no nodes at all or no nodes that match the minimal node requirements of the tier, control node 12 identifies the tiers with a lower priority than the tier needing more processing capacity (96).

[0084] Control node 12 determines which of the nodes of the lower priority tiers meet the minimum requirements of the tier in need of processing capacity (98). Control node 12 may, for example, compare the attributes of each of the nodes assigned to node slots of the lower priority tiers to the node requirements of the tier in need of processing capacity. Lower priority tiers that have the minimum number of computing nodes may be removed from possible tiers from which to harvest an application node. If, however, all the lower priority tiers have the minimum number of computing nodes defined for the respective tier, the lowest priority tier is selected from which to harvest the one or more nodes.

[0085] Control node 12 calculates the processing energy of each of the nodes of the lower priority tiers that meet the minimum requirements (100). The energies of the nodes are calculated using the differences between the node attributes and the

Netflix, Inc. - Ex. 1002, Page 000359
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node requirements of the tier needing additional capacity. Control node 12 selects the computing node with the lowest processing energy that meets the minimum requirements, and assigns the selected computing node to the tier in need of processing capacity (102, 95).

[0086] FIG. 6 is a flow diagram illustrating exemplary operation of control node 12 when harvesting excess node capacity from one of the tiers and returning the harvested computing node to free pool 13. Initially, control node 12 identifies a tier having excess node capacity (110). Control node 12 may, for example, periodically check the node capacity of the tiers to identify any tiers having excess node capacity. Performing a periodic check and removal of excess nodes increases the likelihood that a capable computing node will be in free pool 13 in the event one of the tiers needs additional node capacity.

[0087] When harvesting a node, control node 12 calculates the processing energy of all the nodes in the tier as described above with reference to FIG. 4 (112). Control node 12 identifies the node within the tier with the highest processing energy and returns the identified node to the free pool of nodes (114, 116). As described above, the node with the highest processing energy corresponds to the node whose node attributes are the most in excess of the node requirements of the tier.

[0088] Returning the node to the free pool may involve remotely powering off the computing node and updating the database to associate the harvested node with free pool 13. In addition, control node 12 updates the database to disassociate the returned node with the node slot to which it was assigned. At this point, the node no longer uses the network address associated with the image instance mapped to the node slot. Control node 12 may, therefore, assign a temporary network address to the node while the node is assigned to free pool 13.

[0089] FIG. 7 is a screen illustration of an exemplary user interface 120 presented by control node 12 with which administrator 20 interacts to define tiers for a particular domain. In the example illustrated in FIG. 7, system administrator 20 has selected the "Collage Domain." User interface 120 presents the tiers that are currently in the selected domain. In the example illustrated, the Collage Domain includes three tiers, "test tier 1," "test tier 2," and "test tier 3." As shown in FIG.

Netflix, Inc. - Ex. 1002, Page 000360
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

7, in this example, each of the tiers includes two nodes. In addition, user interface 120 lists the type of software image currently deployed to application nodes for each of the tiers. In the example illustrated, image "applone (1.0.0)" is deployed to the nodes of test tier 1 and image "appltwo (1.0.0)" is deployed to the nodes of test tier 2. System administrator 20 may add one or more tiers to the domain by clicking on new tier button 122.

[0090] FIG. 8 is a screen illustration of an exemplary user interface 130 for defining properties of the tiers. In particular, user interface 130 allows system administrator 20 to input a name for the tier, a description of the tier, and an image associated with the tier. The image associated with the tier refers to a master image from which image instances are generated and deployed to the nodes assigned to the tier.

[0091] When configuring a tier, system administrator 20 may elect to activate email alerts. For example, system administrator 20 may activate the email alerts feature in order to receive email alerts providing system administrator 20 with critical and/or non-critical tier information, such as a notification that a tier has been upgraded, a node of the tier has failed or the like. Furthermore, system administrator 20 may input various policies, such node failure rules. For example, system administrator 20 may identify whether control node 12 should reboot a node in case of failure or whether the failed node should automatically be moved to maintenance pool 17. Similarly, system administrator 20 may identify whether nodes assigned to the tier may be harvested by other tiers.

[0092] User interface 130 may also allow system administrator 20 to input node requirements of a tier. In order to input node requirements of a tier, system administrator 20 may click on the "Requirements" tab 132, causing user interface 130 to present an input area to particular node requirements of the tier.

[0093] FIG. 9 is a screen illustration of an exemplary user interface 140 for viewing and identifying properties of a computing node. User interface 140 allows system administrator 20 to define a name, description, and location (including a rack and slot) of a computing node. In addition user interface 140 may specify user-defined properties of a node, such as whether the computing node has I/O HBA capabilities.

Netflix, Inc. - Ex. 1002, Page 000361
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0094] User interface 140 also displays properties that control node 12 has identified during the computing node inventory process. In this example, user interface 140 presents system administrator 20 with the a CPU node count, a CPU speed, the amount of RAM, the disk size and other characteristics that are identifiable during the automated node inventory. User interface 140 additionally presents interface information to system administrator 20. Specifically, user interface 140 provides system administrator 20 with a list of components and their associated IP and MAC addresses.

[0095] User interface 140 also allows system administrator 20 to define other custom requirements. For example, system administrator 20 may define one or more attributes and add those attributes to the list of node attributes presented to system administrator 20.

[0096] FIG. 10 is a screen illustration of an exemplary user interface 150 for viewing software images. User interface 150 presents to a system administrator or another user a list of images maintained by control node 12 within image repository 26. The image list further includes the status of each image (i.e., either active or inactive), the version of the image, the operating system on which the image should be run, the operating system version on which the image should be run and a brief description of the image.

[0097] System administrator 20 or another user may select an image by clicking on the box in front of the image identifier/name and perform one or more actions on the image. Actions that system administrator 20 may perform on an image include deleting the image, updating the image, and the like. System administrator 20 may select one of the image actions via dropdown menu 152. In some embodiments, user interface 150 may further display other details about the images such as the node to which the images are assigned (if the node status is "active"), the network address associated with the images and the like.

[0098] FIG. 11 is a screen illustration of an exemplary user interface 160 for viewing a hardware inventory report. User interface 160 presents to system administrator 20 or another user a list of the nodes that are currently assigned to a domain. System administrator 20 may elect to view the nodes for the entire domain, for a single tier within the domain or for a single rack within a tier.

Netflix, Inc. - Ex. 1002, Page 000362
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0099] For each node, user interface 160 presents a node ID, a status of the node, the tier to which the node belongs, a hostname associated with the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0100] System administrator 20 or other user may select a node by clicking on the box in front of the node identifier/name and perform one or more actions on the node. Actions that system administrator 20 may perform on the node include deleting the node, updating the node attributes or other properties of the node, and the like. System administrator 20 may select one of the node actions via dropdown menu 162.

[0101] FIG. 12 is a screen illustration of an exemplary user interface 170 for viewing discovered nodes that are located in discovered pool 11. For each node, user interface 170 presents a node ID, a state of the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0102] FIG. 13 is a screen illustration of an exemplary user interface 180 for viewing users of distributed computing system 10. User interface 180 presents a list of users as well as the role assigned to each of the users and the status of each of the users. Thus, system administrator 20 may define different roles to each of the users. For example, a user may be either an operator (i.e., general user) or an administrator. System administrator 20 may add a new user to the list of users by clicking on the "New User" button 182.

[0103] FIG. 14 is a screen illustration of an exemplary user interface 190 for viewing alerts for distributed computing system 10. For each of the alerts, user interface 190 identifies the severity of the alert, whether the alert has been acknowledged, an object associated with the alert, an event associated with the alert, a state of the alert, a user associated with the alert and a date associated with the alert.

[0104] System administrator 20 or other user may select an alert by clicking on the box in front of the logged alert and perform one or more actions on the logged

Netflix, Inc. - Ex. 1002, Page 000363
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

alert. Actions that system administrator 20 may perform include deleting the alert, changing the status of the alert, or the like. System administrator 20 may specify the log actions via dropdown menu 192.

[0105] FIG. 15 is a block diagram illustrating one embodiment of control node 12 in further detail. In the illustrated example, control node 12 includes a monitoring subsystem 202, a service level automation infrastructure (SLAI) 204, and a business logic tier (BLT) 206.

[0106] Monitoring subsystem 202 provides real-time monitoring of the distributed computing system 10. In particular, monitoring subsystem 202 dynamically collects status data 203 from the hardware and software operating within distributed computing system 10, and feeds the status data in the form of monitor inputs 208 to SLAI 204. Monitoring inputs 208 may be viewed as representing the actual state of the fabric defined for the organizational model implemented by distributed computing system 10. Monitoring subsystem 202 may utilize well defined interfaces, e.g., the Simple Network Management Protocol (SNMP) and the Java Management Extensions (JMX), to collect and export real-time monitoring information to SLAI 204.

[0107] SLAI 204 may be viewed as an automation subsystem that provides support for autonomic computing and acts as a central nervous system for the controlled fabric. In general, SLAI 204 receives monitoring inputs 208 from monitoring subsystem 202, analyzes the inputs and outputs appropriate action requests 212 to BLT 206. In one embodiment, SLAI 204 is a cybernetic system that controls the defined fabric via feedback loops. More specifically, administrator 20 may interact with BLT 206 to define an expected state 210 for the fabric. BLT 206 communicates expected state 210 to SLAI 204. SLAI 204 receives the monitoring inputs from monitoring subsystem 202 and applies rules to determine the most effective way of reducing the differences between the expected and actual states for the fabric.

[0108] For example, SLAI 204 may apply a rule to determine that a node within a high priority tier has failed and that the node should be replaced by harvesting a node from a lower priority tier. In this example, SLAI 204 outputs an action request 212 to invoke BLT 206 to move a node from one tier to the other.

Netflix, Inc. - Ex. 1002, Page 000364
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0109] In general, BLT 206 implements high-level business operations on fabrics, domains and tiers. SLAI 204 invokes BLT 206 to bring the actual state of the fabric into accordance with the expected state. In particular, BLT 206 outputs fabric actions 207 to perform the physical fabric changes. In addition, BLT 206 outputs an initial expected state 210 to SLAI 204 and initial monitoring information 214 to SLAI 204 and monitoring subsystem 202, respectively. In addition, BLT 206 outputs notifications 211 to SLAI 204 and monitoring subsystem 202 to indicate the state and monitoring changes to distributed computing system 10,. As one example, BLT 206 may provide control operations that can be used to replace failed nodes. For example, BLT 206 may output an action request indicating that a node having address 10.10.10.10 has been removed from tier ABC and a node having address 10.10.10.11 has been added to tier XYZ. In response, monitoring subsystem 202 stops attempting to collect status data 203 from node 10.10.10.10 and starts monitoring for status data from node 10.10.10.11. In addition, SLAI 204 updates an internal model to automatically associate monitoring inputs from node 10.10.10.11 with tier XYZ.

[0110] FIG. 16 is a block diagram illustrating one embodiment of monitoring subsystem 202. In general, monitoring subsystem 202 dynamically detects and monitors a variety of hardware and software components within the fabric. For example, monitoring subsystem 202 identifies, in a timely and efficient manner, any computing nodes that have failed, i.e., any node that does not respond to a request to a known service. More generally, monitoring subsystem 202 provides a concise, consistent and constantly updating view of the components of the fabric.

[0111] As described further below, monitoring subsystem 202 employs a modular architecture that allows new detection and monitoring collectors 224 to be "plugged-in" for existing and new protocols and for existing and new hardware and software. As illustrated in FIG. 16, monitoring subsystem 202 provides a plug-in architecture that allows different information collectors 224 to be installed. In general, collectors 224 are responsible for protocol-specific collection of monitoring information. The plug-in architecture allows for new protocols to be added by simply adhering to a collector plug-in signature. In this example, monitoring subsystem 202 includes collectors 224A and 224B for collecting

Netflix, Inc. - Ex. 1002, Page 000365
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

information from operating systems and applications executing on nodes within tier A and tier B, respectively.

[0112] In one embodiment, collectors 224 are loaded at startup of control node 12 and are configured with information retrieved from BLT 206. Monitoring engine 222 receives collection requests from SLAI 204, sorts and prioritizes the requests, and invokes the appropriate one of collectors 224 based on the protocol specified in the collection requests. The invoked collector is responsible for collecting the required status data and returning the status data to monitoring engine 222. If the collector is unable to collect the requested status data, the collector returns an error code.

[0113] In one embodiment, collectors 224 are Java code compiled into a jar file and loaded with a class loader at run time. Each of collectors 224 has an associated configuration file written in a data description language, such as the extensible markup language (XML). In addition, a user may interact with BLT 206 to add run-time configuration to dynamically configure collectors 224 for specific computing environments. Each of collectors 224 expose an application programming interface (API) to monitoring engine 222 for communication and data exchange.

[0114] A user, such as a system administrator, specifies the protocol or protocols to be used for monitoring a software image when the image is created. In addition, the users may specify the protocols to be used for monitoring the nodes and each service executing on the nodes. Example protocols supported by the collectors 224 include Secure Shell (SSH), Simple Network Management Protocol (SNMP), Internet Control Message Protocol (ICMP) ping, Java Management Extensions (JMX) and the Hypertext Transfer Protocol (HTTP).

[0115] Some protocols require special privileges, e.g., root privileges, to perform the required data collection. In this case, the corresponding collectors 224 communicate with a separate process that executes as the root. Moreover, some protocols may require deployment and/or configuration of data providers within the fabric. Software agents may, for example, be installed and configured on nodes and configured on other hardware. If needed, custom in-fabric components may be deployed.

Netflix, Inc. - Ex. 1002, Page 000366
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0116] In this example, the modular architecture of monitoring subsystem 202 also supports one or more plug-in interfaces 220 for data collection from a wide range of third-party monitoring systems 228. Third-party monitoring systems 228 monitor portions of the fabric and may be vendor-specific.

[0117] FIG. 17 is a block diagram illustrating one embodiment of SLAI 204 in further detail. In the illustrated embodiment, SLAI 204 is composed of three subsystems: a sensor subsystem 240, an analysis subsystem 244 and an effector subsystem 248.

[0118] In general, sensor subsystem 240 receives actual state data from monitoring subsystem 202 in the form of monitoring inputs 208 and supplies ongoing, dynamic input data to analysis subsystem 244. For example, sensor subsystem 240 is notified of physical changes to distributed computing system 10 by monitoring subsystem 202. Sensor subsystem 240 uses the state data received from monitoring subsystem 202 to maintain ongoing, calculated values that can be sent to analysis subsystem 244 in accordance with scheduler 242.

[0119] In one embodiment, sensor subsystem 240 performs time-based hierarchical data aggregation of the actual state data in accordance with the defined organization model. Sensor subsystem 240 maintains organizational data in a tree-like structure that reflects the current configuration of the hierarchical organization model. Sensor subsystem 240 uses the organizational data to perform the real-time data aggregation and map tiers and domains to specific nodes. Sensor subsystem 240 maintains the organizational data based on notifications 211 received from BLT 206.

[0120] Sensor subsystem 240 sends inputs to analysis subsystem 244 to communicate the aggregated data on a periodic or event-driven basis. Analysis subsystem 244 may register an interest in a particular aggregated data value with sensor subsystem 240 and request updates at a specified frequency. In response, sensor subsystem 240 interacts with monitoring subsystem 202 and scheduler 242 to generate the aggregated data required by analysis subsystem 244.

[0121] Sensor subsystem 240 performs arbitrary data aggregations via instances of plug-in classes (referred to as "triggers") that define the aggregations. Each trigger is registered under a compound name based on the entity being monitored and the

Netflix, Inc. - Ex. 1002, Page 000367
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

type of data being gathered. For example, a trigger may be defined to aggregate and compute an average computing load for a tier every five minutes. Analysis subsystem 244 requests the aggregated data based on the registered names. In some embodiments, analysis subsystem 244 may define calculations directly and pass them to sensor subsystem 240 dynamically.

[0122] Analysis subsystem 244 is composed of a plurality of forward chaining rule engines 246A-246N. In general, rule engines 246 match patterns in a combination of configuration data and monitoring data, which is presented by extraction agent 251 in the form of events. Events contain the aggregated data values that are sent to rule engines 246 in accordance with scheduler 242.

[0123] Sensor subsystem 240 may interact with analysis subsystem 244 via trigger listeners 247 that receives updates from a trigger within sensor subsystem 240 when specified events occur. An event may be based on system state (e.g., a node transitioning to an up or down state) or may be time based.

[0124] Analysis subsystem 244 allows rule sets to be loaded in source form and compiled at load time into discrimination networks. Each rule set specifies trigger-delivered attributes. Upon loading the rule sets, analysis subsystem 244 establishes trigger listeners 247 to receive sensor notifications and update respective working memories of rule engines 246. As illustrated in FIG. 17, each of rule engines 246 may serve a different tier defined within the fabric. Alternatively, multiple rule engines 246 may serve a single tier or a single rule engine may serve multiple tiers.

[0125] Rule engines 246 process the events and invoke action requests via calls to effector subsystem 248. In addition, rule engines 246 provide a call-back interface so that effector subsystem 248 can inform a rule engine when an action has completed. Rule engines 246 prevent a particular rule from re-firing as long as any action invoked by the rule has not finished. In general, rules contain notification calls and service invocations though either may be disabled by configuration of effector subsystem 248. BLT 206 supplies initial system configuration descriptions to seed each of rule engines 246.

[0126] In general, rule engines 246 analyze the events and discover discrepancies between an expected state of the fabric and an actual state. Each of rule engines 246 may be viewed as software that performs logical reasoning using knowledge

Netflix, Inc. - Ex. 1002, Page 000368
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

encoded in high-level condition-action rules. Each of rule engines 246 applies automated reasoning that works forward from preconditions to goals defined by system administrator 20. For example, rule engines 246 may apply modus ponens inferences rules.

[0127] Rule engines 246 output requests to effector subsystem 248 which produce actions requests 212 for BLT 206 to resolve the discrepancies. Effector subsystem 248 performs all operations on behalf of analysis subsystem 244. For example, event generator 250, task invocation module 252 and logger 254 of effector subsystem 248 perform event generation, BLT action invocation and rule logging, respectively. More specifically, task invocation module 252 invokes asynchronous operations within BLT 206. In response, BLT 206 creates a new thread of control for each task which is tracked by a unique task identifier (task id). Rules engine 246 uses the task id to determine when a task completes and, if needed, to re-fire any rules that were pended until completion of the task. These tasks may take arbitrary amounts of time, and rules engine 246 tracks the progress of individual task via change notifications 211 produced by BLT 206.

[0128] Event generator 250 creates persistent event records of the state of processing of SLAI 204 and stores the event records within a database. Clients uses these event records to track progress and determine the current state of the SLAI 204.

[0129] Logger 254 generates detailed trace information about system activities for use in rule development and debugging. The logging level can be raised or lowered as needed without changing operation of SLAI 204.

[0130] FIG. 18 is a block diagram of an example working memory 270 associated with rule engines 246. In this example, working memory 270 includes a read-only first data region 272 that stores the expected state received from BLT 206. Data region 272 is read-only in the sense that it cannot be modified in response to a trigger from sensor subsystem 240 or by rule engines 246 without notification from BLT 206.

[0131] In addition, working memory 270 includes a second data region 274 that is modifiable (i.e., read/write) and may be updated by monitoring subsystem 202 or used internally by rule engines 246. In general, data region 274 stores aggregated

Netflix, Inc. - Ex. 1002, Page 000369
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

data representing the actual state of the fabric and can be updated by sensor subsystem 240 or by rule engines 246. The actual state may consist of a set of property annotations that can be attached to objects received from BLT 206 or to objects locally defined within a rule engine, such as local object 276.

[0132] FIG. 19 is a block diagram illustrating an example embodiment for BLT 206. In this example, BLT 206 includes a set of one or more web service definition language (WSDL) interfaces 300, a report generator 302, a fabric administration interface service 304, a fabric view service 306, a user administration service 308, a task interface 311, a task manager 312 and an event subsystem 315.

[0133] As described, BLT 206 provides the facilities necessary to create and administer the organizational model (e.g., fabric, domains, tiers and nodes) implemented by distributed computing system 10. In general, BLT 206 abstracts access to the persisted configuration state of the fabric, and controls the interactions with interfaces to fabric hardware services. As such, BLT 206 provides fabric management capabilities, such as the ability to create a tier and replace a failed node. WSDL interfaces 300 provide web service interfaces to the functionality of BLT 206 that may be invoked by web service clients 313. Many of WSDL interfaces 300 offered by BLT 206 allow administrator 20 to define goals, such as specifying a goal of the expected state of the fabric. As further described below, rule engines 246 within SLAI 204, in turn, invoke task manger 312 to initiate one or more BLT tasks to achieve the specified goal. In general, web service clients 313 may be presentation layer applications, command line applications, or other clients.

[0134] BLT 206 abstracts all interaction with physical hardware for web service clients 313. BLT 206 is an enabling component for autonomic management behavior, but does not respond to real-time events that either prevent a goal from being achieved or produce a set of deviations between the expected state and the actual state of the system. In contrast, BLT 206 originates goals for autonomic reactions to changing configuration and state. SLAI 204 analyzes and acts upon these goals along with real-time state changes. BLT 206 sets the goals to which

Netflix, Inc. - Ex. 1002, Page 000370
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

SLAI 204 strives to achieve, and provides functionality used by the SLAI in order to achieve the goals.

[0135] In general, BLT 206 does not dictate the steps taken in pursuit of a goal since these are likely to change based on the current state of distributed computing system 10 and changes to configurable policy. SLAI 204 makes these decisions based on the configured rule sets for the fabric and by evaluating monitoring data received from monitoring subsystem 202.

[0136] Fabric administration service 304 implements a set of methods for managing all aspects of the fabric. Example methods include methods for adding, viewing, updating and removing domains, tiers, nodes, notifications, assets, applications, software images, connectors, and monitors. Other example methods include controlling power at a node, and cloning, capturing, importing, exporting or upgrading software images. Rule engines 246 of SLAI 204 may, for example, invoke these methods by issuing action requests 212.

[0137] Task manager 312 receives action requests 212 via task interface 311. In general, , task interface 311 provides an interface for receiving action requests 212 from SLAI 204 or other internal subsystem. In response, task manager 312 manages asynchronous and long running actions and are invoked by SLAI 204 to satisfy a goal or perform an action requested by a client.

[0138] Task manager 312 generates task data 310 that represents identification and status for each task. Task manager 312 returns a task identifier to the calling web service clients 313 or the internal subsystem, e.g., SLAI 204, that initiated the task. Rule engines 246 and web service clients 313 use the task identifiers to track progress and retrieve output, results, and errors associated with achieving the goal.

[0139] In one embodiment, there are no WSDL interfaces 300 for initiating specific tasks. Rather, administrator 20 interacts with BLT 206 though goal interfaces presented by WSDL interfaces 300 to define the goals for the fabric. In contrast, the term task is used to refer to internal system constructs that require no user interaction. Tasks are distinct, low-level units of work that affect the state of the fabric. SLAI 204 may combine tasks to achieve or maintain a goal state.

[0140] For example, administrator 20 can request configuration changes by either adding new goals to an object or by modifying the attributes on existing goals.

Netflix, Inc. - Ex. 1002, Page 000371
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Scheduled goals apply a configuration at a designated time. For example, the goals for a particular tier may specify the minimum, maximum, and target node counts for that tier. As a result, the tier can increase or decrease current node capacity by scheduling goals with different configuration values.

[0141] This may be useful, for example, in scheduling a software image upgrade. As another example, entire domains may transition online and offline per a defined grid schedule. Administrator 20 may mix and match goals on a component to achieve configurations specific to the application and environment. For example, a tier that does not support autonomic node replacement would not be configured with a harvesting goal.

[0142] In some embodiments, goals are either "in force" or "out of force." SLAI 204 only works to achieve and maintain those goals that are currently in force. SLAI 204 may applies a concept of "gravity" as the goals transition from in force to out of force. For example, SLAI 204 may transition a tier offline when an online goal is marked out of force. Some goal types may have prerequisite goals. For example, an image upgrade goal may require as a prerequisite that a tier be transitioned to offline before the image upgrade can be performed. In other embodiments, goals are always in force until modified.

[0143] SLAI 204 may automatically formulate dependencies between goals or may allow a user to specify the dependencies. For example, a user may request that a newly created tier come online. As a result of this goal, SLAI 204 may automatically direct task manager 312 to generate a task of harvesting a target number of nodes to enable the tier. Generally, all goals remain in-force by SLAI 204 until modified by BLT 206. In one embodiment, each goal remains in-force in one of three states: Satisfied, Warning, or Critical depending on how successful SLAI 204 was in achieving the goal at the time the event record was generated and stored.

[0144] In this manner, SLAI 204 controls the life cycle of a goal (i.e., the creation, scheduling, update, deletion of the goal), and provides a common implementation of these and other services such as timeout, event writing, goal conflicts, management of intra-goal dependencies, and tracking tasks to achieving the goals.

Netflix, Inc. - Ex. 1002, Page 000372
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0145] Progress toward a goal is tracked though event subsystem 315. In particular, event subsystem 315 tracks the progress of each in force goal based on the goal identifiers. Tasks executed to achieve a particular goal produce events to communicate result or errors. The events provide a convenient time-based view of all actions and behaviors.

[0146] Examples of goal types that may be defined by administrator 20 include software image management goals, node allocation goals, harvest goals, tier capacity goals, asset requirement goals, tier online / offline goals, and data gathering goals.

[0147] In one embodiment, BLT 206 presents a task interface to SLAI 204 for the creation and management of specific tasks in order to achieve the currently in force goals. In particular, rule engines 246 invoke the task interface based on evaluation of the defined rule sets in view of the expected state and actual state for the fabric. Example task interfaces include interfaces to: reserve node resources; query resources for a node slot; associate or disassociate an image with a node in a tier node slot; allocate, de-allocate, startup or shutdown a node; move a node to a tier; apply, remove or cycle power of a node; create a golden image; create or delete an image instance; and delete an activity, node or tier.

[0148] Report generator 302 provides an extensible mechanism for generating reports 314. Typical reports include image utilization reports that contain information with respect to the number of nodes running each software image, inventory reports detailing both the logical and physical aspects of the fabric, and system event reports showing all events that have occurred within the fabric. Report generator 302 gathers, localizes, formats and displays data into report form for presentation to the user. Report generator 302 may include one or more data gathering modules (not shown) that gather events in accordance with a schedule and update an events table to record the events. The data gathering modules may write the events in XML format.

[0149] FIG. 20 is a block diagram illustrating one embodiment of a rule engine 246 (FIG. 17). In the illustrated embodiment, rule engine 246 includes a rule compiler 344 and an execution engine 346. Each of rules 342 represents a unit of code that conforms to a rule language and expresses a set of triggering conditions

Netflix, Inc. - Ex. 1002, Page 000373
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

and a set of implied actions. When the conditions are met, the actions are eligible to occur. The following is one example of a rule:

```
rule checkTierLoad {
      Tier t where status != "overloaded";
      LoadParameter p where app == t.app && maxload <
      t.load;
} -> {
      modify t {
            status: "overloaded";
      };
}
```

When translated, this example rule marks a tier as overloaded if an application is implemented by the tier and the maximum specified load for the application has been exceeded. Another example rule for outputting a notification that a tier is overloaded and automatically invoking a task within BLT 206 to add a node is:

```
rule tierOverloadNotify {
      Tier t where status == "overloaded";
} -> {
      notify "Tier: " + t + "is overloaded.";
      BLT.addNode(f);
}
```

[0150] Rule compiler 344 compiles each of rules 344 and translates match conditions of the rules into a discrimination network that avoids redundant tests during rule execution. Execution engine 346 handles rule administration, object insertion and retrieval, rule invocation and execution of rule actions. In general, execution engine 346 first matches a current set of rules 342 against a current state of working memory 348 and local objects 350. Execution engine 346 then collects all rules that match as well as the matched objects and selects a particular rule instantiation to fire. Next, execution engine 346 fires (executes) the instantiated rule and propagates any changes to working memory 348. Execution engine 346 repeats the process until no more matching rule instantiations can be found.

[0151] Firing of a rule typically produces a very small number of changes to working memory 348. This allows sophisticated rule engines to scale by retaining match state between cycles. Only the rules and rule instantiations affected by changes get updated, thereby avoiding the bulk of the matching process. One

Netflix, Inc. - Ex. 1002, Page 000374
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

exemplary algorithm that may be used by execution engine 346 to handle the matching process includes the RETE algorithm that creates a decision tree that combines the patterns in all the rules and is intended to improve the speed of forward-chained rule system by limiting the effort required to re-compute a conflict set after a rule is fired. One example of a RETE algorithm is described in Forgy, C.L.: 1982, 'RETE: a fast algorithm for the many pattern/many object pattern match problem'. Artificial Intelligence 19, 1737. Other alternatives include the TREAT algorithms, and LEAPS algorithm, as described by Miranker, D.P.: 'TREAT: A New and Efficient Match Algorithm for AI Production Systems'. ISBN 0934613710 Daniel P. Miranker, David A. Brant, Bernie Lofaso, David Gadbois: On the Performance of Lazy Matching in Production Systems. AAAI 1990: 685692.

[0152] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.

Netflix, Inc. - Ex. 1002, Page 000375
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS:

1. A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network; and

a control node that includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules.

2. The distributed computing system of claim 1, wherein the rule engines are forward-chaining rule engines.

3. The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

4. The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

5. The distributed computing system of claim 4, wherein the monitoring subsystem comprises a set of modular information collectors that collect the status data from the application nodes.

Netflix, Inc. - Ex. 1002, Page 000376
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

6. The distributed computing system of claim 5, wherein each of the information collectors conforms to a different communication protocol.

7. The distributed computing system of claim 5, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

8. The distributed computing system of claim 5, wherein the automation unit registers collection requests with the monitoring subsystem, and the monitoring subsystem includes a monitoring engine that invokes the collectors in response to the collection requests.

9. The distributed computing system of claim 4, wherein the automation unit further comprises a sensor subsystem that receives the state data from the monitoring subsystem and aggregates the state data in accordance with a hierarchical model of deployment of the application nodes.

10. The distributed computing system of claim 4, wherein the sensor subsystem communicates the aggregated data to the rule engines in accordance with a schedule.

11. The distributed computing system of claim 4, an effector subsystem that outputs action requests to the business logic tier in response to application of the rules by the rule engines.

12. The distributed computing system of claim 4, wherein the business logic tier includes a set of web services to control the application nodes.

13. The distributed computing system of claim 4, wherein the business logic tier presents interfaces by which a user defines one or more goals for the expected state of the application nodes.

Netflix, Inc. - Ex. 1002, Page 000377
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

14. The distributed computing system of claim 13, wherein the business logic tier registers the goals with the automation unit for achievement by the rule engines in accordance with the rule set.

15. The distributed computing system of claim 13, wherein the automation unit automatically formulates dependencies between the goals

16. The distributed computing system of claim 13,

wherein the business logic tier further comprises a task manager to invoke tasks in response to requests from the automation unit, and

wherein the business logic tier includes an event subsystem that tracks the progress of each of the tasks.

17. The distributed computing system of claim 13, wherein the goals include at least one of software image management goals, node allocation goals, goals for harvesting nodes, tier capacity goals, asset requirement goals and goals for gathering the status data.

18. The distributed computing system of claim 13, wherein the rules are units of code that conform with a rule language.

19. The distributed computing system of claim 4, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines automatically produce action requests to configure the application nodes in accordance with a hierarchical organization defined by the model.

Netflix, Inc. - Ex. 1002, Page 000378
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

20. The distributed computing system of claim 19, wherein each tier represents a different computing function.

21. A method comprising:

receiving input that defines an expected state for a distributed computing system having a plurality of application nodes interconnected via a communications network;

receiving status data that represents an actual state for the distributed computing system;

processing rules with a set of rule engines to determine operations for reducing any difference between the actual state and the expected state; and

applying the operations to the distributed computing system to control the application nodes in accordance with the rules.

22. The method of claim 21, wherein processing rules comprises processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

23. The method of claim 21, further comprising collecting the status data from the application nodes with a plurality of modular information collectors.

24. The method of claim 21, further comprising:

registering collection requests with a monitoring subsystem; and

executing a monitoring engine that invokes the collectors in response to the collection requests.

25. The method of claim 21, further comprising:

aggregating the state data in accordance with a hierarchical model of deployment of the application nodes; and

processing the aggregated state data with the rule engines.

Netflix, Inc. - Ex. 1002, Page 000379
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

26. The method of claim 21, further comprising:

presenting an interface by which a user defines the expected state as one or more goals; and

registering the goals with the rule engines.

27. The method of claim 26, further comprising:

storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

automatically producing the operations with the rule engines to configure the application nodes in accordance with a hierarchical organization defined by the model.

28. A computer-readable medium comprising instructions that cause a programmable processor to:

process rules with a set of forward-chaining rule engines to determine operations for reducing differences between an actual state of a distributed computing system and an expected state; and

apply the operations to the distributed computing system to provide autonomic control of the application nodes in accordance with the rules.

Netflix, Inc. - Ex. 1002, Page 000380
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

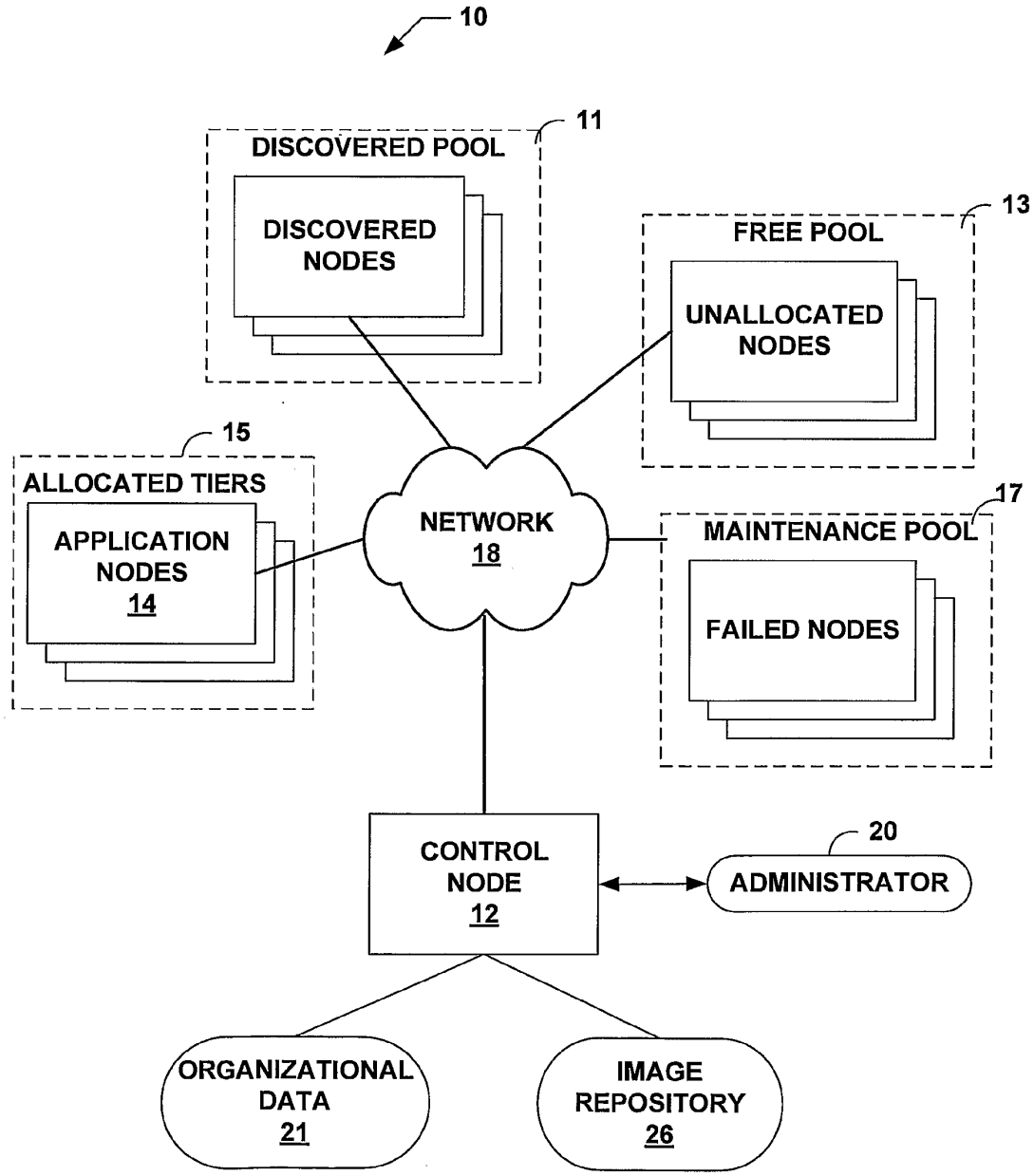

10
11
DISCOVERED POOL
DISCOVERED NODES
13
FREE POOL
UNALLOCATED NODES
15
ALLOCATED TIERS
APPLICATION NODES 14
NETWORK 18
17
MAINTENANCE POOL
FAILED NODES
CONTROL NODE 12
20
ADMINISTRATOR
ORGANIZATIONAL DATA 21
IMAGE REPOSITORY 26

FIG. 1

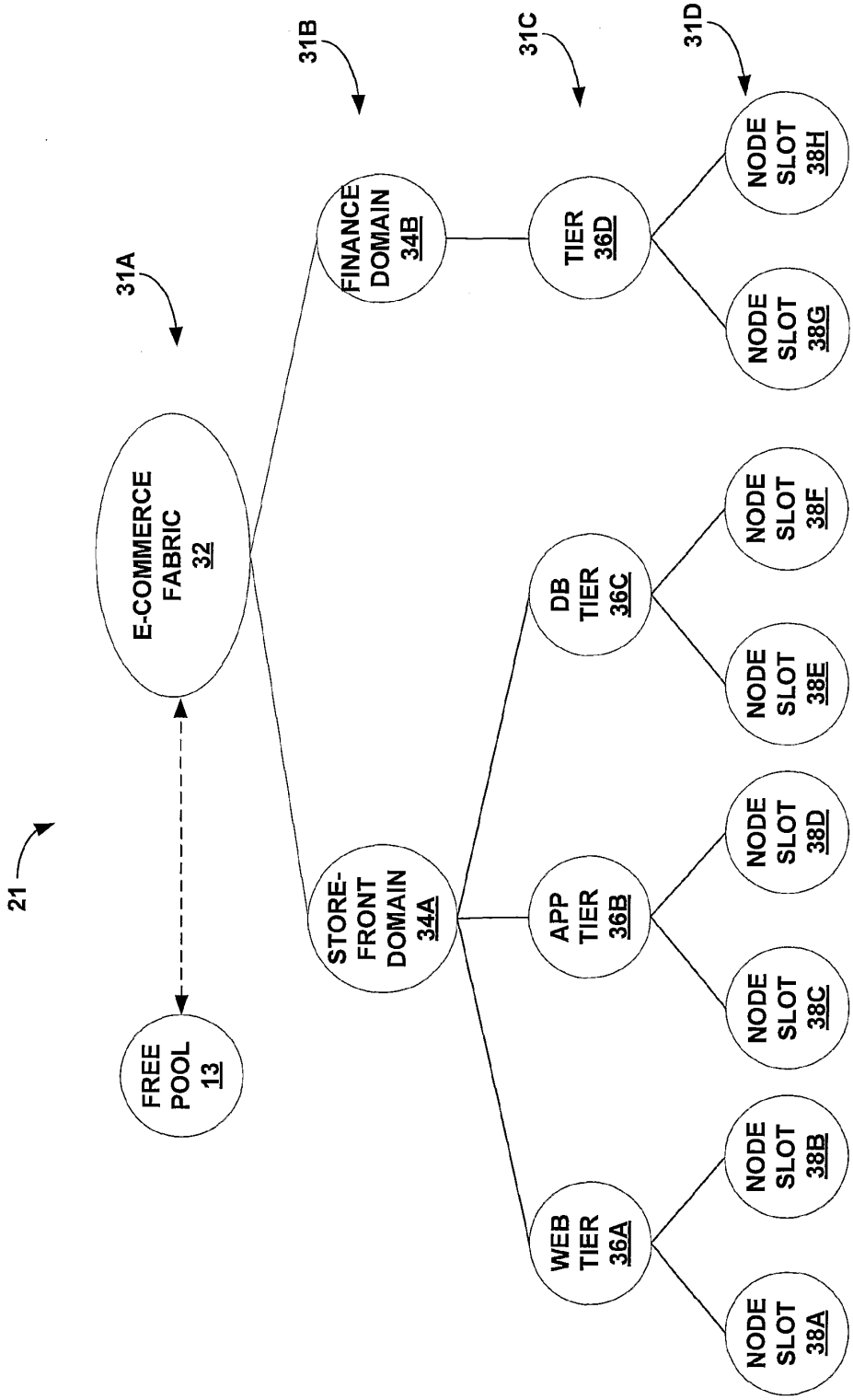

21
E-COMMERCE FABRIC 32
FREE POOL 13
31A
31B
31C
31D
FINANCE DOMAIN 34B
STORE-FRONT DOMAIN 34A
TIER 36D
DB TIER 36C
APP TIER 36B
WEB TIER 36A
NODE SLOT 38H
NODE SLOT 38G
NODE SLOT 38F
NODE SLOT 38E
NODE SLOT 38D
NODE SLOT 38C
NODE SLOT 38B
NODE SLOT 38A

FIG. 2

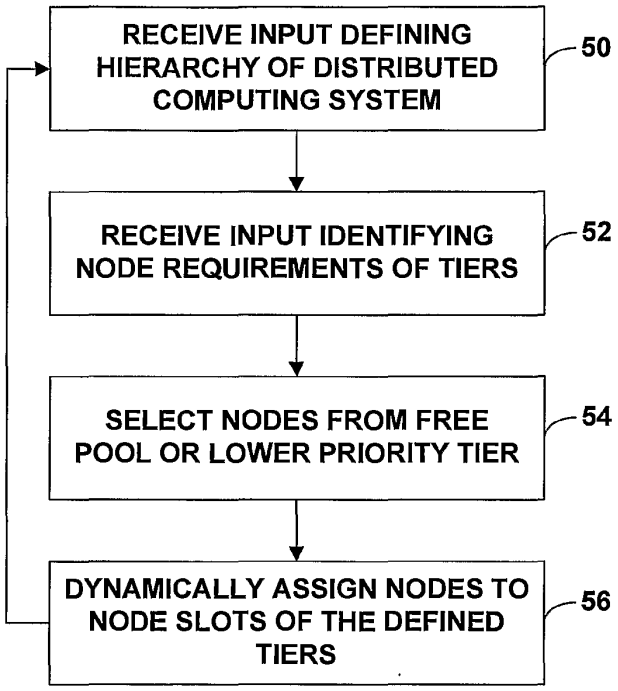

RECEIVE INPUT DEFINING HIERARCHY OF DISTRIBUTED COMPUTING SYSTEM
50
RECEIVE INPUT IDENTIFYING NODE REQUIREMENTS OF TIERS
52
SELECT NODES FROM FREE POOL OR LOWER PRIORITY TIER
54
DYNAMICALLY ASSIGN NODES TO NODE SLOTS OF THE DEFINED TIERS
56

FIG. 3

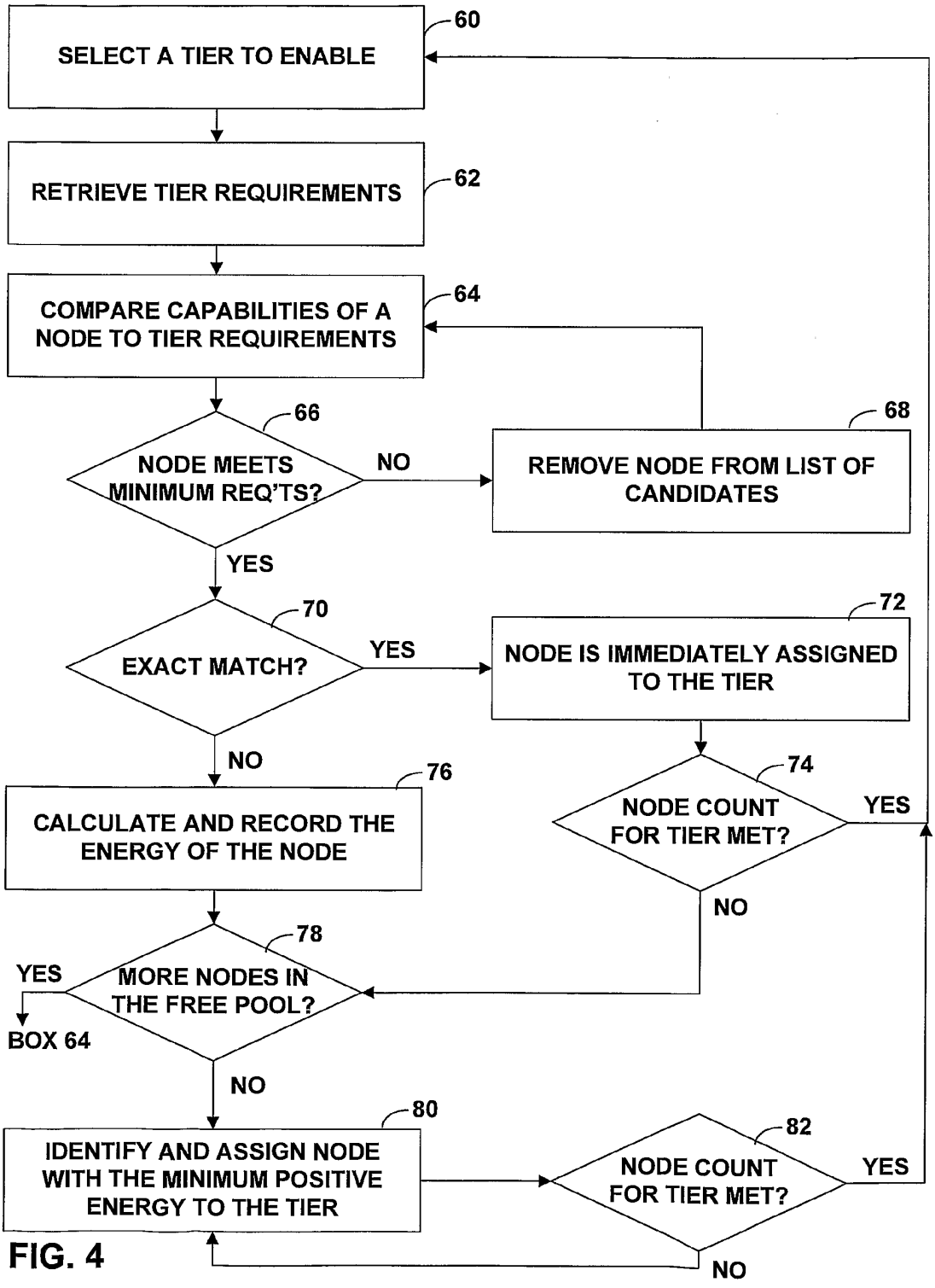

SELECT A TIER TO ENABLE
60
RETRIEVE TIER REQUIREMENTS
62
COMPARE CAPABILITIES OF A NODE TO TIER REQUIREMENTS
64
66
NODE MEETS MINIMUM REQ'TS?
NO
68
REMOVE NODE FROM LIST OF CANDIDATES
YES
70
EXACT MATCH?
YES
72
NODE IS IMMEDIATELY ASSIGNED TO THE TIER
NO
74
NODE COUNT FOR TIER MET?
YES
NO
76
CALCULATE AND RECORD THE ENERGY OF THE NODE
78
MORE NODES IN THE FREE POOL?
YES
BOX 64
NO
80
IDENTIFY AND ASSIGN NODE WITH THE MINIMUM POSITIVE ENERGY TO THE TIER
82
NODE COUNT FOR TIER MET?
YES
NO

FIG. 4

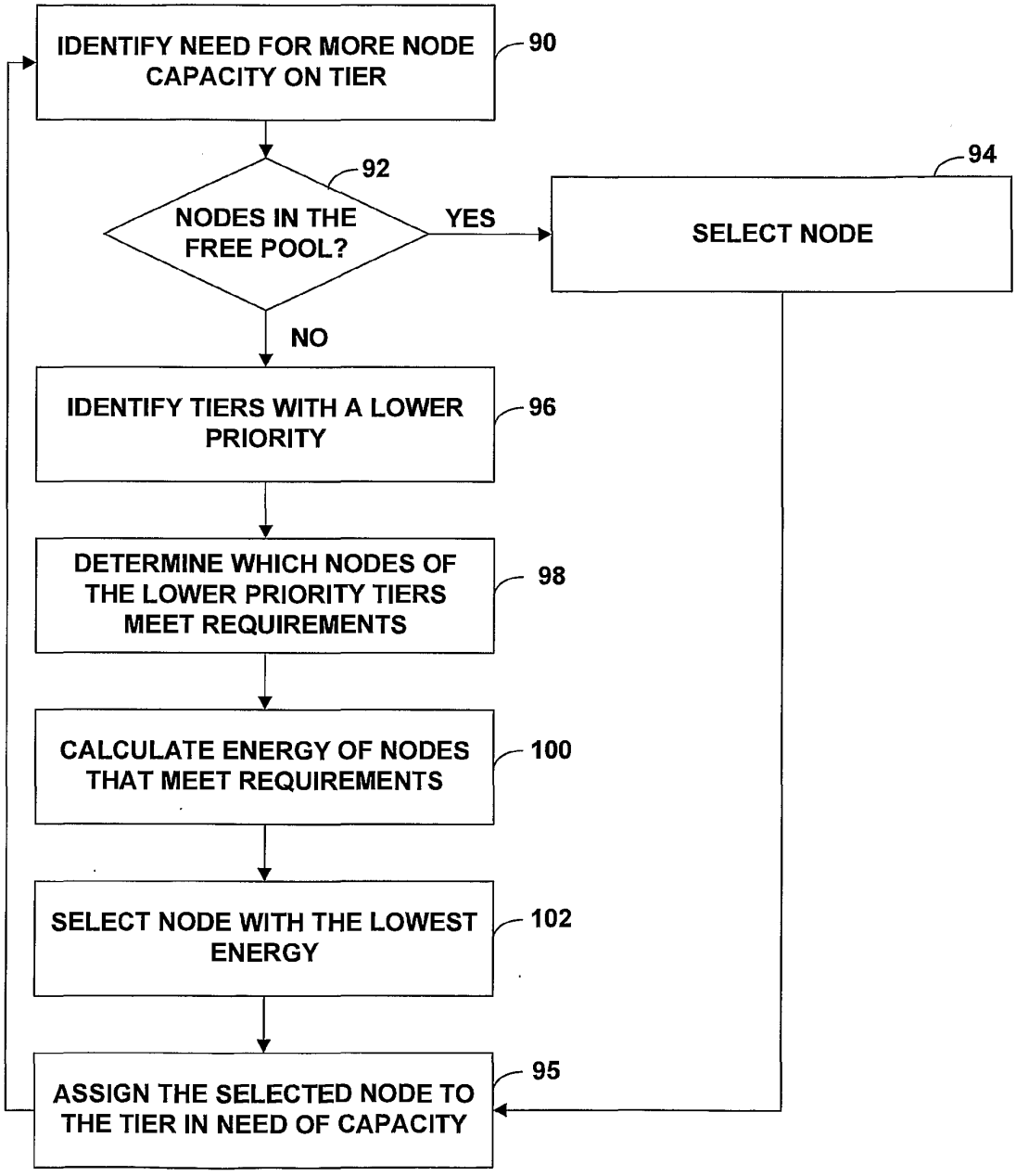

IDENTIFY NEED FOR MORE NODE CAPACITY ON TIER
90
92
NODES IN THE FREE POOL?
YES
94
SELECT NODE
NO
IDENTIFY TIERS WITH A LOWER PRIORITY
96
DETERMINE WHICH NODES OF THE LOWER PRIORITY TIERS MEET REQUIREMENTS
98
CALCULATE ENERGY OF NODES THAT MEET REQUIREMENTS
100
SELECT NODE WITH THE LOWEST ENERGY
102
ASSIGN THE SELECTED NODE TO THE TIER IN NEED OF CAPACITY
95

FIG. 5

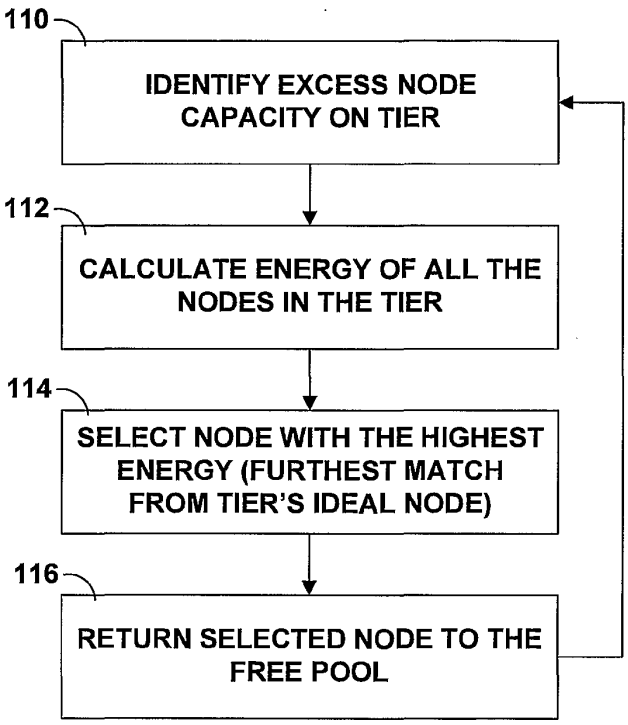

110
IDENTIFY EXCESS NODE CAPACITY ON TIER
112
CALCULATE ENERGY OF ALL THE NODES IN THE TIER
114
SELECT NODE WITH THE HIGHEST ENERGY (FURTHEST MATCH FROM TIER'S IDEAL NODE)
116
RETURN SELECTED NODE TO THE FREE POOL

FIG. 6

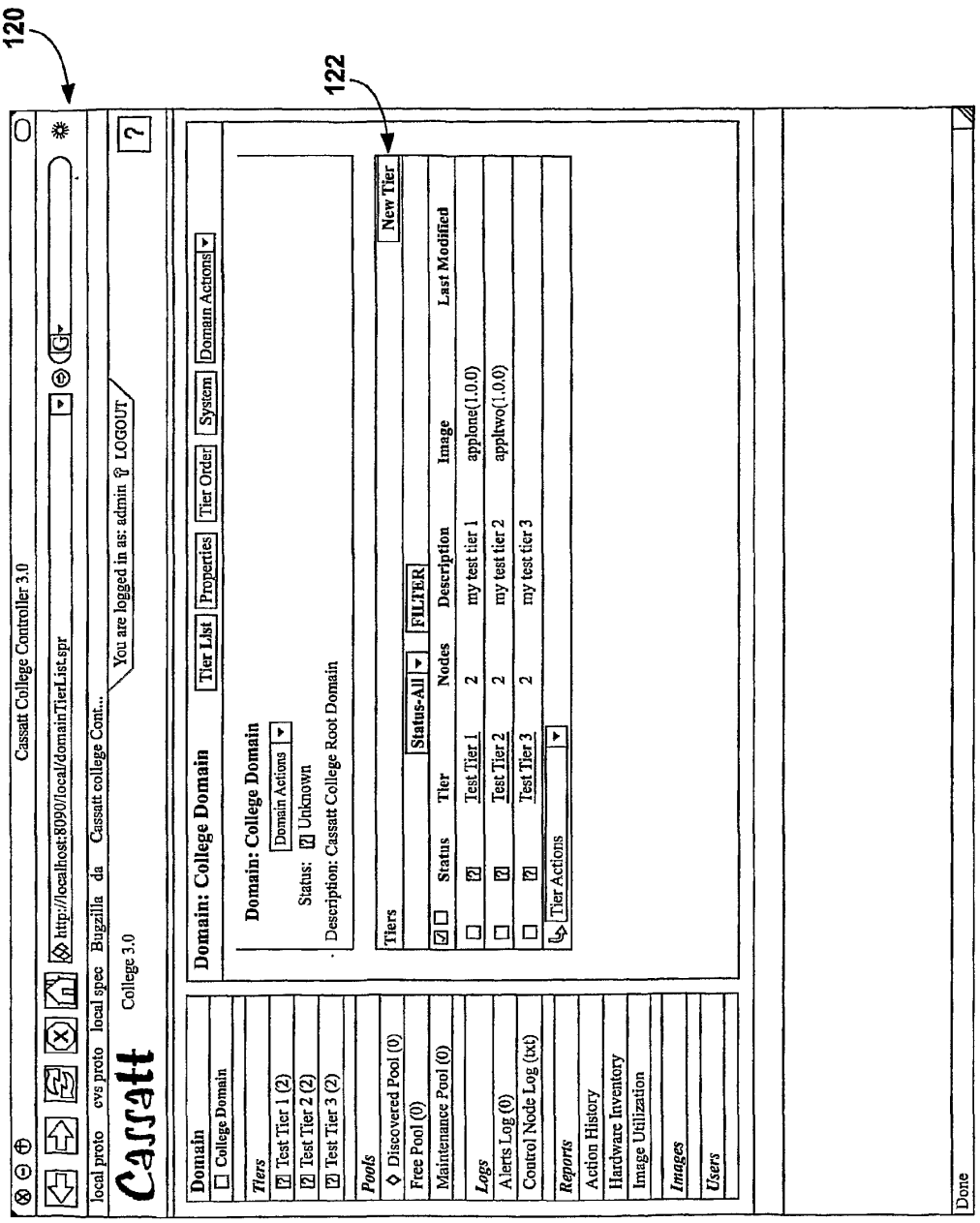


FIG. 7

SUBSTITUTE SHEET (RULE 26)

Netflix, Inc. - Ex. 1002, Page 000387
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

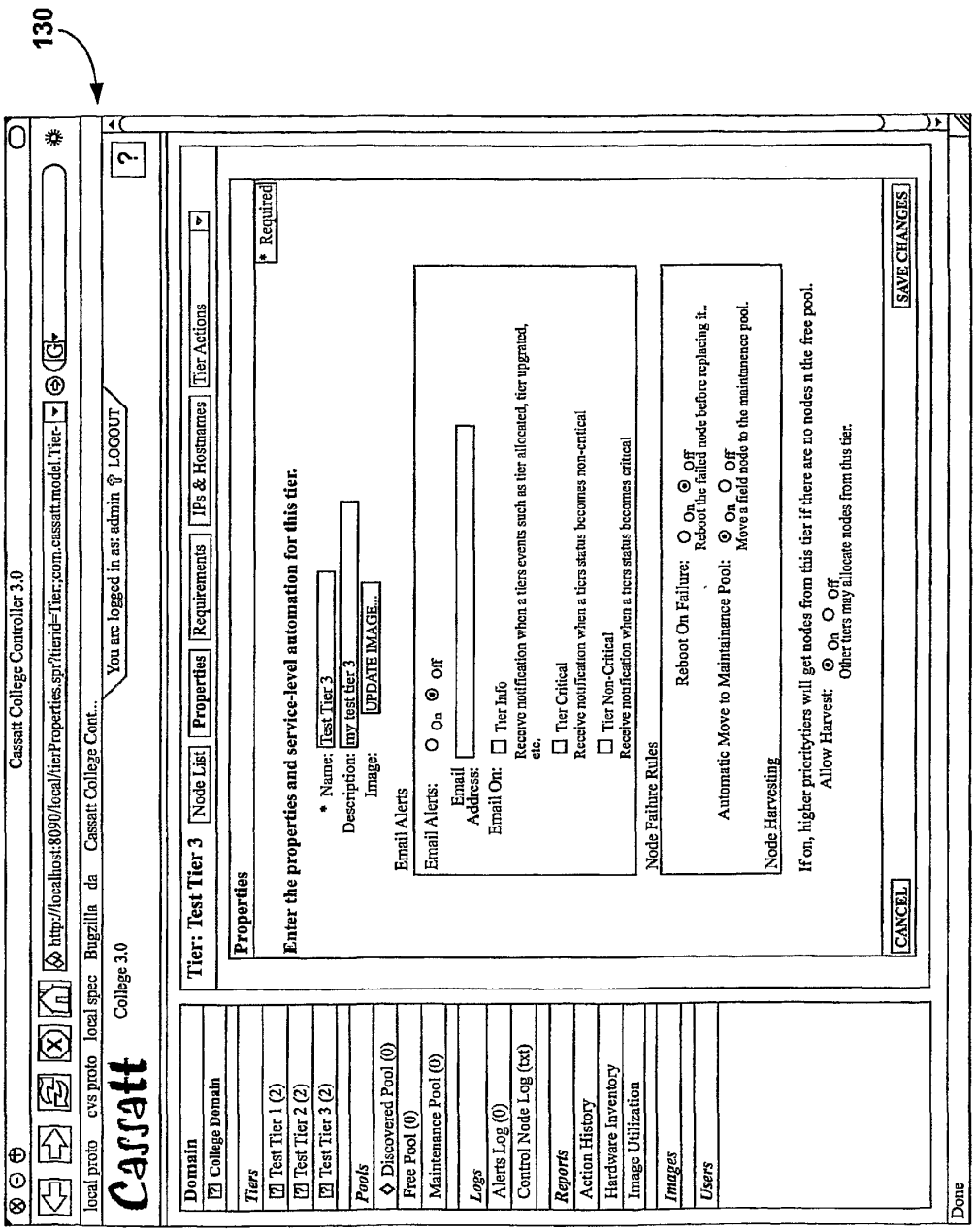


FIG. 8

SUBSTITUTE SHEET (RULE 26)

Netflix, Inc. - Ex. 1002, Page 000388
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

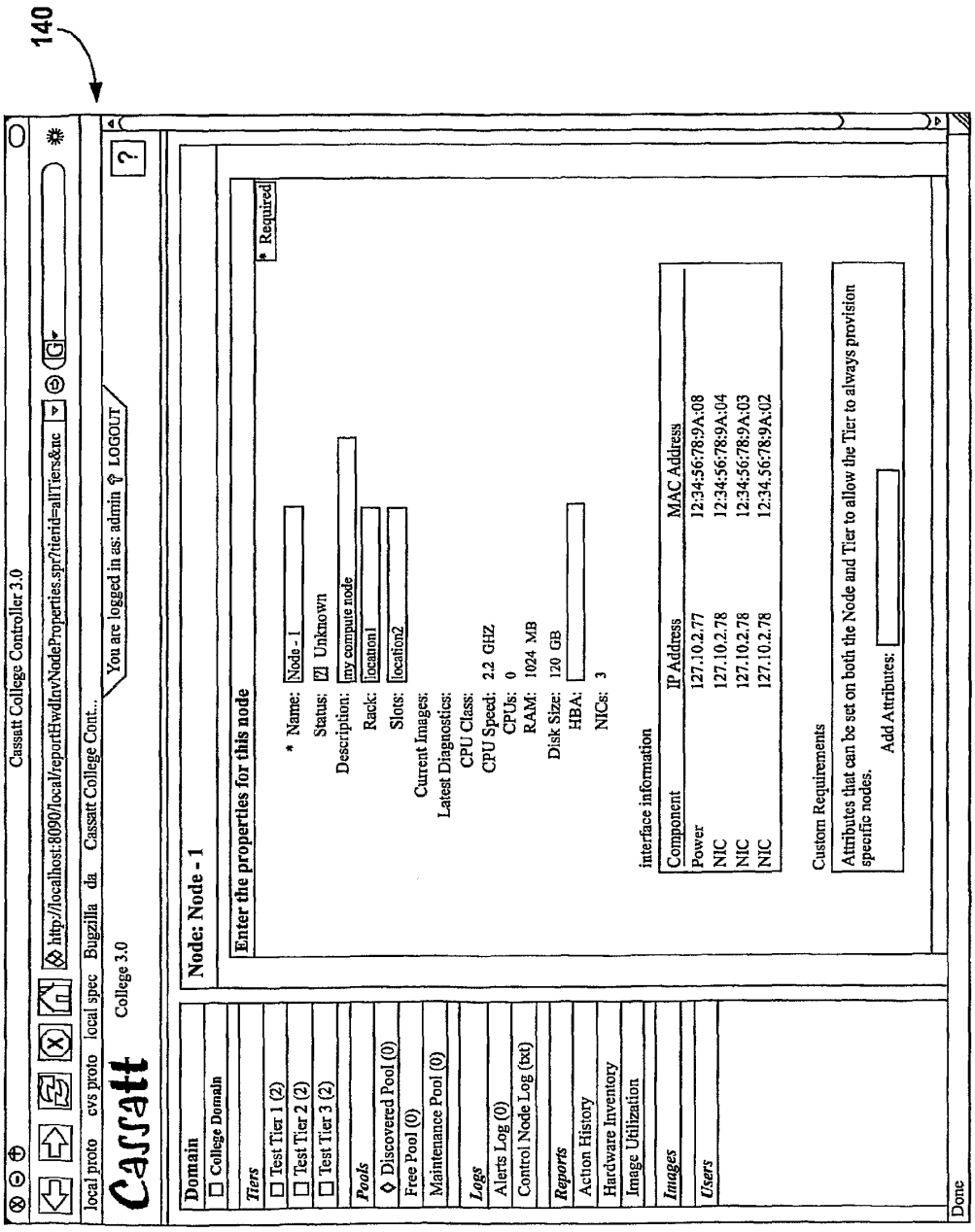


FIG. 9

Netflix, Inc. - Ex. 1002, Page 000389
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

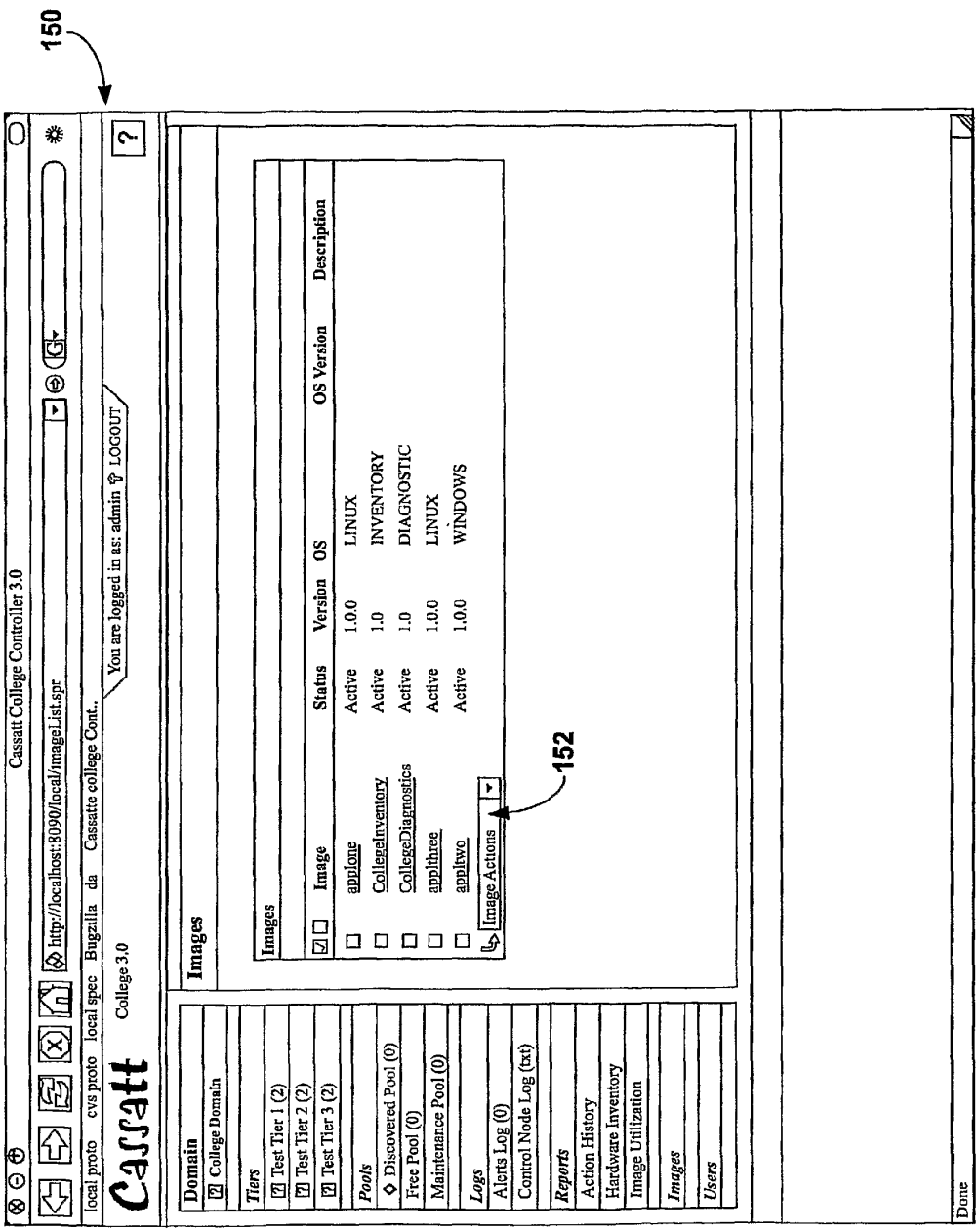


FIG. 10

SUBSTITUTE SHEET (RULE 26)

Netflix, Inc. - Ex. 1002, Page 000390
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

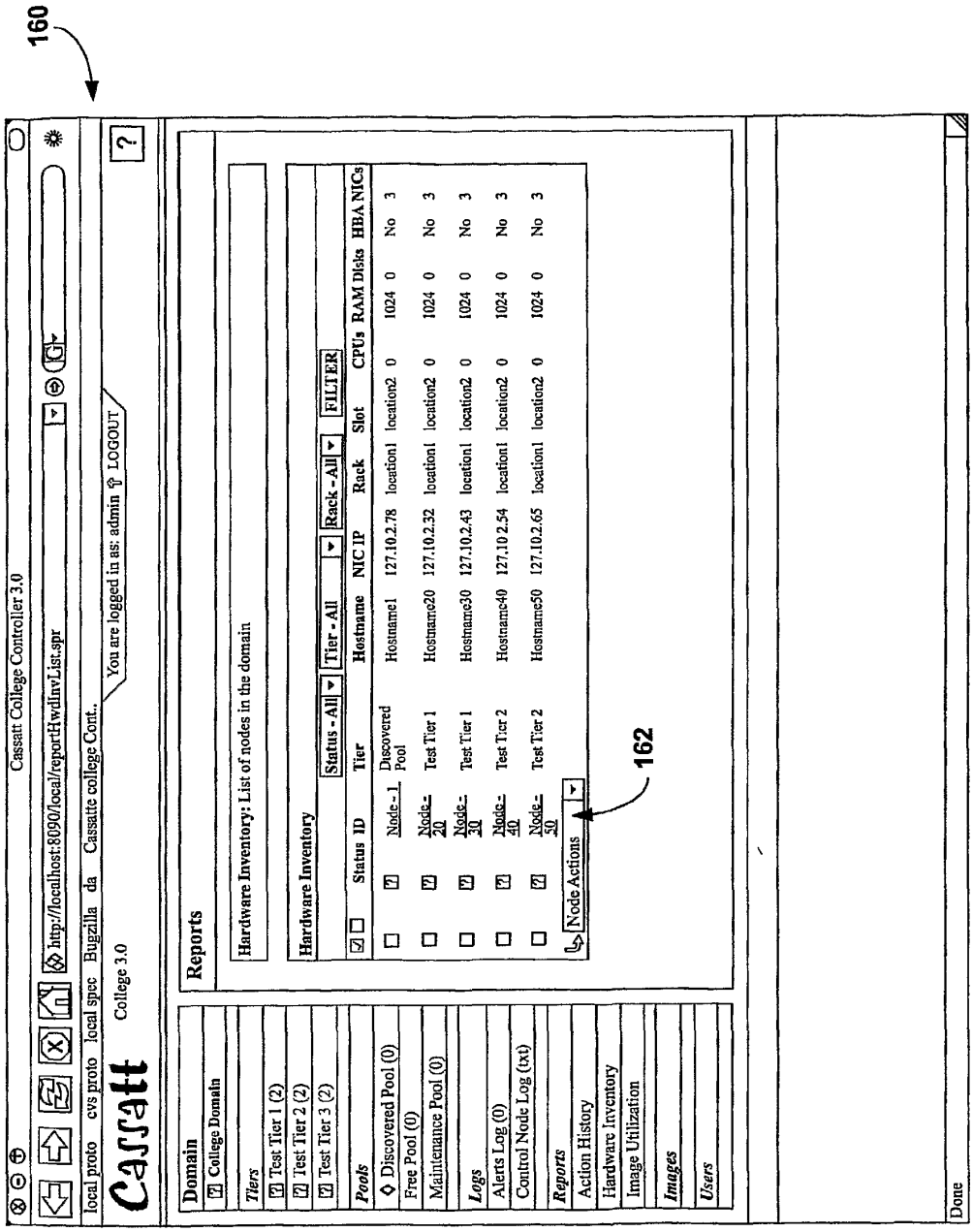


FIG. 11

Netflix, Inc. - Ex. 1002, Page 000391
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

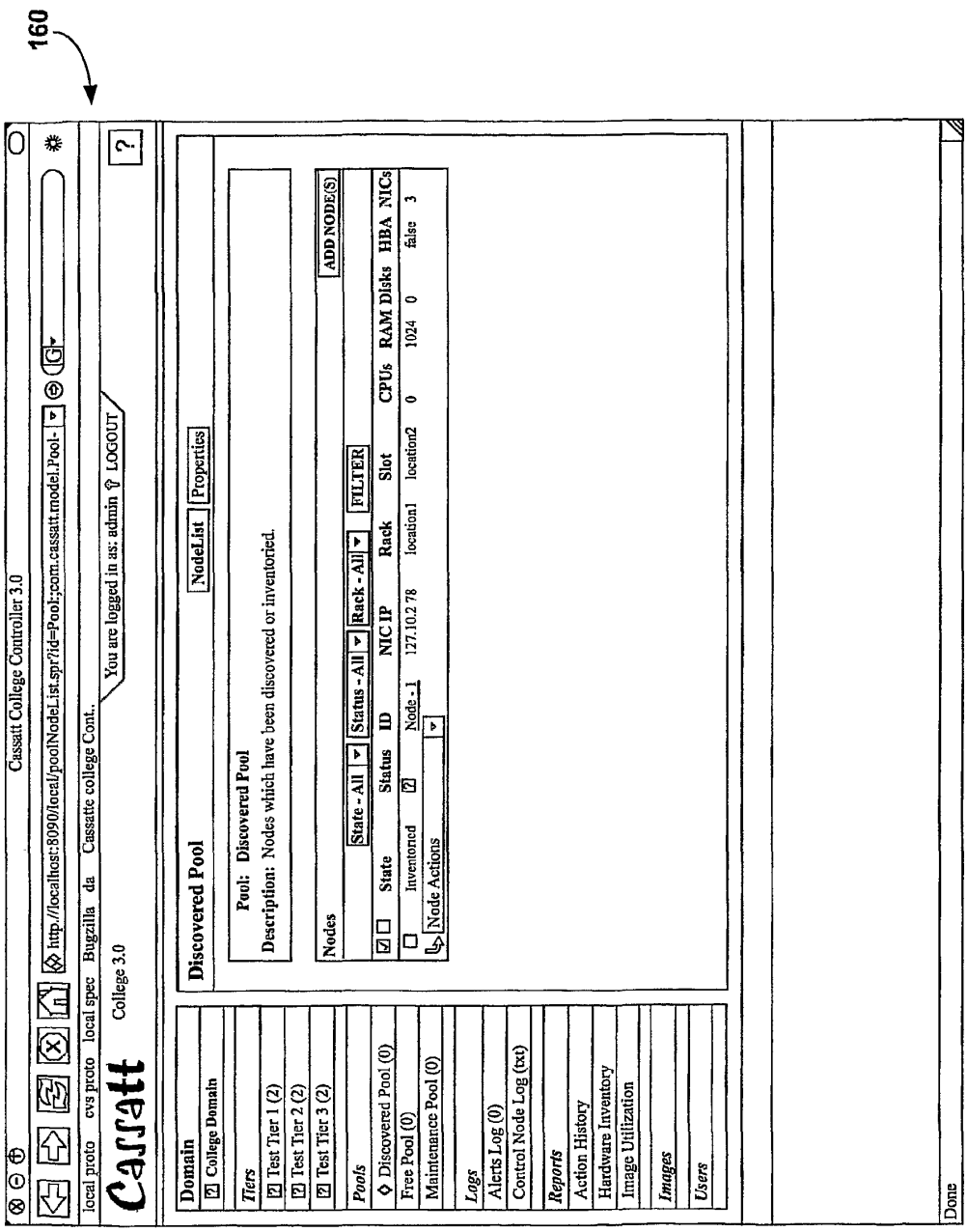


FIG. 12

SUBSTITUTE SHEET (RULE 26)

Netflix, Inc. - Ex. 1002, Page 000392
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

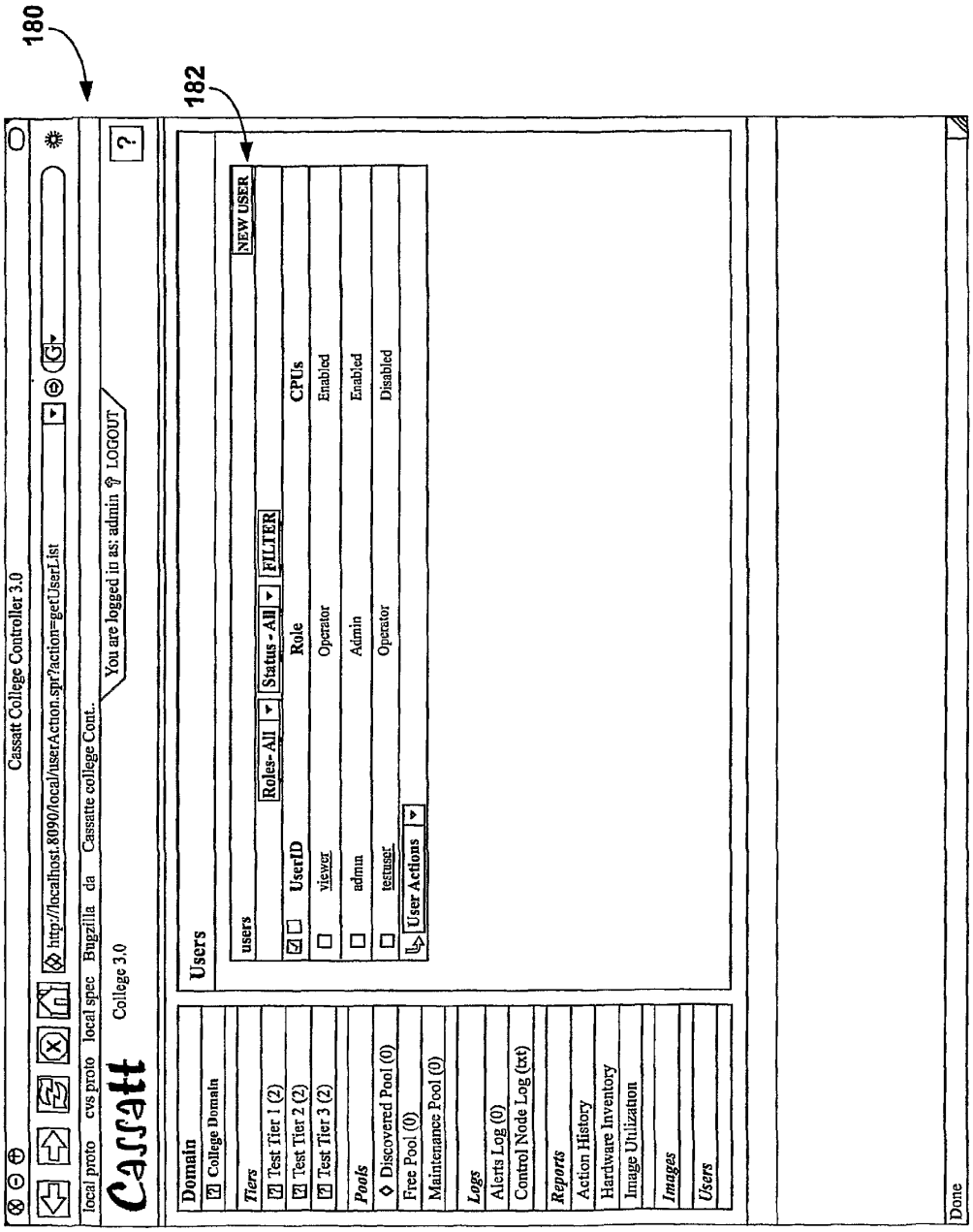


FIG. 13

SUBSTITUTE SHEET (RULE 26)

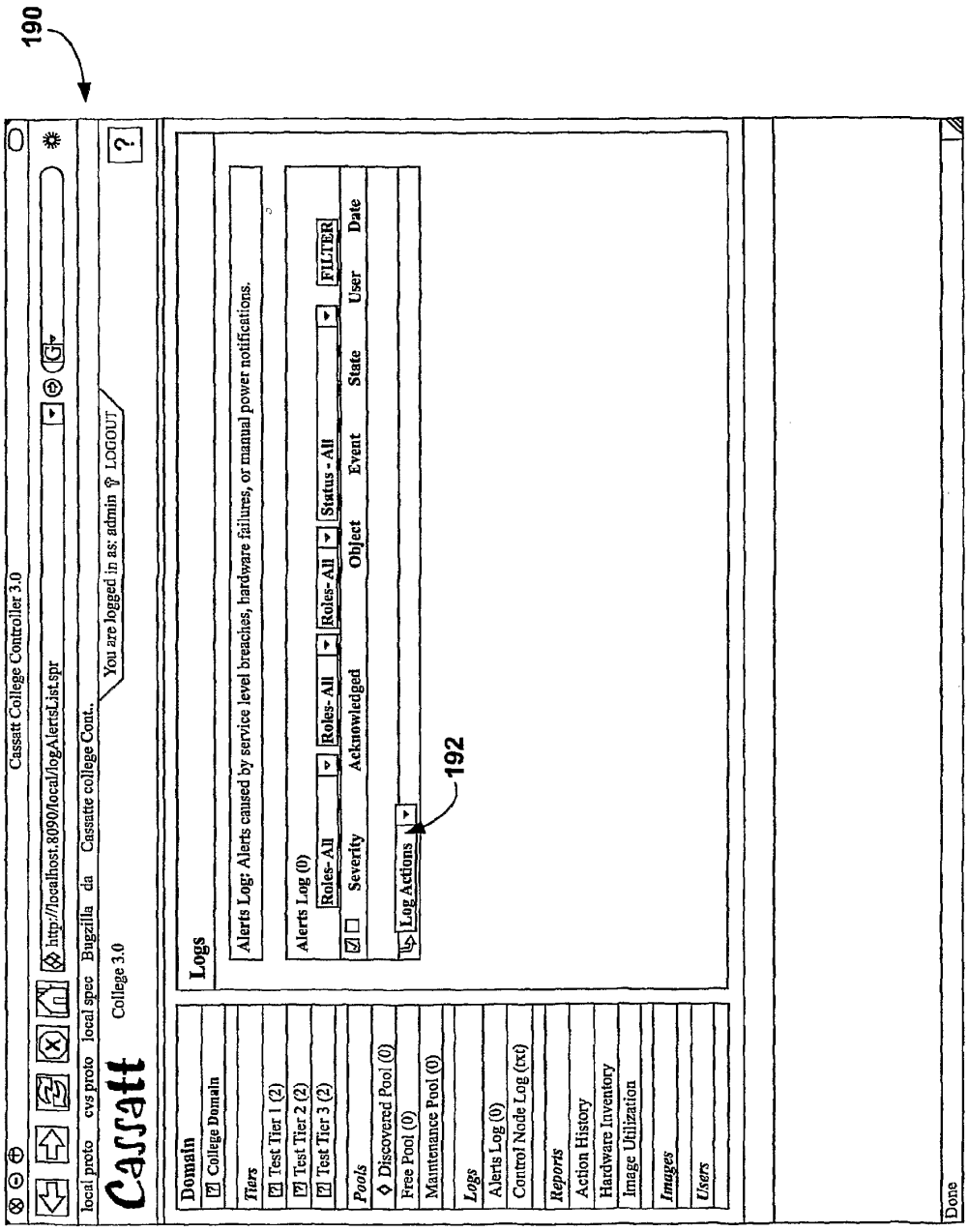


FIG. 14

Netflix, Inc. - Ex. 1002, Page 000394
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

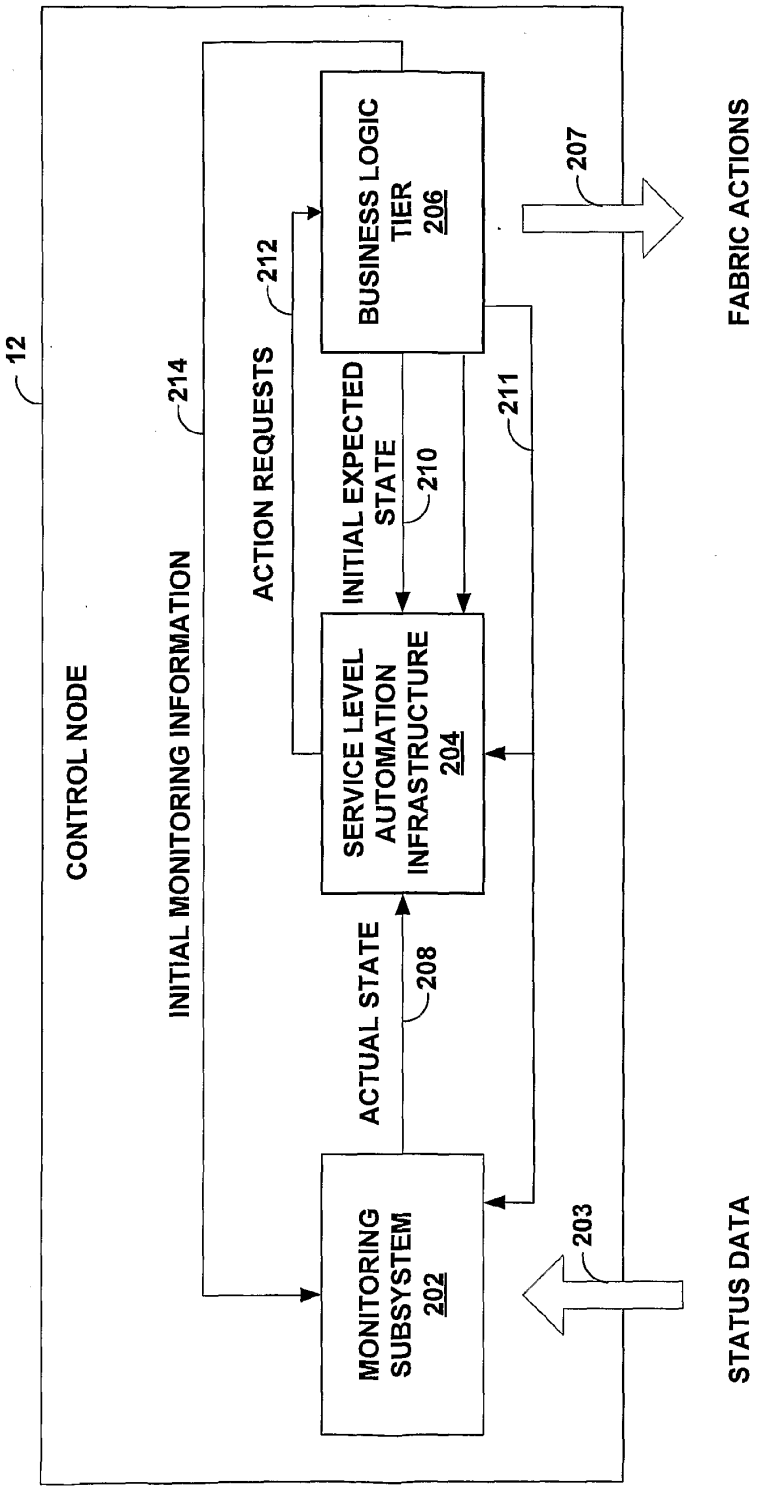

12
CONTROL NODE
INITIAL MONITORING INFORMATION
214
ACTION REQUESTS
212
INITIAL EXPECTED STATE
210
ACTUAL STATE
208
211
BUSINESS LOGIC TIER 206
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
MONITORING SUBSYSTEM 202
207
FABRIC ACTIONS
203
STATUS DATA

FIG. 15

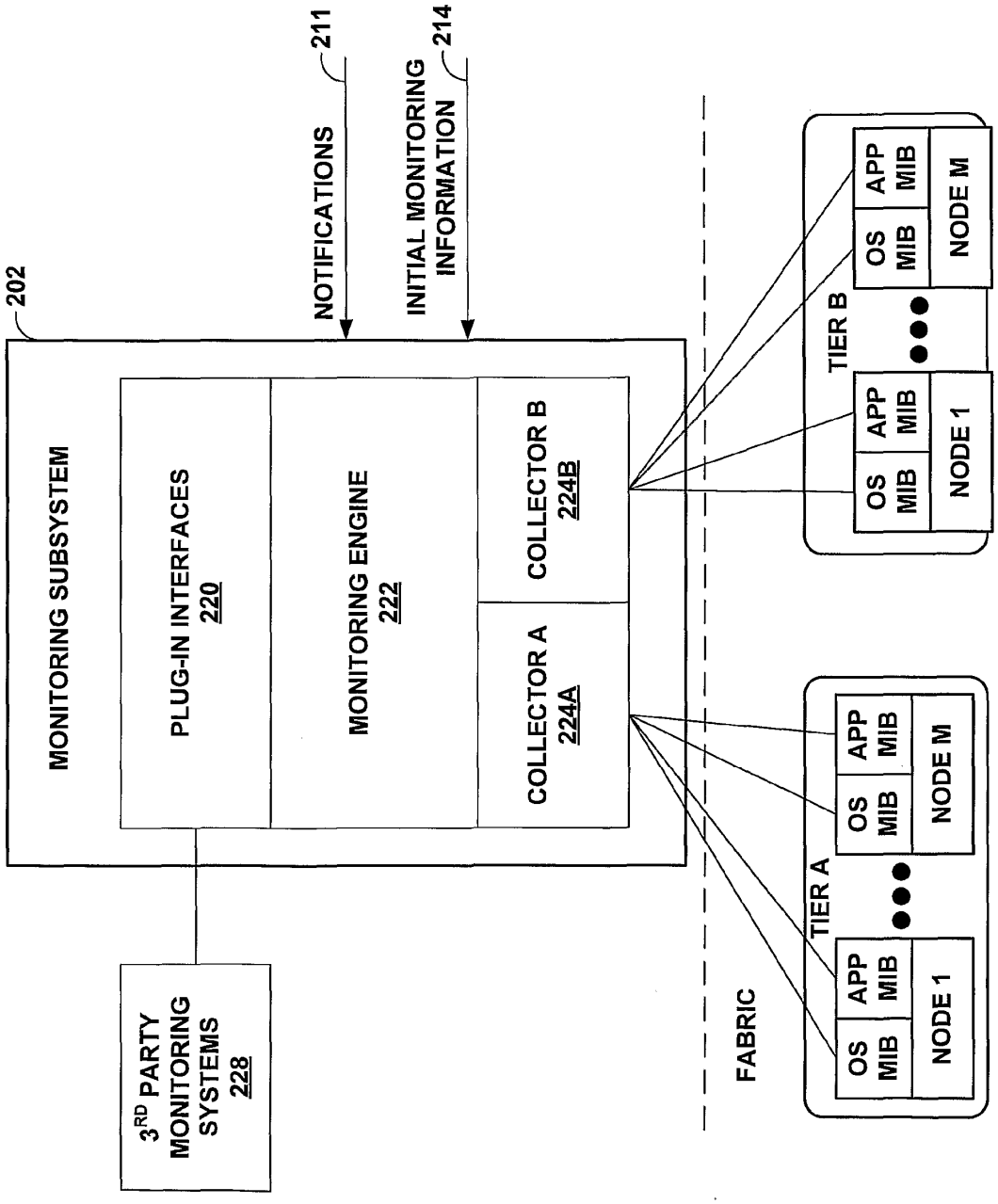

202
MONITORING SUBSYSTEM
PLUG-IN INTERFACES 220
MONITORING ENGINE 222
COLLECTOR A 224A
COLLECTOR B 224B
NOTIFICATIONS 211
INITIAL MONITORING INFORMATION 214
3RD PARTY MONITORING SYSTEMS 228
FABRIC
TIER A
TIER B
OS MIB
APP MIB
NODE 1
NODE M

FIG. 16

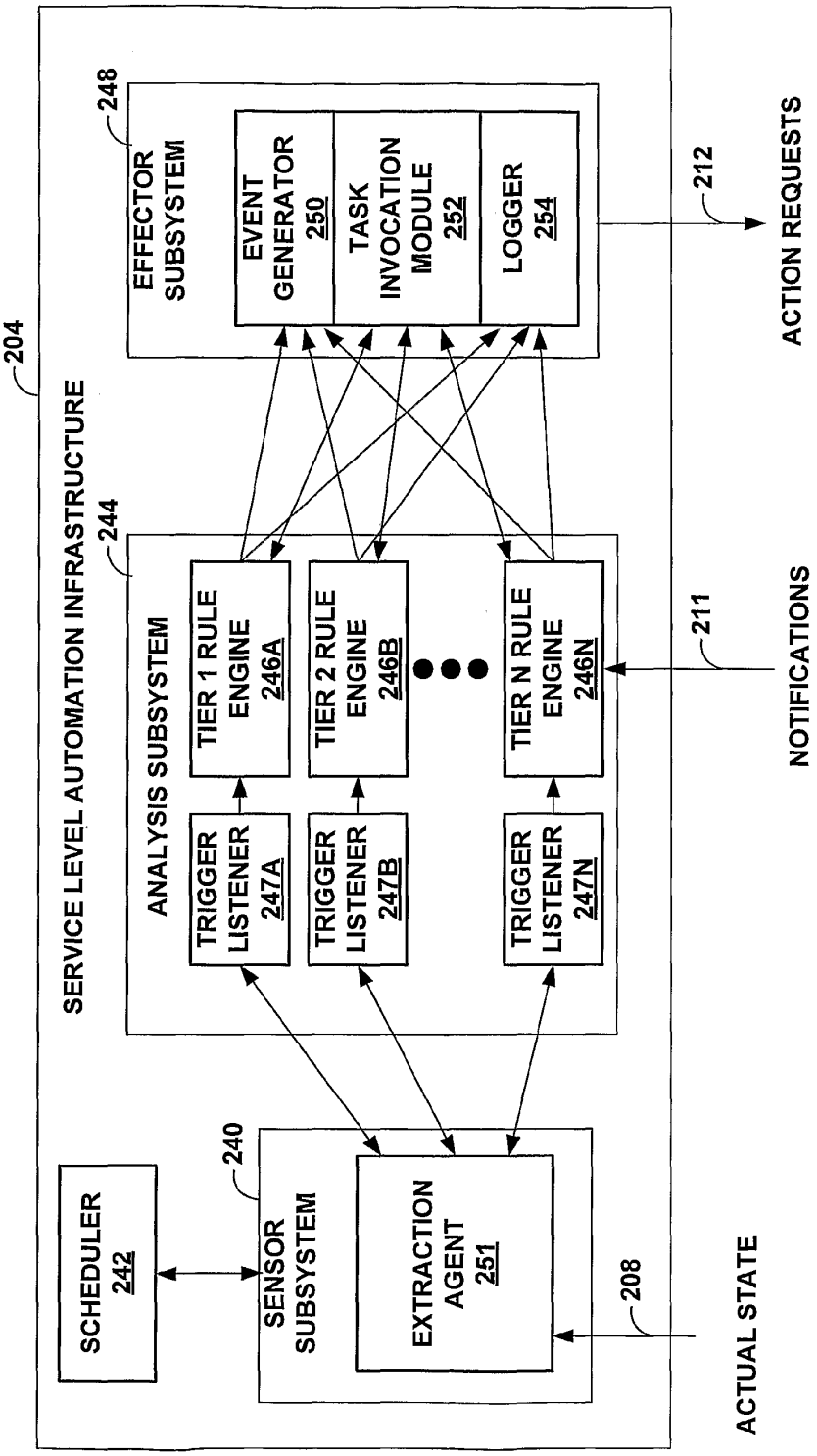

204
SERVICE LEVEL AUTOMATION INFRASTRUCTURE
248
EFFECTOR SUBSYSTEM
EVENT GENERATOR 250
TASK INVOCATION MODULE 252
LOGGER 254
212
ACTION REQUESTS
244
ANALYSIS SUBSYSTEM
TRIGGER LISTENER 247A
TRIGGER LISTENER 247B
TRIGGER LISTENER 247N
TIER 1 RULE ENGINE 246A
TIER 2 RULE ENGINE 246B
TIER N RULE ENGINE 246N
211
NOTIFICATIONS
240
SENSOR SUBSYSTEM
SCHEDULER 242
EXTRACTION AGENT 251
208
ACTUAL STATE

FIG. 17

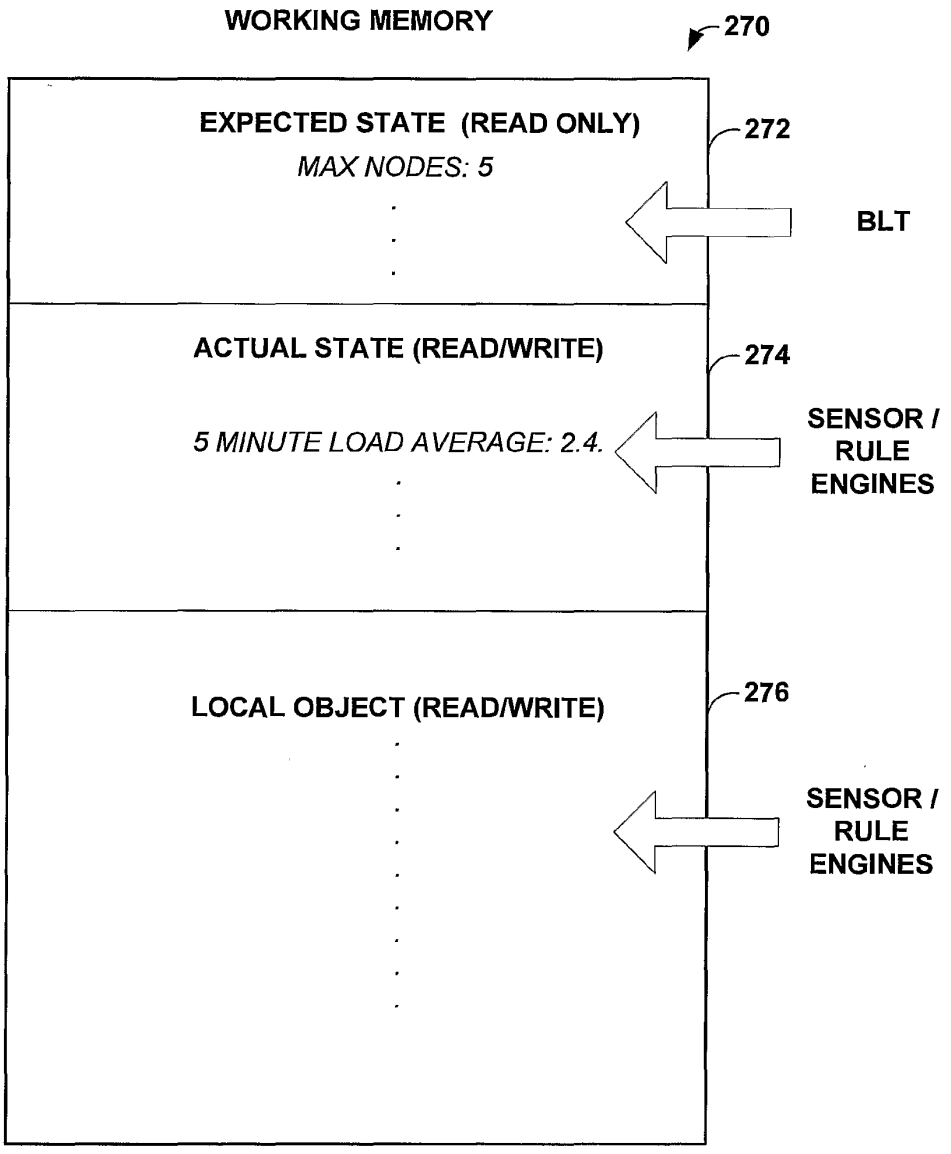

WORKING MEMORY
270
EXPECTED STATE (READ ONLY)
MAX NODES: 5
272
BLT
ACTUAL STATE (READ/WRITE)
274
5 MINUTE LOAD AVERAGE: 2.4.
SENSOR / RULE ENGINES
LOCAL OBJECT (READ/WRITE)
276
SENSOR / RULE ENGINES

FIG. 18

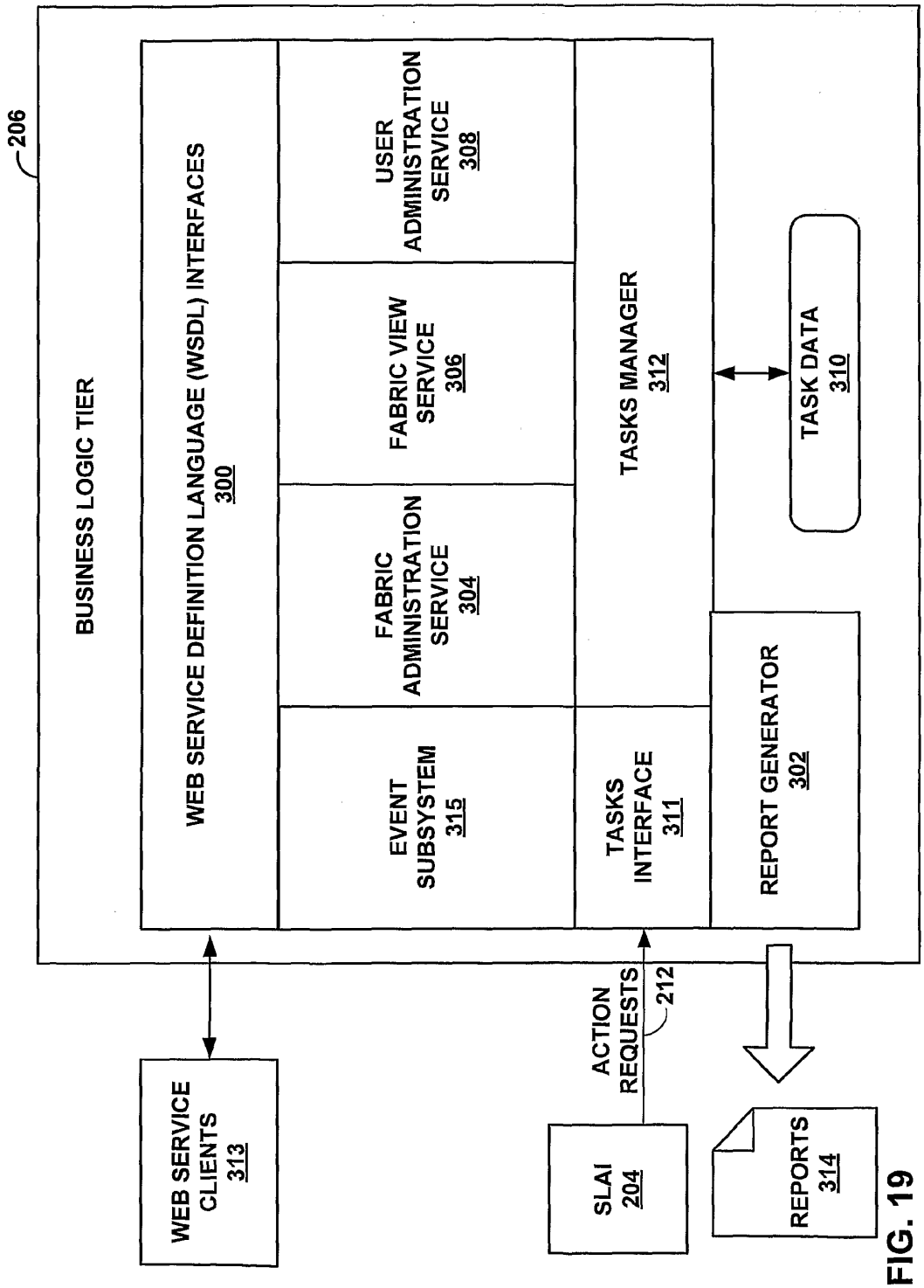

206
BUSINESS LOGIC TIER
WEB SERVICE DEFINITION LANGUAGE (WSDL) INTERFACES 300
USER ADMINISTRATION SERVICE 308
FABRIC VIEW SERVICE 306
FABRIC ADMINISTRATION SERVICE 304
EVENT SUBSYSTEM 315
TASKS MANAGER 312
TASKS INTERFACE 311
REPORT GENERATOR 302
TASK DATA 310
WEB SERVICE CLIENTS 313
ACTION REQUESTS 212
SLAI 204
REPORTS 314

FIG. 19

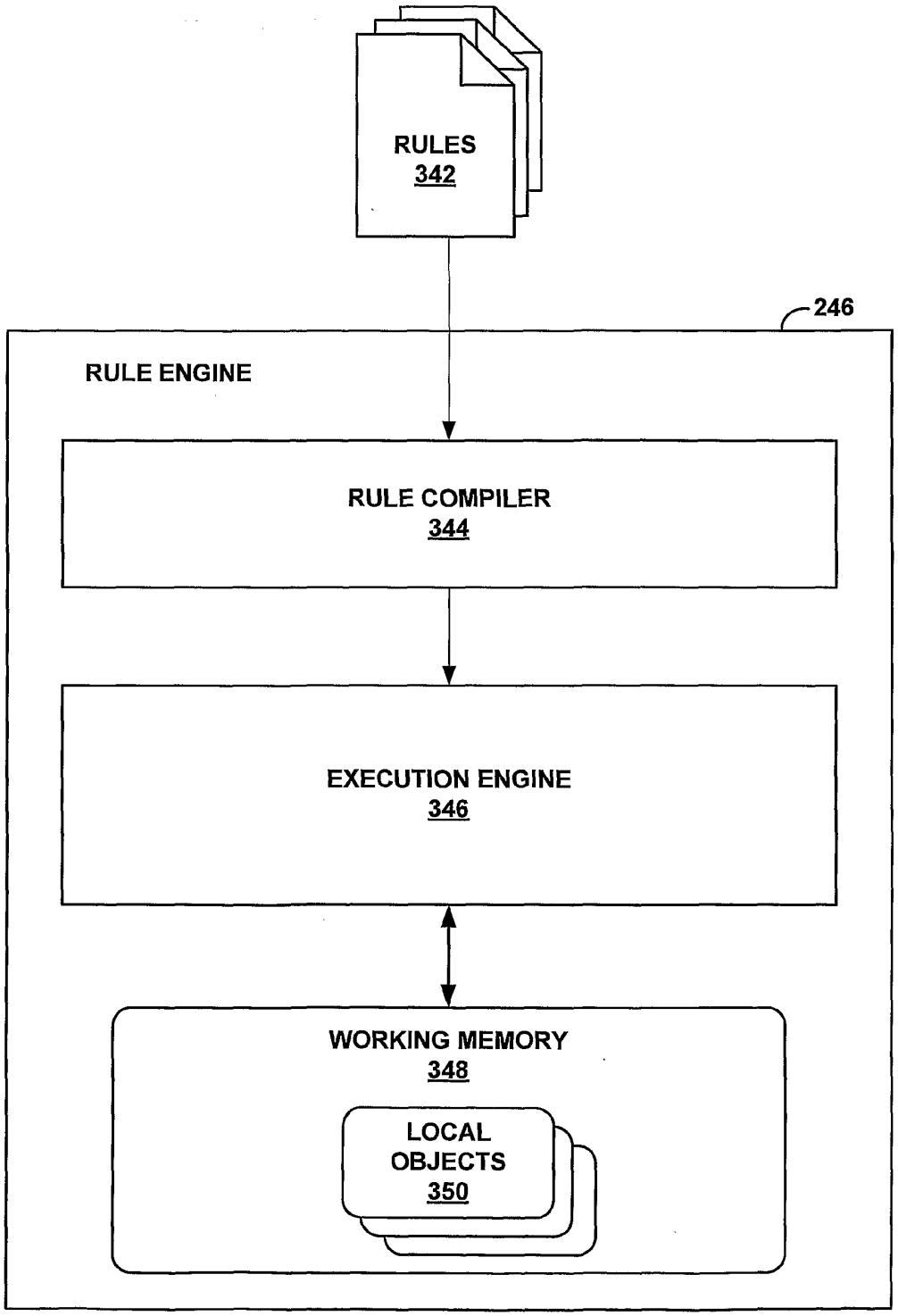

RULES
342
246
RULE ENGINE
RULE COMPILER
344
EXECUTION ENGINE
346
WORKING MEMORY
348
LOCAL
OBJECTS
350

FIG. 20

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003448

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 July 2006	18/07/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ross, C

Form PCT/ISA/210 (second sheet) (April 2005)

Netflix, Inc. - Ex. 1002, Page 000401
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HERNESS E N ET AL: "WebSphere Application Server: a foundation for on demand computing" IBM SYSTEMS JOURNAL IBM USA, vol. 43, no. 2, 2004, pages 213-237, XP002383791 ISSN: 0018-8670 Paragraph starting with "Multitiered distributed business computing": page 215, left-hand column page 220, left-hand column, line 26 - page 221, right-hand column, line 3 Paragraph starting with "Deployment automation": page 222, left-hand column - right-hand column, line 11 page 223, left-hand column, line 5 - line 23 page 224, right-hand column, line 11 - right-hand column, line 3 page 226, left-hand column, line 9 - page 229, left-hand column, line 2 page 229, right-hand column, line 1 - line 13 Paragraph starting with "Monitoring and application response measurement...": page 230, left-hand column	1-28
X	G LODI, F PANZIERI: "QoS-aware Clustering of Application Servers" PROCEEDINGS OF THE 1ST IEEE WORKSHOP ON QUALITY OF SERVICE FOR APPLICATION SERVERS, IN CONJUNCTION WITH THE 23RD INT'L SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS 2004), 17 October 2004 (2004-10-17), pages 1-6, XP002383792 Florianopolis, Brazil page 1, line 1 - line 19 page 2, left-hand column, line 10 - right-hand column, line 18 page 3, right-hand column, line 27 - line 45	1-28
X	US 2003/126265 A1 (AZIZ ASHAR ET AL) 3 July 2003 (2003-07-03) abstract; figure 1c paragraph [0049] paragraph [0052] paragraph [0088] - paragraph [0095] paragraph [0206]	1-28

-/--

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

Netflix, Inc. - Ex. 1002, Page 000402
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003448

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/177176 A1 (HIRSCHFELD ROBERT A ET AL) 18 September 2003 (2003-09-18) abstract column 7, line 4 - line 10 column 9, line 29 - line 52; figure 3 column 13, line 23 - line 53; figures 5a,5b	1-28

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

Netflix, Inc. - Ex. 1002, Page 000403
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/003448

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003126265 A1	03-07-2003	NONE	
US 2003177176 A1	18-09-2003	NONE	

Netflix, Inc. - Ex. 1002, Page 000404
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 August 2006 (10.08.2006)

PCT

(10) International Publication Number
WO 2006/083895 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2006/003439

(22) International Filing Date: 31 January 2006 (31.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

11/047,468	31 January 2005 (31.01.2005)	US
11/074,291	7 March 2005 (07.03.2005)	US
11/176,161	7 July 2005 (07.07.2005)	US

(71) Applicant *(for all designated States except US)*: **CASSATT CORPORATION** [US/US]; 1740 Technology Drive, San Jose, California 95110 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **JACKSON, Jerry, R.** [US/US]; 11270 Wakely Road, Colorado Springs, Colorado 80908 (US).

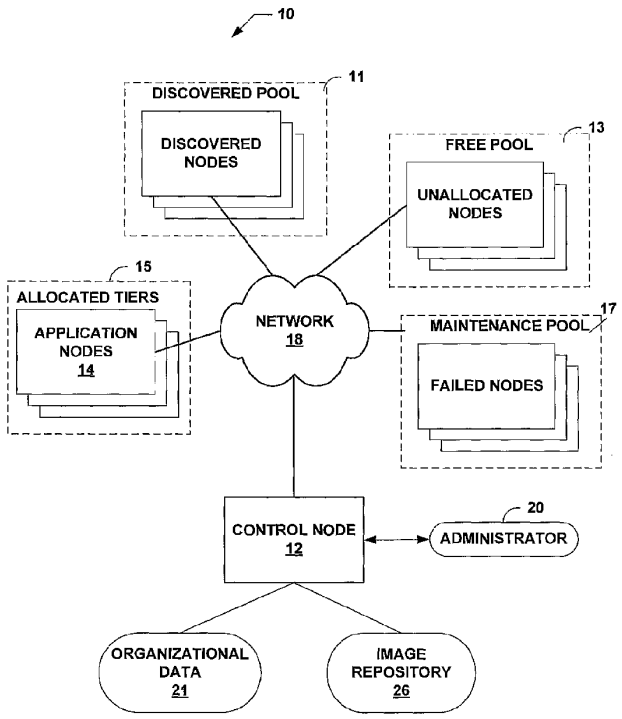
(74) Agent: **SIEFFERT, Kent, J.**; SHUMAKER & SIEFERT, P.A., 8425 SEASONS PARKWAY, Suite 105, St. Paul, Minnesota 55125-4393 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: AUTONOMIC CONTROL OF DISTRIBUTED COMPUTING SYSTEM USING RULE-BASED SENSOR DEFINITIONS



(57) Abstract: A distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of distributed computing system. The control node includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules. For example, the rules may define sensors or methods of configuring the distributed computing system. More specifically, the sensor rules define the conditions under which the automation subsystem may update a model of the distributed computing system based on data received from the distributed computing system.

WO 2006/083895 A1

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

AUTONOMIC CONTROL OF A DISTRIBUTED COMPUTING SYSTEM USING RULE-BASED SENSOR DEFINITIONS

TECHNICAL FIELD

[0001] The invention relates to computing environments and, more specifically, to distributed computing systems.

BACKGROUND

[0002] Distributed computing systems are increasingly being utilized to support business as well as technical applications. Typically, distributed computing systems are constructed from a collection of computing nodes that combine to provide a set of processing services to implement the distributed computing applications. Each of the computing nodes in the distributed computing system is typically a separate, independent computing device interconnected with each of the other computing nodes via a communications medium, e.g., a network.

[0003] One challenge with distributed computing systems is the organization, deployment and administration of such a system within an enterprise environment. For example, it is often difficult to manage the allocation and deployment of enterprise computing functions within the distributed computing system. An enterprise, for example, often includes several business groups, and each group may have competing and variable computing requirements.

SUMMARY

[0004] In general, the invention is directed to a distributed computing system that conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.

[0005] As described herein, the model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes

Netflix, Inc. - Ex. 1002, Page 000407
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

to logically define the hierarchical organization of the distributed computing system. The control nodes are responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

[0006] In one embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network, and a control node. The control node includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules. These rules define preconditions under which the automation system may take the action specified by the rule. For example, the rules may define sensors or methods of configuring the distributed computing system. More specifically, the sensor rules define the conditions under which the automation subsystem may update a model of the distributed computing system based on data received from the distributed computing system.

[0007] In another embodiment, a method comprises receiving input that defines an expected state for a distributed computing system having a plurality of application nodes interconnected via a communications network, and receiving status data that represents an actual state for the distributed computing system. The method further comprises processing rules with a set of rule engines. The rules define sensor rules and configuration rules. The sensor rules determine operations for updating a model of the actual state of the nodes. The configuration rules determine operations for reducing any difference between the actual state and the expected state, and applying the operations to the distributed computing system to control the application nodes in accordance with the rules.

[0008] In another embodiment, a computer-readable medium comprises instructions that cause a processor to process rules with a set of forward-chaining rule engines. These rules may either be sensor rules or configuration rules. The sensor rules define the conditions under which the processor should listen for a certain data pattern, and how to modify a model of the actual state of the distributed computing system when that data pattern occurs. The configuration rules determine operations for reducing differences between the model of the actual state of a distributed computing system and an expected state. The

Netflix, Inc. - Ex. 1002, Page 000408
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

instructions further cause the computer to apply the operations to the distributed computing system to provide autonomic control of the application nodes in accordance with the rules.

[0009] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0010] FIG. 1 is a block diagram illustrating a distributed computing system constructed from a collection of computing nodes.

[0011] FIG. 2 is a schematic diagram illustrating an example of a model of an enterprise that logically defines an enterprise fabric.

[0012] FIG. 3 is a flow diagram that provides a high-level overview of the operation of a control node when configuring the distributed computing system.

[0013] FIG. 4 is a flow diagram illustrating exemplary operation of the control node when assigning computing nodes to node slots of tiers.

[0014] FIG. 5 is a flow diagram illustrating exemplary operation of a control node when adding an additional computing node to a tier to meet additional processing demands.

[0015] FIG. 6 is a flow diagram illustrating exemplary operation of a control node harvesting excess node capacity from one of the tiers and returning the harvested computing node to the free pool.

[0016] FIG. 7 is a screen illustration of an exemplary user interface for defining tiers in a particular domain.

[0017] FIG. 8 is a screen illustration of an exemplary user interface for defining properties of the tiers.

[0018] FIG. 9 is a screen illustration of an exemplary user interface for viewing and identify properties of a computing node.

[0019] FIG. 10 is a screen illustration of an exemplary user interface for viewing software images.

Netflix, Inc. - Ex. 1002, Page 000409
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0020] FIG. 11 is a screen illustration of an exemplary user interface for viewing a hardware inventory report.

[0021] FIG. 12 is a screen illustration of an exemplary user interface for viewing discovered nodes that are located in the free pool.

[0022] FIG. 13 is a screen illustration of an exemplary user interface for viewing users of a distributed computing system.

[0023] FIG. 14 is a screen illustration of an exemplary user interface for viewing alerts for the distributed computing system.

[0024] FIG. 15 is a block diagram illustrating one embodiment of control node that includes a monitoring subsystem, a service level automation infrastructure (SLAI), and a business logic tier (BLT).

[0025] FIG. 16 is a block diagram illustrating one embodiment of the monitoring subsystem.

[0026] FIG. 17 is a block diagram illustrating one embodiment of the SLAI in further detail.

[0027] FIG. 18 is a block diagram of an example working memory associated with rule engines of the SLAI.

[0028] FIG. 19 is a block diagram illustrating an example embodiment for the BLT of the control node.

[0029] FIG. 20 is a block diagram illustrating one embodiment of a rule engine in further detail.

[0030] FIG. 21 is a block diagram illustrating another example embodiment of the SLAI.

[0031] FIG. 22 is a screen illustration of an exemplary user interface for editing sensor and configuration rules.

DETAILED DESCRIPTION

[0032] FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide distributed processing.

Netflix, Inc. - Ex. 1002, Page 000410
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0033] In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated tiers 15 and a maintenance pool 17. In addition, distributed computing system 10 includes at least one control node 12.

[0034] Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

[0035] Discovered pool 11 includes a set of discovered nodes that have been automatically "discovered" within distributed computing system 10 by control node 12. For example, control node 12 may monitor dynamic host communication protocol (DHCP) leases to discover the connection of a node to network 18. Once detected, control node 12 automatically inventories the attributes for the discovered node and reassigns the discovered node to free pool 13. The node attributes identified during the inventory process may include a CPU count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics or other computing resources. Control node 12 may also receive input identifying node attributes not detectable via the automatic inventory, such as whether the node includes I/O, such as HBA. Further details with respect to the automated discovery and inventory processes are described in U.S. Patent Application No. 11/070,851, having attorney docket no. 1072-009US01, entitled "AUTOMATED DISCOVERY AND INVENTORY OF NODES WITHIN AN AUTONOMIC DISTRIBUTED COMPUTING SYSTEM," filed March 2, 2005.

[0036] Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated tiers 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to add an application node to allocated tiers 15 to increase processing capacity of distributed computing system 10.

Netflix, Inc. - Ex. 1002, Page 000411
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0037] In general, allocated tiers 15 include one or more tiers of application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory). Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated tiers 15.

[0038] Control node 12 provides the system support functions for managing distributed computing system 10. More specifically, control node 12 manages the roles of each computing node within distributed computing system 10 and the execution of software applications within the distributed computing system. In general, distributed computing system 10 includes at least one control node 12, but may utilize additional control nodes to assist with the management functions.

[0039] Other control nodes 12 (not shown in FIG. 1) are optional and may be associated with a different subset of the computing nodes within distributed computing system 10. Moreover, control node 12 may be replicated to provide primary and backup administration functions, thereby allowing for graceful handling a failover in the event control node 12 fails.

[0040] Network 18 provides a communications interconnect for control node 12 and application nodes 14, as well as discovered nodes, unallocated nodes and failed nodes. Communications network 18 permits internode communications among the computing nodes as the nodes perform interrelated operations and functions. Communications network 18 may comprise, for example, direct connections between one or more of the computing nodes, one or more customer networks maintained by an enterprise, local area networks (LANs), wide area networks (WANs) or a combination thereof. Communications network 18 may include a number of switches, routers, firewalls, load balancers, and the like.

[0041] In one embodiment, each of the computing nodes within distributed computing system 10 executes a common general-purpose operating system. One example of a general-purpose operating system is the Windows™ operating system

Netflix, Inc. - Ex. 1002, Page 000412
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

provided by Microsoft Corporation. In some embodiments, the general-purpose operating system such as the Linux kernel may be used.

[0042] In the example of FIG. 1, control node 12 is responsible for software image management. The term "software image" refers to a complete set of software loaded on an individual computing node, including the operating system and all boot code, middleware and application files. System administrator 20 may interact with control node 12 and identify the particular types of software images to be associated with application nodes 14. Alternatively, administration software executing on control node 12 may automatically identify the appropriate software images to be deployed to application nodes 14 based on the input received from system administrator 20. For example, control node 12 may determine the type of software image to load onto an application node 14 based on the functions assigned to the node by system administrator 20. Application nodes14 may be divided into a number of groups based on their assigned functionality. As one example, application nodes 14 may be divided into a first group to provide web server functions, a second group to provide business application functions and a third group to provide database functions. The application nodes 14 of each group may be associated with different software images.

[0043] Control node 12 provides for the efficient allocation and management of the various software images within distributed computing system 10. In some embodiments, control node 12 generates a "golden image" for each type of software image that may be deployed on one or more of application nodes 14. As described herein, the term "golden image" refers to a reference copy of a complete software stack.

[0044] System administrator 20 may create a golden image by installing an operating system, middleware and software applications on a computing node and then making a complete copy of the installed software. In this manner, a golden image may be viewed as a "master copy" of the software image for a particular computing function. Control node 12 maintains a software image repository 26 that stores the golden images associated with distributed computing system 10.

[0045] Control node 12 may create a copy of a golden image, referred to as an "image instance," for each possible image instance that may be deployed within

Netflix, Inc. - Ex. 1002, Page 000413
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

distributed computing system 10 for a similar computing function. In other words, control node 12 pre-generates a set of K image instances for a golden image, where K represents the maximum number of image instances for which distributed computing system 10 is configured for the particular type of computing function. For a given computing function, control node 12 may create the complete set of image instance even if not all of the image instances will be initially deployed. Control node 12 creates different sets of image instances for different computing functions, and each set may have a different number of image instances depending on the maximum number of image instances that may be deployed for each set. Control node 12 stores the image instances within software image repository 26. Each image instance represents a collection of bits that may be deployed on an application node.

[0046] Further details of software image management are described in co-pending U.S. patent application Ser. No. 11/046,133, entitled "MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005 and co-pending U.S. patent application Ser. No. 11/046,152, entitled "UPDATING SOFTWARE IMAGES ASSOCIATED WITH A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005.

[0047] In general, distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

[0048] As used herein, the "fabric" level generally refers to the logical constructs that allow for definition, deployment, partitioning and management of distinct enterprise applications. In other words, fabric refers to the integrated set of hardware, system software and application software that can be "knitted" together to form a complete enterprise system. In general, the fabric level consists of two elements: fabric components or fabric payload. Control node 12 provides fabric management and fabric services as described herein.

Netflix, Inc. - Ex. 1002, Page 000414
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0049] In contrast, a "domain" is a logical abstraction for containment and management within the fabric. The domain provides a logical unit of fabric allocation that enables the fabric to be partitioned amongst multiple uses, e.g. different business services.

[0050] Domains are comprised of tiers, such as a 4-tier application model (web server, application server, business logic, persistence layer) or a single tier monolithic application. Fabric domains contain the free pool of devices available for assignment to tiers.

[0051] A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

[0052] The lowest level, the node level, includes the physical components of the fabric. This includes computing nodes that, as described above, provide operating environments for system applications and enterprise software applications. In addition, the node level may include network devices (e.g., Ethernet switches, load balancers and firewalls) used in creating the infrastructure of network 18. The node level may further include network storage nodes that are network connected to the fabric.

[0053] System administrator 20 accesses administration software executing on control node 12 to logically define the hierarchical organization of distributed computing system 10. For example, system administrator 20 may provide organizational data 21 to develop a model for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.

[0054] More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a

Netflix, Inc. - Ex. 1002, Page 000415
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

customer relation management (CRM) service. System administrator 20 further defines one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a web tier, an application tier and a database tier. In this manner, distributed computing system 10 may be configured to automatically provide web server functions, business application functions and database functions.

[0055] For each of the tiers, control node 12 creates a number of "node slots" equal to the maximum number of application nodes 14 that may be deployed. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot. The node slots may, for instance, identify a particular software image instance associated with an application node 14 as well as a network address associated with that particular image instance.

[0056] In this manner, each of the tiers include one or more node slots that reference particular software image instances to boot on the application nodes 14 to which each software image instance is assigned. The application nodes 14 to which control node 12A assigns the image instances temporarily inherit the network address assigned to the image instance for as long as the image instance is deployed on that particular application node. If for some reason the image instance is moved to a different application node 14, control node 12A moves the network address to that new application node.

[0057] System administrator 20 may further define specific node requirements for each tier of the fabric. For example, the node requirements specified by system administrator 20 may include a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics that may be detected on the individual computing nodes. System administrator 20 may also specify user-defined hardware attributes of the computing nodes, such as whether I/O (like HBA) is required. The user-defined hardware attributes are typically not capable of detection during an automatic inventory. In this manner, system administrator 20 creates a list of attributes that

Netflix, Inc. - Ex. 1002, Page 000416
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the tier requires of its candidate computing nodes. In addition, particular node requirements may be defined for software image instances.

[0058] In addition to the node requirements described above, system administrator 20 may further define policies that are used when re-provisioning computing nodes within the fabric. System administrator 20 may define policies regarding tier characteristics, such as a minimum number of nodes a tier requires, an indication of whether or not a failed node is dynamically replaced by a node from free pool 13, a priority for each tier relative to other tiers, an indication of whether or not a tier allows nodes to be re-provisioned to other tiers to satisfy processing requirements by other tiers of a higher priority or other policies. Control node 12 uses the policy information input by system administrator 20 to re-provision computing nodes to meet tier processing capacity demands.

[0059] After receiving input from system administrator 20 defining the architecture and policy of the enterprise fabric, control node 12 identifies unallocated nodes within free pool 13 that satisfy required node attributes. Control node 12 automatically assigns unallocated nodes from free pool 13 to respective tier node slots of a tier. As will be described in detail herein, in one embodiment, control node 12 may assign computing nodes to the tiers in a "best fit" fashion. Particularly, control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20. The assignment of the computing nodes may occur on a tier-by-tier basis beginning with a tier with the highest priority and ending with a tier with the lowest priority. Alternatively, or in addition, assignment of computing nodes may be based on dependencies defined between tiers.

[0060] As will be described in detail below, control node 12 may automatically add unallocated nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to the free pool when the tier has excess capacity, transfer nodes from tier to tier to meet processing demands, or replace failed nodes with nodes from the free pool. Thus, computing resources, i.e., computing nodes, may be automatically shared between tiers and domains within the fabric based on user-defined policies to dynamically address high-processing demands, failures and other events.

Netflix, Inc. - Ex. 1002, Page 000417
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0061] FIG. 2 is a schematic diagram illustrating an example embodiment of organizational data 21 that defines a model logically representing an enterprise fabric in accordance with the invention. In the example illustrated in FIG. 2, control node 12 (FIG. 1) maintains organizational data 21 to define a simple e-commerce fabric 32.

[0062] In this example, e-commerce fabric 32 includes a storefront domain 34A and a financial planning domain 34B. Storefront domain 34A corresponds to the enterprise storefront domain and allows customers to find and purchase products over a network, such as the Internet. Financial planning domain 34B allows one or more employees to perform financial planning tasks for the enterprise.

[0063] Tier level 31C includes one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. For example, storefront domain 34A includes a web server tier (labeled "web tier") 36A, a business application tier (labeled "app tier") 36B, and a database tier (labeled "DB tier") 36C. Web server tier 36A, business application tier 36B and database tier 36C interact with one another to present a customer with an online storefront application and services. For example, the customer may interact with web server tier 36A via a web browser. When the customer searches for a product, web server tier 36A may interacts with business application tier 36B, which may in turn access a database tier 36C. Similarly, financial planning domain 34B includes a financial planning tier 36D that provides subcomponents of applications and services of the financial planning domain 34B. Thus, in this example, a domain may include a single tier.

[0064] Tier level 31D includes one or more logical node slots 38A-38H ("node slots 38") within each of the tiers. Each of node slots 38 include node specific information, such as software resources for an application node 14 that is assigned to a respective one of the node slots 38. Node slots 38 may, for instance, identify particular software image instances within image repository 26 and map the identified software image instances to respective application nodes 14. As an example, node slots 38A and 38B belonging to web server tier 36A may reference particular software image instances used to boot two application nodes 14 to provide web server functions. Similarly, the other node slots 38 may reference

Netflix, Inc. - Ex. 1002, Page 000418
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

software image instances to provide business application functions, database functions, or financial application functions depending upon the tier to which the node slots are logically associated.

[0065] Although in the example of FIG. 2, there are two node slots 38 corresponding to each tier, the tiers may include any number of node slots depending on the processing capacity needed on the tier. Furthermore, not all of node slots 38 may be currently assigned to an application node 14. For example, node slot 28B may be associated with an inactive software image instance and, when needed, may be assigned to an application node 14 for deployment of the software image instance.

[0066] In this example, organizational data 21 associates free node pool 13 with the highest-level of the model, i.e., e-commerce fabric 32. As described above, control node 12 may automatically assign unallocated nodes from free node pool 13 to at least a portion of tier node slots 38 of tiers 36 as needed using the "best fit" algorithm described above or another algorithm. Additionally, control node 12 may also add nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.

[0067] Although not illustrated, the model for the enterprise fabric may include multiple free node pools. For example, the model may associate free node pools with individual domains at the domain level or with individual tier levels. In this manner, administrator 20 may define policies for the model such that unallocated computing nodes of free node pools associated with domains or tiers may only be used within the domain or tier to which they are assigned. In this manner, a portion of the computing nodes may be shared between domains of the entire fabric while other computing nodes may be restricted to particular domains or tiers.

[0068] FIG. 3 is a flow diagram that provides a high-level overview of the operation of control node 12 when configuring distributed computing system 10. Initially, control node 12 receives input from a system administrator defining the hierarchical organization of distributed computing system 10 (50). In one example, control node 12 receives input that defines a model that specifies a

Netflix, Inc. - Ex. 1002, Page 000419
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

number of hierarchically arranged nodes as described in detail in FIG. 2. Particularly, the defined architecture of distributed computing system 10 includes an overall fabric having a number of hierarchically arranged domains, tiers and node slots.

[0069] During this process, control node 12 may receive input specifying node requirements of each of the tiers of the hierarchical model (52). As described above, administrator 20 may specify a list of attributes, e.g., a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), or local disk characteristics, that the tiers require of their candidate computing nodes. In addition, control node 12 may further receive user-defined custom attributes, such as requiring the node to have I/O, such as HBA connectivity. The node requirements or attributes defined by system administrator 20 may each include a name used to identify the characteristic, a data type (e.g., integer, long, float or string), and a weight to define the importance of the requirement.

[0070] Control node 12 identifies the attributes for all candidate computing nodes within free pool 13 or a lower priority tier (54). As described above, control node 12 may have already discovered the computing nodes and inventoried the candidate computing nodes to identify hardware characteristics of all candidate computing nodes. Additionally, control node 12 may receive input from system administrator 20 identifying specialized capabilities of one or more computing nodes that are not detectable by the inventory process.

[0071] Control node 12 dynamically assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a "best fit" algorithm.

[0072] FIG. 4 is a flow diagram illustrating exemplary operation of control node 12 when assigning computing nodes to node slots of tiers. Initially, control node

Netflix, Inc. - Ex. 1002, Page 000420
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

12 selects a tier to enable (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

[0073] Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each of the tiers. Control node 12 retrieves the node requirements for the selected tier from the database.

[0074] In addition, control node 12 accesses the database and retrieves the computing node attributes of one of the unallocated computing nodes of free pool 13. Control node 12 compares the node requirements of the tier to the node attributes of the selected computing node (64).

[0075] Based on the comparison, control node 12 determines whether the node attributes of the computing node meets the minimum node requirements of the tier (66). If the node attributes of the selected computing node do not meet the minimum node requirements of the tier, then the computing node is removed from the list of candidate nodes for this particular tier (68). Control node 12 repeats the process by retrieving the node attributes of another of the computing nodes of the free pool and compares the node requirements of the tier to the node attributes of the computing node.

[0076] If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the node requirements of the tier (70). If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).

[0077] Control node 12 then determines whether the node count for the tier is met (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the

Netflix, Inc. - Ex. 1002, Page 000421
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

ideal number of nodes defined by system administrator 20. When the node count for the tier is met, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority, and repeats the process until all defined tiers are enabled, i.e., populated with application nodes (60).

[0078] If the node attributes of the selected computing node and the node requirements of the tier are not a perfect match control node 12 calculates and records a "processing energy" of the node (76). As used herein, the term "processing energy" refers to a numerical representation of the difference between the node attributes of a selected node and the node requirements of the tier. A positive processing energy indicates the node attributes more than satisfy the node requirements of the tier. The magnitude of the processing energy represents the degree to which the node requirements exceed the tier requirements.

[0079] After computing and recording the processing energy of the nodes, control node 12 determines whether there are more candidate nodes in free pool 13 (78). If there are additional candidate nodes, control node 12 repeats the process by retrieving the computing node attributes of another one of the computing nodes of the free pool of computing nodes and comparing the node requirements of the tier to the node attributes of the computing node (64).

[0080] When all of the candidate computing nodes in the free pool have been examined, control node 12 selects the candidate computing node having the minimum positive processing energy and assigns the selected computing node to a node slot of the tier (80). Control node 12 determines whether the minimum node count for the tier is met (82). If the minimum node count for the tier has not been met, control node 12 assigns the computing node with the next lowest calculated processing energy to the tier (80). Control node 12 repeats this process until the node count is met. At this point, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority (60).

[0081] In the event there are an insufficient number of computing nodes in free pool 13, or an insufficient number of computing nodes that meet the tier requirements, control node 12 notifies system administrator 20. System administrator 20 may add more nodes to free pool 13, add more capable nodes to the free pool, reduce the node requirements of the tier so more of the unallocated

Netflix, Inc. - Ex. 1002, Page 000422
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

nodes meet the requirements, or reduce the configured minimum node counts for the tiers.

[0082] FIG. 5 is a flow diagram illustrating exemplary operation of control node 12 when adding an additional computing node to a tier to meet increased processing demands. Initially, control node 12 or system administrator 20 identifies a need for additional processing capacity on one of the tiers (90). Control node 12 may, for example, identify a high processing load on the tier or receive input from a system administrator identifying the need for additional processing capacity on the tier.

[0083] Control node 12 then determines whether there are any computing nodes in the free pool of nodes that meet the minimum node requirements of the tier (92). When there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and assigns the node to the tier (95). As described in detail with respect to FIG. 4, control node 12 may determine whether there are any nodes that have node attributes that are an exact match to the node requirements of the tier. If an exact match is found, the corresponding computing node is assigned to a node slot of the tier. If no exact match is found, control node 12 computes the processing energy for each node and assigns the computing node with the minimum processing energy to the tier. Control node 12 remotely powers on the assigned node and remotely boots the node with the image instance associated with the node slot. Additionally, the booted computing node inherits the network address associated with the node slot.

[0084] If there are no adequate computing nodes in the free pool, i.e., no nodes at all or no nodes that match the minimal node requirements of the tier, control node 12 identifies the tiers with a lower priority than the tier needing more processing capacity (96).

[0085] Control node 12 determines which of the nodes of the lower priority tiers meet the minimum requirements of the tier in need of processing capacity (98). Control node 12 may, for example, compare the attributes of each of the nodes assigned to node slots of the lower priority tiers to the node requirements of the tier in need of processing capacity. Lower priority tiers that have the minimum

Netflix, Inc. - Ex. 1002, Page 000423
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

number of computing nodes may be removed from possible tiers from which to harvest an application node. If, however, all the lower priority tiers have the minimum number of computing nodes defined for the respective tier, the lowest priority tier is selected from which to harvest the one or more nodes.

[0086] Control node 12 calculates the processing energy of each of the nodes of the lower priority tiers that meet the minimum requirements (100). The energies of the nodes are calculated using the differences between the node attributes and the node requirements of the tier needing additional capacity. Control node 12 selects the computing node with the lowest processing energy that meets the minimum requirements, and assigns the selected computing node to the tier in need of processing capacity (102, 95).

[0087] FIG. 6 is a flow diagram illustrating exemplary operation of control node 12 when harvesting excess node capacity from one of the tiers and returning the harvested computing node to free pool 13. Initially, control node 12 identifies a tier having excess node capacity (110). Control node 12 may, for example, periodically check the node capacity of the tiers to identify any tiers having excess node capacity. Performing a periodic check and removal of excess nodes increases the likelihood that a capable computing node will be in free pool 13 in the event one of the tiers needs additional node capacity.

[0088] When harvesting a node, control node 12 calculates the processing energy of all the nodes in the tier as described above with reference to FIG. 4 (112). Control node 12 identifies the node within the tier with the highest processing energy and returns the identified node to the free pool of nodes (114, 116). As described above, the node with the highest processing energy corresponds to the node whose node attributes are the most in excess of the node requirements of the tier.

[0089] Returning the node to the free pool may involve remotely powering off the computing node and updating the database to associate the harvested node with free pool 13. In addition, control node 12 updates the database to disassociate the returned node with the node slot to which it was assigned. At this point, the node no longer uses the network address associated with the image instance mapped to

Netflix, Inc. - Ex. 1002, Page 000424
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the node slot. Control node 12 may, therefore, assign a temporary network address to the node while the node is assigned to free pool 13.

[0090] FIG. 7 is a screen illustration of an exemplary user interface 120 presented by control node 12 with which administrator 20 interacts to define tiers for a particular domain. In the example illustrated in FIG. 7, system administrator 20 has selected the "Collage Domain." User interface 120 presents the tiers that are currently in the selected domain. In the example illustrated, the Collage Domain includes three tiers, "test tier 1," "test tier 2," and "test tier 3." As shown in FIG. 7, in this example, each of the tiers includes two nodes. In addition, user interface 120 lists the type of software image currently deployed to application nodes for each of the tiers. In the example illustrated, image "applone (1.0.0)" is deployed to the nodes of test tier 1 and image "appltwo (1.0.0)" is deployed to the nodes of test tier 2. System administrator 20 may add one or more tiers to the domain by clicking on new tier button 122.

[0091] FIG. 8 is a screen illustration of an exemplary user interface 130 for defining properties of the tiers. In particular, user interface 130 allows system administrator 20 to input a name for the tier, a description of the tier, and an image associated with the tier. The image associated with the tier refers to a master image from which image instances are generated and deployed to the nodes assigned to the tier.

[0092] When configuring a tier, system administrator 20 may elect to activate email alerts. For example, system administrator 20 may activate the email alerts feature in order to receive email alerts providing system administrator 20 with critical and/or non-critical tier information, such as a notification that a tier has been upgraded, a node of the tier has failed or the like. Furthermore, system administrator 20 may input various policies, such node failure rules. For example, system administrator 20 may identify whether control node 12 should reboot a node in case of failure or whether the failed node should automatically be moved to maintenance pool 17. Similarly, system administrator 20 may identify whether nodes assigned to the tier may be harvested by other tiers.

[0093] User interface 130 may also allow system administrator 20 to input node requirements of a tier. In order to input node requirements of a tier, system

Netflix, Inc. - Ex. 1002, Page 000425
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

administrator 20 may click on the "Requirements" tab 132, causing user interface 130 to present an input area to particular node requirements of the tier.

[0094] FIG. 9 is a screen illustration of an exemplary user interface 140 for viewing and identifying properties of a computing node. User interface 140 allows system administrator 20 to define a name, description, and location (including a rack and slot) of a computing node. In addition user interface 140 may specify user-defined properties of a node, such as whether the computing node has I/O HBA capabilities.

[0095] User interface 140 also displays properties that control node 12 has identified during the computing node inventory process. In this example, user interface 140 presents system administrator 20 with the a CPU node count, a CPU speed, the amount of RAM, the disk size and other characteristics that are identifiable during the automated node inventory. User interface 140 additionally presents interface information to system administrator 20. Specifically, user interface 140 provides system administrator 20 with a list of components and their associated IP and MAC addresses.

[0096] User interface 140 also allows system administrator 20 to define other custom requirements. For example, system administrator 20 may define one or more attributes and add those attributes to the list of node attributes presented to system administrator 20.

[0097] FIG. 10 is a screen illustration of an exemplary user interface 150 for viewing software images. User interface 150 presents to a system administrator or another user a list of images maintained by control node 12 within image repository 26. The image list further includes the status of each image (i.e., either active or inactive), the version of the image, the operating system on which the image should be run, the operating system version on which the image should be run and a brief description of the image.

[0098] System administrator 20 or another user may select an image by clicking on the box in front of the image identifier/name and perform one or more actions on the image. Actions that system administrator 20 may perform on an image include deleting the image, updating the image, and the like. System administrator 20 may select one of the image actions via dropdown menu 152. In some embodiments,

Netflix, Inc. - Ex. 1002, Page 000426
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

user interface 150 may further display other details about the images such as the node to which the images are assigned (if the node status is "active"), the network address associated with the images and the like.

[0099] FIG. 11 is a screen illustration of an exemplary user interface 160 for viewing a hardware inventory report. User interface 160 presents to system administrator 20 or another user a list of the nodes that are currently assigned to a domain. System administrator 20 may elect to view the nodes for the entire domain, for a single tier within the domain or for a single rack within a tier.

[0100] For each node, user interface 160 presents a node ID, a status of the node, the tier to which the node belongs, a hostname associated with the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0101] System administrator 20 or other user may select a node by clicking on the box in front of the node identifier/name and perform one or more actions on the node. Actions that system administrator 20 may perform on the node include deleting the node, updating the node attributes or other properties of the node, and the like. System administrator 20 may select one of the node actions via dropdown menu 162.

[0102] FIG. 12 is a screen illustration of an exemplary user interface 170 for viewing discovered nodes that are located in discovered pool 11. For each node, user interface 170 presents a node ID, a state of the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0103] FIG. 13 is a screen illustration of an exemplary user interface 180 for viewing users of distributed computing system 10. User interface 180 presents a list of users as well as the role assigned to each of the users and the status of each of the users. Thus, system administrator 20 may define different roles to each of the users. For example, a user may be either an operator (i.e., general user) or an administrator. System administrator 20 may add a new user to the list of users by clicking on the "New User" button 182.

Netflix, Inc. - Ex. 1002, Page 000427
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0104] FIG. 14 is a screen illustration of an exemplary user interface 190 for viewing alerts for distributed computing system 10. For each of the alerts, user interface 190 identifies the severity of the alert, whether the alert has been acknowledged, an object associated with the alert, an event associated with the alert, a state of the alert, a user associated with the alert and a date associated with the alert.

[0105] System administrator 20 or other user may select an alert by clicking on the box in front of the logged alert and perform one or more actions on the logged alert. Actions that system administrator 20 may perform include deleting the alert, changing the status of the alert, or the like. System administrator 20 may specify the log actions via dropdown menu 192.

[0106] FIG. 15 is a block diagram illustrating one embodiment of control node 12 in further detail. In the illustrated example, control node 12 includes a monitoring subsystem 202, a service level automation infrastructure (SLAI) 204, and a business logic tier (BLT) 206.

[0107] Monitoring subsystem 202 provides real-time monitoring of the distributed computing system 10. In particular, monitoring subsystem 202 dynamically collects status data 203 from the hardware and software operating within distributed computing system 10, and feeds the status data in the form of monitor inputs 208 to SLAI 204. Monitoring inputs 208 may be viewed as representing the actual state of the fabric defined for the organizational model implemented by distributed computing system 10. Monitoring subsystem 202 may utilize well defined interfaces, e.g., the Simple Network Management Protocol (SNMP) and the Java Management Extensions (JMX), to collect and export real-time monitoring information to SLAI 204.

[0108] SLAI 204 may be viewed as an automation subsystem that provides support for autonomic computing and acts as a central nervous system for the controlled fabric. In general, SLAI 204 receives monitoring inputs 208 from monitoring subsystem 202, analyzes the inputs and outputs appropriate action requests 212 to BLT 206. In one embodiment, SLAI 204 is a cybernetic system that controls the defined fabric via feedback loops. More specifically, administrator 20 may interact with BLT 206 to define an expected state 210 for the fabric. BLT 206

Netflix, Inc. - Ex. 1002, Page 000428
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

communicates expected state 210 to SLAI 204. SLAI 204 receives the monitoring inputs from monitoring subsystem 202 and applies rules to determine the most effective way of reducing the differences between the expected and actual states for the fabric.

[0109] For example, SLAI 204 may apply a rule to determine that a node within a high priority tier has failed and that the node should be replaced by harvesting a node from a lower priority tier. In this example, SLAI 204 outputs an action request 212 to invoke BLT 206 to move a node from one tier to the other.

[0110] In general, BLT 206 implements high-level business operations on fabrics, domains and tiers. SLAI 204 invokes BLT 206 to bring the actual state of the fabric into accordance with the expected state. In particular, BLT 206 outputs fabric actions 207 to perform the physical fabric changes. In addition, BLT 206 outputs an initial expected state 210 to SLAI 204 and initial monitoring information 214 to SLAI 204 and monitoring subsystem 202, respectively. In addition, BLT 206 outputs notifications 211 to SLAI 204 and monitoring subsystem 202 to indicate the state and monitoring changes to distributed computing system 10. As one example, BLT 206 may provide control operations that can be used to replace failed nodes. For example, BLT 206 may output an action request indicating that a node having address 10.10.10.10 has been removed from tier ABC and a node having address 10.10.10.11 has been added to tier XYZ. In response, monitoring subsystem 202 stops attempting to collect status data 203 from node 10.10.10.10 and starts monitoring for status data from node 10.10.10.11. In addition, SLAI 204 updates an internal model to automatically associate monitoring inputs from node 10.10.10.11 with tier XYZ.

[0111] FIG. 16 is a block diagram illustrating one embodiment of monitoring subsystem 202. In general, monitoring subsystem 202 dynamically detects and monitors a variety of hardware and software components within the fabric. For example, monitoring subsystem 202 identifies, in a timely and efficient manner, any computing nodes that have failed, i.e., any node that does not respond to a request to a known service. More generally, monitoring subsystem 202 provides a concise, consistent and constantly updating view of the components of the fabric.

Netflix, Inc. - Ex. 1002, Page 000429
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0112] As described further below, monitoring subsystem 202 employs a modular architecture that allows new detection and monitoring collectors 224 to be "plugged-in" for existing and new protocols and for existing and new hardware and software. As illustrated in FIG. 16, monitoring subsystem 202 provides a plug-in architecture that allows different information collectors 224 to be installed. In general, collectors 224 are responsible for protocol-specific collection of monitoring information. The plug-in architecture allows for new protocols to be added by simply adhering to a collector plug-in signature. In this example, monitoring subsystem 202 includes collectors 224A and 224B for collecting information from operating systems and applications executing on nodes within tier A and tier B, respectively.

[0113] In one embodiment, collectors 224 are loaded at startup of control node 12 and are configured with information retrieved from BLT 206. Monitoring engine 222 receives collection requests from SLAI 204, sorts and prioritizes the requests, and invokes the appropriate one of collectors 224 based on the protocol specified in the collection requests. The invoked collector is responsible for collecting the required status data and returning the status data to monitoring engine 222. If the collector is unable to collect the requested status data, the collector returns an error code.

[0114] In one embodiment, collectors 224 are Java code compiled into a jar file and loaded with a class loader at run time. Each of collectors 224 has an associated configuration file written in a data description language, such as the extensible markup language (XML). In addition, a user may interact with BLT 206 to add run-time configuration to dynamically configure collectors 224 for specific computing environments. Each of collectors 224 expose an application programming interface (API) to monitoring engine 222 for communication and data exchange.

[0115] A user, such as a system administrator, specifies the protocol or protocols to be used for monitoring a software image when the image is created. In addition, the users may specify the protocols to be used for monitoring the nodes and each service executing on the nodes. Example protocols supported by the collectors 224 include Secure Shell (SSH), Simple Network Management Protocol (SNMP),

Netflix, Inc. - Ex. 1002, Page 000430
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Internet Control Message Protocol (ICMP) ping, Java Management Extensions (JMX) and the Hypertext Transfer Protocol (HTTP).

[0116] Some protocols require special privileges, e.g., root privileges, to perform the required data collection. In this case, the corresponding collectors 224 communicate with a separate process that executes as the root. Moreover, some protocols may require deployment and/or configuration of data providers within the fabric. Software agents may, for example, be installed and configured on nodes and configured on other hardware. If needed, custom in-fabric components may be deployed.

[0117] In this example, the modular architecture of monitoring subsystem 202 also supports one or more plug-in interfaces 220 for data collection from a wide range of third-party monitoring systems 228. Third-party monitoring systems 228 monitor portions of the fabric and may be vendor-specific.

[0118] FIG. 17 is a block diagram illustrating one embodiment of SLAI 204 in further detail. In the illustrated embodiment, SLAI 204 is composed of three subsystems: a sensor subsystem 240, an analysis subsystem 244 and an effector subsystem 248.

[0119] In general, sensor subsystem 240 receives actual state data from monitoring subsystem 202 in the form of monitoring inputs 208 and supplies ongoing, dynamic input data to analysis subsystem 244. For example, sensor subsystem 240 is notified of physical changes to distributed computing system 10 by monitoring subsystem 202. Sensor subsystem 240 uses the state data received from monitoring subsystem 202 to maintain ongoing, calculated values that can be sent to analysis subsystem 244 in accordance with scheduler 242.

[0120] In one embodiment, sensor subsystem 240 performs time-based hierarchical data aggregation of the actual state data in accordance with the defined organization model. Sensor subsystem 240 maintains organizational data in a tree-like structure that reflects the current configuration of the hierarchical organization model. Sensor subsystem 240 uses the organizational data to perform the real-time data aggregation and map tiers and domains to specific nodes. Sensor subsystem 240 maintains the organizational data based on notifications 211 received from BLT 206.

Netflix, Inc. - Ex. 1002, Page 000431
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0121] Sensor subsystem 240 sends inputs to analysis subsystem 244 to communicate the aggregated data on a periodic or event-driven basis. Analysis subsystem 244 may register an interest in a particular aggregated data value with sensor subsystem 240 and request updates at a specified frequency. In response, sensor subsystem 240 interacts with monitoring subsystem 202 and scheduler 242 to generate the aggregated data required by analysis subsystem 244.

[0122] Sensor subsystem 240 performs arbitrary data aggregations via instances of plug-in classes (referred to as "triggers") that define the aggregations. Each trigger is registered under a compound name based on the entity being monitored and the type of data being gathered. For example, a trigger may be defined to aggregate and compute an average computing load for a tier every five minutes. Analysis subsystem 244 requests the aggregated data based on the registered names. In some embodiments, analysis subsystem 244 may define calculations directly and pass them to sensor subsystem 240 dynamically.

[0123] Analysis subsystem 244 is composed of a plurality of forward chaining rule engines 246A-246N. In general, rule engines 246 match patterns in a combination of configuration data and monitoring data, which is presented by extraction agent 251 in the form of events. Events contain the aggregated data values that are sent to rule engines 246 in accordance with scheduler 242.

[0124] Sensor subsystem 240 may interact with analysis subsystem 244 via trigger listeners 247 that receives updates from a trigger within sensor subsystem 240 when specified events occur. An event may be based on system state (e.g., a node transitioning to an up or down state) or may be time based.

[0125] Analysis subsystem 244 allows rule sets to be loaded in source form and compiled at load time into discrimination networks. Each rule set specifies trigger-delivered attributes. Upon loading the rule sets, analysis subsystem 244 establishes trigger listeners 247 to receive sensor notifications and update respective working memories of rule engines 246. As illustrated in FIG. 17, each of rule engines 246 may serve a different tier defined within the fabric. Alternatively, multiple rule engines 246 may serve a single tier or a single rule engine may serve multiple tiers.

[0126] Rule engines 246 process the events and invoke action requests via calls to effector subsystem 248. In addition, rule engines 246 provide a call-back interface

Netflix, Inc. - Ex. 1002, Page 000432
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

so that effector subsystem 248 can inform a rule engine when an action has completed. Rule engines 246 prevent a particular rule from re-firing as long as any action invoked by the rule has not finished. In general, rules contain notification calls and service invocations though either may be disabled by configuration of effector subsystem 248. BLT 206 supplies initial system configuration descriptions to seed each of rule engines 246.

[0127] In general, rule engines 246 analyze the events and discover discrepancies between an expected state of the fabric and an actual state. Each of rule engines 246 may be viewed as software that performs logical reasoning using knowledge encoded in high-level condition-action rules. Each of rule engines 246 applies automated reasoning that works forward from preconditions to goals defined by system administrator 20. For example, rule engines 246 may apply modus ponens inferences rules.

[0128] Rule engines 246 output requests to effector subsystem 248 which produce actions requests 212 for BLT 206 to resolve the discrepancies. Effector subsystem 248 performs all operations on behalf of analysis subsystem 244. For example, event generator 250, task invocation module 252 and logger 254 of effector subsystem 248 perform event generation, BLT action invocation and rule logging, respectively. More specifically, task invocation module 252 invokes asynchronous operations within BLT 206. In response, BLT 206 creates a new thread of control for each task which is tracked by a unique task identifier (task id). Rules engine 246 uses the task id to determine when a task completes and, if needed, to re-fire any rules that were pended until completion of the task. These tasks may take arbitrary amounts of time, and rules engine 246 tracks the progress of individual task via change notifications 211 produced by BLT 206.

[0129] Event generator 250 creates persistent event records of the state of processing of SLAI 204 and stores the event records within a database. Clients uses these event records to track progress and determine the current state of the SLAI 204.

[0130] Logger 254 generates detailed trace information about system activities for use in rule development and debugging. The logging level can be raised or lowered as needed without changing operation of SLAI 204.

Netflix, Inc. - Ex. 1002, Page 000433
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0131] FIG. 18 is a block diagram of an example working memory 270 associated with rule engines 246. In this example, working memory 270 includes a read-only first data region 272 that stores the expected state received from BLT 206. Data region 272 is read-only in the sense that it cannot be modified in response to a trigger from sensor subsystem 240 or by rule engines 246 without notification from BLT 206.

[0132] In addition, working memory 270 includes a second data region 274 that is modifiable (i.e., read/write) and may be updated by monitoring subsystem 202 or used internally by rule engines 246. In general, data region 274 stores aggregated data representing the actual state of the fabric and can be updated by sensor subsystem 240 or by rule engines 246. The actual state may consist of a set of property annotations that can be attached to objects received from BLT 206 or to objects locally defined within a rule engine, such as local object 276.

[0133] FIG. 19 is a block diagram illustrating an example embodiment for BLT 206. In this example, BLT 206 includes a set of one or more web service definition language (WSDL) interfaces 300, a report generator 302, a fabric administration interface service 304, a fabric view service 306, a user administration service 308, a task interface 311, a task manager 312 and an event subsystem 315.

[0134] As described, BLT 206 provides the facilities necessary to create and administer the organizational model (e.g., fabric, domains, tiers and nodes) implemented by distributed computing system 10. In general, BLT 206 abstracts access to the persisted configuration state of the fabric, and controls the interactions with interfaces to fabric hardware services. As such, BLT 206 provides fabric management capabilities, such as the ability to create a tier and replace a failed node. WSDL interfaces 300 provide web service interfaces to the functionality of BLT 206 that may be invoked by web service clients 313. Many of WSDL interfaces 300 offered by BLT 206 allow administrator 20 to define goals, such as specifying a goal of the expected state of the fabric. As further described below, rule engines 246 within SLAI 204, in turn, invoke task manger 312 to initiate one or more BLT tasks to achieve the specified goal. In general,

Netflix, Inc. - Ex. 1002, Page 000434
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

web service clients 313 may be presentation layer applications, command line applications, or other clients.

[0135] BLT 206 abstracts all interaction with physical hardware for web service clients 313. BLT 206 is an enabling component for autonomic management behavior, but does not respond to real-time events that either prevent a goal from being achieved or produce a set of deviations between the expected state and the actual state of the system. In contrast, BLT 206 originates goals for autonomic reactions to changing configuration and state. SLAI 204 analyzes and acts upon these goals along with real-time state changes. BLT 206 sets the goals to which SLAI 204 strives to achieve, and provides functionality used by the SLAI in order to achieve the goals.

[0136] In general, BLT 206 does not dictate the steps taken in pursuit of a goal since these are likely to change based on the current state of distributed computing system 10 and changes to configurable policy. SLAI 204 makes these decisions based on the configured rule sets for the fabric and by evaluating monitoring data received from monitoring subsystem 202.

[0137] Fabric administration service 304 implements a set of methods for managing all aspects of the fabric. Example methods include methods for adding, viewing, updating and removing domains, tiers, nodes, notifications, assets, applications, software images, connectors, and monitors. Other example methods include controlling power at a node, and cloning, capturing, importing, exporting or upgrading software images. Rule engines 246 of SLAI 204 may, for example, invoke these methods by issuing action requests 212.

[0138] Task manager 312 receives action requests 212 via task interface 311. In general, task interface 311 provides an interface for receiving action requests 212 from SLAI 204 or other internal subsystem. In response, task manager 312 manages asynchronous and long running actions and are invoked by SLAI 204 to satisfy a goal or perform an action requested by a client.

[0139] Task manager 312 generates task data 310 that represents identification and status for each task. Task manager 312 returns a task identifier to the calling web service clients 313 or the internal subsystem, e.g., SLAI 204, that initiated the task.

Netflix, Inc. - Ex. 1002, Page 000435
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Rule engines 246 and web service clients 313 use the task identifiers to track progress and retrieve output, results, and errors associated with achieving the goal.

[0140] In one embodiment, there are no WSDL interfaces 300 for initiating specific tasks. Rather, administrator 20 interacts with BLT 206 though goal interfaces presented by WSDL interfaces 300 to define the goals for the fabric. In contrast, the term task is used to refer to internal system constructs that require no user interaction. Tasks are distinct, low-level units of work that affect the state of the fabric. SLAI 204 may combine tasks to achieve or maintain a goal state.

[0141] For example, administrator 20 can request configuration changes by either adding new goals to an object or by modifying the attributes on existing goals. Scheduled goals apply a configuration at a designated time. For example, the goals for a particular tier may specify the minimum, maximum, and target node counts for that tier. As a result, the tier can increase or decrease current node capacity by scheduling goals with different configuration values.

[0142] This may be useful, for example, in scheduling a software image upgrade. As another example, entire domains may transition online and offline per a defined grid schedule. Administrator 20 may mix and match goals on a component to achieve configurations specific to the application and environment. For example, a tier that does not support autonomic node replacement would not be configured with a harvesting goal.

[0143] In some embodiments, goals are either "in force" or "out of force." SLAI 204 only works to achieve and maintain those goals that are currently in force. SLAI 204 may applies a concept of "gravity" as the goals transition from in force to out of force. For example, SLAI 204 may transition a tier offline when an online goal is marked out of force. Some goal types may have prerequisite goals. For example, an image upgrade goal may require as a prerequisite that a tier be transitioned to offline before the image upgrade can be performed. In other embodiments, goals are always in force until modified.

[0144] SLAI 204 may automatically formulate dependencies between goals or may allow a user to specify the dependencies. For example, a user may request that a newly created tier come online. As a result of this goal, SLAI 204 may automatically direct task manager 312 to generate a task of harvesting a target

Netflix, Inc. - Ex. 1002, Page 000436
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

number of nodes to enable the tier. Generally, all goals remain in-force by SLAI 204 until modified by BLT 206. In one embodiment, each goal remains in-force in one of three states: Satisfied, Warning, or Critical depending on how successful SLAI 204 was in achieving the goal at the time the event record was generated and stored.

[0145] In this manner, SLAI 204 controls the life cycle of a goal (i.e., the creation, scheduling, update, deletion of the goal), and provides a common implementation of these and other services such as timeout, event writing, goal conflicts, management of intra-goal dependencies, and tracking tasks to achieving the goals.

[0146] Progress toward a goal is tracked though event subsystem 315. In particular, event subsystem 315 tracks the progress of each in force goal based on the goal identifiers. Tasks executed to achieve a particular goal produce events to communicate result or errors. The events provide a convenient time-based view of all actions and behaviors.

[0147] Examples of goal types that may be defined by administrator 20 include software image management goals, node allocation goals, harvest goals, tier capacity goals, asset requirement goals, tier online / offline goals, and data gathering goals.

[0148] In one embodiment, BLT 206 presents a task interface to SLAI 204 for the creation and management of specific tasks in order to achieve the currently in force goals. In particular, rule engines 246 invoke the task interface based on evaluation of the defined rule sets in view of the expected state and actual state for the fabric. Example task interfaces include interfaces to: reserve node resources; query resources for a node slot; associate or disassociate an image with a node in a tier node slot; allocate, de-allocate, startup or shutdown a node; move a node to a tier; apply, remove or cycle power of a node; create a golden image; create or delete an image instance; and delete an activity, node or tier.

[0149] Report generator 302 provides an extensible mechanism for generating reports 314. Typical reports include image utilization reports that contain information with respect to the number of nodes running each software image, inventory reports detailing both the logical and physical aspects of the fabric, and system event reports showing all events that have occurred within the fabric.

Netflix, Inc. - Ex. 1002, Page 000437
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Report generator 302 gathers, localizes, formats and displays data into report form for presentation to the user. Report generator 302 may include one or more data gathering modules (not shown) that gather events in accordance with a schedule and update an events table to record the events. The data gathering modules may write the events in XML format.

[0150] FIG. 20 is a block diagram illustrating one embodiment of a rule engine 246 (FIG. 17). In the illustrated embodiment, rule engine 246 includes a rule compiler 344 and an execution engine 346. Each of rules 342 represents a unit of code that conforms to a rule language and expresses a set of triggering conditions and a set of implied actions. When the conditions are met, the actions are eligible to occur. The following is one example of a configuration rule:

```
rule checkTierLoad {
      Tier t where status != "overloaded";
      LoadParameter p where app == t.app && maxload <
      t.load;
} -> {
      modify t {
            status: "overloaded";
      };
}
```

When translated, this example rule marks a tier as overloaded if an application is implemented by the tier and the maximum specified load for the application has been exceeded. Another example rule for outputting a notification that a tier is overloaded and automatically invoking a task within BLT 206 to add a node is:

```
rule tierOverloadNotify {
      Tier t where status == "overloaded";
} -> {
      notify "Tier: " + t + "is overloaded.";
      BLT.addNode(f);
}
```

[0151] Rule compiler 344 compiles each of rules 344 and translates match conditions of the rules into a discrimination network that avoids redundant tests during rule execution. Execution engine 346 handles rule administration, object insertion and retrieval, rule invocation and execution of rule actions. In general, execution engine 346 first matches a current set of rules 342 against a current state

Netflix, Inc. - Ex. 1002, Page 000438
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

of working memory 348 and local objects 350. Execution engine 346 then collects all rules that match as well as the matched objects and selects a particular rule instantiation to fire. Next, execution engine 346 fires (executes) the instantiated rule and propagates any changes to working memory 348. Execution engine 346 repeats the process until no more matching rule instantiations can be found.

[0152] Firing of a rule typically produces a very small number of changes to working memory 348. This allows sophisticated rule engines to scale by retaining match state between cycles. Only the rules and rule instantiations affected by changes are updated, thereby avoiding the bulk of the matching process. One exemplary algorithm that may be used by execution engine 346 to handle the matching process includes the RETE algorithm that creates a decision tree that combines the patterns in all the rules and is intended to improve the speed of forward-chained rule system by limiting the effort required to re-compute a conflict set after a rule is fired. One example of a RETE algorithm is described in Forgy, C.L.: 1982, 'RETE: a fast algorithm for the many pattern/many object pattern match problem', Artificial Intelligence 19, 1737. Other alternatives include the TREAT algorithms, and LEAPS algorithm, as described by Miranker, D.P.: 'TREAT: A New and Efficient Match Algorithm for AI Production Systems'. ISBN 0934613710 Daniel P. Miranker, David A. Brant, Bernie Lofaso, David Gadbois: On the Performance of Lazy Matching in Production Systems. AAAI 1990: 685692.

[0153] FIG. 21 is a block diagram illustrating an alternative embodiment of SLAI 204 (FIG. 17). In FIG. 21, rules engines 246 has been replaced with a single rules engine 360 and the separate sensor subsystem 240 has been removed. In this embodiment, the functionality of sensor subsystem 240 has been folded into analysis subsystem 244 by defining sensors in terms of rules.

[0154] As described earlier, a sensor is a software module that listens to a stream of monitoring data produced by monitoring subsystem 202. When the sensor detects a certain pattern of data, the sensor takes some action. However, it is not always necessary for the sensor to be listening and responding. Rather there may be only specific times when SLAI 204 needs the information provided by the sensor.

Netflix, Inc. - Ex. 1002, Page 000439
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0155] As described in further detail below, in this embodiment SLAI 204 defines sensors be defined in terms of one or more rules. When certain preconditions are true, the sensor becomes active, i.e., enters a monitoring state. Thereafter, the sensor performs some action (i.e., "fires") whenever the sensor detects a certain pattern of data in the stream of information describing the actual state of the system 208. In particular, the sensor action changes or updates a model in working memory 348, and fires when one or more rules are satisfied. This model in working memory 348 reflects the actual state of the distributed computing system. Moreover, a sensor may be defined as a set of one or more forward-chaining rules linked to form an overall triggering sequence.

[0156] Whenever the preconditions defining the sensor are not true, the sensor becomes inactive, i.e., ceases monitoring the data.

[0157] For example, if administrator 20 wished to define a sensor that automatically updates a variable in working memory 348 when the status of a node changes for a particular tier node slot. Moreover, assume administrator 20 is only interested in this change when the node deployed to the slot is currently in the NODE_ENABLED or ONLINE states and the target state of the node is ONLINE, or when the target state is OFFLINE and the current state is ONLINE, or when the node has not been interacted with. In this case, administrator 20 could write a rule describing this sensor using the following pseudo-code:

```
sensor nodeStateSensor {
    NodeGoal goal;
    NodeConnectorIVO connector where .uid ==
goal#connectorUid;
    test (
            (goal#targetState > NODE_ENABLED &&
                goal#state >= NODE_ENABLED) ||
            (goal#targetState < NODE_STARTED &&
                goal#state > NODE_ENABLED) ||
            goal#hasInteracted == false
      );
} --> {
    update connector#state when "NodeOnlineStatus" changed;
}
```

[0158] This exemplary rule first defines a sensor called "nodeStateSensor." Second, the code describes the objects to which this sensor applies. In this case, a "NodeConnectorIVO" object is a particular tier node slot and a "NodeGoal" is a business goal defined by BLT 206. Thus, this rule specifies that the defined sensor applies to the particular tier node slot having unique identifier ("Uid") matching

Netflix, Inc. - Ex. 1002, Page 000440
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the unique identifier of a "NodeConnectorIVO" object. Third, the rule describes three alternative conditions in the "test" portion. Last, the rule contains the action section indicated by the portion after the arrow. In this case, the action specifies updating the state property of the connector in working memory 348 whenever the online status of the connector changes. The sensor continues to update the online status of the connector so long as the preconditions of the "test" portion are true.

[0159] Rule compiler 344 (FIG. 20) compiles the forward-chaining sensor-defining rules along with the configuration rules. As a result of compilation, analysis subsystem 244 generates a series of sensors 362 that monitor incoming actual state data 208. Furthermore, the compilation integrates the rules and the match conditions of the sensor-defining rules into the discrimination network. As before, execution engine 346 handles rule administration, object insertion and retrieval, rule invocation and execution of rule actions. Similarly, execution engine 346 matches rules against working memory and local objects. In this manner, sensor-defining rules may be created and acted upon by execution engine 346 as any other forward-chaining rule.

[0160] FIG. 22 is a screen illustration of an exemplary user interface 370 for creating and editing sensor-defining rules. System administrator 20 may use a text editor, such as Emacs, to create and maintain the rules. Administrator 20 may configure the text editor to format sensor rules with color and indentation. In one embodiment, the text editor saves the rules as a text file. Rule compiler 344 parses the text file and generates a discrimination network. In the example of FIG. 22, user interface 370 presents three sensor-defining rules: "nodeStateSensor" 372, "applicationParameterSensor" 374, and "loadAverageSensor" 376.

[0161] In this manner, administrator 20 may utilize sensor-defining rules to more easily govern the behavior of control node 12 and the distributed computing system. Administrator 20 need not to separately create and configure sensors as the sensor definition may be integrated within the forward-chaining rules engine that provides overall autonomic control. In addition, administrator 20 can use the same programming constructions to describe sensors and substantive rules.

[0162] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.

Netflix, Inc. - Ex. 1002, Page 000441
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS:

1. A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network; and

a control node that includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules,

wherein the rules define one or more sensors that when processed by the rules engine update a model of the application nodes based on status data received from the application nodes.

2. The distributed computing system of claim 1, wherein the rule engines are forward-chaining rule engines.

3. The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

4. The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

Netflix, Inc. - Ex. 1002, Page 000442
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

5. The distributed computing system of claim 4, wherein the monitoring subsystem comprises a set of modular information collectors that collect the status data from the application nodes.

6. The distributed computing system of claim 5, wherein each of the information collectors conforms to a different communication protocol.

7. The distributed computing system of claim 5, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

8. The distributed computing system of claim 5, wherein the automation unit registers collection requests with the monitoring subsystem, and the monitoring subsystem includes a monitoring engine that invokes the collectors in response to the collection requests.

9. The distributed computing system of claim 4, wherein the automation unit further comprises a sensor subsystem that receives the state data from the monitoring subsystem and alters the model of the application nodes in accordance with the state data and in accordance with the rules defining the sensors.

10. The distributed computing system of claim 9, wherein the sensor subsystem communicates the modified model to the rule engines in accordance with a schedule.

11. The distributed computing system of claim 4, an effector subsystem that outputs action requests to the business logic tier in response to application of the rules by the rule engines.

12. The distributed computing system of claim 4, wherein the business logic tier includes a set of web services to control the application nodes.

Netflix, Inc. - Ex. 1002, Page 000443
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

13. The distributed computing system of claim 4, wherein the business logic tier presents interfaces by which a user defines one or more goals for the expected state of the application nodes.

14. The distributed computing system of claim 13, wherein the business logic tier registers the goals with the automation unit for achievement by the rule engines in accordance with the rule set.

15. The distributed computing system of claim 1, wherein the rules are units of code that conform with a rule language.

16. The distributed computing system of claim 13, wherein the automation unit automatically formulates dependencies between the goals.

17. The distributed computing system of claim 13,

wherein the business logic tier further comprises a task manager to invoke tasks in response to requests from the automation unit, and

wherein the business logic tier includes an event subsystem that tracks the progress of each of the tasks.

18. The distributed computing system of claim 13, wherein the goals include at least one of software image management goals, node allocation goals, goals for harvesting nodes, tier capacity goals, asset requirement goals and goals for gathering the status data.

19. The distributed computing system of claim 1, wherein the control node presents interfaces by which a user defines the sensors.

Netflix, Inc. - Ex. 1002, Page 000444
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

20. The distributed computing system of claim 4, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines automatically produce action requests to configure the application nodes in accordance with a hierarchical organization defined by the model.

21. The distributed computing system of claim 20, wherein each tier represents a different computing function.

22. The distributed computing system of claim 1, wherein each of the rules engines comprise:

a rule compiler to generate a discrimination network based on a set of match conditions supplied by a set of rules;

a set of sensors to update a model of the actual state of the distributed computing system, wherein the set of rules defines each sensor in the set of sensors and wherein each sensor only updates the model when the discrimination network indicates that a set of match conditions associated with the sensor have been satisfied;

a memory to store the model of the actual state; and

an execution engine to perform an action associated with a rule in the set of rules, wherein the discrimination network indicates that the exists a set of match conditions associated with the rule where all match conditions in the set of match conditions are satisfied.

Netflix, Inc. - Ex. 1002, Page 000445
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

23. A method comprising:

receiving input that defines an expected state for a distributed computing system having a plurality of application nodes interconnected via a communications network;

receiving status data from the distributed computing system regarding an actual state of the nodes;

processing sensor-defining rules with a set of rule engines to determine operations for updating a model of the actual state of the nodes, based on received status data;

processing rules with the set of rule engines to determine operations for reducing any difference between the expected state and the actual state as interpreted by the sensor-defining rules; and

applying the operations to the distributed computing system to control the application nodes in accordance with the rules.

24. The method of claim 23, wherein processing rules comprises processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

25. The method of claim 23, further comprising collecting the status data from the application nodes with a plurality of modular information collectors.

26. The method of claim 23, further comprising:

registering collection requests with a monitoring subsystem; and

executing a monitoring engine that invokes the collectors in response to the collection requests.

27. The method of claim 23, further comprising:

aggregating the state data in accordance with a hierarchical model of deployment of the application nodes; and

processing the aggregated state data with the rule engines.

Netflix, Inc. - Ex. 1002, Page 000446
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

28. The method of claim 23, further comprising:

presenting an interface by which a user defines the expected state as one or more goals; and

registering the goals with the rule engines.

29. The method of claim 28, further comprising:

storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

automatically producing the operations with the rule engines to configure the application nodes in accordance with a hierarchical organization defined by the model.

30. The method of claim 23, further comprising:

presenting an interface by which a user provides the sensor-defining rules; and

registering the sensor-defining rules with the rule engines.

31. The method of claim 23, wherein processing sensor-defining rules comprises:

providing the sensor-defining rules to a compiler, wherein each sensor-defining rule comprises a match condition and an action;

receiving a discrimination network from the compiler based on the match condition of each sensor-defining rule;

referencing the discrimination network an a model of the current state of the distributed computing system to determine a set of active sensors;

updating the model of the current state of the distributed computing system with the action specified by a sensor in the set of active sensors when the status data matches a pattern associated with the sensor in the set of active sensors.

Netflix, Inc. - Ex. 1002, Page 000447
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

32. A computer-readable medium comprising instructions that cause a programmable processor to:

process sensor-defining rules with a set of forward-chaining rule engines to determine the conditions under which to listen for a pattern in a stream of data from components of a distributed computing system and how to modify a model of an actual state of the distributed computing system a what action to perform when the pattern occurs;

process rules with the set of forward-chaining rule engines to determine operations for reducing differences between the model of the actual state of the distributed computing system and an expected state based on the action performed as a result of a sensor-defining rule; and

apply the operations to the distributed computing system to provide autonomic control of the application nodes in accordance with the rules.

Netflix, Inc. - Ex. 1002, Page 000448
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

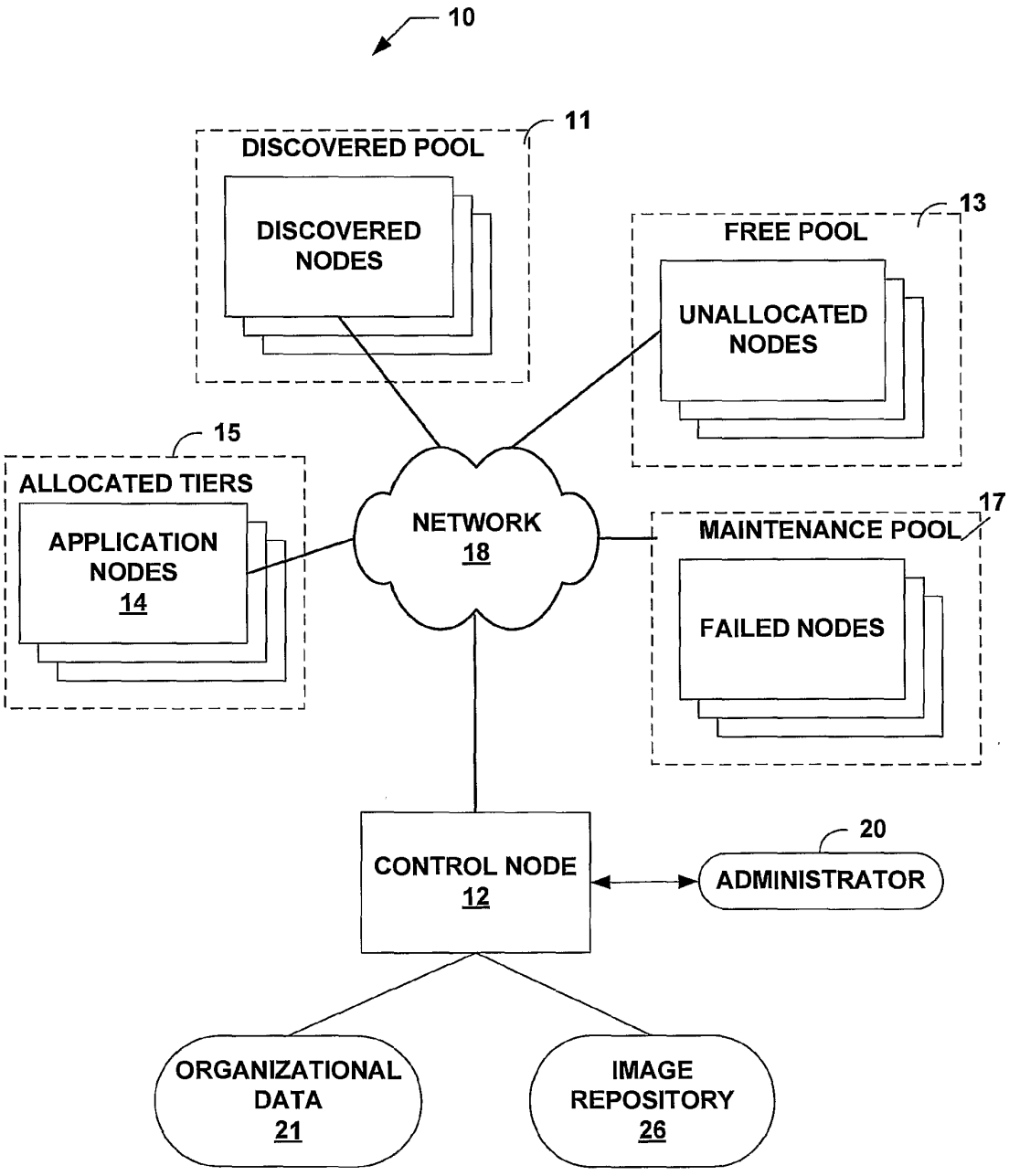

10
11
DISCOVERED POOL
DISCOVERED NODES
13
FREE POOL
UNALLOCATED NODES
15
ALLOCATED TIERS
APPLICATION NODES 14
NETWORK 18
17
MAINTENANCE POOL
FAILED NODES
CONTROL NODE 12
20
ADMINISTRATOR
ORGANIZATIONAL DATA 21
IMAGE REPOSITORY 26

FIG. 1

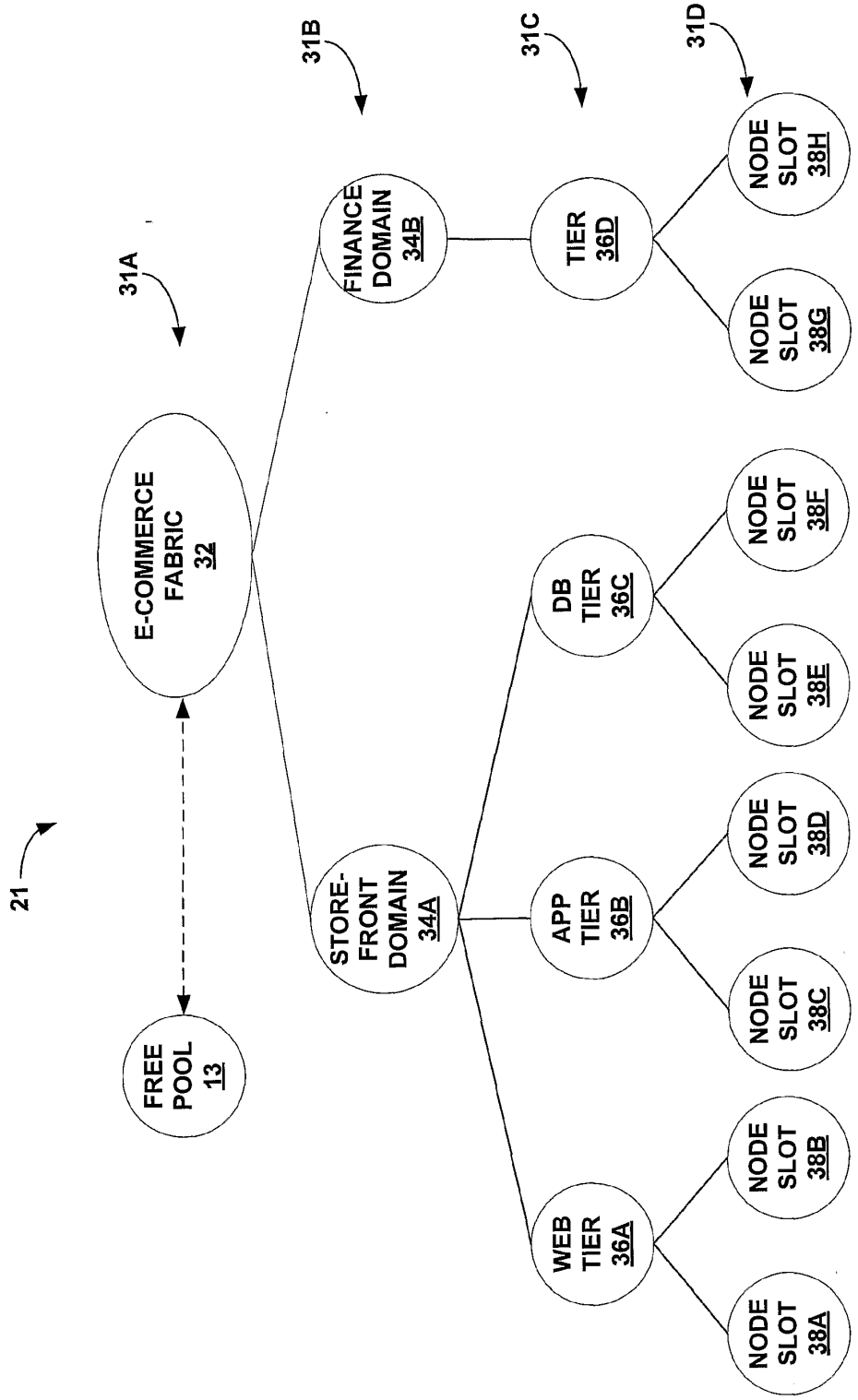

21
E-COMMERCE FABRIC 32
FREE POOL 13
31A
31B
31C
31D
FINANCE DOMAIN 34B
STORE-FRONT DOMAIN 34A
TIER 36D
DB TIER 36C
APP TIER 36B
WEB TIER 36A
NODE SLOT 38H
NODE SLOT 38G
NODE SLOT 38F
NODE SLOT 38E
NODE SLOT 38D
NODE SLOT 38C
NODE SLOT 38B
NODE SLOT 38A

FIG. 2

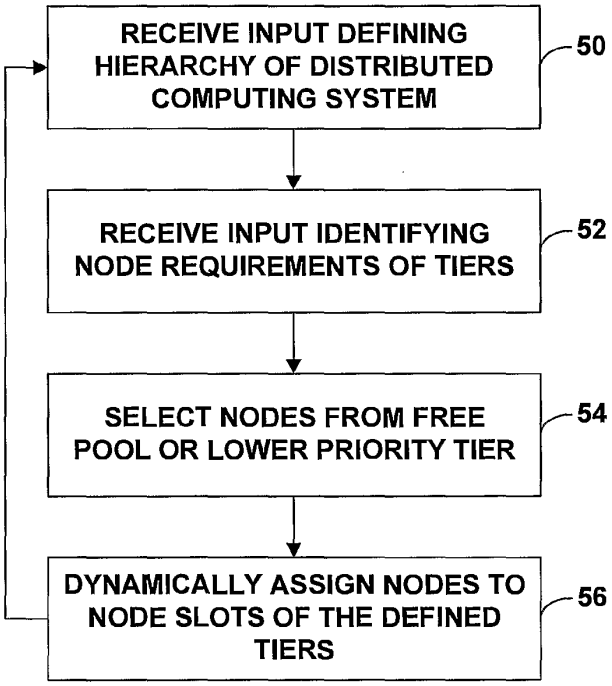

RECEIVE INPUT DEFINING HIERARCHY OF DISTRIBUTED COMPUTING SYSTEM
50
RECEIVE INPUT IDENTIFYING NODE REQUIREMENTS OF TIERS
52
SELECT NODES FROM FREE POOL OR LOWER PRIORITY TIER
54
DYNAMICALLY ASSIGN NODES TO NODE SLOTS OF THE DEFINED TIERS
56

FIG. 3

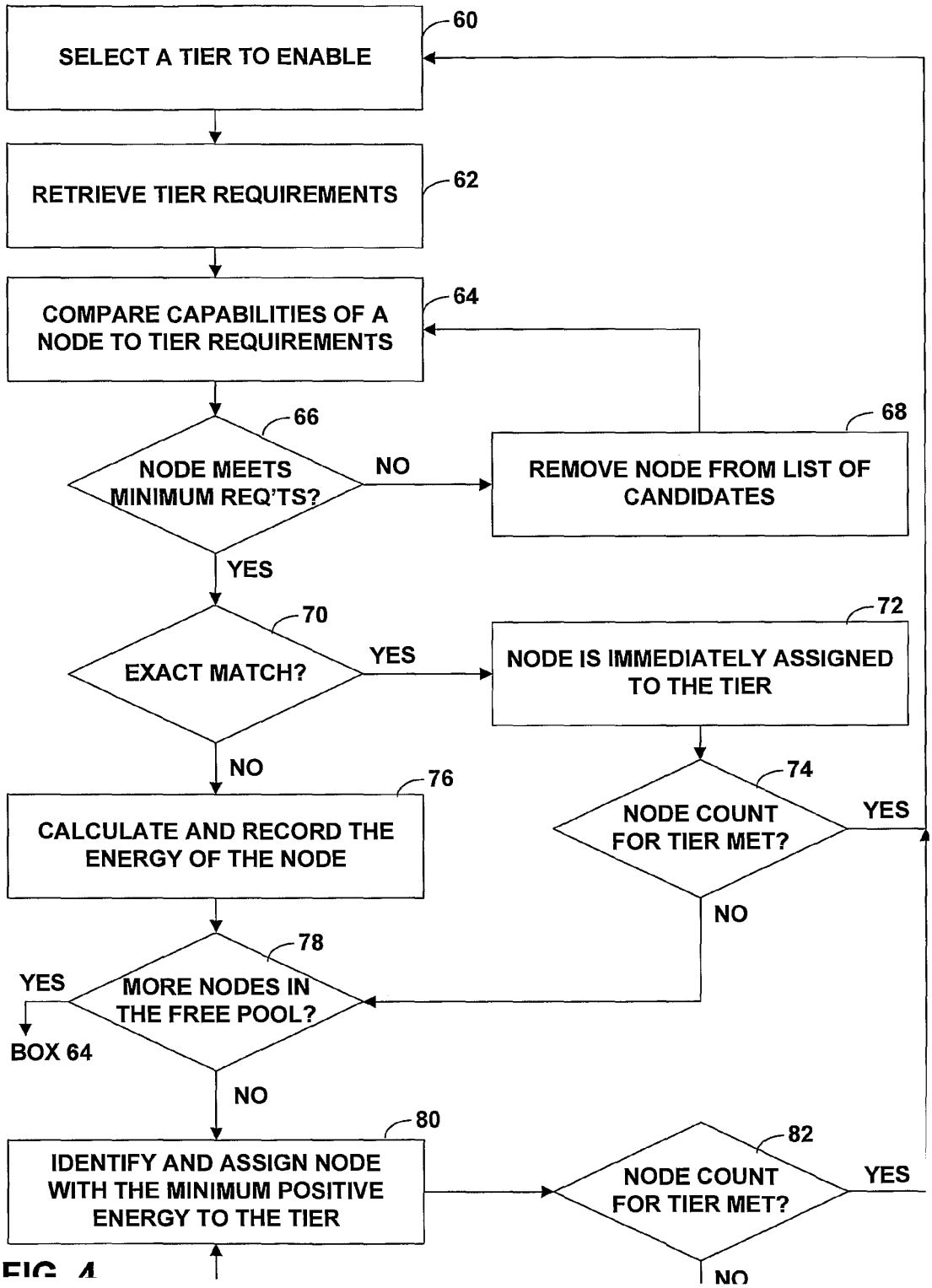

SELECT A TIER TO ENABLE
60
RETRIEVE TIER REQUIREMENTS
62
COMPARE CAPABILITIES OF A NODE TO TIER REQUIREMENTS
64
NODE MEETS MINIMUM REQ'TS?
66
NO
REMOVE NODE FROM LIST OF CANDIDATES
68
YES
EXACT MATCH?
70
YES
NODE IS IMMEDIATELY ASSIGNED TO THE TIER
72
NO
CALCULATE AND RECORD THE ENERGY OF THE NODE
76
NODE COUNT FOR TIER MET?
74
YES
NO
MORE NODES IN THE FREE POOL?
78
YES
BOX 64
NO
IDENTIFY AND ASSIGN NODE WITH THE MINIMUM POSITIVE ENERGY TO THE TIER
80
NODE COUNT FOR TIER MET?
82
YES
NO

FIG. 4

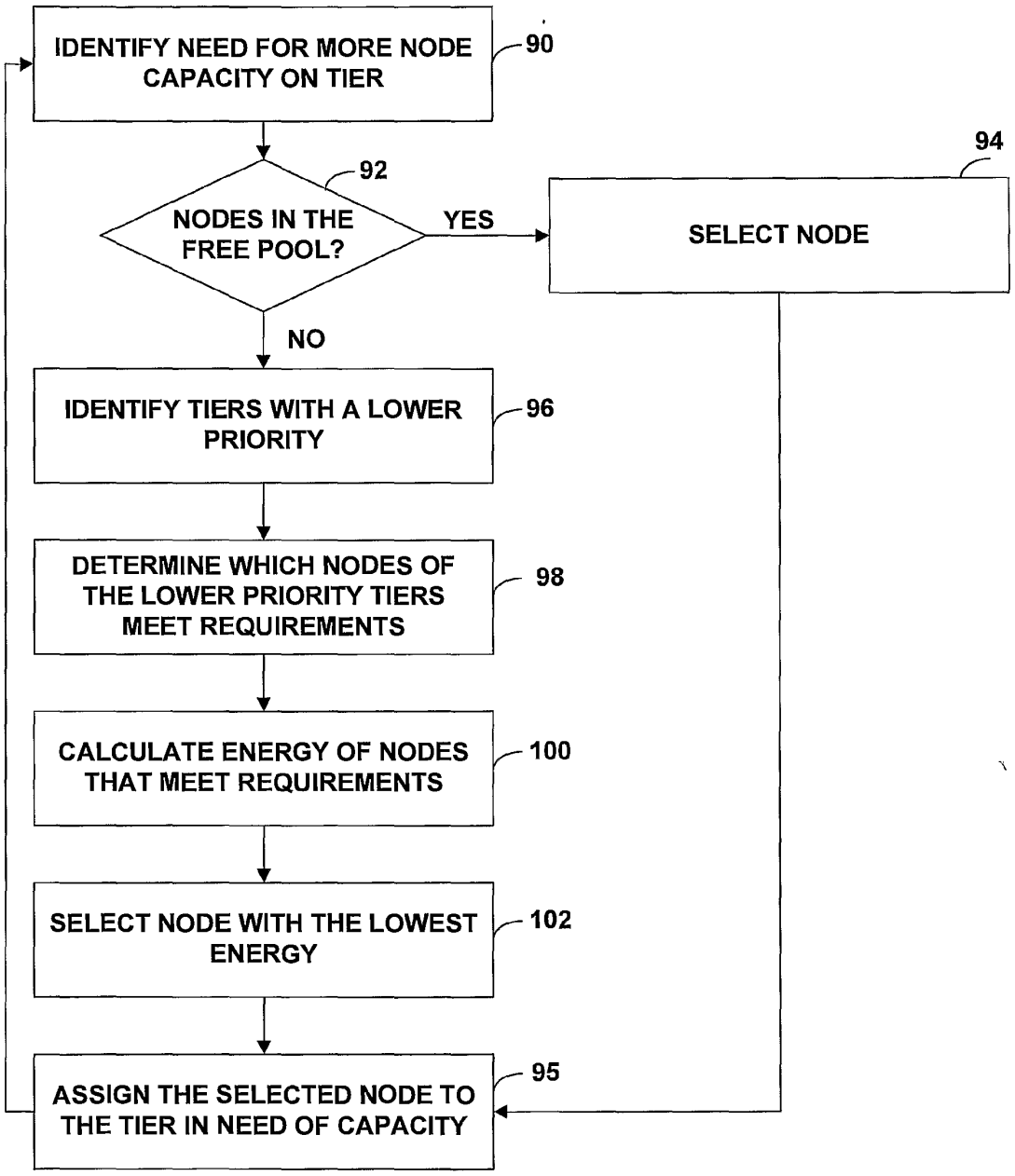

IDENTIFY NEED FOR MORE NODE CAPACITY ON TIER
90
92
NODES IN THE FREE POOL?
YES
94
SELECT NODE
NO
IDENTIFY TIERS WITH A LOWER PRIORITY
96
DETERMINE WHICH NODES OF THE LOWER PRIORITY TIERS MEET REQUIREMENTS
98
CALCULATE ENERGY OF NODES THAT MEET REQUIREMENTS
100
SELECT NODE WITH THE LOWEST ENERGY
102
ASSIGN THE SELECTED NODE TO THE TIER IN NEED OF CAPACITY
95

FIG. 5

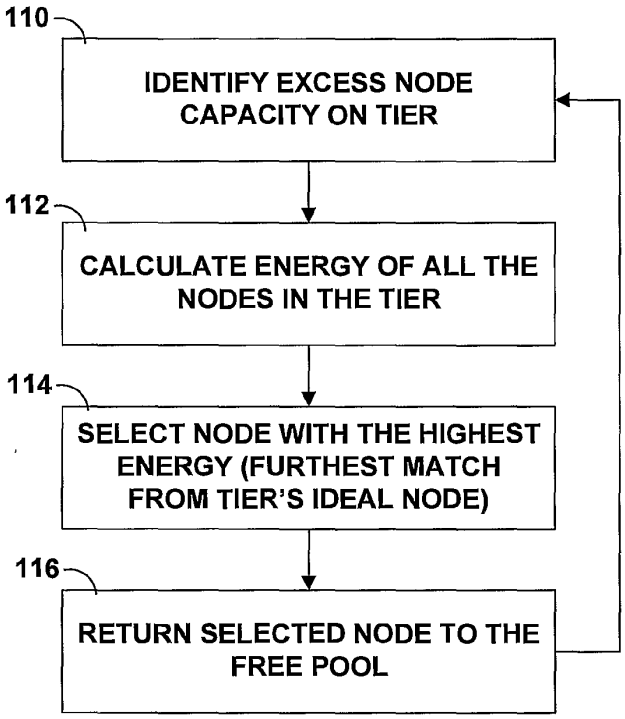

110
IDENTIFY EXCESS NODE CAPACITY ON TIER
112
CALCULATE ENERGY OF ALL THE NODES IN THE TIER
114
SELECT NODE WITH THE HIGHEST ENERGY (FURTHEST MATCH FROM TIER'S IDEAL NODE)
116
RETURN SELECTED NODE TO THE FREE POOL

FIG. 6

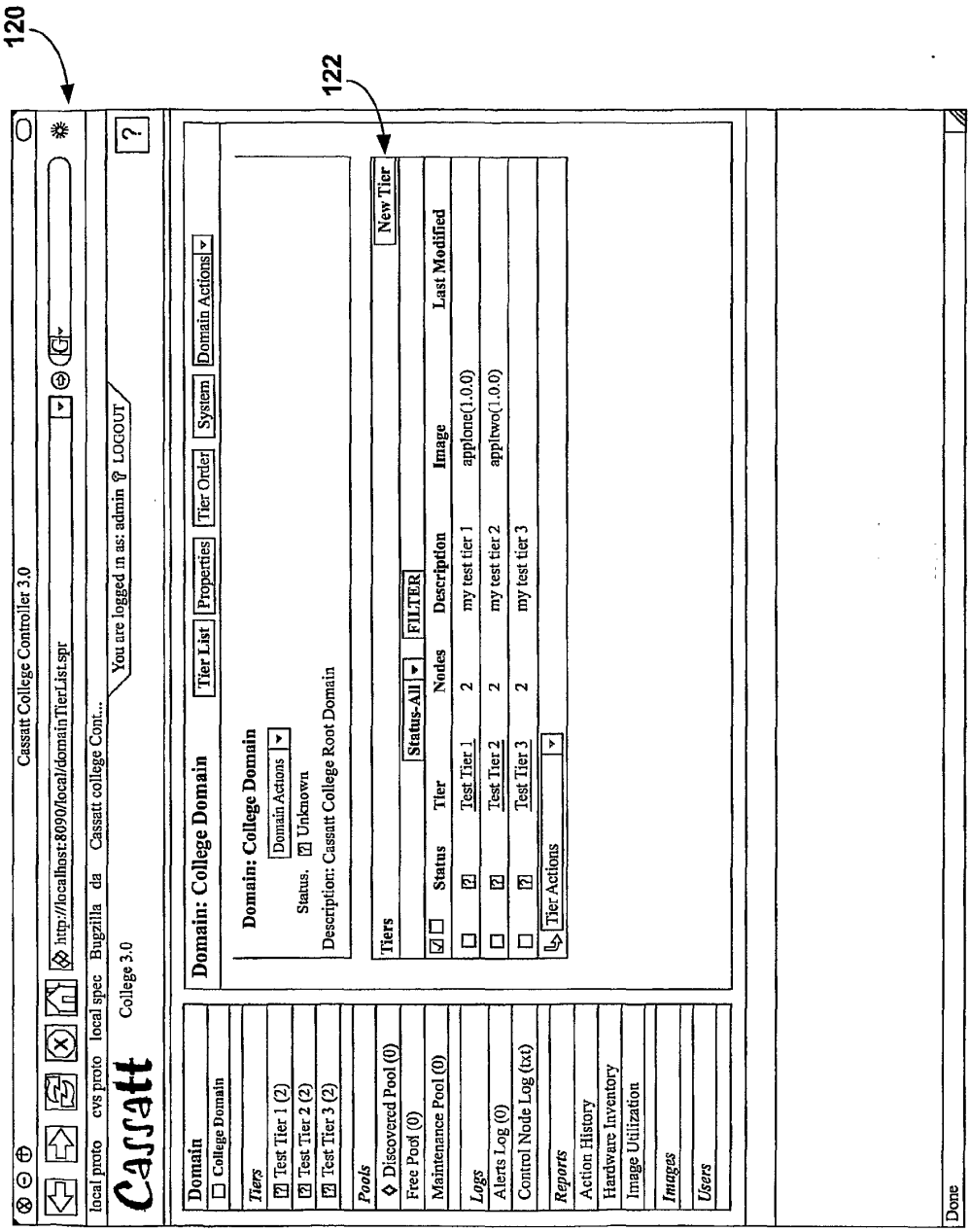

120
122
Cassatt College Controller 3.0
http://localhost:8090/local/domainTierList.spr
local proto
cvs proto
local spec
Bugzilla
da
Cassatt college Cont...
Cassatt
College 3.0
You are logged in as: admin LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Domain: College Domain
Tier List
Properties
Tier Order
System
Domain Actions
Domain: College Domain
Domain Actions
Status: Unknown
Description: Cassatt College Root Domain
Tiers
New Tier
Status-All
FILTER
Status
Tier
Nodes
Description
Image
Last Modified
Test Tier 1
2
my test tier 1
applone(1.0.0)
Test Tier 2
2
my test tier 2
applwo(1.0.0)
Test Tier 3
2
my test tier 3
Tier Actions
Done

FIG. 7

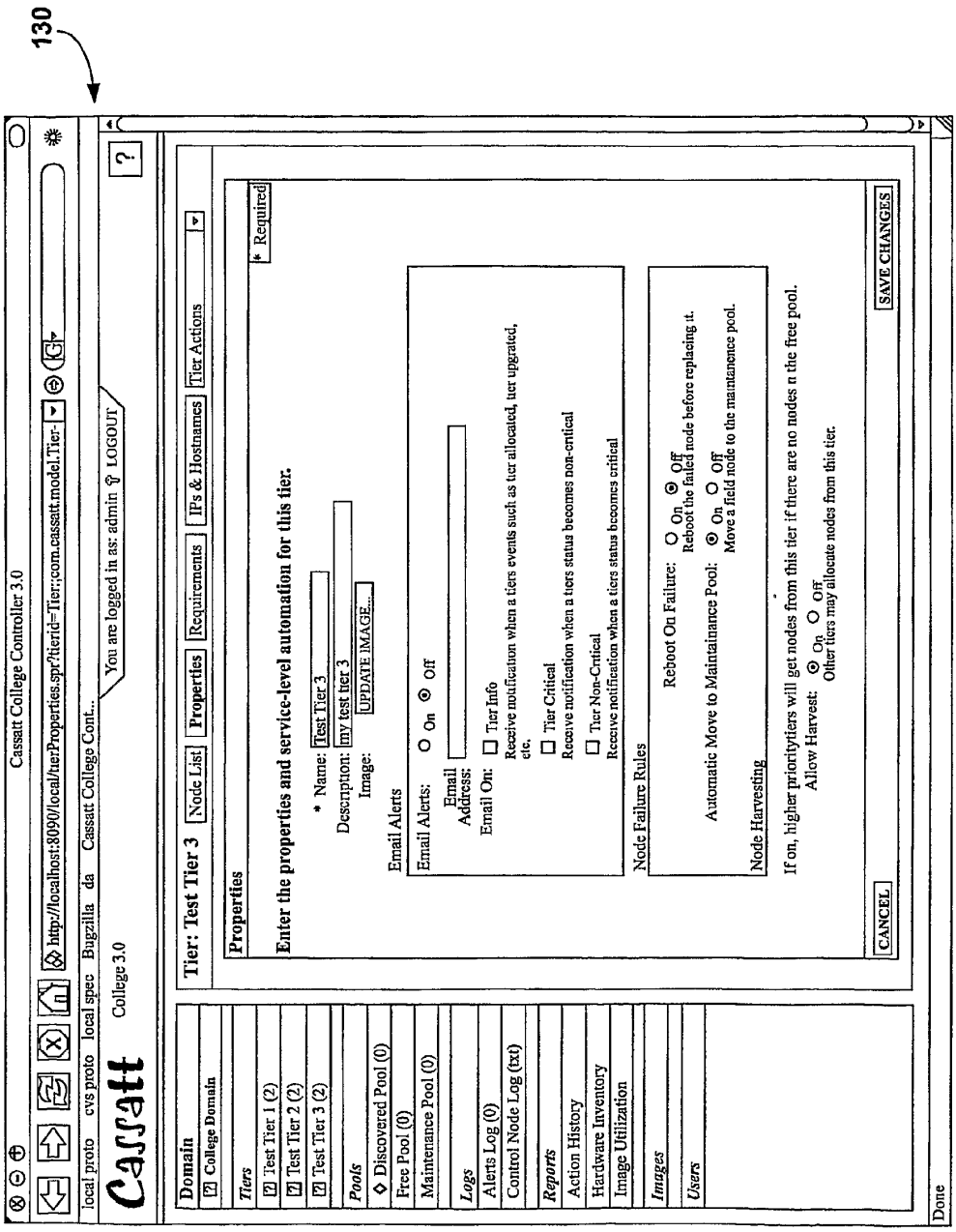

130
Cassatt College Controller 3.0
http://localhost:8090/local/tierProperties.spr?tierid=Tier:;com.cassatt.model.Tier:
local proto
cvs proto
local spec
Bugzilla
da
Cassatt College Cont...
Cassatt
College 3.0
You are logged in as: admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Tier: Test Tier 3
Node List
Properties
Requirements
IPs & Hostnames
Tier Actions
Properties
* Required
Enter the properties and service-level automation for this tier.
* Name: Test Tier 3
Description: my test tier 3
Image: UPDATE IMAGE...
Email Alerts
Email Alerts: On Off
Email Address:
Email On: Tier Info
Receive notification when a tiers events such as tier allocated, tier upgrated, etc.,
Tier Critical
Receive notification when a tiers status becomes non-critical
Tier Non-Critical
Receive notification when a tiers status becomes critical
Node Failure Rules
Reboot On Failure: On Off
Reboot the failed node before replacing it.
Automatic Move to Maintainance Pool: On Off
Move a field node to the maintanence pool.
Node Harvesting
If on, higher priority tiers will get nodes from this tier if there are no nodes n the free pool.
Allow Harvest: On Off
Other tiers may allocate nodes from this tier.
CANCEL
SAVE CHANGES
Done

FIG. 8

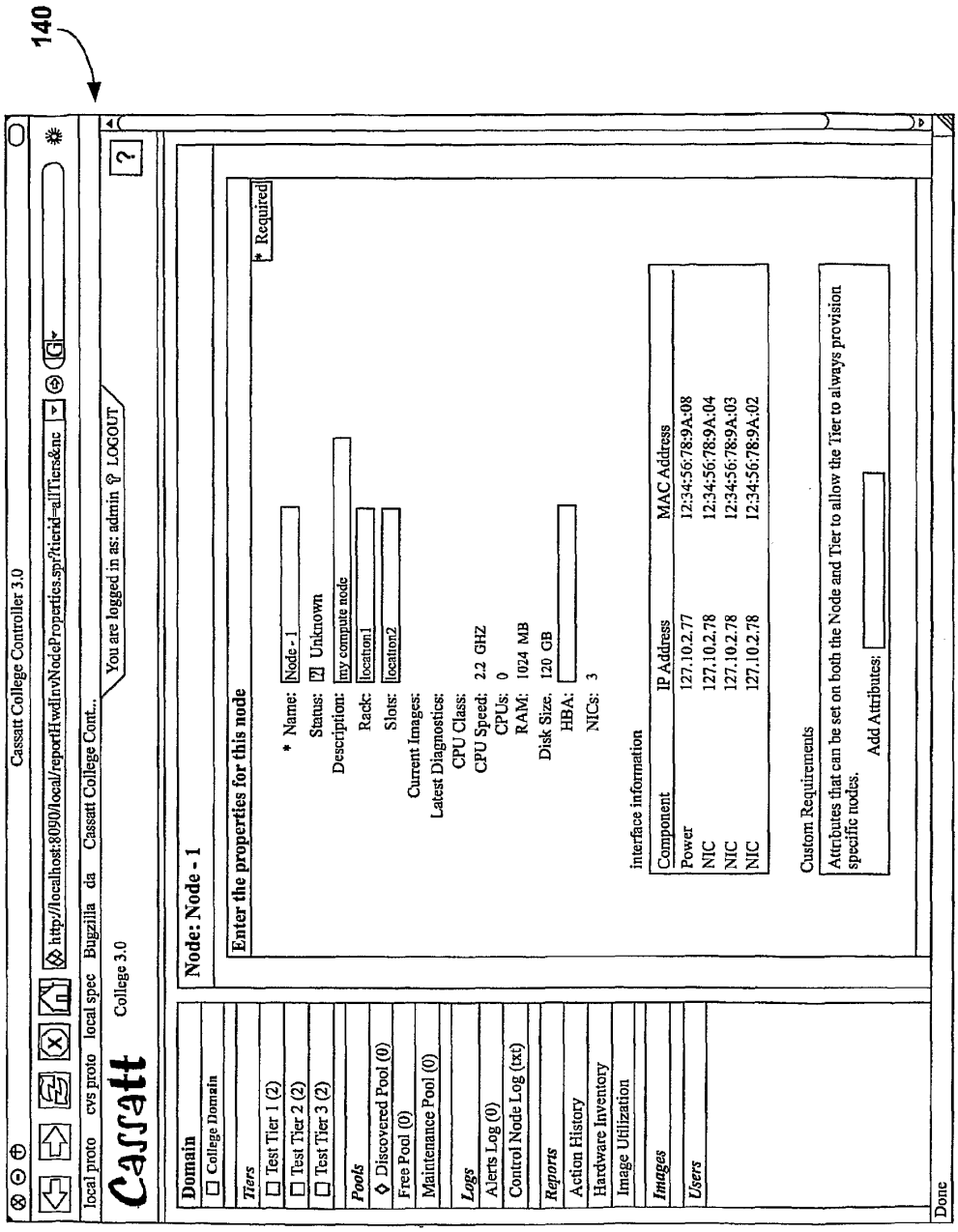

140
Cassatt College Controller 3.0
http://localhost:8090/local/reportHwdInvNodeProperties.spr?tierid=allTiers&nc
local proto
cvs proto
local spec
Bugzilla
da
Cassatt College Cont...
Cassatt
College 3.0
You are logged in as: admin LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Node: Node - 1
Enter the properties for this node
* Required
* Name: Node - 1
Status: Unknown
Description: my compute node
Rack: location1
Slots: location2
Current Images:
Latest Diagnostics:
CPU Class:
CPU Speed: 2.2 GHZ
CPUs: 0
RAM: 1024 MB
Disk Size: 120 GB
HBA:
NICs: 3
interface information
Component
IP Address
MAC Address
Power 127.10.2.77 12:34:56:78:9A:08
NIC 127.10.2.78 12:34:56:78:9A:04
NIC 127.10.2.78 12:34:56:78:9A:03
NIC 127.10.2.78 12:34:56:78:9A:02
Custom Requirements
Attributes that can be set on both the Node and Tier to allow the Tier to always provision specific nodes.
Add Attributes:
Done

FIG. 9

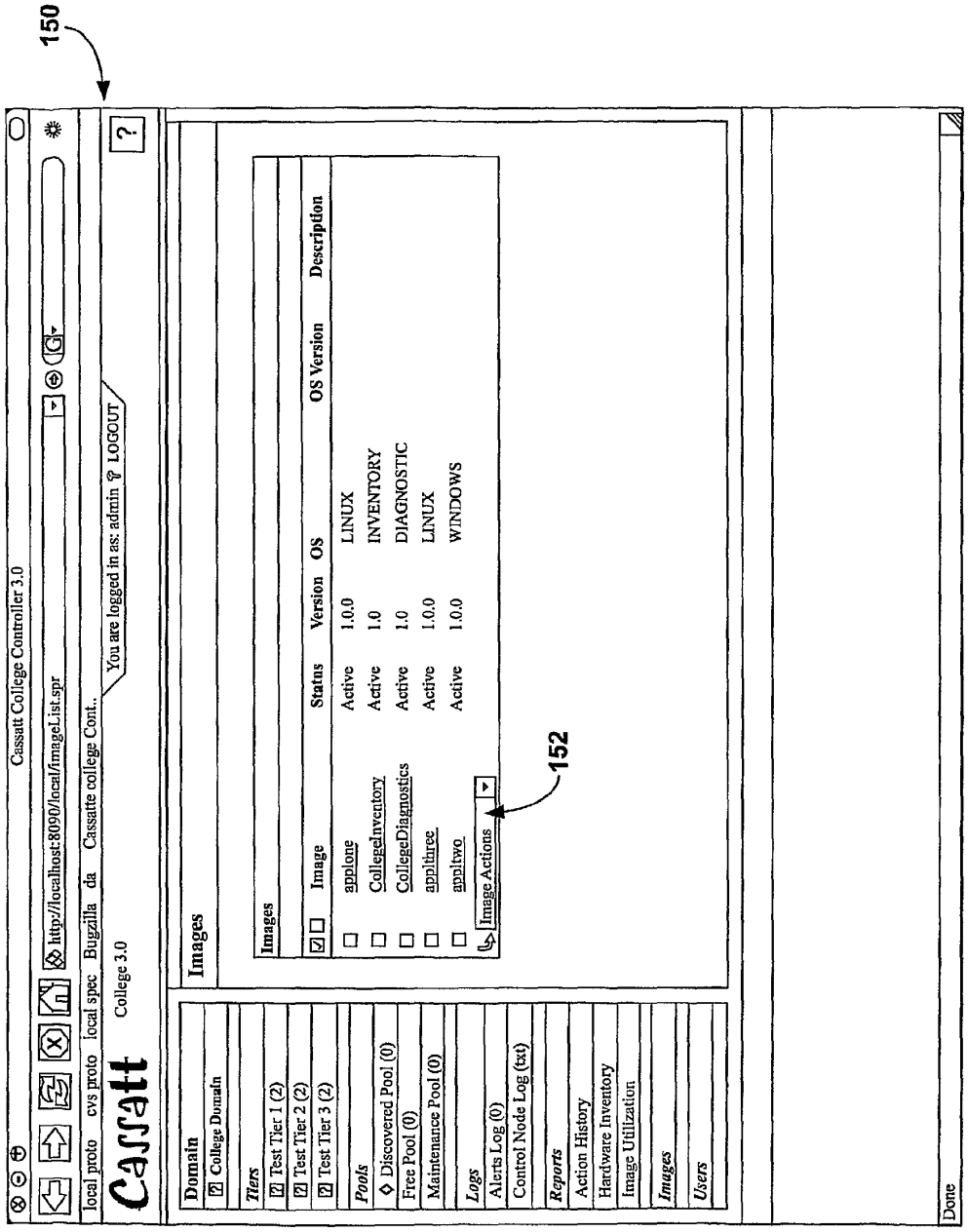

150
Cassatt College Controller 3.0
http://localhost:8090/local/imageList.spr
local proto
cvs proto
local spec
Bugzilla
da
Cassatte college Cont..
Cassatt
College 3.0
You are logged in as: admin
LOGOUT
?
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Images
Images
Image
Status
Version
OS
OS Version
Description
applone
Active
1.0.0
LINUX
CollegeInventory
Active
1.0
INVENTORY
CollegeDiagnostics
Active
1.0
DIAGNOSTIC
applthree
Active
1.0.0
LINUX
apptwo
Active
1.0.0
WINDOWS
Image Actions
152
Done

FIG. 10

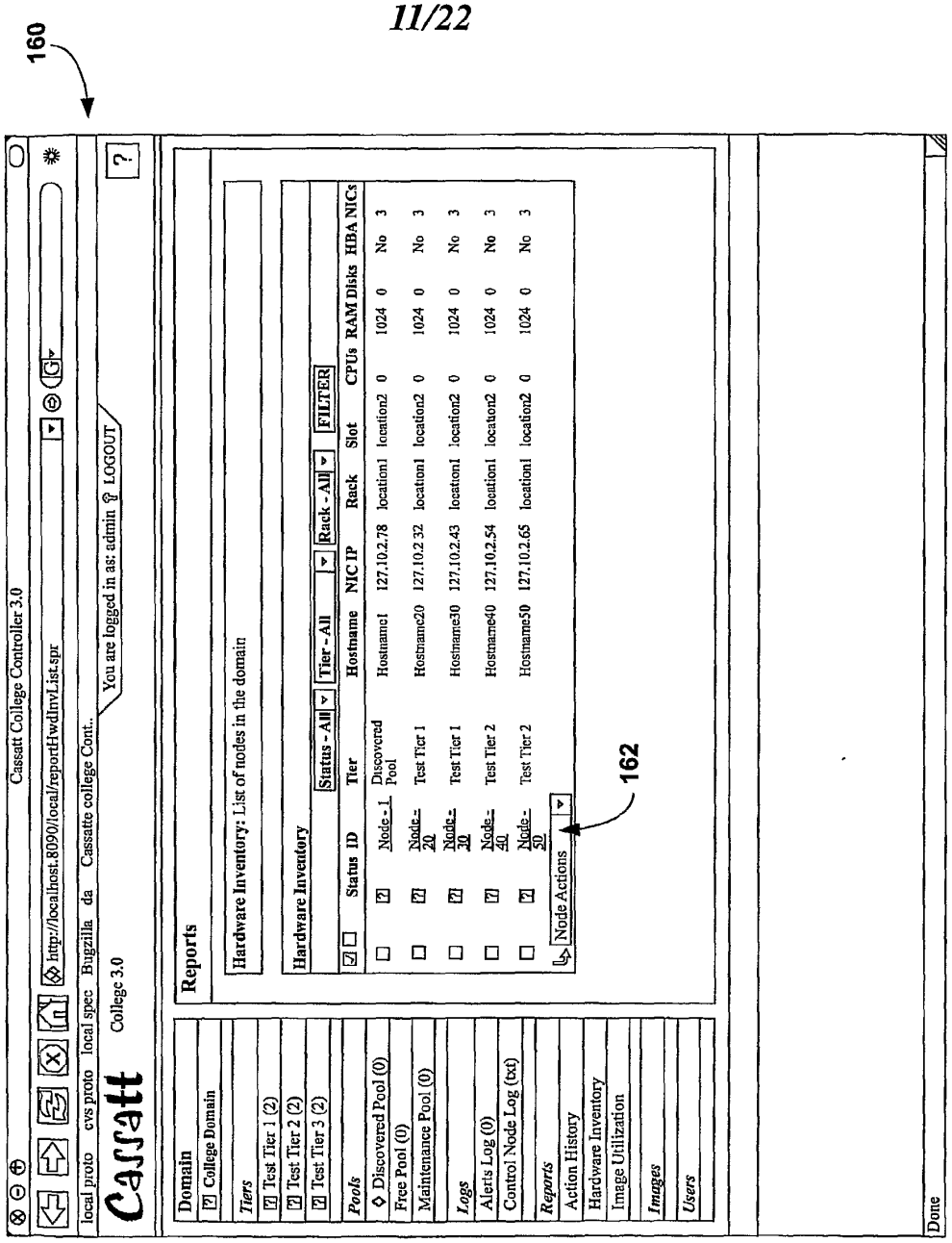

160
Cassatt College Controller 3.0
http://localhost:8090/local/reportHwdInvList.spr
local proto cvs proto local spec Bugzilla da Cassatte college Cont...
Cassatt College 3.0
You are logged in as: admin LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Reports
Hardware Inventory: List of nodes in the domain
Hardware Inventory
Status - All
Tier - All
Rack - All
FILTER
Status ID Tier Hostname NIC IP Rack Slot CPUs RAM Disks HBA NICs
Node - 1 Discovered Pool Hostname1 127.10.2.78 location1 location2 0 1024 0 No 3
Node - 20 Test Tier 1 Hostname20 127.10.2.32 location1 location2 0 1024 0 No 3
Node - 30 Test Tier 1 Hostname30 127.10.2.43 location1 location2 0 1024 0 No 3
Node - 40 Test Tier 2 Hostname40 127.10.2.54 location1 location2 0 1024 0 No 3
Node - 50 Test Tier 2 Hostname50 127.10.2.65 location1 location2 0 1024 0 No 3
Node Actions
162
Done

FIG. 11

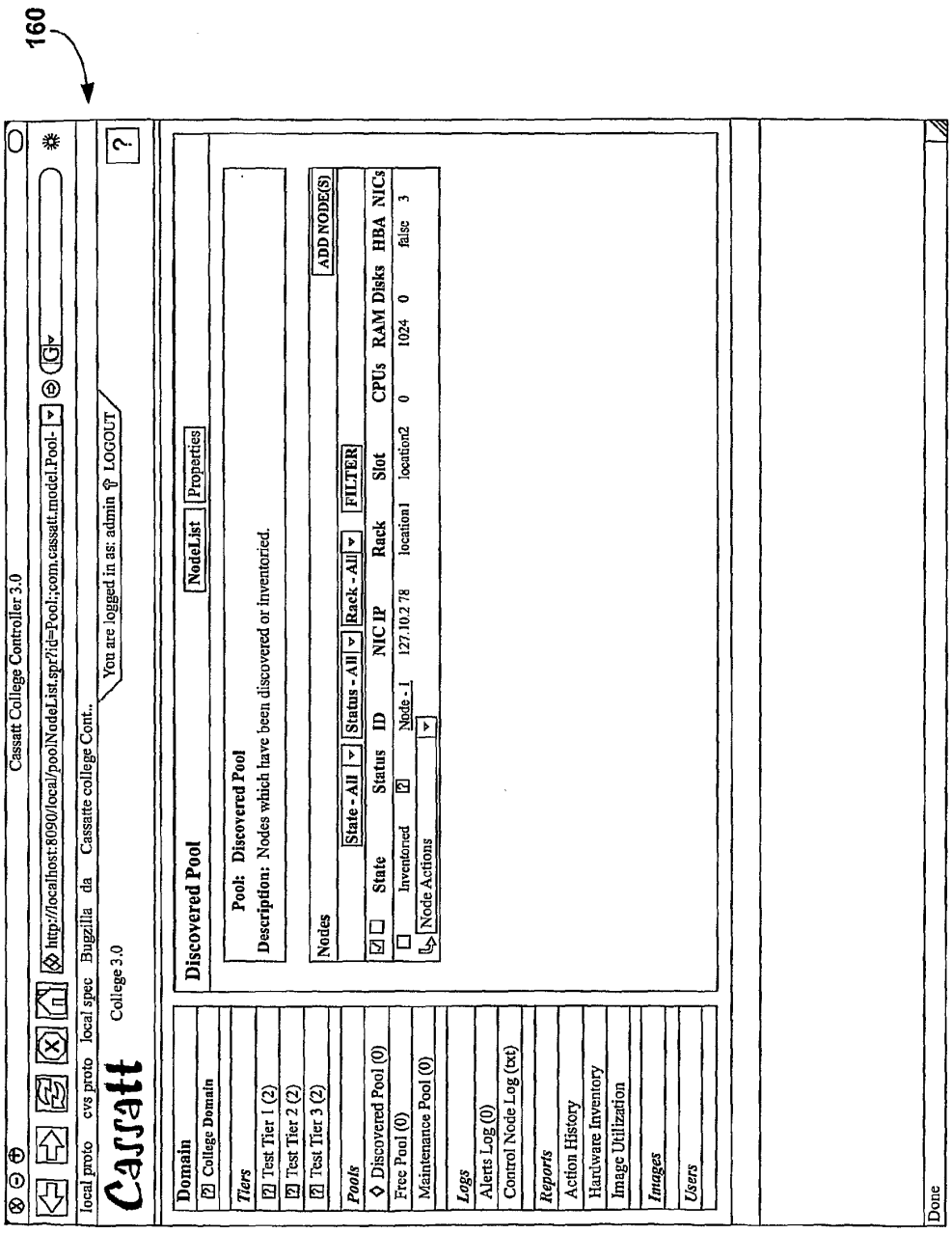

160
Cassatt College Controller 3.0
http://localhost:8090/local/poolNodeList.spr?id=Pool:;com.cassatt.model.Pool-
local proto cvs proto local spec Bugzilla da Cassatte college Cont..
Cassatt
College 3.0
You are logged in as: admin LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Discovered Pool
NodeList
Properties
Pool: Discovered Pool
Description: Nodes which have been discovered or inventoried.
Nodes
ADD NODE(S)
State - All
Status - All
Rack - All
FILTER
State
Status
ID
NIC IP
Rack
Slot
CPUs
RAM
Disks
HBA
NICs
Inventoried
Node - 1
127.10.2.78
location1
location2
0
1024
0
false
3
Node Actions
Done

FIG. 12

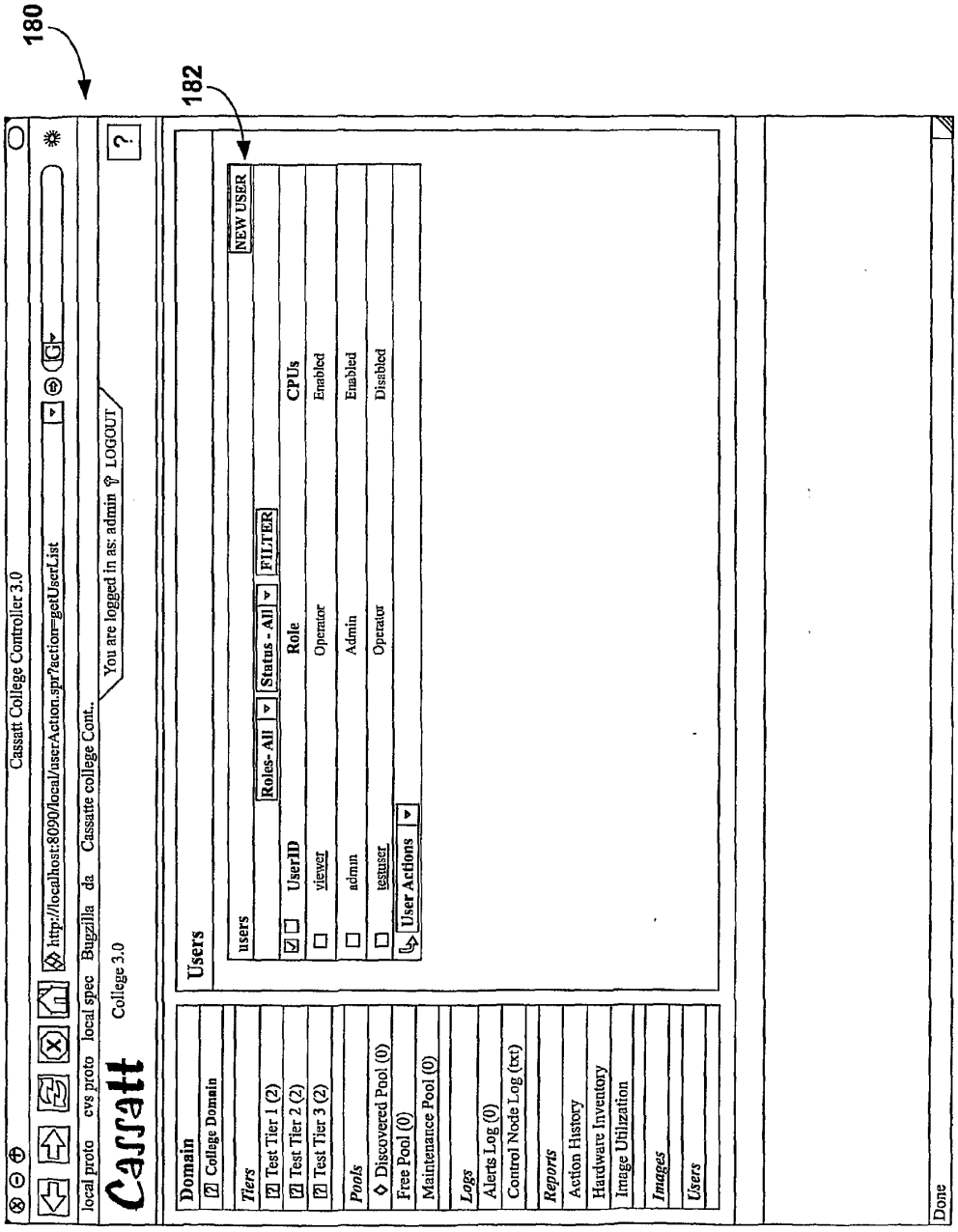

180
182
Cassatt College Controller 3.0
http://localhost:8090/local/userAction.spr?action=getUserList
local proto
cvs proto
local spec
Bugzilla
da
Cassatte college Cont..
Cassatt
College 3.0
You are logged in as: admin
LOGOUT
?
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Users
users
Roles- All
Status - All
FILTER
NEW USER
UserID
Role
CPUs
viewer
Operator
Enabled
admin
Admin
Enabled
testuser
Operator
Disabled
User Actions
Done

FIG. 13

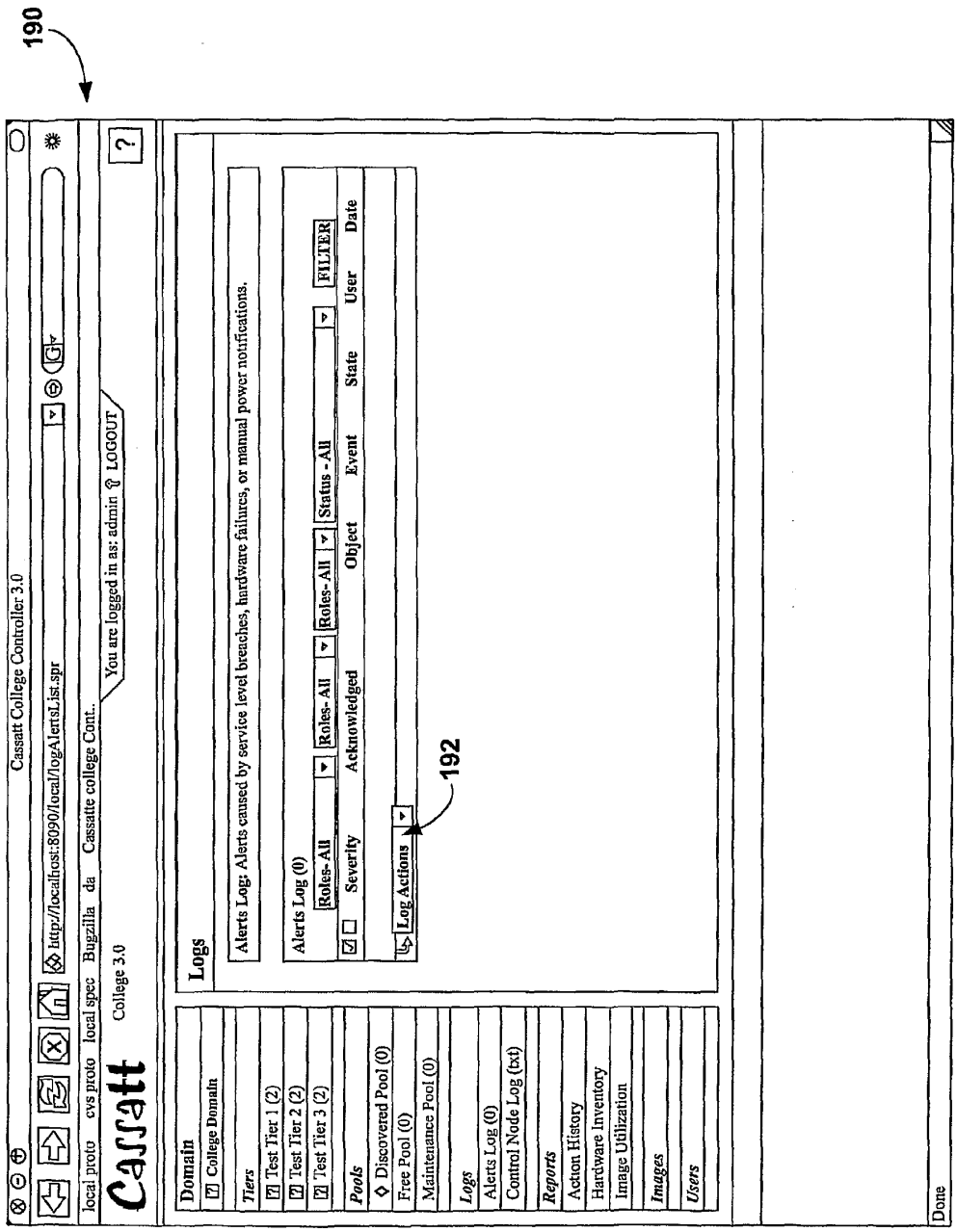

190
Cassatt College Controller 3.0
http://localhost:8090/local/logAlertsList.spr
local proto cvs proto local spec Bugzilla da Cassatte college Cont..
Cassatt
College 3.0
You are logged in as: admin LOGOUT
?
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Logs
Alerts Log: Alerts caused by service level breaches, hardware failures, or manual power notifications.
Alerts Log (0)
Roles- All
Roles- All
Roles- All
Status - All
FILTER
Severity
Acknowledged
Object
Event
State
User
Date
Log Actions
192
Done

FIG. 14

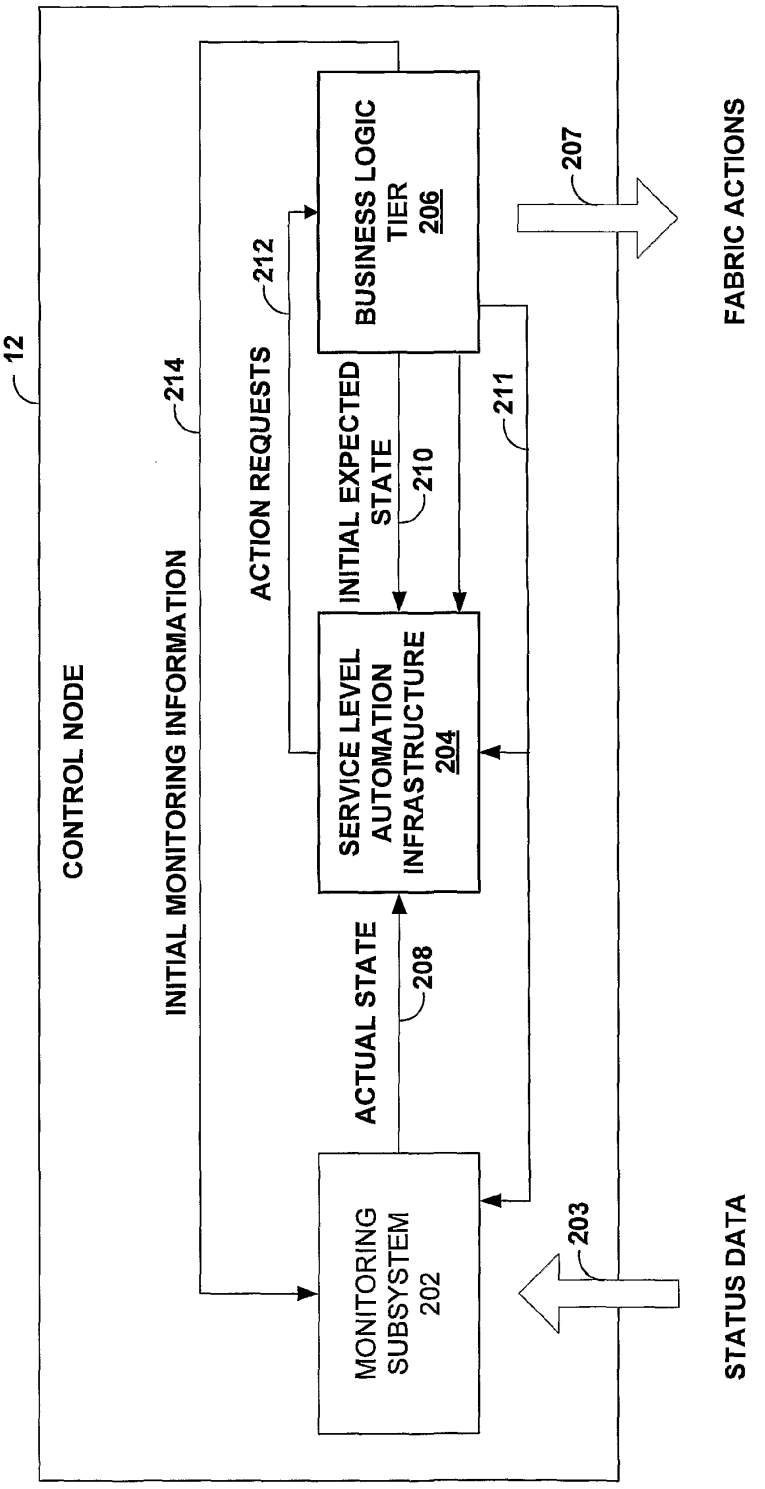

12
CONTROL NODE
214
INITIAL MONITORING INFORMATION
212
ACTION REQUESTS
BUSINESS LOGIC TIER 206
207
FABRIC ACTIONS
INITIAL EXPECTED STATE
210
211
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
ACTUAL STATE
208
MONITORING SUBSYSTEM 202
203
STATUS DATA

FIG. 15

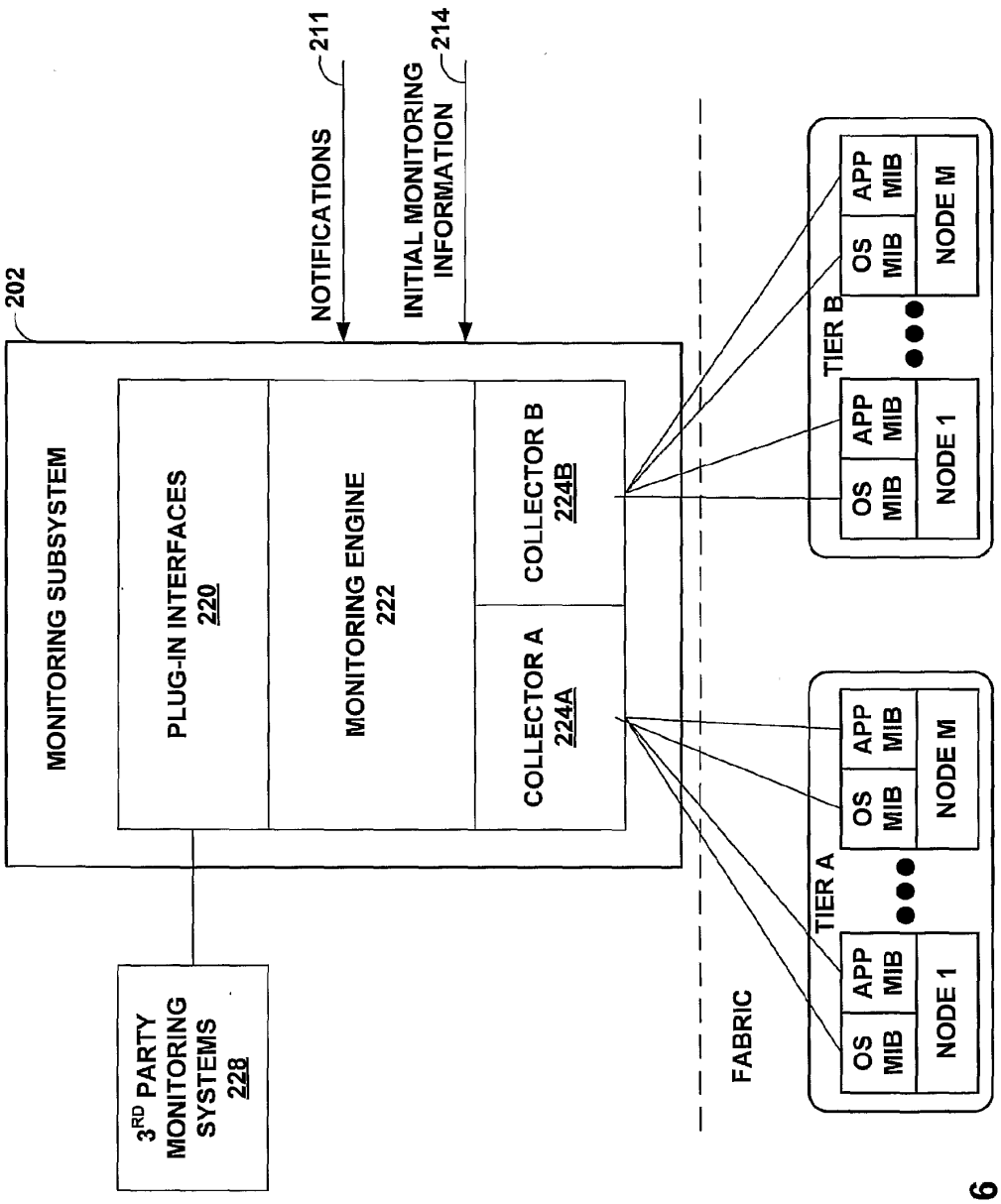

202
NOTIFICATIONS
211
INITIAL MONITORING INFORMATION
214
MONITORING SUBSYSTEM
PLUG-IN INTERFACES 220
MONITORING ENGINE 222
COLLECTOR A 224A
COLLECTOR B 224B
3RD PARTY MONITORING SYSTEMS 228
FABRIC
TIER A
TIER B
OS MIB
APP MIB
NODE 1
NODE M

FIG. 16

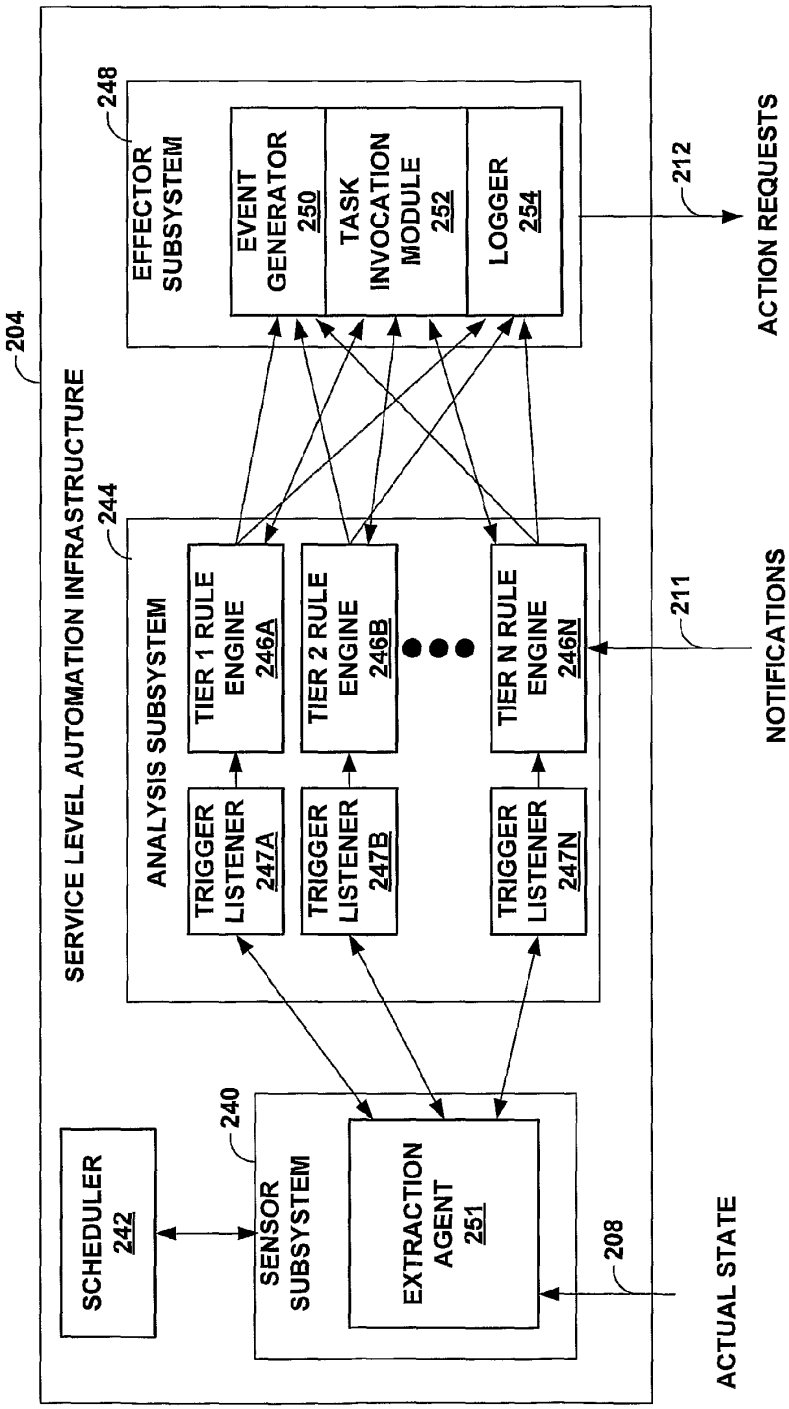

204
SERVICE LEVEL AUTOMATION INFRASTRUCTURE
248
EFFECTOR SUBSYSTEM
EVENT GENERATOR 250
TASK INVOCATION MODULE 252
LOGGER 254
212
ACTION REQUESTS
244
ANALYSIS SUBSYSTEM
TIER 1 RULE ENGINE 246A
TIER 2 RULE ENGINE 246B
TIER N RULE ENGINE 246N
TRIGGER LISTENER 247A
TRIGGER LISTENER 247B
TRIGGER LISTENER 247N
211
NOTIFICATIONS
240
SENSOR SUBSYSTEM
SCHEDULER 242
EXTRACTION AGENT 251
208
ACTUAL STATE

FIG. 17

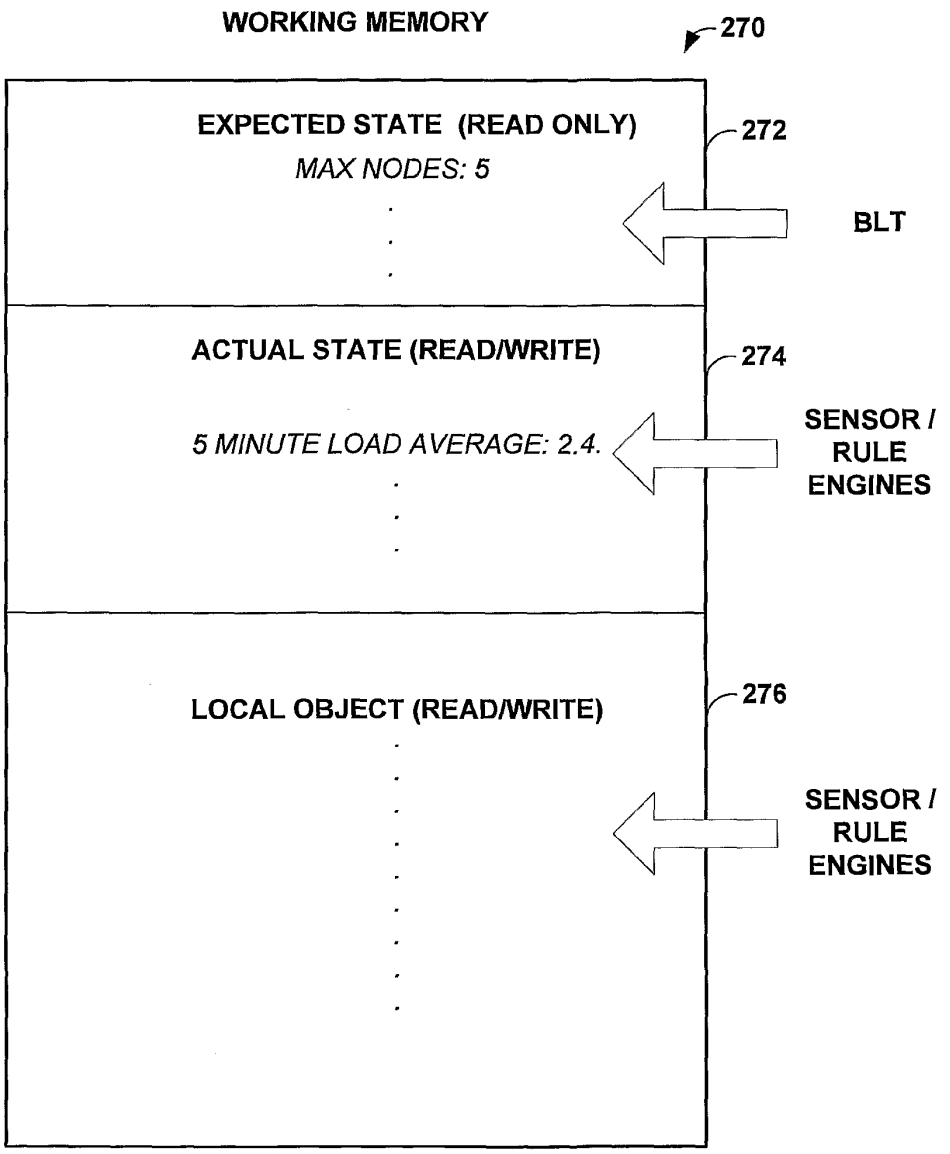

WORKING MEMORY
270
EXPECTED STATE (READ ONLY)
MAX NODES: 5
272
BLT
ACTUAL STATE (READ/WRITE)
274
5 MINUTE LOAD AVERAGE: 2.4.
SENSOR / RULE ENGINES
LOCAL OBJECT (READ/WRITE)
276
SENSOR / RULE ENGINES

FIG. 18

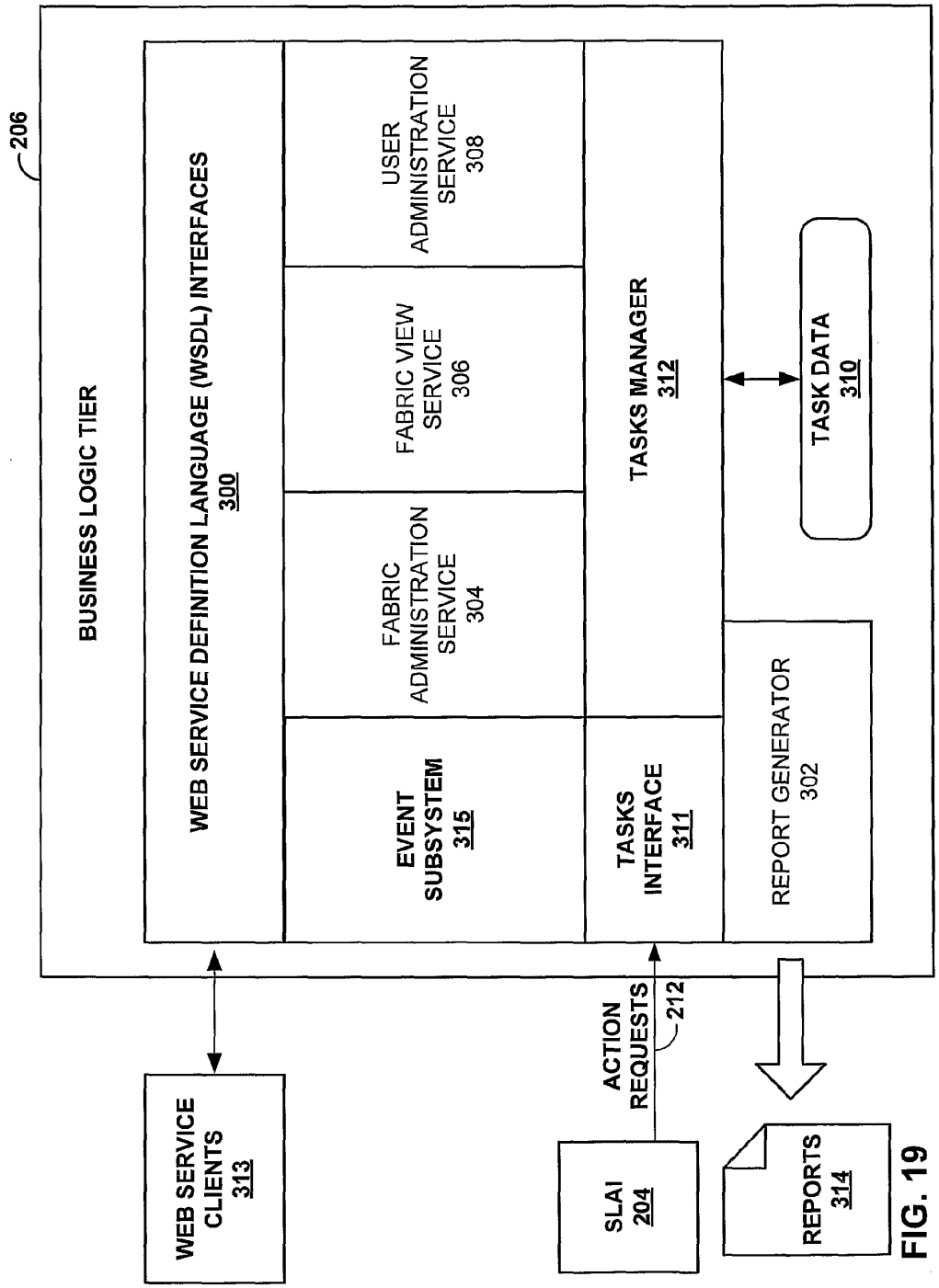

206
BUSINESS LOGIC TIER
WEB SERVICE DEFINITION LANGUAGE (WSDL) INTERFACES 300
FABRIC ADMINISTRATION SERVICE 304
FABRIC VIEW SERVICE 306
USER ADMINISTRATION SERVICE 308
EVENT SUBSYSTEM 315
TASKS MANAGER 312
TASKS INTERFACE 311
REPORT GENERATOR 302
TASK DATA 310
WEB SERVICE CLIENTS 313
SLAI 204
ACTION REQUESTS 212
REPORTS 314

FIG. 19

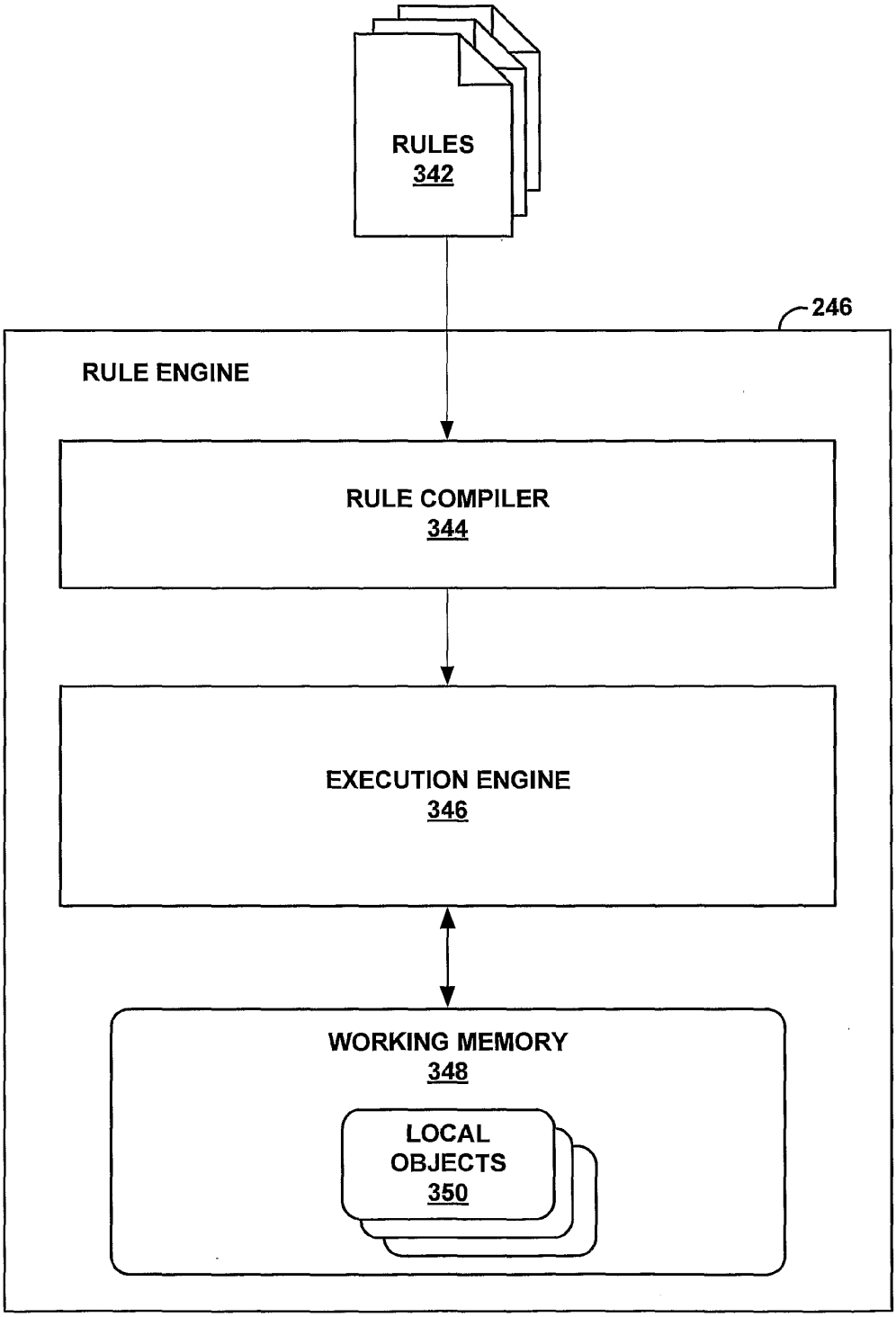

RULES
342
246
RULE ENGINE
RULE COMPILER
344
EXECUTION ENGINE
346
WORKING MEMORY
348
LOCAL
OBJECTS
350

FIG. 20

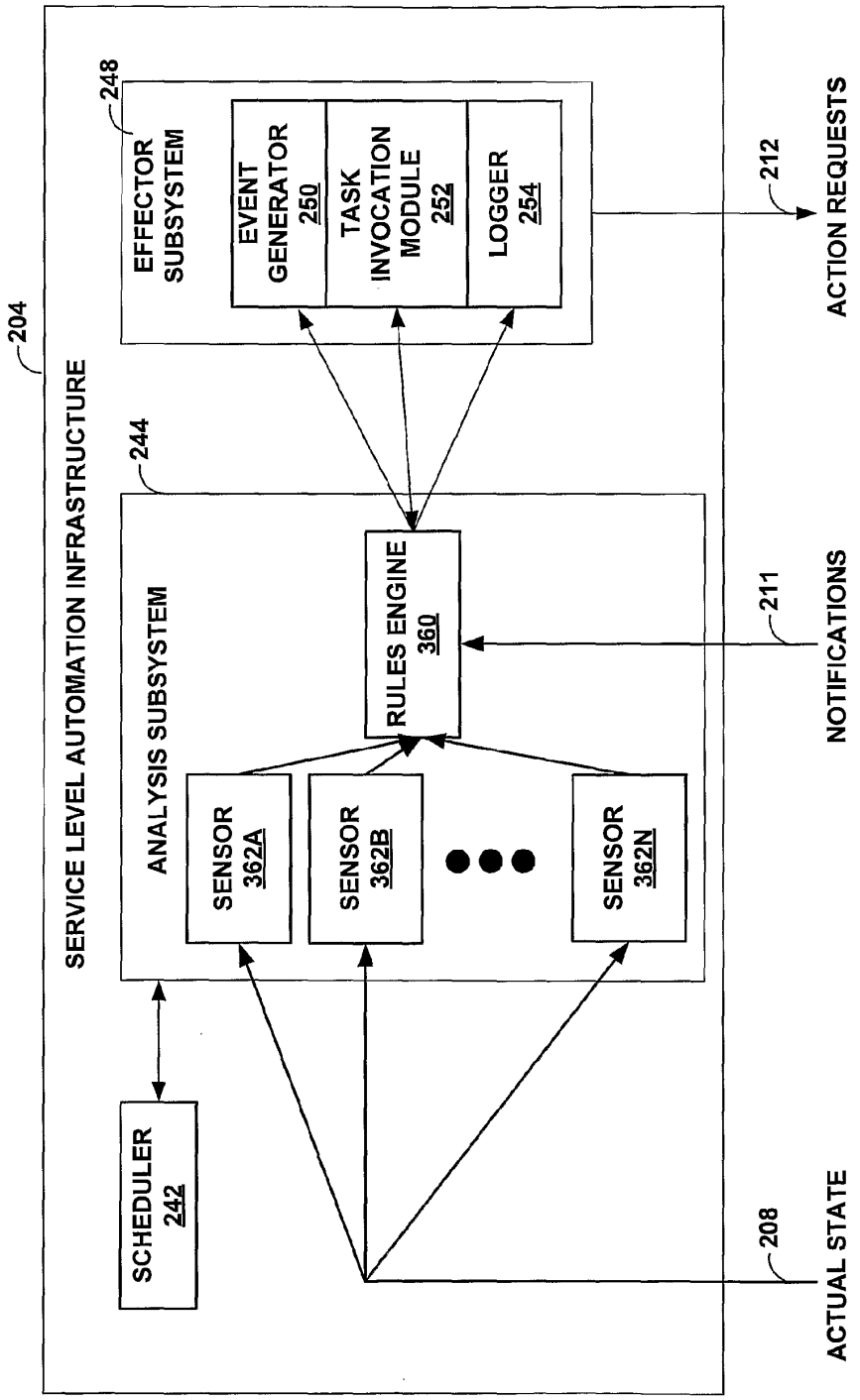

204
SERVICE LEVEL AUTOMATION INFRASTRUCTURE
248
EFFECTOR SUBSYSTEM
EVENT GENERATOR 250
TASK INVOCATION MODULE 252
LOGGER 254
212
ACTION REQUESTS
244
ANALYSIS SUBSYSTEM
RULES ENGINE 360
211
NOTIFICATIONS
SENSOR 362A
SENSOR 362B
SENSOR 362N
SCHEDULER 242
208
ACTUAL STATE

FIG. 21

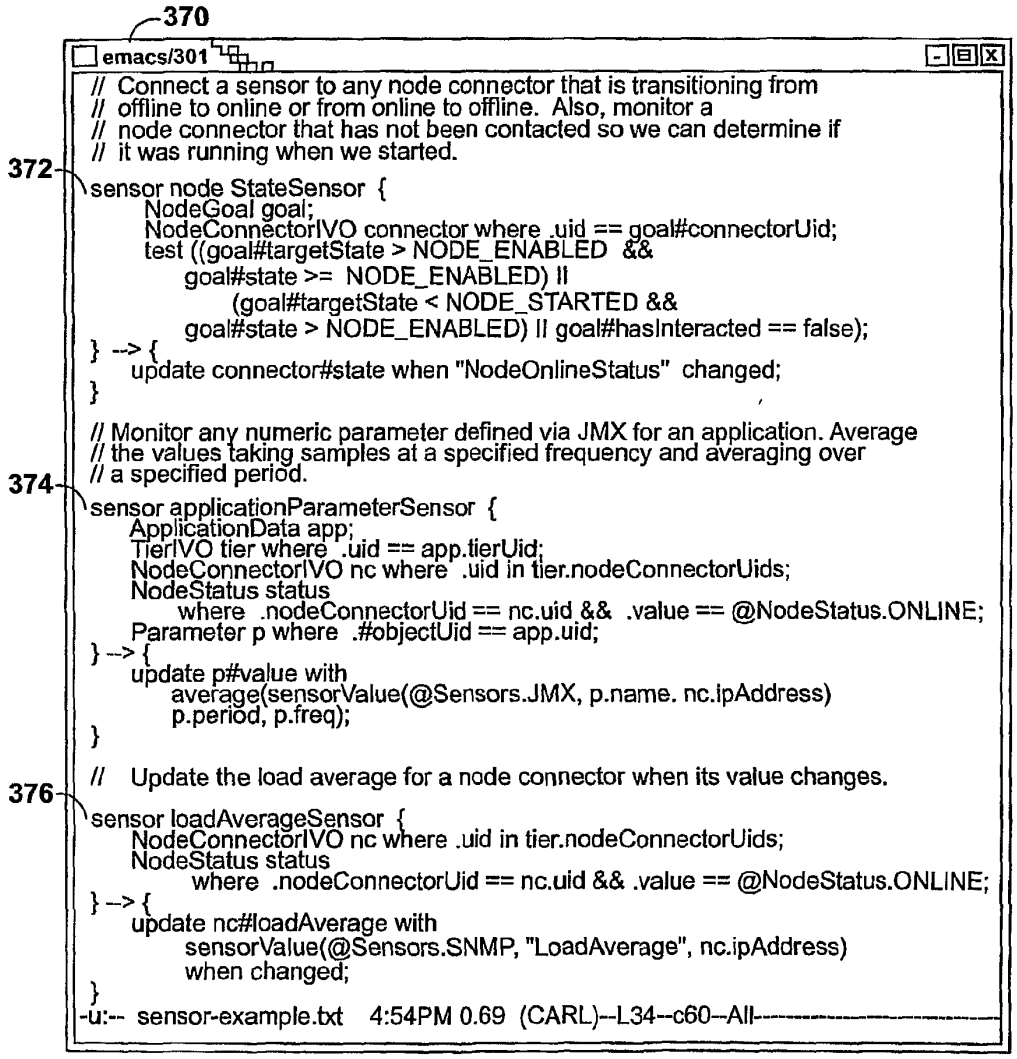

370
emacs/301
// Connect a sensor to any node connector that is transitioning from
// offline to online or from online to offline. Also, monitor a
// node connector that has not been contacted so we can determine if
// it was running when we started.
372
sensor node StateSensor {
NodeGoal goal;
NodeConnectorIVO connector where .uid == goal#connectorUid;
test ((goal#targetState > NODE_ENABLED &&
goal#state >= NODE_ENABLED) ||
(goal#targetState < NODE_STARTED &&
goal#state > NODE_ENABLED) || goal#hasInteracted == false);
} --> {
update connector#state when "NodeOnlineStatus" changed;
}
// Monitor any numeric parameter defined via JMX for an application. Average
// the values taking samples at a specified frequency and averaging over
374
// a specified period.
sensor applicationParameterSensor {
ApplicationData app;
TierIVO tier where .uid == app.tierUid;
NodeConnectorIVO nc where .uid in tier.nodeConnectorUids;
NodeStatus status
where .nodeConnectorUid == nc.uid && .value == @NodeStatus.ONLINE;
Parameter p where .#objectUid == app.uid;
} --> {
update p#value with
average(sensorValue(@Sensors.JMX, p.name. nc.ipAddress)
p.period, p.freq);
}
// Update the load average for a node connector when its value changes.
376
sensor loadAverageSensor {
NodeConnectorIVO nc where .uid in tier.nodeConnectorUids;
NodeStatus status
where .nodeConnectorUid == nc.uid && .value == @NodeStatus.ONLINE;
} --> {
update nc#loadAverage with
sensorValue(@Sensors.SNMP, "LoadAverage", nc.ipAddress)
when changed;
}
-u:-- sensor-example.txt 4:54PM 0.69 (CARL)--L34--c60--All----------

FIG. 22

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003439

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 July 2006	18/07/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ross, C

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003439

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HERNESS E N ET AL: "WebSphere Application Server: a foundation for on demand computing" IBM SYSTEMS JOURNAL IBM USA, vol. 43, no. 2, 2004, pages 213-237, XP002383791 ISSN: 0018-8670 Paragraph starting with "Multitiered distributed business computing": page 215, left-hand column page 220, left-hand column, line 26 - page 221, right-hand column, line 3 Paragraph starting with "Deployment automation": page 222, left-hand column - right-hand column, line 11 page 223, left-hand column, line 5 - line 23 page 224, right-hand column, line 11 - right-hand column, line 3 page 226, left-hand column, line 9 - page 229, left-hand column, line 2 page 229, right-hand column, line 1 - line 13 Paragraph starting with "Monitoring and application response measurement...": page 230, left-hand column	1-32
X	G LODI, F PANZIERI: "QoS-aware Clustering of Application Servers" PROCEEDINGS OF THE 1ST IEEE WORKSHOP ON QUALITY OF SERVICE FOR APPLICATION SERVERS, IN CONJUNCTION WITH THE 23RD INT'L SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS 2004), 17 October 2004 (2004-10-17), pages 1-6, XP002383792 Florianopolis, Brazil page 1, line 1 - line 19 page 2, left-hand column, line 10 - right-hand column, line 18 page 3, right-hand column, line 27 - line 45	1-32
X	US 2003/126265 A1 (AZIZ ASHAR ET AL) 3 July 2003 (2003-07-03) abstract; figure 1c paragraph [0049] paragraph [0052] paragraph [0088] - paragraph [0095] paragraph [0206]	1-32

-/--

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003439

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/177176 A1 (HIRSCHFELD ROBERT A ET AL) 18 September 2003 (2003-09-18) abstract column 7, line 4 - line 10 column 9, line 29 - line 52; figure 3 column 13, line 23 - line 53; figures 5a,5b	1-32
A	US 2003/140282 A1 (KALER CHRISTOPHER G ET AL) 24 July 2003 (2003-07-24) abstract paragraph [0197] - paragraph [0205] paragraph [0211] - paragraph [0228]	1-32

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/003439

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003126265 A1	03-07-2003	NONE	
US 2003177176 A1	18-09-2003	NONE	
US 2003140282 A1	24-07-2003	US 2003061541 A1	27-03-2003
		US 2003126509 A1	03-07-2003
		US 2003140280 A1	24-07-2003
		US 6467052 B1	15-10-2002

Netflix, Inc. - Ex. 1002, Page 000474
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 August 2006 (10.08.2006)

PCT

(10) International Publication Number
WO 2006/083727 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2006/003028

(22) International Filing Date: 31 January 2006 (31.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

11/047,468	31 January 2005 (31.01.2005)	US
11/074,291	7 March 2005 (07.03.2005)	US
11/176,161	7 July 2005 (07.07.2005)	US
11/191,384	28 July 2005 (28.07.2005)	US

(71) Applicant *(for all designated States except US)*: **CASSATT CORPORATION** [US/US]; 1740 Technology Drive, San Jose, California 95110 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **EMEIS, Mark, W.** [US/US]; 20250 Silver Horn Lane, Monument, Colorado 80132 (US). **JACKSON, Jerry, R.** [US/US]; 11270 Wakely Road, Colorado Springs, Colorado 80908 (US).

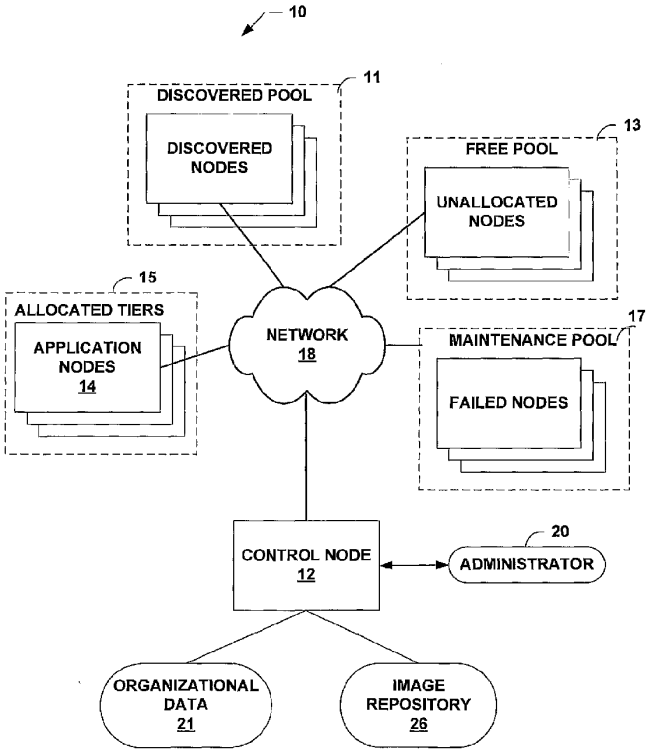
(74) Agent: **SIEFFERT, Kent, J.**; Shumaker & Sieffert, P.A., 8425 Seasons Parkway, Suite 105, St.paul, Minnesota 55125-4393 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: APPLICATION GOVERNOR PROVIDING APPLICATION-LEVEL AUTONOMIC CONTROL WITHIN A DISTRIBUTED COMPUTING SYSTEM



(57) Abstract: A distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of distributed computing system. The control node includes an automation subsystem having one or more rule engines that provide autonomic control of the application nodes in accordance with a set of one or more rules. A pluggable, application-specific application governor is selected and installed within the control node to provide an application- independent interface through which the rule engines interact to control the deployment, execution and monitoring of the applications within the distributed computing system.

WO 2006/083727 A1

FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

APPLICATION GOVERNOR PROVIDING APPLICATION-LEVEL AUTONOMIC CONTROL WITHIN A DISTRIBUTED COMPUTING SYSTEM

TECHNICAL FIELD

[0001] The invention relates to computing environments and, more specifically, to distributed computing systems.

BACKGROUND

[0002] Distributed computing systems are increasingly being utilized to support business as well as technical applications. Typically, distributed computing systems are constructed from a collection of computing nodes that combine to provide a set of processing services to implement the distributed computing applications. Each of the computing nodes in the distributed computing system is typically a separate, independent computing device interconnected with each of the other computing nodes via a communications medium, e.g., a network.

[0003] One challenge with distributed computing systems is the organization, deployment and administration of such a system within an enterprise environment. For example, it is often difficult to manage the allocation and deployment of enterprise computing functions within the distributed computing system. An enterprise, for example, often includes several business groups, and each group may have competing and variable computing requirements.

SUMMARY

[0004] In general, the invention is directed to a distributed computing system that conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.

[0005] As described herein, the model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the

Netflix, Inc. - Ex. 1002, Page 000477
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of the distributed computing system. The control nodes are responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

[0006] In one embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network; and a control node. The control node comprises an automation subsystem having one or more rule engines that provide autonomic control of the deployment of a set of applications to the application nodes in accordance with a set of one or more rules, and an application governor that performs actions with respect to the applications in response to action requests from the rules engine.

[0007] In another embodiment, a method comprises receiving status data for a distributed computing system having a plurality of application nodes interconnected via a communications network, wherein the status data represents an actual state of the application nodes. The method further comprises processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system, and directing an application governor to perform the operations to provide autonomic control over the deployment and execution of a set of applications within the distributed computing system in accordance with the rules.

[0008] In another embodiment, the invention is directed to a computer-readable medium comprising instructions. The instructions cause the processor to install an application governor as a pluggable module within a framework provided by a control node for a distributed computing system, wherein the distributed computing system includes a plurality of application nodes interconnected via a communications network. The instructions further cause the processor to receive status data representing an actual state of the application nodes, and process the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system. The instructions cause the processor to direct the application

Netflix, Inc. - Ex. 1002, Page 000478
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

governor to perform the operations to provide autonomic control over deployment and execution of a set of applications within the distributed computing system in accordance with the rules.

[0009] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0010] FIG. 1 is a block diagram illustrating a distributed computing system constructed from a collection of computing nodes.

[0011] FIG. 2 is a schematic diagram illustrating an example of a model of an enterprise that logically defines an enterprise fabric.

[0012] FIG. 3 is a flow diagram that provides a high-level overview of the operation of a control node when configuring the distributed computing system.

[0013] FIG. 4 is a flow diagram illustrating exemplary operation of the control node when assigning computing nodes to node slots of tiers.

[0014] FIG. 5 is a flow diagram illustrating exemplary operation of a control node when adding an additional computing node to a tier to meet additional processing demands.

[0015] FIG. 6 is a flow diagram illustrating exemplary operation of a control node harvesting excess node capacity from one of the tiers and returning the harvested computing node to the free pool.

[0016] FIG. 7 is a screen illustration of an exemplary user interface for defining tiers in a particular domain.

[0017] FIG. 8 is a screen illustration of an exemplary user interface for defining properties of the tiers.

[0018] FIG. 9 is a screen illustration of an exemplary user interface for viewing and identify properties of a computing node.

[0019] FIG. 10 is a screen illustration of an exemplary user interface for viewing software images.

Netflix, Inc. - Ex. 1002, Page 000479
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0020] FIG. 11 is a screen illustration of an exemplary user interface for viewing a hardware inventory report.

[0021] FIG. 12 is a screen illustration of an exemplary user interface for viewing discovered nodes that are located in the free pool.

[0022] FIG. 13 is a screen illustration of an exemplary user interface for viewing users of a distributed computing system.

[0023] FIG. 14 is a screen illustration of an exemplary user interface for viewing alerts for the distributed computing system.

[0024] FIG. 15 is a block diagram illustrating one embodiment of control node that includes a monitoring subsystem, a service level automation infrastructure (SLAI), and a business logic tier (BLT).

[0025] FIG. 16 is a block diagram illustrating one embodiment of the monitoring subsystem.

[0026] FIG. 17 is a block diagram illustrating one embodiment of the SLAI in further detail.

[0027] FIG. 18 is a block diagram of an example working memory associated with rule engines of the SLAI.

[0028] FIG. 19 is a block diagram illustrating an example embodiment for the BLT of the control node.

[0029] FIG. 20 is a block diagram illustrating one embodiment of a rule engine in further detail.

[0030] FIG. 21 is a block diagram illustrating another example embodiment of the control node.

[0031] FIG. 22 is a flowchart illustrating an exemplary mode of initiating a control node utilizing an application matrix.

[0032] FIG. 23 is a block diagram illustrating an exemplary application matrix.

DETAILED DESCRIPTION

[0033] FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide distributed processing.

Netflix, Inc. - Ex. 1002, Page 000480
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0034] In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated tiers 15 and a maintenance pool 17. In addition, distributed computing system 10 includes at least one control node 12.

[0035] Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

[0036] Discovered pool 11 includes a set of discovered nodes that have been automatically "discovered" within distributed computing system 10 by control node 12. For example, control node 12 may monitor dynamic host communication protocol (DHCP) leases to discover the connection of a node to network 18. Once detected, control node 12 automatically inventories the attributes for the discovered node and reassigns the discovered node to free pool 13. The node attributes identified during the inventory process may include a CPU count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics or other computing resources. Control node 12 may also receive input identifying node attributes not detectable via the automatic inventory, such as whether the node includes I/O, such as HBA. Further details with respect to the automated discovery and inventory processes are described in U.S. Patent Application No. 11/070,851, having attorney docket no. 1072-009US01, entitled "AUTOMATED DISCOVERY AND INVENTORY OF NODES WITHIN AN AUTONOMIC DISTRIBUTED COMPUTING SYSTEM," filed March 2, 2005.

[0037] Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated tiers 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to add an application node to allocated tiers 15 to increase processing capacity of distributed computing system 10.

Netflix, Inc. - Ex. 1002, Page 000481
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0038] In general, allocated tiers 15 include one or more tiers of application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory). Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated tiers 15.

[0039] Control node 12 provides the system support functions for managing distributed computing system 10. More specifically, control node 12 manages the roles of each computing node within distributed computing system 10 and the execution of software applications within the distributed computing system. In general, distributed computing system 10 includes at least one control node 12, but may utilize additional control nodes to assist with the management functions.

[0040] Other control nodes 12 (not shown in FIG. 1) are optional and may be associated with a different subset of the computing nodes within distributed computing system 10. Moreover, control node 12 may be replicated to provide primary and backup administration functions, thereby allowing for graceful handling a failover in the event control node 12 fails.

[0041] Network 18 provides a communications interconnect for control node 12 and application nodes 14, as well as discovered nodes, unallocated nodes and failed nodes. Communications network 18 permits internode communications among the computing nodes as the nodes perform interrelated operations and functions. Communications network 18 may comprise, for example, direct connections between one or more of the computing nodes, one or more customer networks maintained by an enterprise, local area networks (LANs), wide area networks (WANs) or a combination thereof. Communications network 18 may include a number of switches, routers, firewalls, load balancers, and the like.

[0042] In one embodiment, each of the computing nodes within distributed computing system 10 executes a common general-purpose operating system. One example of a general-purpose operating system is the Windows™ operating system

Netflix, Inc. - Ex. 1002, Page 000482
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

provided by Microsoft Corporation. In some embodiments, the general-purpose operating system such as the Linux kernel may be used.

[0043] In the example of FIG. 1, control node 12 is responsible for software image management. The term "software image" refers to a complete set of software loaded on an individual computing node to provide an execution environment for one or more applications. The software image including the operating system and all boot code, middleware files, and may include application files. As described below embodiments of the invention provide application-level autonomic control over the deployment, execution and monitoring of applications onto software images associated with application nodes 14.

[0044] System administrator 20 may interact with control node 12 and identify the particular types of software images to be associated with application nodes 14. Alternatively, administration software executing on control node 12 may automatically identify the appropriate software images to be deployed to application nodes 14 based on the input received from system administrator 20. For example, control node 12 may determine the type of software image to load onto an application node 14 based on the functions assigned to the node by system administrator 20. Application nodes14 may be divided into a number of groups based on their assigned functionality. As one example, application nodes 14 may be divided into a first group to provide web server functions, a second group to provide business application functions and a third group to provide database functions. The application nodes 14 of each group may be associated with different software images.

[0045] Control node 12 provides for the efficient allocation and management of the various software images within distributed computing system 10. In some embodiments, control node 12 generates a "golden image" for each type of software image that may be deployed on one or more of application nodes 14. As described herein, the term "golden image" refers to a reference copy of a complete software stack for providing an execution environment for applications.

[0046] System administrator 20 may create a golden image by installing an operating system, middleware and software applications on a computing node and then making a complete copy of the installed software. In this manner, a golden

Netflix, Inc. - Ex. 1002, Page 000483
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

image may be viewed as a "master copy" of the software image for a particular computing function. Control node 12 maintains a software image repository 26 that stores the golden images associated with distributed computing system 10.

[0047] Control node 12 may create a copy of a golden image, referred to as an "image instance," for each possible image instance that may be deployed within distributed computing system 10 for a similar computing function. In other words, control node 12 pre-generates a set of K image instances for a golden image, where K represents the maximum number of image instances for which distributed computing system 10 is configured for the particular type of computing function. For a given computing function, control node 12 may create the complete set of image instance even if not all of the image instances will be initially deployed. Control node 12 creates different sets of image instances for different computing functions, and each set may have a different number of image instances depending on the maximum number of image instances that may be deployed for each set. Control node 12 stores the image instances within software image repository 26. Each image instance represents a collection of bits that may be deployed on an application node.

[0048] Further details of software image management are described in co-pending U.S. patent application Ser. No. 11/046,133, entitled "MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005 and co-pending U.S. patent application Ser. No. 11/046,152, entitled "UPDATING SOFTWARE IMAGES ASSOCIATED WITH A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005.

[0049] In general, distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

[0050] As used herein, the "fabric" level generally refers to the logical constructs that allow for definition, deployment, partitioning and management of distinct enterprise applications. In other words, fabric refers to the integrated set of

Netflix, Inc. - Ex. 1002, Page 000484
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

hardware, system software and application software that can be "knitted" together to form a complete enterprise system. In general, the fabric level consists of two elements: fabric components or fabric payload. Control node 12 provides fabric management and fabric services as described herein.

[0051] In contrast, a "domain" is a logical abstraction for containment and management within the fabric. The domain provides a logical unit of fabric allocation that enables the fabric to be partitioned amongst multiple uses, e.g. different business services.

[0052] Domains are comprised of tiers, such as a 4-tier application model (web server, application server, business logic, persistence layer) or a single tier monolithic application. Fabric domains contain the free pool of devices available for assignment to tiers.

[0053] A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

[0054] The lowest level, the node level, includes the physical components of the fabric. This includes computing nodes that, as described above, provide operating environments for system applications and enterprise software applications. In addition, the node level may include network devices (e.g., Ethernet switches, load balancers and firewalls) used in creating the infrastructure of network 18. The node level may further include network storage nodes that are network connected to the fabric.

[0055] System administrator 20 accesses administration software executing on control node 12 to logically define the hierarchical organization of distributed computing system 10. For example, system administrator 20 may provide organizational data 21 to develop a model for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, develop a model

Netflix, Inc. - Ex. 1002, Page 000485
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.

[0056] More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation management (CRM) service. System administrator 20 further defines one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a web tier, an application tier and a database tier. In this manner, distributed computing system 10 may be configured to automatically provide web server functions, business application functions and database functions.

[0057] For each of the tiers, control node 12 creates a number of "node slots" equal to the maximum number of application nodes 14 that may be deployed. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot. The node slots may, for instance, identify a particular software image instance associated with an application node 14 as well as a network address associated with that particular image instance.

[0058] In this manner, each of the tiers include one or more node slots that reference particular software image instances to boot on the application nodes 14 to which each software image instance is assigned. The application nodes 14 to which control node 12 assigns the image instances temporarily inherit the network address assigned to the image instance for as long as the image instance is deployed on that particular application node. If for some reason the image instance is moved to a different application node 14, control node 12A moves the network address to that new application node.

[0059] System administrator 20 may further define specific node requirements for each tier of the fabric. For example, the node requirements specified by system administrator 20 may include a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics that may be detected on the individual computing nodes. System

Netflix, Inc. - Ex. 1002, Page 000486
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

administrator 20 may also specify user-defined hardware attributes of the computing nodes, such as whether I/O (like HBA) is required. The user-defined hardware attributes are typically not capable of detection during an automatic inventory. In this manner, system administrator 20 creates a list of attributes that the tier requires of its candidate computing nodes. In addition, particular node requirements may be defined for software image instances.

[0060] In addition to the node requirements described above, system administrator 20 may further define policies that are used when re-provisioning computing nodes within the fabric. System administrator 20 may define policies regarding tier characteristics, such as a minimum number of nodes a tier requires, an indication of whether or not a failed node is dynamically replaced by a node from free pool 13, a priority for each tier relative to other tiers, an indication of whether or not a tier allows nodes to be re-provisioned to other tiers to satisfy processing requirements by other tiers of a higher priority or other policies. Control node 12 uses the policy information input by system administrator 20 to re-provision computing nodes to meet tier processing capacity demands.

[0061] After receiving input from system administrator 20 defining the architecture and policy of the enterprise fabric, control node 12 identifies unallocated nodes within free pool 13 that satisfy required node attributes. Control node 12 automatically assigns unallocated nodes from free pool 13 to respective tier node slots of a tier. As will be described in detail herein, in one embodiment, control node 12 may assign computing nodes to the tiers in a "best fit" fashion. Particularly, control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20. The assignment of the computing nodes may occur on a tier-by-tier basis beginning with a tier with the highest priority and ending with a tier with the lowest priority. Alternatively, or in addition, assignment of computing nodes may be based on dependencies defined between tiers.

[0062] As will be described in detail below, control node 12 may automatically add unallocated nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to the free pool when the tier has excess capacity, transfer nodes from tier to tier to meet processing demands, or

Netflix, Inc. - Ex. 1002, Page 000487
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

replace failed nodes with nodes from the free pool. Thus, computing resources, i.e., computing nodes, may be automatically shared between tiers and domains within the fabric based on user-defined policies to dynamically address high-processing demands, failures and other events.

[0063] FIG. 2 is a schematic diagram illustrating an example embodiment of organizational data 21 that defines a model logically representing an enterprise fabric in accordance with the invention. In the example illustrated in FIG. 2, control node 12 (FIG. 1) maintains organizational data 21 to define a simple e-commerce fabric 32.

[0064] In this example, e-commerce fabric 32 includes a storefront domain 34A and a financial planning domain 34B. Storefront domain 34A corresponds to the enterprise storefront domain and allows customers to find and purchase products over a network, such as the Internet. Financial planning domain 34B allows one or more employees to perform financial planning tasks for the enterprise.

[0065] Tier level 31C includes one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. For example, storefront domain 34A includes a web server tier (labeled "web tier") 36A, a business application tier (labeled "app tier") 36B, and a database tier (labeled "DB tier") 36C. Web server tier 36A, business application tier 36B and database tier 36C interact with one another to present a customer with an online storefront application and services. For example, the customer may interact with web server tier 36A via a web browser. When the customer searches for a product, web server tier 36A may interacts with business application tier 36B, which may in turn access a database tier 36C. Similarly, financial planning domain 34B includes a financial planning tier 36D that provides subcomponents of applications and services of the financial planning domain 34B. Thus, in this example, a domain may include a single tier.

[0066] Tier level 31D includes one or more logical node slots 38A-38H ("node slots 38") within each of the tiers. Each of node slots 38 include node specific information, such as software resources for an application node 14 that is assigned to a respective one of the node slots 38. Node slots 38 may, for instance, identify particular software image instances within image repository 26 and map the

Netflix, Inc. - Ex. 1002, Page 000488
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

identified software image instances to respective application nodes 14. As an example, node slots 38A and 38B belonging to web server tier 36A may reference particular software image instances used to boot two application nodes 14 to provide web server functions. Similarly, the other node slots 38 may reference software image instances to provide business application functions, database functions, or financial application functions depending upon the tier to which the node slots are logically associated.

[0067] Although in the example of FIG. 2, there are two node slots 38 corresponding to each tier, the tiers may include any number of node slots depending on the processing capacity needed on the tier. Furthermore, not all of node slots 38 may be currently assigned to an application node 14. For example, node slot 28B may be associated with an inactive software image instance and, when needed, may be assigned to an application node 14 for deployment of the software image instance.

[0068] In this example, organizational data 21 associates free node pool 13 with the highest-level of the model, i.e., e-commerce fabric 32. As described above, control node 12 may automatically assign unallocated nodes from free node pool 13 to at least a portion of tier node slots 38 of tiers 36 as needed using the "best fit" algorithm described above or another algorithm. Additionally, control node 12 may also add nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.

[0069] Although not illustrated, the model for the enterprise fabric may include multiple free node pools. For example, the model may associate free node pools with individual domains at the domain level or with individual tier levels. In this manner, administrator 20 may define policies for the model such that unallocated computing nodes of free node pools associated with domains or tiers may only be used within the domain or tier to which they are assigned. In this manner, a portion of the computing nodes may be shared between domains of the entire fabric while other computing nodes may be restricted to particular domains or tiers.

Netflix, Inc. - Ex. 1002, Page 000489
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0070] FIG. 3 is a flow diagram that provides a high-level overview of the operation of control node 12 when configuring distributed computing system 10. Initially, control node 12 receives input from a system administrator defining the hierarchical organization of distributed computing system 10 (50). In one example, control node 12 receives input that defines a model that specifies a number of hierarchically arranged nodes as described in detail in FIG. 2. Particularly, the defined architecture of distributed computing system 10 includes an overall fabric having a number of hierarchically arranged domains, tiers and node slots.

[0071] During this process, control node 12 may receive input specifying node requirements of each of the tiers of the hierarchical model (52). As described above, administrator 20 may specify a list of attributes, e.g., a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), or local disk characteristics, that the tiers require of their candidate computing nodes. In addition, control node 12 may further receive user-defined custom attributes, such as requiring the node to have I/O, such as HBA connectivity. The node requirements or attributes defined by system administrator 20 may each include a name used to identify the characteristic, a data type (e.g., integer, long, float or string), and a weight to define the importance of the requirement.

[0072] Control node 12 identifies the attributes for all candidate computing nodes within free pool 13 or a lower priority tier (54). As described above, control node 12 may have already discovered the computing nodes and inventoried the candidate computing nodes to identify hardware characteristics of all candidate computing nodes. Additionally, control node 12 may receive input from system administrator 20 identifying specialized capabilities of one or more computing nodes that are not detectable by the inventory process.

[0073] Control node 12 dynamically assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the

Netflix, Inc. - Ex. 1002, Page 000490
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a "best fit" algorithm.

[0074] FIG. 4 is a flow diagram illustrating exemplary operation of control node 12 when assigning computing nodes to node slots of tiers. Initially, control node 12 selects a tier to enable (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

[0075] Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each of the tiers. Control node 12 retrieves the node requirements for the selected tier from the database.

[0076] In addition, control node 12 accesses the database and retrieves the computing node attributes of one of the unallocated computing nodes of free pool 13. Control node 12 compares the node requirements of the tier to the node attributes of the selected computing node (64).

[0077] Based on the comparison, control node 12 determines whether the node attributes of the computing node meets the minimum node requirements of the tier (66). If the node attributes of the selected computing node do not meet the minimum node requirements of the tier, then the computing node is removed from the list of candidate nodes for this particular tier (68). Control node 12 repeats the process by retrieving the node attributes of another of the computing nodes of the free pool and compares the node requirements of the tier to the node attributes of the computing node.

[0078] If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the node requirements of the tier (70). If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned

Netflix, Inc. - Ex. 1002, Page 000491
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).

[0079] Control node 12 then determines whether the node count for the tier is met (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20. When the node count for the tier is met, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority, and repeats the process until all defined tiers are enabled, i.e., populated with application nodes (60).

[0080] If the node attributes of the selected computing node and the node requirements of the tier are not a perfect match control node 12 calculates and records a "processing energy" of the node (76). As used herein, the term "processing energy" refers to a numerical representation of the difference between the node attributes of a selected node and the node requirements of the tier. A positive processing energy indicates the node attributes more than satisfy the node requirements of the tier. The magnitude of the processing energy represents the degree to which the node requirements exceed the tier requirements.

[0081] After computing and recording the processing energy of the nodes, control node 12 determines whether there are more candidate nodes in free pool 13 (78). If there are additional candidate nodes, control node 12 repeats the process by retrieving the computing node attributes of another one of the computing nodes of the free pool of computing nodes and comparing the node requirements of the tier to the node attributes of the computing node (64).

[0082] When all of the candidate computing nodes in the free pool have been examined, control node 12 selects the candidate computing node having the minimum positive processing energy and assigns the selected computing node to a node slot of the tier (80). Control node 12 determines whether the minimum node count for the tier is met (82). If the minimum node count for the tier has not been met, control node 12 assigns the computing node with the next lowest calculated processing energy to the tier (80). Control node 12 repeats this process until the

Netflix, Inc. - Ex. 1002, Page 000492
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node count is met. At this point, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority (60).

[0083] In the event there are an insufficient number of computing nodes in free pool 13, or an insufficient number of computing nodes that meet the tier requirements, control node 12 notifies system administrator 20. System administrator 20 may add more nodes to free pool 13, add more capable nodes to the free pool, reduce the node requirements of the tier so more of the unallocated nodes meet the requirements, or reduce the configured minimum node counts for the tiers.

[0084] FIG. 5 is a flow diagram illustrating exemplary operation of control node 12 when adding an additional computing node to a tier to meet increased processing demands. Initially, control node 12 or system administrator 20 identifies a need for additional processing capacity on one of the tiers (90). Control node 12 may, for example, identify a high processing load on the tier or receive input from a system administrator identifying the need for additional processing capacity on the tier.

[0085] Control node 12 then determines whether there are any computing nodes in the free pool of nodes that meet the minimum node requirements of the tier (92). When there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and assigns the node to the tier (95). As described in detail with respect to FIG. 4, control node 12 may determine whether there are any nodes that have node attributes that are an exact match to the node requirements of the tier. If an exact match is found, the corresponding computing node is assigned to a node slot of the tier. If no exact match is found, control node 12 computes the processing energy for each node and assigns the computing node with the minimum processing energy to the tier. Control node 12 remotely powers on the assigned node and remotely boots the node with the image instance associated with the node slot. Additionally, the booted computing node inherits the network address associated with the node slot.

[0086] If there are no adequate computing nodes in the free pool, i.e., no nodes at all or no nodes that match the minimal node requirements of the tier, control node

Netflix, Inc. - Ex. 1002, Page 000493
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

12 identifies the tiers with a lower priority than the tier needing more processing capacity (96).

[0087] Control node 12 determines which of the nodes of the lower priority tiers meet the minimum requirements of the tier in need of processing capacity (98). Control node 12 may, for example, compare the attributes of each of the nodes assigned to node slots of the lower priority tiers to the node requirements of the tier in need of processing capacity. Lower priority tiers that have the minimum number of computing nodes may be removed from possible tiers from which to harvest an application node. If, however, all the lower priority tiers have the minimum number of computing nodes defined for the respective tier, the lowest priority tier is selected from which to harvest the one or more nodes.

[0088] Control node 12 calculates the processing energy of each of the nodes of the lower priority tiers that meet the minimum requirements (100). The energies of the nodes are calculated using the differences between the node attributes and the node requirements of the tier needing additional capacity. Control node 12 selects the computing node with the lowest processing energy that meets the minimum requirements, and assigns the selected computing node to the tier in need of processing capacity (102, 95).

[0089] FIG. 6 is a flow diagram illustrating exemplary operation of control node 12 when harvesting excess node capacity from one of the tiers and returning the harvested computing node to free pool 13. Initially, control node 12 identifies a tier having excess node capacity (110). Control node 12 may, for example, periodically check the node capacity of the tiers to identify any tiers having excess node capacity. Performing a periodic check and removal of excess nodes increases the likelihood that a capable computing node will be in free pool 13 in the event one of the tiers needs additional node capacity.

[0090] When harvesting a node, control node 12 calculates the processing energy of all the nodes in the tier as described above with reference to FIG. 4 (112). Control node 12 identifies the node within the tier with the highest processing energy and returns the identified node to the free pool of nodes (114, 116). As described above, the node with the highest processing energy corresponds to the

Netflix, Inc. - Ex. 1002, Page 000494
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node whose node attributes are the most in excess of the node requirements of the tier.

[0091] Returning the node to the free pool may involve remotely powering off the computing node and updating the database to associate the harvested node with free pool 13. In addition, control node 12 updates the database to disassociate the returned node with the node slot to which it was assigned. At this point, the node no longer uses the network address associated with the image instance mapped to the node slot. Control node 12 may, therefore, assign a temporary network address to the node while the node is assigned to free pool 13.

[0092] FIG. 7 is a screen illustration of an exemplary user interface 120 presented by control node 12 with which administrator 20 interacts to define tiers for a particular domain. In the example illustrated in FIG. 7, system administrator 20 has selected the "Collage Domain." User interface 120 presents the tiers that are currently in the selected domain. In the example illustrated, the Collage Domain includes three tiers, "test tier 1," "test tier 2," and "test tier 3." As shown in FIG. 7, in this example, each of the tiers includes two nodes. In addition, user interface 120 lists the type of software image currently deployed to application nodes for each of the tiers. In the example illustrated, image "applone (1.0.0)" is deployed to the nodes of test tier 1 and image "appltwo (1.0.0)" is deployed to the nodes of test tier 2. System administrator 20 may add one or more tiers to the domain by clicking on new tier button 122.

[0093] FIG. 8 is a screen illustration of an exemplary user interface 130 for defining properties of the tiers. In particular, user interface 130 allows system administrator 20 to input a name for the tier, a description of the tier, and an image associated with the tier. The image associated with the tier refers to a golden image from which image instances are generated and deployed to the nodes assigned to the tier.

[0094] When configuring a tier, system administrator 20 may elect to activate email alerts. For example, system administrator 20 may activate the email alerts feature in order to receive email alerts providing system administrator 20 with critical and/or non-critical tier information, such as a notification that a tier has been upgraded, a node of the tier has failed or the like. Furthermore, system

Netflix, Inc. - Ex. 1002, Page 000495
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

administrator 20 may input various policies, such node failure rules. For example, system administrator 20 may identify whether control node 12 should reboot a node in case of failure or whether the failed node should automatically be moved to maintenance pool 17. Similarly, system administrator 20 may identify whether nodes assigned to the tier may be harvested by other tiers.

[0095] User interface 130 may also allow system administrator 20 to input node requirements of a tier. In order to input node requirements of a tier, system administrator 20 may click on the "Requirements" tab 132, causing user interface 130 to present an input area to particular node requirements of the tier.

[0096] FIG. 9 is a screen illustration of an exemplary user interface 140 for viewing and identifying properties of a computing node. User interface 140 allows system administrator 20 to define a name, description, and location (including a rack and slot) of a computing node. In addition user interface 140 may specify user-defined properties of a node, such as whether the computing node has I/O HBA capabilities.

[0097] User interface 140 also displays properties that control node 12 has identified during the computing node inventory process. In this example, user interface 140 presents system administrator 20 with the a CPU node count, a CPU speed, the amount of RAM, the disk size and other characteristics that are identifiable during the automated node inventory. User interface 140 additionally presents interface information to system administrator 20. Specifically, user interface 140 provides system administrator 20 with a list of components and their associated IP and MAC addresses.

[0098] User interface 140 also allows system administrator 20 to define other custom requirements. For example, system administrator 20 may define one or more attributes and add those attributes to the list of node attributes presented to system administrator 20.

[0099] FIG. 10 is a screen illustration of an exemplary user interface 150 for viewing software images. User interface 150 presents to a system administrator or another user a list of images maintained by control node 12 within image repository 26. The image list further includes the status of each image (i.e., either active or inactive), the version of the image, the operating system on which the

Netflix, Inc. - Ex. 1002, Page 000496
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

image should be run, the operating system version on which the image should be run and a brief description of the image.

[0100] System administrator 20 or another user may select an image by clicking on the box in front of the image identifier/name and perform one or more actions on the image. Actions that system administrator 20 may perform on an image include deleting the image, updating the image, and the like. System administrator 20 may select one of the image actions via dropdown menu 152. In some embodiments, user interface 150 may further display other details about the images such as the node to which the images are assigned (if the node status is "active"), the network address associated with the images and the like.

[0101] FIG. 11 is a screen illustration of an exemplary user interface 160 for viewing a hardware inventory report. User interface 160 presents to system administrator 20 or another user a list of the nodes that are currently assigned to a domain. System administrator 20 may elect to view the nodes for the entire domain, for a single tier within the domain or for a single rack within a tier.

[0102] For each node, user interface 160 presents a node ID, a status of the node, the tier to which the node belongs, a hostname associated with the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0103] System administrator 20 or other user may select a node by clicking on the box in front of the node identifier/name and perform one or more actions on the node. Actions that system administrator 20 may perform on the node include deleting the node, updating the node attributes or other properties of the node, and the like. System administrator 20 may select one of the node actions via dropdown menu 162.

[0104] FIG. 12 is a screen illustration of an exemplary user interface 170 for viewing discovered nodes that are located in discovered pool 11. For each node, user interface 170 presents a node ID, a state of the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

Netflix, Inc. - Ex. 1002, Page 000497
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0105] FIG. 13 is a screen illustration of an exemplary user interface 180 for viewing users of distributed computing system 10. User interface 180 presents a list of users as well as the role assigned to each of the users and the status of each of the users. Thus, system administrator 20 may define different roles to each of the users. For example, a user may be either an operator (i.e., general user) or an administrator. System administrator 20 may add a new user to the list of users by clicking on the "New User" button 182.

[0106] FIG. 14 is a screen illustration of an exemplary user interface 190 for viewing alerts for distributed computing system 10. For each of the alerts, user interface 190 identifies the severity of the alert, whether the alert has been acknowledged, an object associated with the alert, an event associated with the alert, a state of the alert, a user associated with the alert and a date associated with the alert.

[0107] System administrator 20 or other user may select an alert by clicking on the box in front of the logged alert and perform one or more actions on the logged alert. Actions that system administrator 20 may perform include deleting the alert, changing the status of the alert, or the like. System administrator 20 may specify the log actions via dropdown menu 192.

[0108] FIG. 15 is a block diagram illustrating one embodiment of control node 12 in further detail. In the illustrated example, control node 12 includes a monitoring subsystem 202, a service level automation infrastructure (SLAI) 204, and a business logic tier (BLT) 206.

[0109] Monitoring subsystem 202 provides real-time monitoring of the distributed computing system 10. In particular, monitoring subsystem 202 dynamically collects status data 203 from the hardware and software operating within distributed computing system 10, and feeds the status data in the form of monitor inputs 208 to SLAI 204. Monitoring inputs 208 may be viewed as representing the actual state of the fabric defined for the organizational model implemented by distributed computing system 10. Monitoring subsystem 202 may utilize well defined interfaces, e.g., the Simple Network Management Protocol (SNMP) and the Java Management Extensions (JMX), to collect and export real-time monitoring information to SLAI 204.

Netflix, Inc. - Ex. 1002, Page 000498
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0110] SLAI 204 may be viewed as an automation subsystem that provides support for autonomic computing and acts as a central nervous system for the controlled fabric. In general, SLAI 204 receives monitoring inputs 208 from monitoring subsystem 202, analyzes the inputs and outputs appropriate action requests 212 to BLT 206. In one embodiment, SLAI 204 is a cybernetic system that controls the defined fabric via feedback loops. More specifically, administrator 20 may interact with BLT 206 to define an expected state 210 for the fabric. BLT 206 communicates expected state 210 to SLAI 204. SLAI 204 receives the monitoring inputs from monitoring subsystem 202 and applies rules to determine the most effective way of reducing the differences between the expected and actual states for the fabric.

[0111] For example, SLAI 204 may apply a rule to determine that a node within a high priority tier has failed and that the node should be replaced by harvesting a node from a lower priority tier. In this example, SLAI 204 outputs an action request 212 to invoke BLT 206 to move a node from one tier to the other.

[0112] In general, BLT 206 implements high-level business operations on fabrics, domains and tiers. SLAI 204 invokes BLT 206 to bring the actual state of the fabric into accordance with the expected state. In particular, BLT 206 outputs fabric actions 207 to perform the physical fabric changes. In addition, BLT 206 outputs an initial expected state 210 to SLAI 204 and initial monitoring information 214 to SLAI 204 and monitoring subsystem 202, respectively. In addition, BLT 206 outputs notifications 211 to SLAI 204 and monitoring subsystem 202 to indicate the state and monitoring changes to distributed computing system 10. As one example, BLT 206 may provide control operations that can be used to replace failed nodes. For example, BLT 206 may output an action request indicating that a node having address 10.10.10.10 has been removed from tier ABC and a node having address 10.10.10.11 has been added to tier XYZ. In response, monitoring subsystem 202 stops attempting to collect status data 203 from node 10.10.10.10 and starts monitoring for status data from node 10.10.10.11. In addition, SLAI 204 updates an internal model to automatically associate monitoring inputs from node 10.10.10.11 with tier XYZ.

Netflix, Inc. - Ex. 1002, Page 000499
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0113] FIG. 16 is a block diagram illustrating one embodiment of monitoring subsystem 202. In general, monitoring subsystem 202 dynamically detects and monitors a variety of hardware and software components within the fabric. For example, monitoring subsystem 202 identifies, in a timely and efficient manner, any computing nodes that have failed, i.e., any node that does not respond to a request to a known service. More generally, monitoring subsystem 202 provides a concise, consistent and constantly updating view of the components of the fabric.

[0114] As described further below, monitoring subsystem 202 employs a modular architecture that allows new detection and monitoring collectors 224 to be "plugged-in" for existing and new protocols and for existing and new hardware and software. As illustrated in FIG. 16, monitoring subsystem 202 provides a plug-in architecture that allows different information collectors 224 to be installed. In general, collectors 224 are responsible for protocol-specific collection of monitoring information. The plug-in architecture allows for new protocols to be added by simply adhering to a collector plug-in signature. In this example, monitoring subsystem 202 includes collectors 224A and 224B for collecting information from operating systems and applications executing on nodes within tier A and tier B, respectively.

[0115] In one embodiment, collectors 224 are loaded at startup of control node 12 and are configured with information retrieved from BLT 206. Monitoring engine 222 receives collection requests from SLAI 204, sorts and prioritizes the requests, and invokes the appropriate one of collectors 224 based on the protocol specified in the collection requests. The invoked collector is responsible for collecting the required status data and returning the status data to monitoring engine 222. If the collector is unable to collect the requested status data, the collector returns an error code.

[0116] In one embodiment, collectors 224 are Java code compiled into a jar file and loaded with a class loader at run time. Each of collectors 224 has an associated configuration file written in a data description language, such as the extensible markup language (XML). In addition, a user may interact with BLT 206 to add run-time configuration to dynamically configure collectors 224 for specific computing environments. Each of collectors 224 expose an application

Netflix, Inc. - Ex. 1002, Page 000500
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

programming interface (API) to monitoring engine 222 for communication and data exchange.

[0117] A user, such as a system administrator, specifies the protocol or protocols to be used for monitoring a software image when the image is created. In addition, the users may specify the protocols to be used for monitoring the nodes and each service executing on the nodes. Example protocols supported by the collectors 224 include Secure Shell (SSH), Simple Network Management Protocol (SNMP), Internet Control Message Protocol (ICMP) ping, Java Management Extensions (JMX) and the Hypertext Transfer Protocol (HTTP).

[0118] Some protocols require special privileges, e.g., root privileges, to perform the required data collection. In this case, the corresponding collectors 224 communicate with a separate process that executes as the root. Moreover, some protocols may require deployment and/or configuration of data providers within the fabric. Software agents may, for example, be installed and configured on nodes and configured on other hardware. If needed, custom in-fabric components may be deployed.

[0119] In this example, the modular architecture of monitoring subsystem 202 also supports one or more plug-in interfaces 220 for data collection from a wide range of third-party monitoring systems 228. Third-party monitoring systems 228 monitor portions of the fabric and may be vendor-specific.

[0120] FIG. 17 is a block diagram illustrating one embodiment of SLAI 204 in further detail. In the illustrated embodiment, SLAI 204 is composed of three subsystems: a sensor subsystem 240, an analysis subsystem 244 and an effector subsystem 248.

[0121] In general, sensor subsystem 240 receives actual state data from monitoring subsystem 202 in the form of monitoring inputs 208 and supplies ongoing, dynamic input data to analysis subsystem 244. For example, sensor subsystem 240 is notified of physical changes to distributed computing system 10 by monitoring subsystem 202. Sensor subsystem 240 uses the state data received from monitoring subsystem 202 to maintain ongoing, calculated values that can be sent to analysis subsystem 244 in accordance with scheduler 242.

Netflix, Inc. - Ex. 1002, Page 000501
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0122] In one embodiment, sensor subsystem 240 performs time-based hierarchical data aggregation of the actual state data in accordance with the defined organization model. Sensor subsystem 240 maintains organizational data in a tree-like structure that reflects the current configuration of the hierarchical organization model. Sensor subsystem 240 uses the organizational data to perform the real-time data aggregation and map tiers and domains to specific nodes. Sensor subsystem 240 maintains the organizational data based on notifications 211 received from BLT 206.

[0123] Sensor subsystem 240 sends inputs to analysis subsystem 244 to communicate the aggregated data on a periodic or event-driven basis. Analysis subsystem 244 may register an interest in a particular aggregated data value with sensor subsystem 240 and request updates at a specified frequency. In response, sensor subsystem 240 interacts with monitoring subsystem 202 and scheduler 242 to generate the aggregated data required by analysis subsystem 244.

[0124] Sensor subsystem 240 performs arbitrary data aggregations via instances of plug-in classes (referred to as "triggers") that define the aggregations. Each trigger is registered under a compound name based on the entity being monitored and the type of data being gathered. For example, a trigger may be defined to aggregate and compute an average computing load for a tier every five minutes. Analysis subsystem 244 requests the aggregated data based on the registered names. In some embodiments, analysis subsystem 244 may define calculations directly and pass them to sensor subsystem 240 dynamically.

[0125] Analysis subsystem 244 is composed of a plurality of forward chaining rule engines 246A-246N. In general, rule engines 246 match patterns in a combination of configuration data and monitoring data, which is presented by extraction agent 251 in the form of events. Events contain the aggregated data values that are sent to rule engines 246 in accordance with scheduler 242.

[0126] Sensor subsystem 240 may interact with analysis subsystem 244 via trigger listeners 247 that receives updates from a trigger within sensor subsystem 240 when specified events occur. An event may be based on system state (e.g., a node transitioning to an up or down state) or may be time based.

Netflix, Inc. - Ex. 1002, Page 000502
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0127] Analysis subsystem 244 allows rule sets to be loaded in source form and compiled at load time into discrimination networks. Each rule set specifies trigger-delivered attributes. Upon loading the rule sets, analysis subsystem 244 establishes trigger listeners 247 to receive sensor notifications and update respective working memories of rule engines 246. As illustrated in FIG. 17, each of rule engines 246 may serve a different tier defined within the fabric. Alternatively, multiple rule engines 246 may serve a single tier or a single rule engine may serve multiple tiers.

[0128] Rule engines 246 process the events and invoke action requests via calls to effector subsystem 248. In addition, rule engines 246 provide a call-back interface so that effector subsystem 248 can inform a rule engine when an action has completed. Rule engines 246 prevent a particular rule from re-firing as long as any action invoked by the rule has not finished. In general, rules contain notification calls and service invocations though either may be disabled by configuration of effector subsystem 248. BLT 206 supplies initial system configuration descriptions to seed each of rule engines 246.

[0129] In general, rule engines 246 analyze the events and discover discrepancies between an expected state of the fabric and an actual state. Each of rule engines 246 may be viewed as software that performs logical reasoning using knowledge encoded in high-level condition-action rules. Each of rule engines 246 applies automated reasoning that works forward from preconditions to goals defined by system administrator 20. For example, rule engines 246 may apply modus ponens inferences rules.

[0130] Rule engines 246 output requests to effector subsystem 248 which produce actions requests 212 for BLT 206 to resolve the discrepancies. Effector subsystem 248 performs all operations on behalf of analysis subsystem 244. For example, event generator 250, task invocation module 252 and logger 254 of effector subsystem 248 perform event generation, BLT action invocation and rule logging, respectively. More specifically, task invocation module 252 invokes asynchronous operations within BLT 206. In response, BLT 206 creates a new thread of control for each task which is tracked by a unique task identifier (task id). Rules engine 246 uses the task id to determine when a task completes and, if needed, to re-fire any rules that were pended until completion of the task. These tasks may take

Netflix, Inc. - Ex. 1002, Page 000503
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

arbitrary amounts of time, and rules engine 246 tracks the progress of individual task via change notifications 211 produced by BLT 206.

[0131] Event generator 250 creates persistent event records of the state of processing of SLAI 204 and stores the event records within a database. Clients uses these event records to track progress and determine the current state of the SLAI 204.

[0132] Logger 254 generates detailed trace information about system activities for use in rule development and debugging. The logging level can be raised or lowered as needed without changing operation of SLAI 204.

[0133] FIG. 18 is a block diagram of an example working memory 270 associated with rule engines 246. In this example, working memory 270 includes a read-only first data region 272 that stores the expected state received from BLT 206. Data region 272 is read-only in the sense that it cannot be modified in response to a trigger from sensor subsystem 240 or by rule engines 246 without notification from BLT 206.

[0134] In addition, working memory 270 includes a second data region 274 that is modifiable (i.e., read/write) and may be updated by monitoring subsystem 202 or used internally by rule engines 246. In general, data region 274 stores aggregated data representing the actual state of the fabric and can be updated by sensor subsystem 240 or by rule engines 246. The actual state may consist of a set of property annotations that can be attached to objects received from BLT 206 or to objects locally defined within a rule engine, such as local object 276.

[0135] FIG. 19 is a block diagram illustrating an example embodiment for BLT 206. In this example, BLT 206 includes a set of one or more web service definition language (WSDL) interfaces 300, a report generator 302, a fabric administration interface service 304, a fabric view service 306, a user administration service 308, a task interface 311, a task manager 312 and an event subsystem 315.

[0136] As described, BLT 206 provides the facilities necessary to create and administer the organizational model (e.g., fabric, domains, tiers and nodes) implemented by distributed computing system 10. In general, BLT 206 abstracts access to the persisted configuration state of the fabric, and controls the

Netflix, Inc. - Ex. 1002, Page 000504
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

interactions with interfaces to fabric hardware services. As such, BLT 206 provides fabric management capabilities, such as the ability to create a tier and replace a failed node. WSDL interfaces 300 provide web service interfaces to the functionality of BLT 206 that may be invoked by web service clients 313. Many of WSDL interfaces 300 offered by BLT 206 allow administrator 20 to define goals, such as specifying a goal of the expected state of the fabric. As further described below, rule engines 246 within SLAI 204, in turn, invoke task manger 312 to initiate one or more BLT tasks to achieve the specified goal. In general, web service clients 313 may be presentation layer applications, command line applications, or other clients.

[0137] BLT 206 abstracts all interaction with physical hardware for web service clients 313. BLT 206 is an enabling component for autonomic management behavior, but does not respond to real-time events that either prevent a goal from being achieved or produce a set of deviations between the expected state and the actual state of the system. In contrast, BLT 206 originates goals for autonomic reactions to changing configuration and state. SLAI 204 analyzes and acts upon these goals along with real-time state changes. BLT 206 sets the goals to which SLAI 204 strives to achieve, and provides functionality used by the SLAI in order to achieve the goals.

[0138] In general, BLT 206 does not dictate the steps taken in pursuit of a goal since these are likely to change based on the current state of distributed computing system 10 and changes to configurable policy. SLAI 204 makes these decisions based on the configured rule sets for the fabric and by evaluating monitoring data received from monitoring subsystem 202.

[0139] Fabric administration service 304 implements a set of methods for managing all aspects of the fabric. Example methods include methods for adding, viewing, updating and removing domains, tiers, nodes, notifications, assets, applications, software images, connectors, and monitors. Other example methods include controlling power at a node, and cloning, capturing, importing, exporting or upgrading software images. Rule engines 246 of SLAI 204 may, for example, invoke these methods by issuing action requests 212.

Netflix, Inc. - Ex. 1002, Page 000505
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0140] Task manager 312 receives action requests 212 via task interface 311. In general, task interface 311 provides an interface for receiving action requests 212 from SLAI 204 or other internal subsystem. In response, task manager 312 manages asynchronous and long running actions that are invoked by SLAI 204 to satisfy a goal or perform an action requested by a client.

[0141] Task manager 312 generates task data 310 that represents identification and status for each task. Task manager 312 returns a task identifier to the calling web service clients 313 or the internal subsystem, e.g., SLAI 204, that initiated the task. Rule engines 246 and web service clients 313 use the task identifiers to track progress and retrieve output, results, and errors associated with achieving the goal.

[0142] In one embodiment, there are no WSDL interfaces 300 for initiating specific tasks. Rather, administrator 20 interacts with BLT 206 though goal interfaces presented by WSDL interfaces 300 to define the goals for the fabric. In contrast, the term task is used to refer to internal system constructs that require no user interaction. Tasks are distinct, low-level units of work that affect the state of the fabric. SLAI 204 may combine tasks to achieve or maintain a goal state.

[0143] For example, administrator 20 can request configuration changes by either adding new goals to an object or by modifying the attributes on existing goals. Scheduled goals apply a configuration at a designated time. For example, the goals for a particular tier may specify the minimum, maximum, and target node counts for that tier. As a result, the tier can increase or decrease current node capacity by scheduling goals with different configuration values.

[0144] This may be useful, for example, in scheduling a software image upgrade. As another example, entire domains may transition online and offline per a defined grid schedule. Administrator 20 may mix and match goals on a component to achieve configurations specific to the application and environment. For example, a tier that does not support autonomic node replacement would not be configured with a harvesting goal.

[0145] In some embodiments, goals are either "in force" or "out of force." SLAI 204 only works to achieve and maintain those goals that are currently in force. SLAI 204 may applies a concept of "gravity" as the goals transition from in force to out of force. For example, SLAI 204 may transition a tier offline when an

Netflix, Inc. - Ex. 1002, Page 000506
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

online goal is marked out of force. Some goal types may have prerequisite goals. For example, an image upgrade goal may require as a prerequisite that a tier be transitioned to offline before the image upgrade can be performed. In other embodiments, goals are always in force until modified.

[0146] SLAI 204 may automatically formulate dependencies between goals or may allow a user to specify the dependencies. For example, a user may request that a newly created tier come online. As a result of this goal, SLAI 204 may automatically direct task manager 312 to generate a task of harvesting a target number of nodes to enable the tier. Generally, all goals remain in-force by SLAI 204 until modified by BLT 206. In one embodiment, each goal remains in-force in one of three states: Satisfied, Warning, or Critical depending on how successful SLAI 204 was in achieving the goal at the time the event record was generated and stored.

[0147] In this manner, SLAI 204 controls the life cycle of a goal (i.e., the creation, scheduling, update, deletion of the goal), and provides a common implementation of these and other services such as timeout, event writing, goal conflicts, management of intra-goal dependencies, and tracking tasks to achieving the goals.

[0148] Progress toward a goal is tracked though event subsystem 315. In particular, event subsystem 315 tracks the progress of each in force goal based on the goal identifiers. Tasks executed to achieve a particular goal produce events to communicate result or errors. The events provide a convenient time-based view of all actions and behaviors.

[0149] Examples of goal types that may be defined by administrator 20 include software image management goals, node allocation goals, harvest goals, tier capacity goals, asset requirement goals, tier online / offline goals, and data gathering goals.

[0150] In one embodiment, BLT 206 presents a task interface to SLAI 204 for the creation and management of specific tasks in order to achieve the currently in force goals. In particular, rule engines 246 invoke the task interface based on evaluation of the defined rule sets in view of the expected state and actual state for the fabric. Example task interfaces include interfaces to: reserve node resources; query resources for a node slot; associate or disassociate an image with a node in a tier

Netflix, Inc. - Ex. 1002, Page 000507
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node slot; allocate, de-allocate, startup or shutdown a node; move a node to a tier; apply, remove or cycle power of a node; create a golden image; create or delete an image instance; and delete an activity, node or tier.

[0151] Report generator 302 provides an extensible mechanism for generating reports 314. Typical reports include image utilization reports that contain information with respect to the number of nodes running each software image, inventory reports detailing both the logical and physical aspects of the fabric, and system event reports showing all events that have occurred within the fabric. Report generator 302 gathers, localizes, formats and displays data into report form for presentation to the user. Report generator 302 may include one or more data gathering modules (not shown) that gather events in accordance with a schedule and update an events table to record the events. The data gathering modules may write the events in XML format.

[0152] FIG. 20 is a block diagram illustrating one embodiment of a rule engine 246 (FIG. 17). In the illustrated embodiment, rule engine 246 includes a rule compiler 344 and an execution engine 346. Each of rules 342 represents a unit of code that conforms to a rule language and expresses a set of triggering conditions and a set of implied actions. When the conditions are met, the actions are eligible to occur. The following is one example of a configuration rule:

```
rule checkTierLoad {
      Tier t where status != "overloaded";
      LoadParameter p where app == t.app && maxload <
      t.load;
} -> {
      modify t {
            status: "overloaded";
      };
}
```

When translated, this example rule marks a tier as overloaded if an application is implemented by the tier and the maximum specified load for the application has been exceeded. Another example rule for outputting a notification that a tier is overloaded and automatically invoking a task within BLT 206 to add a node is:

```
rule tierOverloadNotify {
      Tier t where status == "overloaded";
} -> {
```


Netflix, Inc. - Ex. 1002, Page 000508
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
        notify "Tier: " + t + "is overloaded.";
        BLT.addNode(f);
    }
```

[0153] Rule compiler 344 compiles each of rules 344 and translates match conditions of the rules into a discrimination network that avoids redundant tests during rule execution. Execution engine 346 handles rule administration, object insertion and retrieval, rule invocation and execution of rule actions. In general, execution engine 346 first matches a current set of rules 342 against a current state of working memory 348 and local objects 350. Execution engine 346 then collects all rules that match as well as the matched objects and selects a particular rule instantiation to fire. Next, execution engine 346 fires (executes) the instantiated rule and propagates any changes to working memory 348. Execution engine 346 repeats the process until no more matching rule instantiations can be found.

[0154] Firing of a rule typically produces a very small number of changes to working memory 348. This allows sophisticated rule engines to scale by retaining match state between cycles. Only the rules and rule instantiations affected by changes are updated, thereby avoiding the bulk of the matching process. One exemplary algorithm that may be used by execution engine 346 to handle the matching process includes the RETE algorithm that creates a decision tree that combines the patterns in all the rules and is intended to improve the speed of forward-chained rule system by limiting the effort required to re-compute a conflict set after a rule is fired. One example of a RETE algorithm is described in Forgy, C.L.: 1982, 'RETE: a fast algorithm for the many pattern/many object pattern match problem', Artificial Intelligence 19, 1737. Other alternatives include the TREAT algorithms, and LEAPS algorithm, as described by Miranker, D.P.: 'TREAT: A New and Efficient Match Algorithm for AI Production Systems'. ISBN 0934613710 Daniel P. Miranker, David A. Brant, Bernie Lofaso, David Gadbois: On the Performance of Lazy Matching in Production Systems. AAAI 1990: 685692.

[0155] FIG. 21 is a block diagram illustrating an alternative embodiment of control unit 12 (FIG. 15). In this embodiment, control unit 12 operates substantially as described above, but includes an application matrix 350, an application governor 352, a configuration processor 354, and an application service level automation

Netflix, Inc. - Ex. 1002, Page 000509
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

infrastructure ("application SLAI") 358. As described below, application matrix 350, application governor 352, configuration processor 354, and application SLAI 358 provide a framework that allows control unit 12 to autonomically control the deployment, execution and monitoring of applications across application nodes 14 of distributed computing system 10.

[0156] Application matrix 350 contains all the information needed by control unit 12 to interact with one or more applications or application servers and provide autonomic control over a set of applications. Specifically, application matrix 350 provides a logical definition for deploying and controlling the set of applications to one or more tiers within distributed computing system 10. In one embodiment, application matrix 350 is an electronic document that conforms to a data description language, e.g., the extensible markup language (XML). Application SLAI 358 includes an application rules engine 355 dedicated to processing application-level rules, i.e., forward-chaining rules to provide autonomic control over the applications defined within application matrix 350. Like rules engine 246, application rules engine 355 contains a rule compiler, an execution engine, and a working memory. In order to give effect to changes in application matrix 350, application SLAI 358 automatically updates application rules engine 355 and monitoring subsystem 202. In particular, application matrix 350 sends an alert whenever application matrix 350 changes. In response to this alert, application SLAI 358 captures application-specific attributes from application matrix 350. Specifically, application SLAI 358 captures configuration attributes and rule attributes contained in application matrix 350. Application SLAI 358 transfers any new rule attributes to the working memory of application rules engine 355 to provide autonomic control over the deployment, monitoring and the execution of the applications defined within application matrix 350. In addition, application SLAI 358 updates monitoring subsystem 202 to collect information required to control the applications In this manner, administrator 20 may continue to add new application definitions and configurations to application matrix 350 after distributed control node 12 has started.

[0157] As described in further detail below, configuration processor 354 is a software module that generates an application matrix entry based on an application

Netflix, Inc. - Ex. 1002, Page 000510
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

definition and application configuration properties of a "staged" application. A staged application is an application that has been deployed in a staging environment and customized for subsequent deployment within distributed computing system 10. After creating the application matrix entry, administrator 20 may insert the application matrix entry into application matrix 350.

[0158] Configuration processor 354 is "pluggable." That is, administrator 20 can "plug in" different implementations of configuration processor 354 as needed. For example, administrator 20 may need to "plug in" a different implementation of configuration processor 354 to handle applications that do not use an application server.

[0159] Application governor 352 is a software engine that performs application-level actions based on requests received from application rules engine 355. In this manner, BLT 206 effects fabric-level actions (e.g., deployment and monitoring of nodes and images) based on request from fabric-level rules engines 246 (FIG. 17), while matrix governor 352 performs application-level actions (e.g., deployment and monitoring of applications) based on requests from application rules engine 355.

[0160] Application governor 352 uses application matrix 350 as a source of parameters when carrying out the application-level operations requested by application rules engine 355. For example, application rules engine 355 may detect that a node to which the application is not deployed is ready for use by the application. As a result, application rules engine 355 directs application governor 352 to handle the details of deploying the application to the node. In turn, application governor 352 accesses application matrix 350 to retrieve application-specific parameters necessary to deploy the application. Storing application-specific parameters in application matrix 350 allows the application-specific parameters to change without having to recompile the rules within working memory of application rules engine 355.

[0161] Application governor 352 performs a similar procedure to undeploy an application. That is, application rules engine 355 may detect that a second application needs to use a node more than a first application that is currently deployed to the node. In this situation, application rules engine 355 sends an

Netflix, Inc. - Ex. 1002, Page 000511
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

instruction to application governor 352 to undeploy the first application and deploy the second application. To carry out this instruction, application governor 352 accesses application matrix 350 to discover configuration parameters of both applications. Application governor 352 then uses the discovered configuration parameters to communicate with the applications.

[0162] Like configuration processor 354, application governor 352 is also "pluggable." That is, administrator 20 can easily install or remove implementations of application governor 352 depending on the circumstances. Because application governor 352, and configuration processor 354 represent interchangeable, plug-in modules, the other parts of control unit 12 and system 10, including application rules engine 355, can remain generic and application neutral while providing autonomic control over distributed computing system 10.

[0163] FIG. 22 provides a conceptual view of an exemplary application matrix 350. Although application matrix 350 is typically represented in an encoded, electronic document (e.g., an XML document), FIG. 22 provides a conceptual view for ease of illustration.

[0164] In this example, application matrix 350 contains seven columns and two rows. Each row represents a different application entry for deployment within distributed computing system 10. In FIG. 22, only the first entry is shown in detail.

[0165] Each of the columns represents a different category of elements that make up an application's logical definition. In this example, the columns include:

(1) an application column 360 that includes elements generally related to the deployment of the application,

(2) an application nodes column 362 that contains elements related to the tier node slots to which the application may be assigned,

(3) a services column 364 that contains elements related to the executable services launched when the application is deployed,

(4) a node monitor values column 366 that contains elements related to attributes of the nodes that are to be monitored after the application is deployed to that node,

Netflix, Inc. - Ex. 1002, Page 000512
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

(5) a service monitored attributes column 368 that contains elements related to attributes of the services that are to be monitored after the application is deployed,

(6) a service levels column 370 that contains elements related to attributes for use when constructing rules to monitor execution of the services, and

(7) a deployment constraints column 372 that contains elements related to attributes for use when constructing rules to control deployment of the application. Different types of applications may have different elements in each column, and different numbers of columns.

[0166] In the example of FIG. 22, application matrix 350 contains two applications 360 "DataDomain" 374 and "PortalDomain" 376. DataDomain application 374 has eleven attributes that define how control node 12 accesses and launches the application. For instance, the "adminIP" and "adminPort" attributes instruct governor 352 as to which the server and port hosts the administrative part of the logically defined application. Other attributes like "maxNodes" and "minNodes" instruct application rules engine 355 to run the application no less than minNodes and no more than maxNodes. Applications other than application 374 may have different numbers or types of attributes. In XML format, the attributes of application 374 may appear as follows:

```
<WebServerDomain
      name="DataDomain"
      adminIP="172.31.64.201"
      adminPort="1100"
      adminTier="Web Admin"
      clusterName="PathCluster"
      expectedStartupDelay="120"
      loadDelay="120"
      maxNodes="2"
      minNodes="1"
      nodeManagerPort="5811"
      nodeTier="Web App"
>
```

[0167] In addition to these attributes, application 374 contains a series of elements (columns 362 – 372). In general, application nodes 362 contain a list of all of the available tier-node slots to which control node 12 may deploy application 374. In this example, two tier-node slots are specified. In XML, managed servers 362 appears as:

Netflix, Inc. - Ex. 1002, Page 000513
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
<ManagedServers IP="172.31.64.201" name="managedServer_0"
state="STOPPED" />
<ManagedServers IP="172.31.64.201" name="managedServer_1"
state="STOPPED" />
```

[0168] Each services 364 element identifies a service that control node 12 launches when deploying application 374 on a given application node. A service element comprises a service name and a path to a file containing the executable service. Governor 352 uses the path to locate and launch the service. For example, the following XML code indicates that governor 352 must access the file at "/lib/worklistApp/worklistApp.ear".

```
<services name="Worklist User Interface"
path="/lib/worklistApp/worklistApp.ear" />
```

[0169] Node Monitored Values 366 elements represent characteristics for use in constructing rules for monitoring nodes to which the applications are deployed. Similarly, Service Monitored Values 368 elements represent characteristics for use in constructing rules for monitoring services that are launched once the application is deployed. In this example, a "nodeMonitoredValues" element defines characteristics of a particular node that are to be monitored. For instance, the amount of free memory in a node is one example of a characteristic listed as a "nodeMonitoredValues" element. On the other hand, a "serviceMonitoredValues" element is specific attribute of a service that is to be monitored. For example, the number of pending operating system-level requests for the service, the number of idle threads, etc., could be service monitored values. In a XML rendition of application matrix 350, node monitored values and service monitored values could appear as follows:

```
<nodeMonitoredValues name="Load5Average" />
<nodeMonitoredValues name="PercentMemoryFree" />
<serviceMonitoredValues name="PendingRequests" />
<serviceMonitoredValues name="ExecuteThreadIdleCount" />
<serviceMonitoredValues name="ExecuteThreadTotalCount" />
```

[0170] Deployment constraint elements 372 specify characteristics of a node under which application rules engine 355 should (or should not) deploy a service to the node. In this example, a deployment constraint element has five attributes: "attribute", "expression", "frequency", "maxThreshold", "minThreshold", and "period." The "attribute" attribute names the deployment constraint. The "expression" attribute specifies an arithmetic expression manipulating a monitored

Netflix, Inc. - Ex. 1002, Page 000514
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

value. For example, the expression could be "PercentMemoryFree * 100", meaning monitor the value of the "PercentMemoryFree" node monitored value multiplied by 100. The "expression" attribute may specify one or more node monitored values. The "frequency" attribute informs application rules engine 355 how frequently to check the monitored value. The "maxThreshold" attribute tells application rules engine 355 to invoke a rule when the value of the expression exceeds the value specified by the "maxThreshold" attribute. Similarly, the "minThreshold" attribute tells application rules engine 355 to invoke the rule when the value of the expression drops below the value specified by the "minThreshold" attribute. Finally, the "period" attribute informs application rules engine 355 of the period over which to collect monitored value. For example, a deployment constraint element may specify that application rules engine 355 should monitor the PercentMemoryFree attribute of a node every 15 seconds for 60 seconds. If the value of PercentMemoryFree * 100 should drop below 1.0 (i.e. 1% of memory free) for 60 seconds, then application rules engine 355 should not deploy the application to that node. In XML, this rule would be represented as:

```
<deploymentConstraints attribute="FreeMemory"
expression="PercentMemoryFree*100" frequency="15"
maxThreshold="-1.0" minThreshold="1.0" period="60" />
```

[0171] Service level elements 370 have the same five attributes as deployment constraints elements: "attribute", "expression", "frequency", "maxThreshold", "minThreshold", and "period". However, the "expression" attribute deals with service monitored attributes rather than node monitored attributes. For example, a service level element may specify that application rules engine 355 check the "pendingRequest" service-monitored attribute every 15 seconds for 30 seconds. Then, if there are more than 20 pending requests for more than 30 seconds, application rules engine 355 should take the action of starting a new application. On the other hand, if there are fewer than 5 pending requests for 30 seconds, application rules engine 355 enables the action to remove the application to free up space on a node. Such a service level element could be represented as:

```
<serviceLevels attribute="PendingRequests"
expression="PendingRequests" frequency="15"
maxThreshold="20.0" minThreshold="5.0" period="30" />
```


Netflix, Inc. - Ex. 1002, Page 000515
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0172] Put together, an XML representation of an application matrix logically defining a single application for deployment within autonomically controlled distributed computing system 10 may appear as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<appMatrixRegister>
<WebServerDomain adminIP="172.31.64.201" adminPort="1100"
adminTier="Web Admin" clusterName="PathCluster"
expectedStartupDelay="120" loadDelay="120" maxNodes="2"
minNodes="1" name="DataDomain" nodeManagerPort="5811"
nodeTier="Web App">
    <ManagedServers IP="172.31.64.201" name="managedServer_0"
    state="STOPPED" />

    <ManagedServers IP="172.31.64.201" name="managedServer_1"
    state="STOPPED" />
    <applications name="AI Design-time" path="/lib/ai-
    designtime.ear" />

    <applications name="System EJBs" path="/lib/ejbs.ear" />

    <applications name="Worklist Worker User Interface"
    path="/lib/worklist/worklist.ear" />

    <applications name="DBMS_ADK" path="/lib/DBMS_ADK.ear" />

    <applications name="UserApp"
    path="/user_projects/domains/userApp.ear" />

    <nodeMonitoredValues name="Load5Average" />

    <nodeMonitoredValues name="PercentMemoryFree" />

    <serviceMonitoredValues name="PendingRequests" />

    <serviceMonitoredValues name="ExecuteThreadIdleCount" />

    <serviceMonitoredValues name="ExecuteThreadTotalCount" />
```


Netflix, Inc. - Ex. 1002, Page 000516
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
<deploymentConstraints attribute="LoadAverage"
expression="Load5Average" frequency="15" maxThreshold="4.0"
minThreshold="-1.0" period="15" />

<deploymentConstraints attribute="FreeMemory"
expression="PercentMemoryFree*100" frequency="15"
maxThreshold="-1.0" minThreshold="1.0" period="60" />

<serviceLevels attribute="BusyThreadPercentage"
expression="(ExecuteThreadTotalCount-
ExecuteThreadIdleCount)*100/ExecuteThreadTotalCount"
frequency="15" maxThreshold="20.0" minThreshold="5.0"
period="30" />

<serviceLevels attribute="PendingRequests"
expression="PendingRequests" frequency="15"
maxThreshold="20.0" minThreshold="5.0" period="30" />
</WebserverDomain>
</appMatrixRegister>
```

[0173] FIG. 23 is a flowchart illustrating an exemplary series of general steps that are performed to add an application to application matrix 350 in accordance with the principles of this invention. Administrator 20 first stages the application within a staging environment including deploying the application on a desired image, creating a domain, specifying any services and external system connectivity (e.g., connections to databases, queues) (380).

[0174] During this process, administrator 20 directs the application to generate an application definition file that represents an application-specific configuration based on the environment specified by the administrator. The application definition file contains information specific to an installation instance of the application. For example, the application definition file typically specifies the locations (paths) for all software components of the application as staged within the staging environment, a list of files to execute, connectivity information for the application, and other information that may be recorded by the configured application.

[0175] After staging the application, administrator 20 defines a set of application configuration properties that specify desired behavior of the application within

Netflix, Inc. - Ex. 1002, Page 000517
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

distributed computing system 10 (382). Examples of such application configuration properties include a minimum number of nodes, minimum resource requirements for a node, deployment timing information, network addresses of tier node slots that are able to execute the application and other properties.

[0176] Next, administrator 20 directs configuration processor 354 to generate an application entry for the application using the application definition and the application configuration properties (384). The application entry contains configuration attributes used by application governor 352 to interact with the application. In addition, the application entry contains rule attributes used by application rules engine 355 that define how control node 12 monitors the deployment and execution of the application within distributed computing environment 10.

[0177] After configuration processor 354 creates the application entry, administrator 20 may modify the application entry (386). In particular, administrator 20 may update start scripts or shell scripts for the application due to path changes between the staging environment and distributed computing system 10.

[0178] Once administrator 20 has finished modifying the application entry, administrator 20 inserts the application entry into application matrix 350 (388). Because application matrix 350 detects that it has changed, application matrix 350 sends an alert to application SLAI 358 (390).

[0179] In response to the alert, application SLAI 358 may update application rules engine 355 and monitoring subsystem 202 (392). In particular, application SLAI 358 automatically scans application matrix 350. If application SLAI 358 detects new rule attributes, application SLAI 358 creates local objects reflecting the new rule attributes in the working memory of application rules engine 355. In addition, if application SLAI 358 detects new monitored values, application SLAI 358 updates monitoring subsystem 202 to add new monitoring collectors 224 (FIG. 16).

[0180] After application SLAI 358 updates application rules engine 355 and monitoring subsystem 202, control node 12 has autonomic control over the deployment of applications to tiers based on the configuration of application matrix 350 (394). For example, control node 12 deploys images, deploys applications,

Netflix, Inc. - Ex. 1002, Page 000518
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

monitors the state of nodes and the execution of the applications, and applies tier-level rules as well as application-specific rules. Application SLAI 358 continues to listen for alerts from application matrix 350.

[0181] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.

Netflix, Inc. - Ex. 1002, Page 000519
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS:

1. A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network; and

a control node comprising:

an automation subsystem having one or more rule engines that provide autonomic control of the deployment, undeployment, start of execution, and termination of execution of a set of applications to the application nodes in accordance with a set of one or more rules; and

an application governor that performs actions with respect to the applications in response to action requests from the rules engine.

2. The distributed computing system of claim 1, wherein the rules engines comprises:

a first rules engine that applies a set of application-specific rules to produce action requests to the application governor to control deployment of the applications.

a second rules engine that is application neutral, where the second rules engine applies a set of application-independent rules to produce action requests to control the application nodes.

3. The distributed computing system of claim 1, further comprising:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes,

wherein at least one of the rules engines analyze the status data from the monitoring subsystem and applies the set of rules to produce the action requests to the application governor to control deployment of the applications.

Netflix, Inc. - Ex. 1002, Page 000520
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

4. The distributed computing system of claim 1, wherein in response to the action requests the application governor performs a set of operations that deploy, undeploy, start, or stop the applications.

5. The distributed computing system of claim 4, wherein the application governor contains operations to start or stop execution of the applications on one or more of the application nodes.

6. The distributed computing system of claim 4, wherein at least one of the rules engines invokes operations in the application governor to automatically control the deployment and execution of at least one of the applications when a set of preconditions trigger one of the rules.

7. The distributed computing system of claim 1,

wherein the application governor is a pluggable module that is specified to the set of applications, and

wherein the control node provides a framework to receive any of a plurality of pluggable application governors that can be installed and removed from the control node to provide autonomic control over different groups of applications within the distributed computing system.

8. The distributed computing system of claim 1, wherein the application governor accesses an application matrix that contains a logical definition of the applications.

9. The distributed computing system of claim 7, wherein the application matrix includes parameters for controlling the deploying of the applications within the distributed computing system.

Netflix, Inc. - Ex. 1002, Page 000521
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

10. The distributed computing system of claim 7,

wherein the automation subsystem accesses the application matrix and automatically updates the rule engines to autonomically control the deployment of the applications to the application nodes.

11. The distributed computing system of claim 10, wherein the automation subsystem accesses the application matrix and automatically updates the rule engines to monitor the execution of the applications when deployed to the application nodes.

12. The distributed computing system of claim 8, wherein the application matrix includes deployment constraints that define criteria for use by the rules engine when selecting a set of the application nodes to which to deploy the applications.

13. The distributed computing system of claim 11, wherein the application matrix includes elements that set a desired minimum or maximum numbers of the application nodes onto which the control node may deploy the application.

14. The distributed computing system of claim 1, wherein the rule engines are forward-chaining rule engines.

15. The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

Netflix, Inc. - Ex. 1002, Page 000522
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

16. The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and applies the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

17. The distributed computing system of claim 16, wherein the monitoring subsystem comprises a set of modular information collectors that collect the status data from the application nodes.

18. The distributed computing system of claim 16, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

19. The distributed computing system of claim 16, wherein the automation unit registers collection requests with the monitoring subsystem, and the monitoring subsystem includes a monitoring engine that invokes the collectors in response to the collection requests.

20. The distributed computing system of claim 16, wherein the business logic tier presents interfaces by which a user defines one or more goals for the expected state of the application nodes.

21. The distributed computing system of claim 1, wherein the rules are units of code that conform to a rule language.

Netflix, Inc. - Ex. 1002, Page 000523
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

22. The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines produce the action requests to control the deployment of the application in accordance with the hierarchical organization of the distributed computing system defined by the model.

23. The distributed computing system of claim 22, wherein each tier represents a different computing function.

24. The distributed computing system of claim 1, wherein the application governor provides an application-independent interface to the rules engines, and performs application-specific operations in response to the action requests.

25. A method comprising:

receiving status data for a distributed computing system having a plurality of application nodes interconnected via a communications network, wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system; and

directing an application governor to perform the operations to provide autonomic control over the deployment and execution of a set of applications within the distributed computing system in accordance with the rules.

26. The method of claim 25, further comprising installing the application governor as a pluggable module within a framework provided by a control node for the distributed computing system.

Netflix, Inc. - Ex. 1002, Page 000524
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

27. The method of claim 25, further comprising selecting the application governor from a plurality of pluggable application governors that can be installed and removed from a control node of the distributed computing system to provide autonomic control over different groups of applications.

28. The method of claim 25, wherein processing the status data comprises processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

29. The method of claim 25, further comprising collecting the status data from the application nodes with a plurality of modular information collectors.

30. The method of claim 25, further comprising:

aggregating the state data in accordance with a hierarchical model of deployment of the application nodes; and

processing the aggregated state data with the rule engines.

31. The method of claim 25, further comprising:

storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

automatically producing the operations with the rule engines to control the applications in accordance with a hierarchical organization defined by the model.

32. The method of claim 25, further comprising

accessing with the application governor an application matrix that contains a logical definition of the applications to retrieve one or more data elements; and

taking at least one action with the application governor with respect to the applications based on the retrieved data elements.

Netflix, Inc. - Ex. 1002, Page 000525
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

33. The method of claim 25, wherein processing the status data comprises:

automatically incorporating application-specific rule attributes based on the application matrix into the rule engines; and

processing the set of application-specific rule attributes with the rules engines to determine the operations, wherein the rules engine are application independent.

34. A computer-readable medium comprising instructions that cause a programmable processor to:

install an application governor as a pluggable module within a framework provided by a control node for a distributed computing system, wherein the distributed computing system includes a plurality of application nodes interconnected via a communications network;

receiving status data representing an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system; and

directing the application governor to perform the operations to provide autonomic control over deployment and execution of a set of applications within the distributed computing system in accordance with the rules.

Netflix, Inc. - Ex. 1002, Page 000526
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

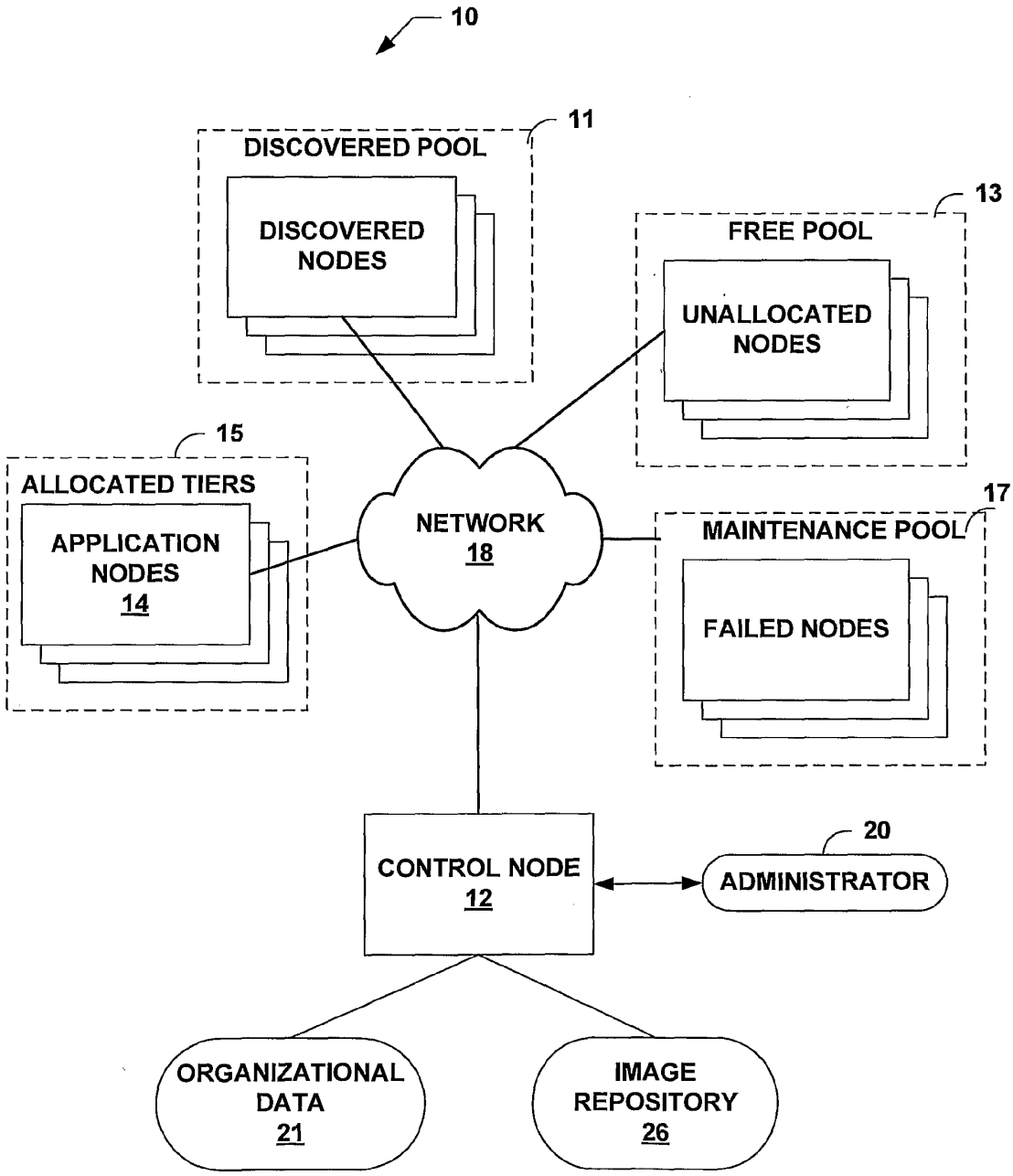

10
11
DISCOVERED POOL
DISCOVERED NODES
13
FREE POOL
UNALLOCATED NODES
15
ALLOCATED TIERS
APPLICATION NODES 14
NETWORK 18
17
MAINTENANCE POOL
FAILED NODES
CONTROL NODE 12
20
ADMINISTRATOR
ORGANIZATIONAL DATA 21
IMAGE REPOSITORY 26

FIG. 1

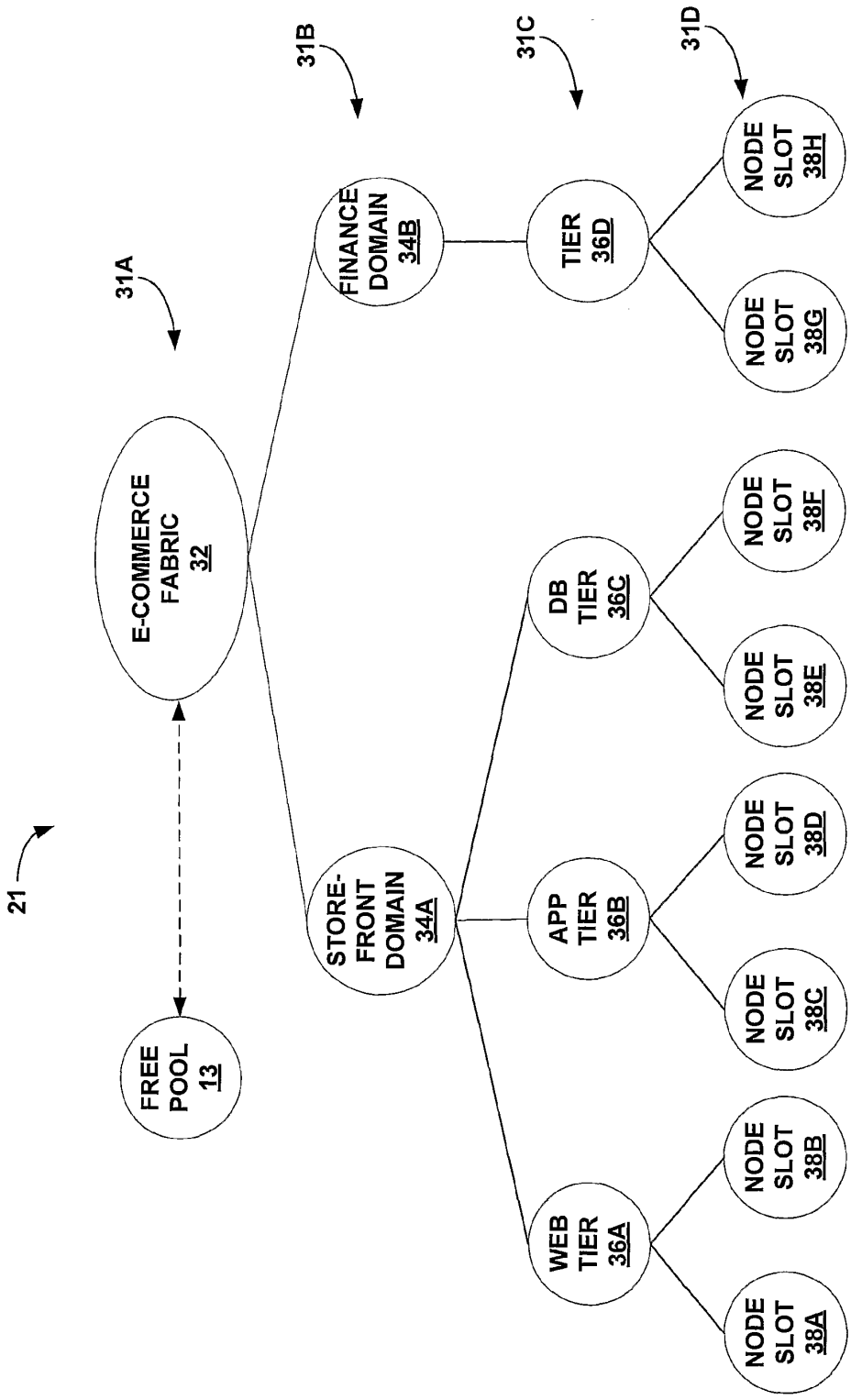

21
E-COMMERCE FABRIC 32
31A
FREE POOL 13
FINANCE DOMAIN 34B
STORE-FRONT DOMAIN 34A
31B
TIER 36D
DB TIER 36C
APP TIER 36B
WEB TIER 36A
31C
NODE SLOT 38H
NODE SLOT 38G
NODE SLOT 38F
NODE SLOT 38E
NODE SLOT 38D
NODE SLOT 38C
NODE SLOT 38B
NODE SLOT 38A
31D

FIG. 2

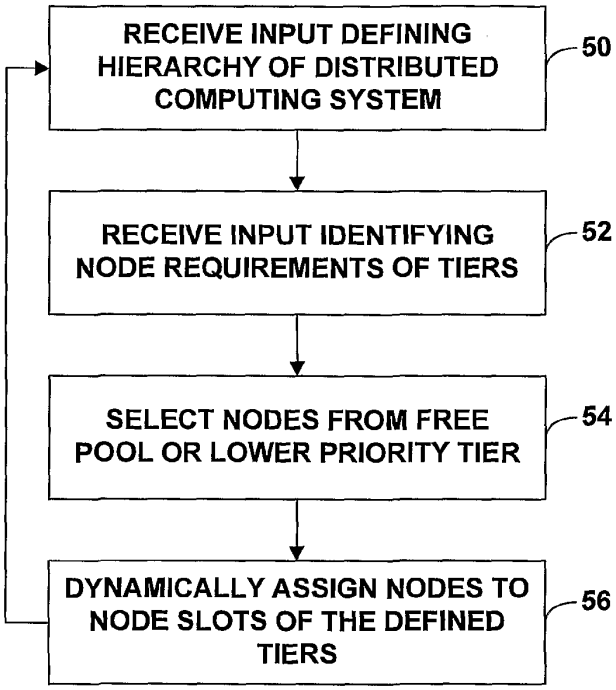

RECEIVE INPUT DEFINING HIERARCHY OF DISTRIBUTED COMPUTING SYSTEM
50
RECEIVE INPUT IDENTIFYING NODE REQUIREMENTS OF TIERS
52
SELECT NODES FROM FREE POOL OR LOWER PRIORITY TIER
54
DYNAMICALLY ASSIGN NODES TO NODE SLOTS OF THE DEFINED TIERS
56

FIG. 3

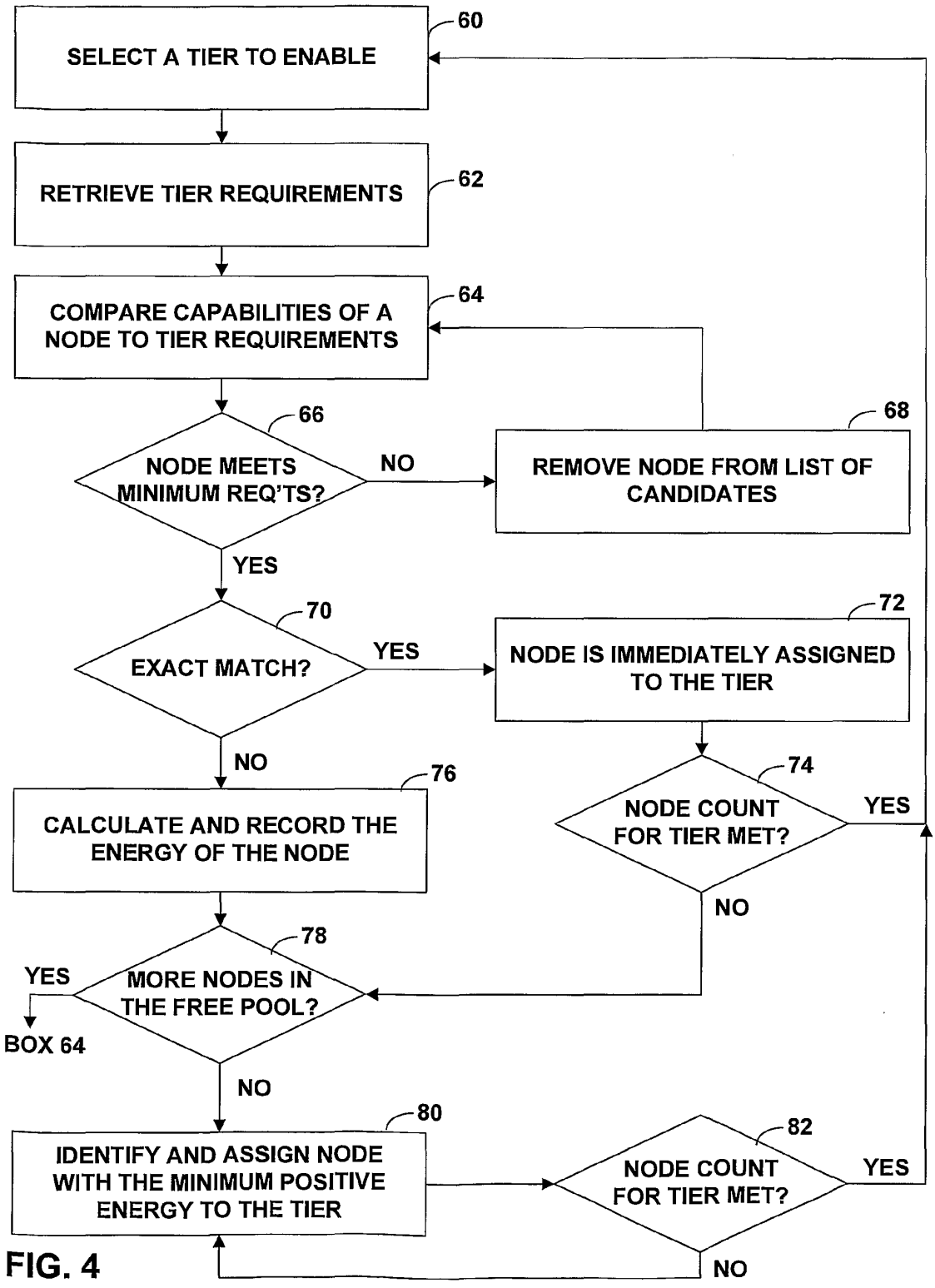

SELECT A TIER TO ENABLE
60
RETRIEVE TIER REQUIREMENTS
62
COMPARE CAPABILITIES OF A NODE TO TIER REQUIREMENTS
64
NODE MEETS MINIMUM REQ'TS?
66
NO
REMOVE NODE FROM LIST OF CANDIDATES
68
YES
EXACT MATCH?
70
YES
NODE IS IMMEDIATELY ASSIGNED TO THE TIER
72
NO
CALCULATE AND RECORD THE ENERGY OF THE NODE
76
NODE COUNT FOR TIER MET?
74
YES
NO
MORE NODES IN THE FREE POOL?
78
YES
BOX 64
NO
IDENTIFY AND ASSIGN NODE WITH THE MINIMUM POSITIVE ENERGY TO THE TIER
80
NODE COUNT FOR TIER MET?
82
YES
NO

FIG. 4

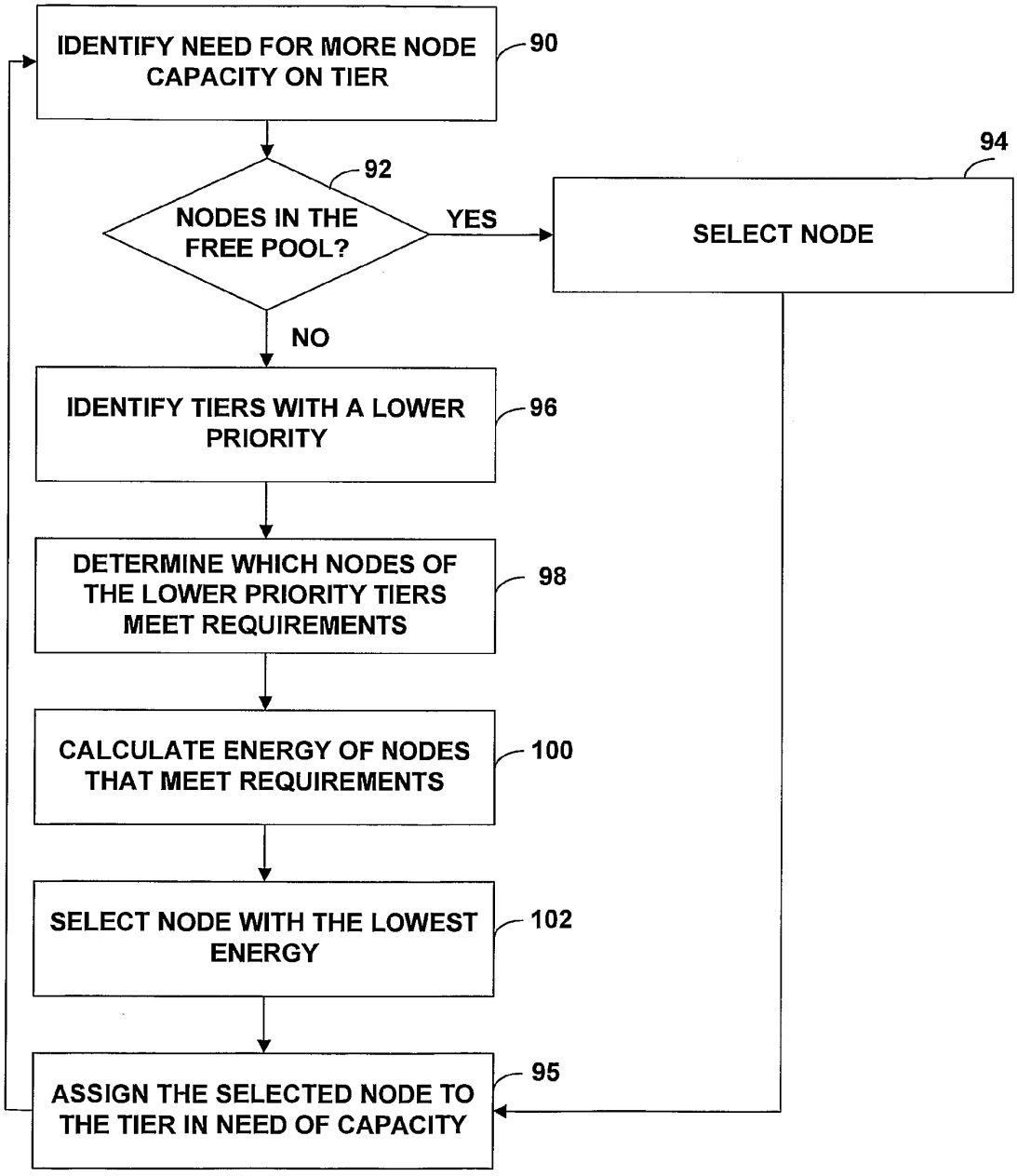

IDENTIFY NEED FOR MORE NODE CAPACITY ON TIER
90
NODES IN THE FREE POOL?
92
YES
SELECT NODE
94
NO
IDENTIFY TIERS WITH A LOWER PRIORITY
96
DETERMINE WHICH NODES OF THE LOWER PRIORITY TIERS MEET REQUIREMENTS
98
CALCULATE ENERGY OF NODES THAT MEET REQUIREMENTS
100
SELECT NODE WITH THE LOWEST ENERGY
102
ASSIGN THE SELECTED NODE TO THE TIER IN NEED OF CAPACITY
95

FIG. 5

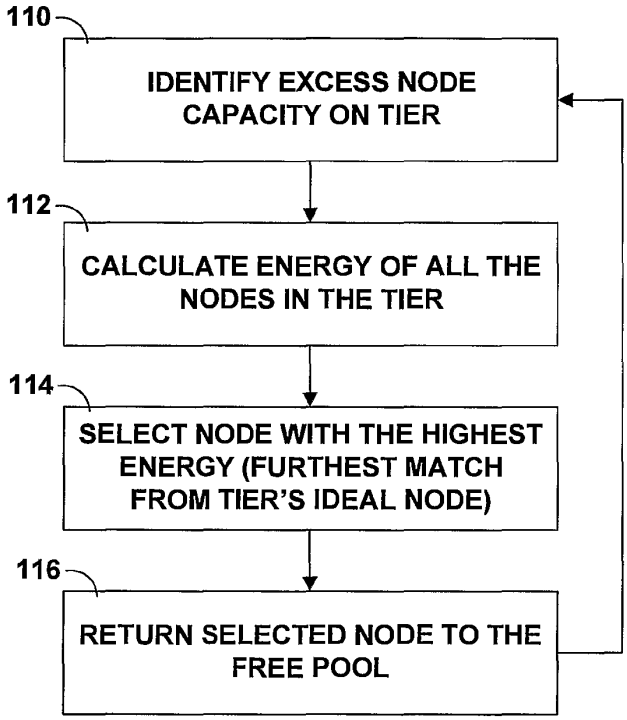

110
IDENTIFY EXCESS NODE CAPACITY ON TIER
112
CALCULATE ENERGY OF ALL THE NODES IN THE TIER
114
SELECT NODE WITH THE HIGHEST ENERGY (FURTHEST MATCH FROM TIER'S IDEAL NODE)
116
RETURN SELECTED NODE TO THE FREE POOL

FIG. 6

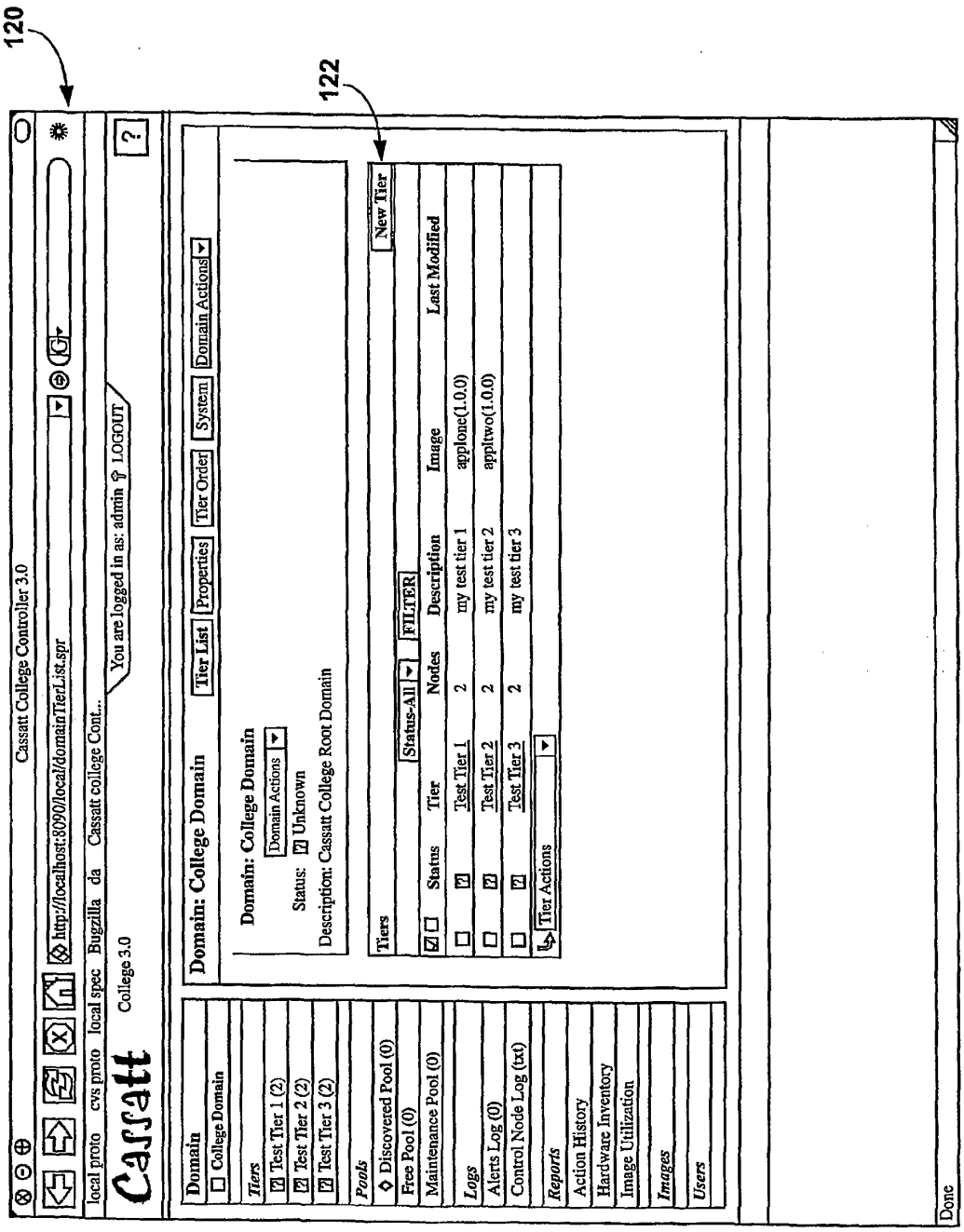


FIG. 7

SUBSTITUTE SHEET (RULE 26)

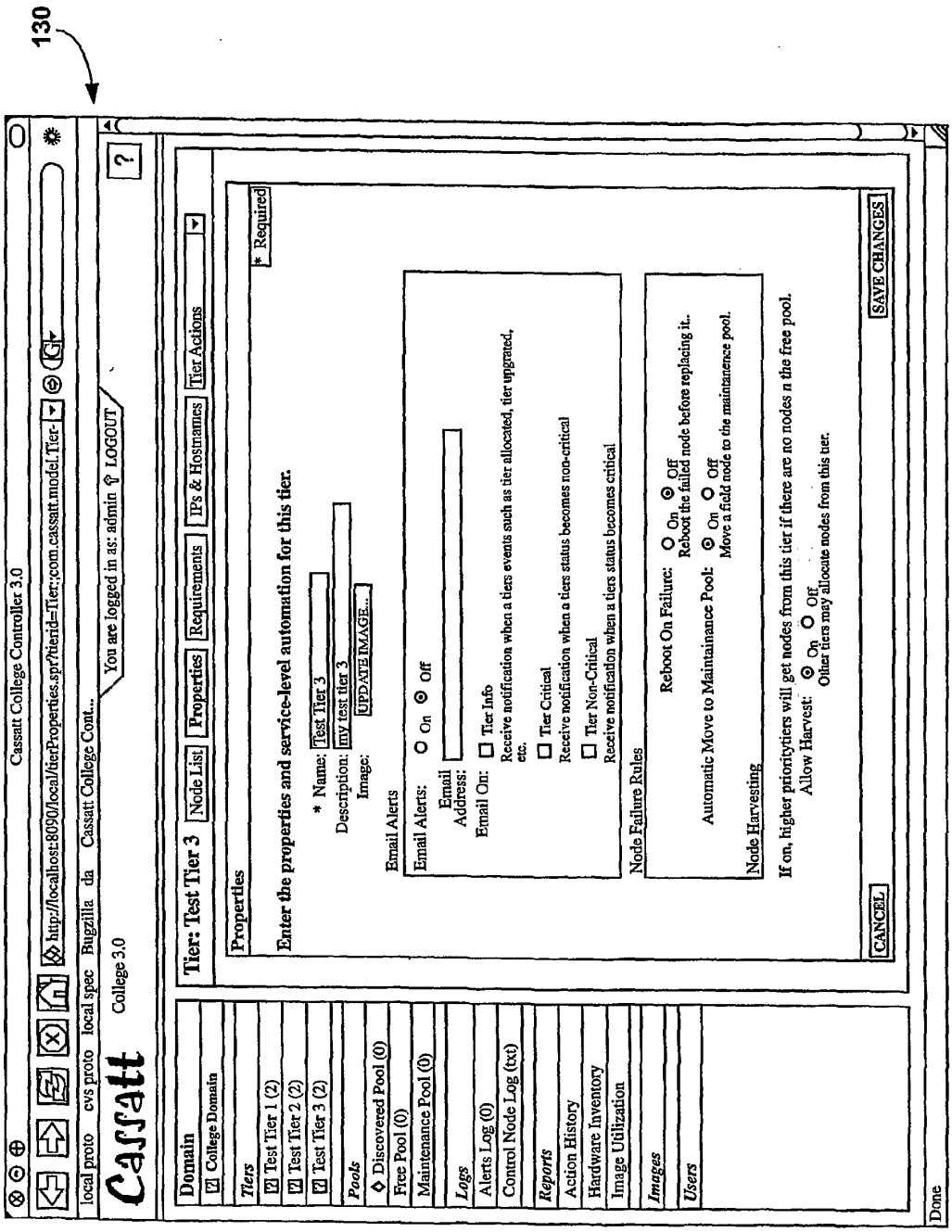


FIG. 8

Netflix, Inc. - Ex. 1002, Page 000534
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

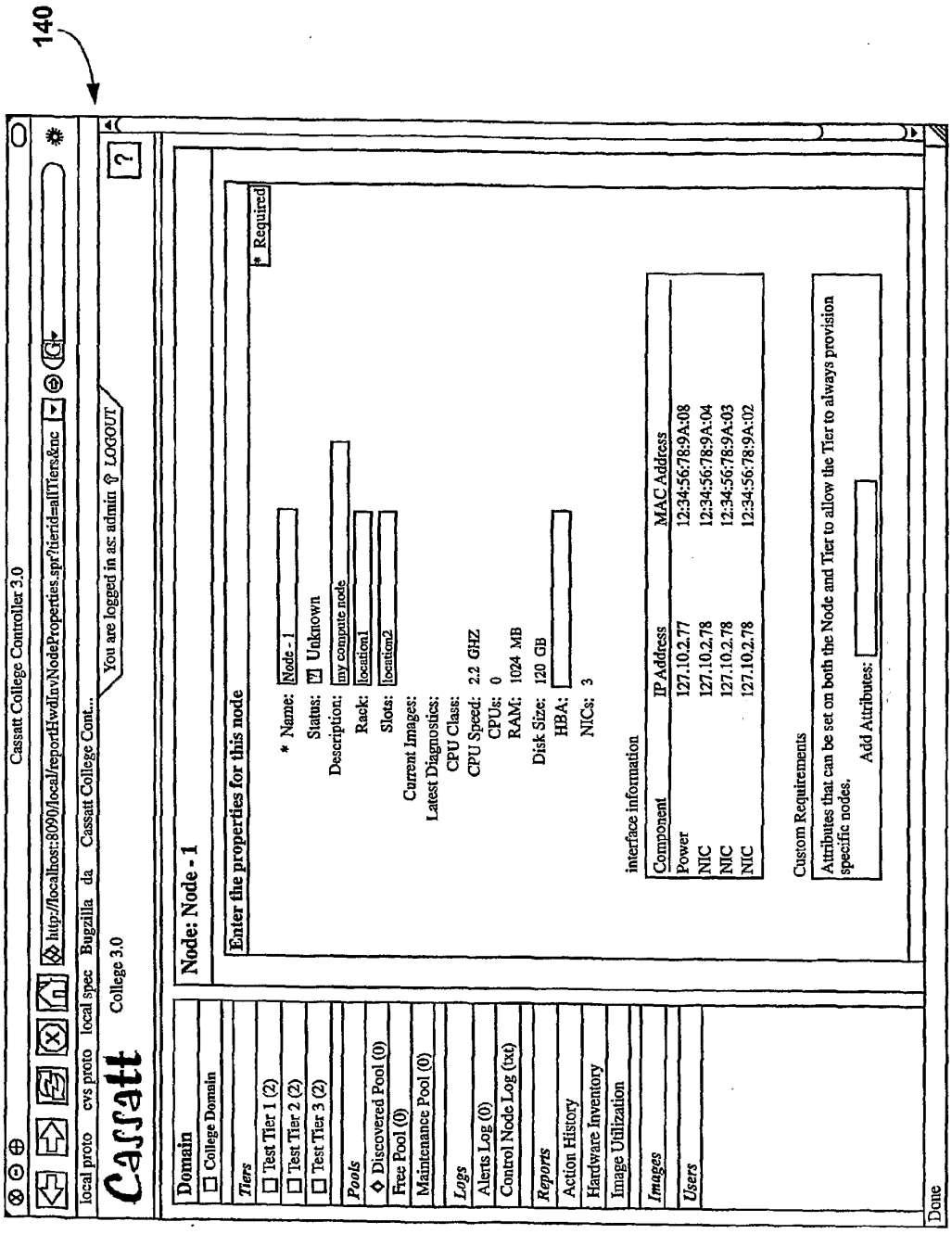


FIG. 9

Netflix, Inc. - Ex. 1002, Page 000535
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

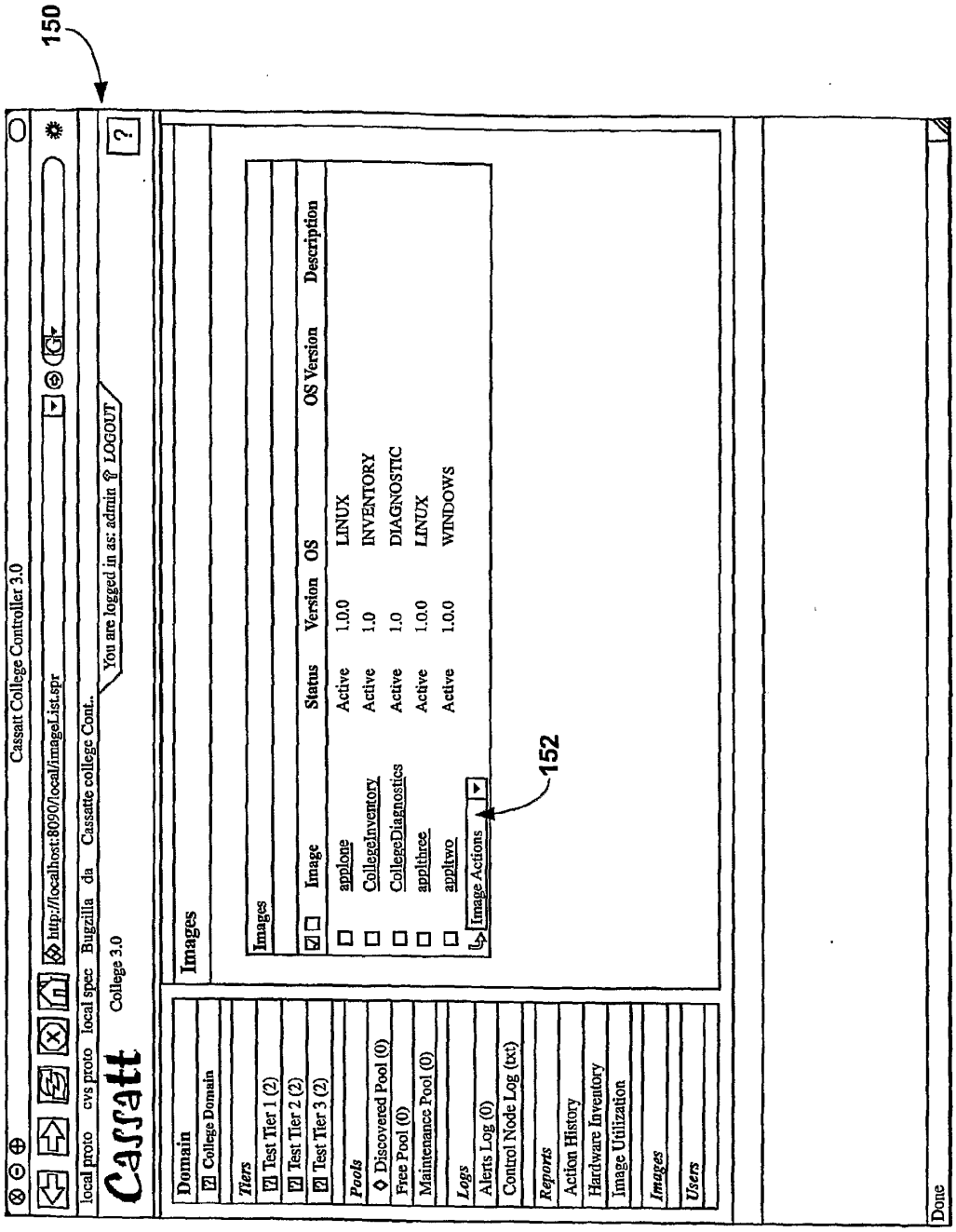


FIG. 10

Netflix, Inc. - Ex. 1002, Page 000536
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

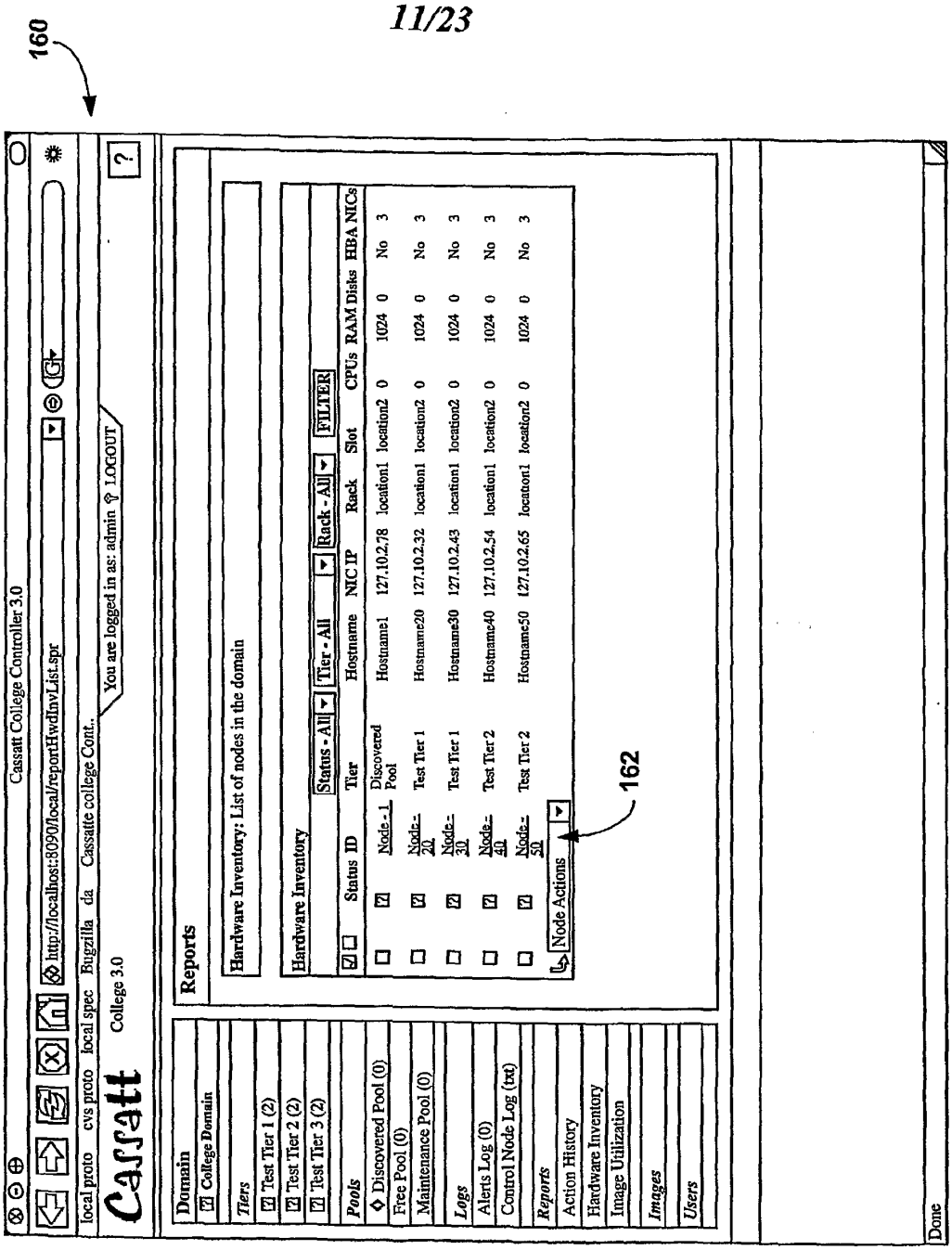


FIG. 11

Netflix, Inc. - Ex. 1002, Page 000537
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

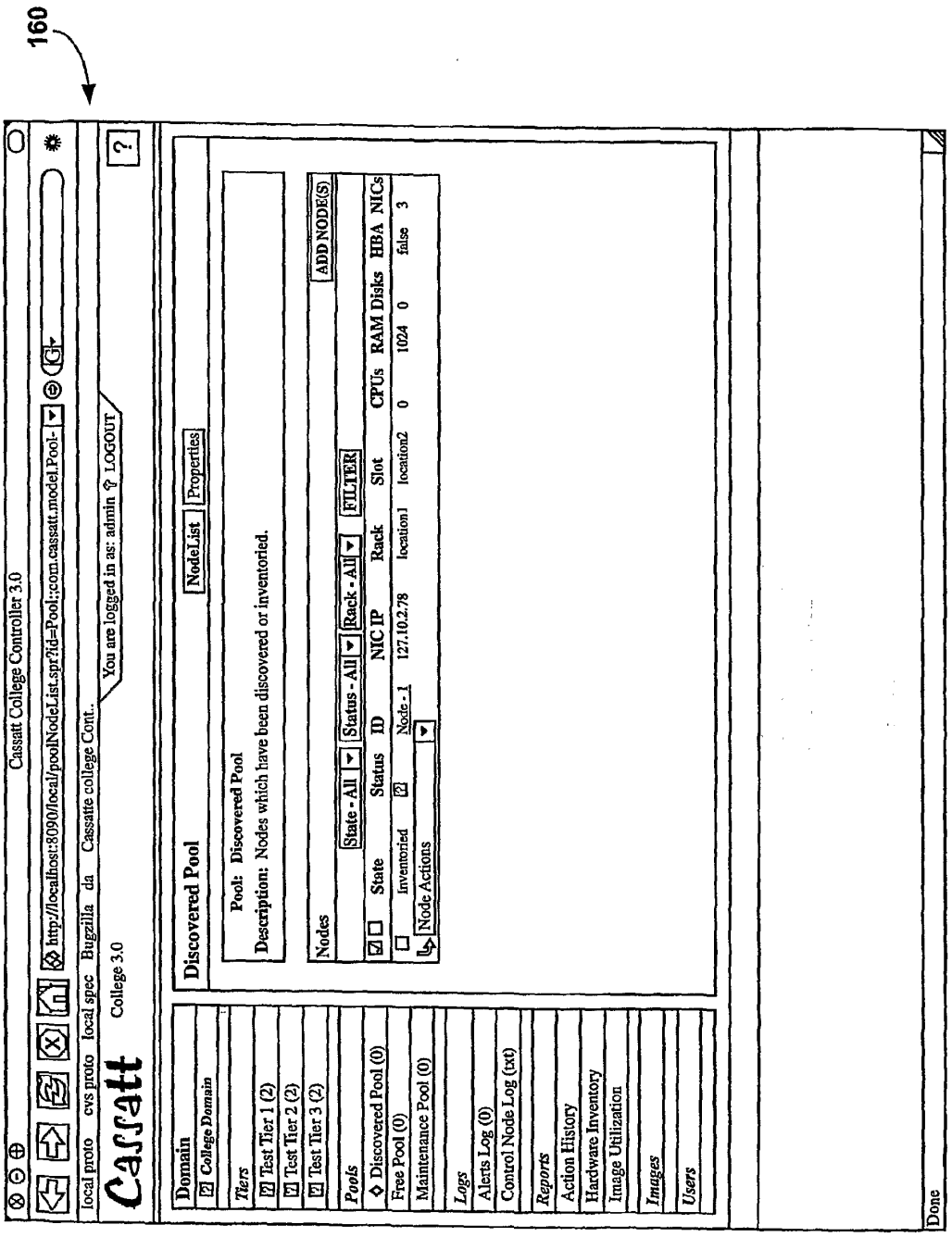


FIG. 12

Netflix, Inc. - Ex. 1002, Page 000538
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

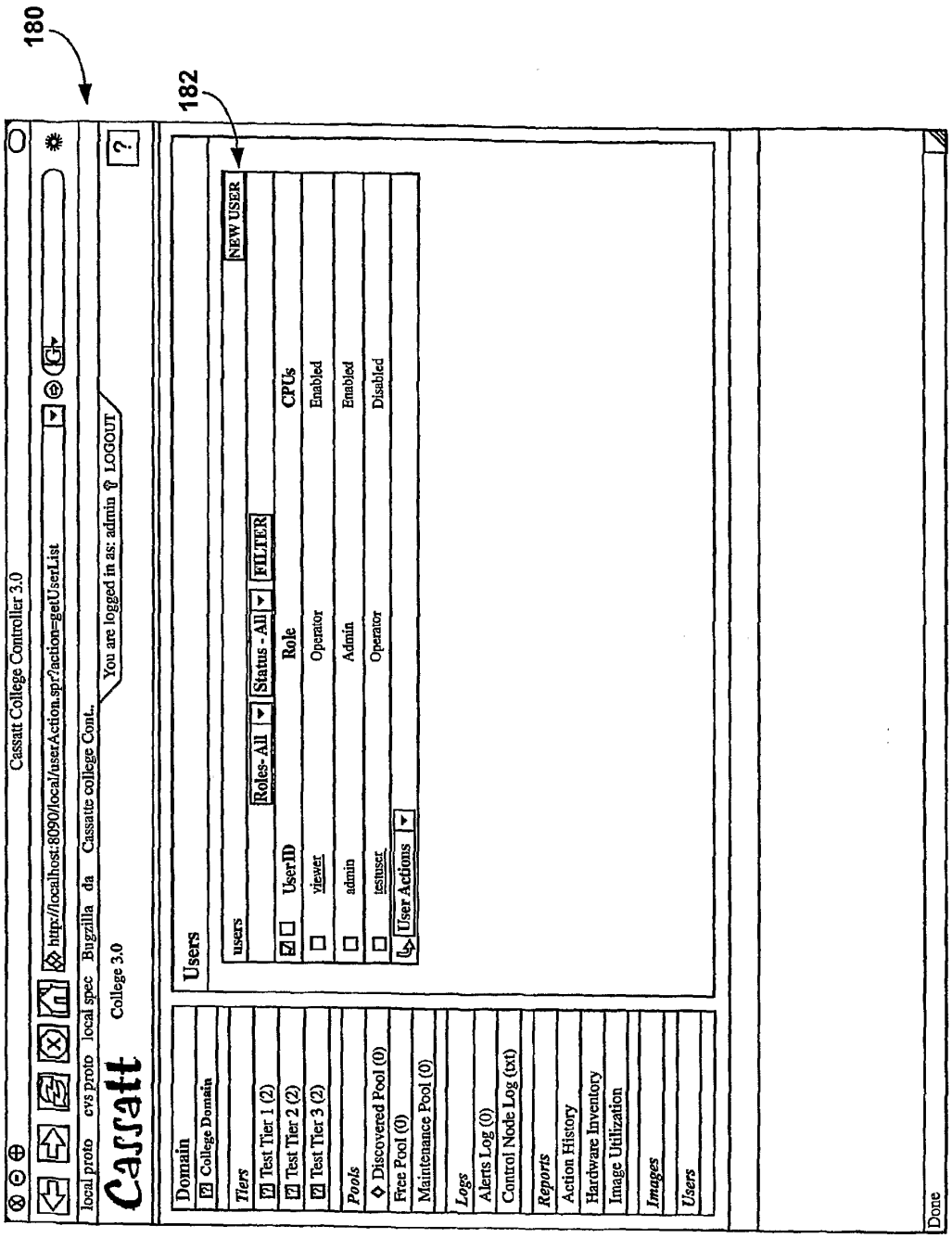


FIG. 13

Netflix, Inc. - Ex. 1002, Page 000539
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

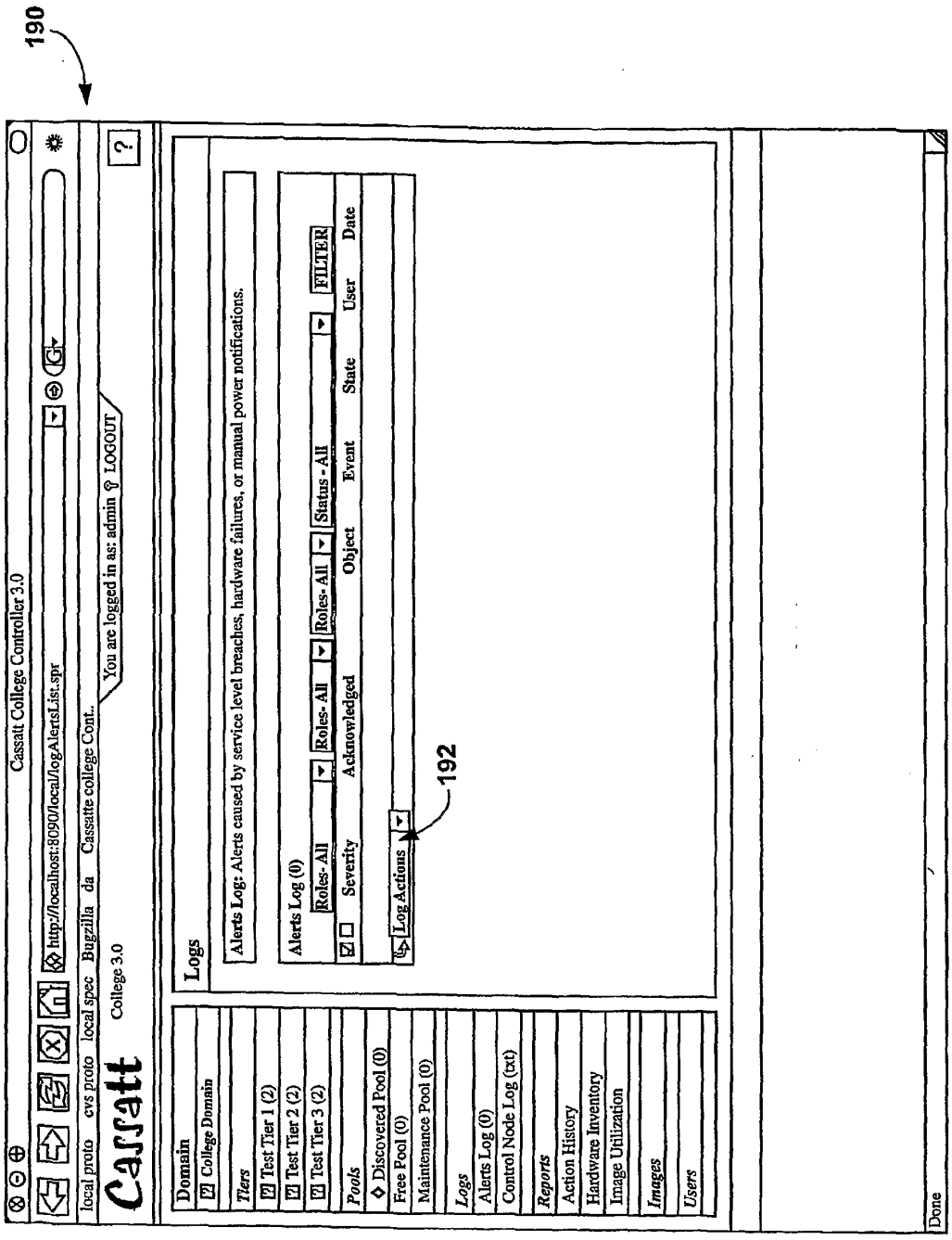


FIG. 14

Netflix, Inc. - Ex. 1002, Page 000540
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

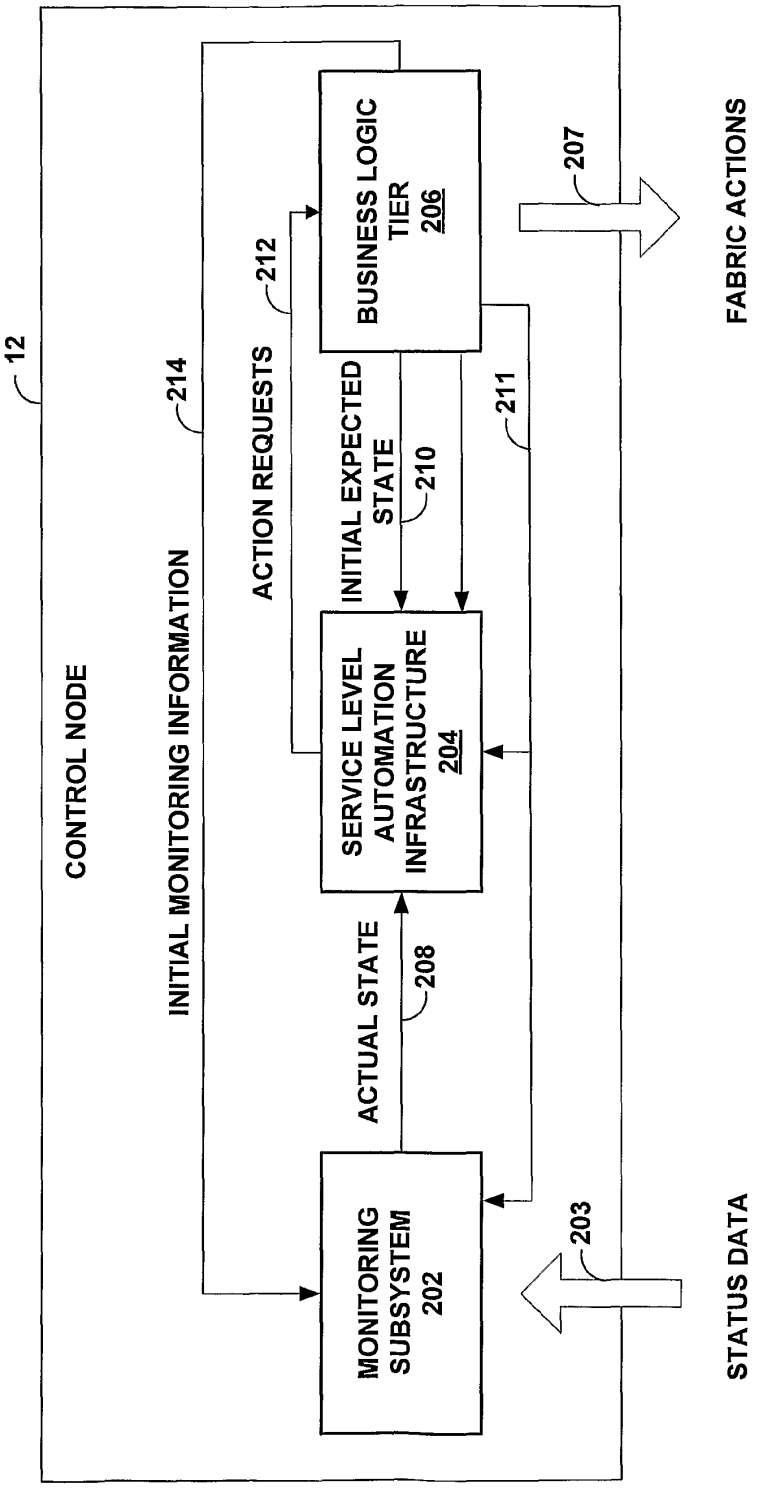

12
CONTROL NODE
214
INITIAL MONITORING INFORMATION
212
ACTION REQUESTS
BUSINESS LOGIC TIER 206
207
FABRIC ACTIONS
INITIAL EXPECTED STATE
210
211
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
ACTUAL STATE
208
MONITORING SUBSYSTEM 202
203
STATUS DATA

FIG. 15

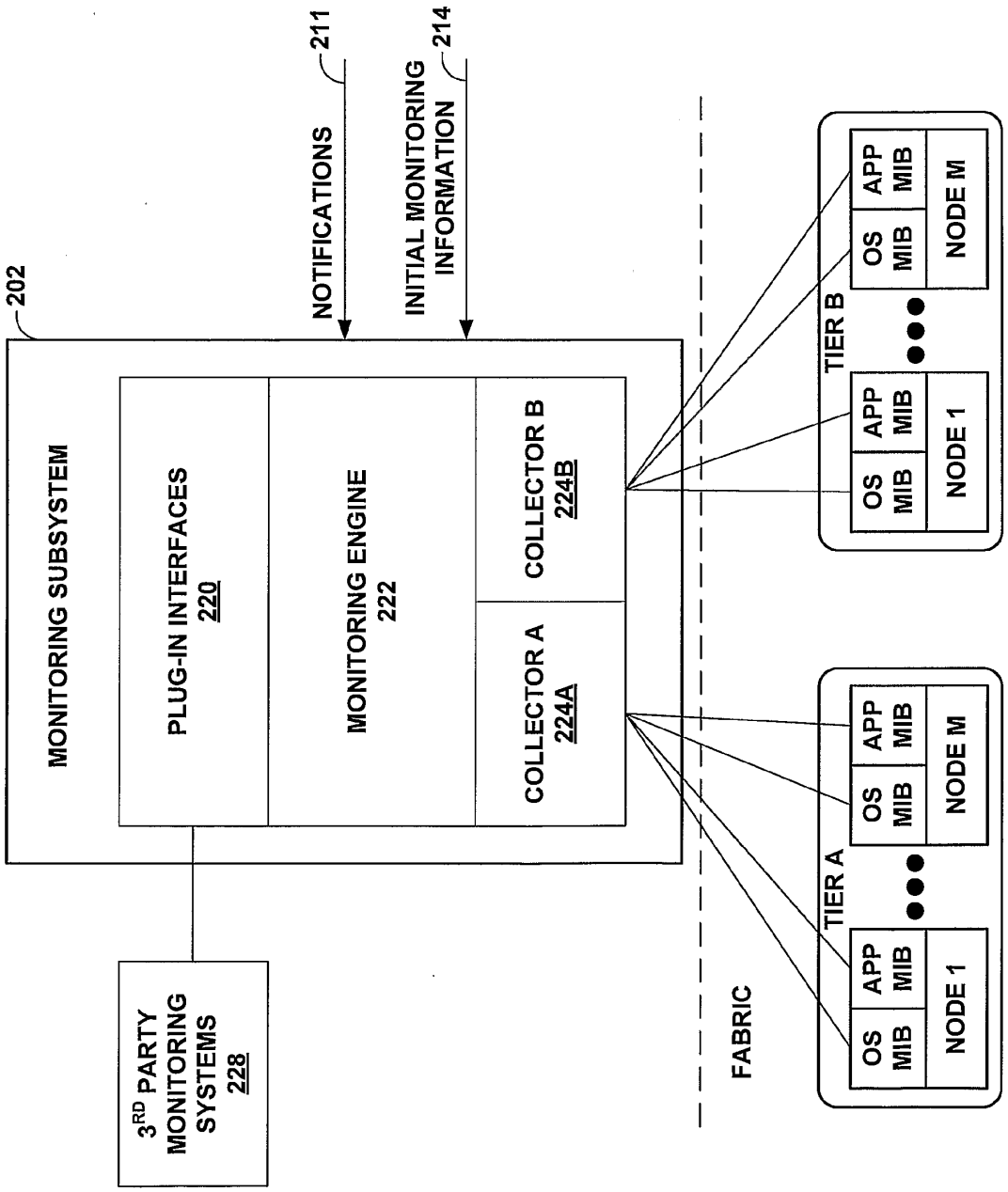

202
MONITORING SUBSYSTEM
PLUG-IN INTERFACES 220
MONITORING ENGINE 222
COLLECTOR A 224A
COLLECTOR B 224B
NOTIFICATIONS 211
INITIAL MONITORING INFORMATION 214
3RD PARTY MONITORING SYSTEMS 228
FABRIC
TIER A
TIER B
OS MIB
APP MIB
NODE 1
NODE M

FIG. 16

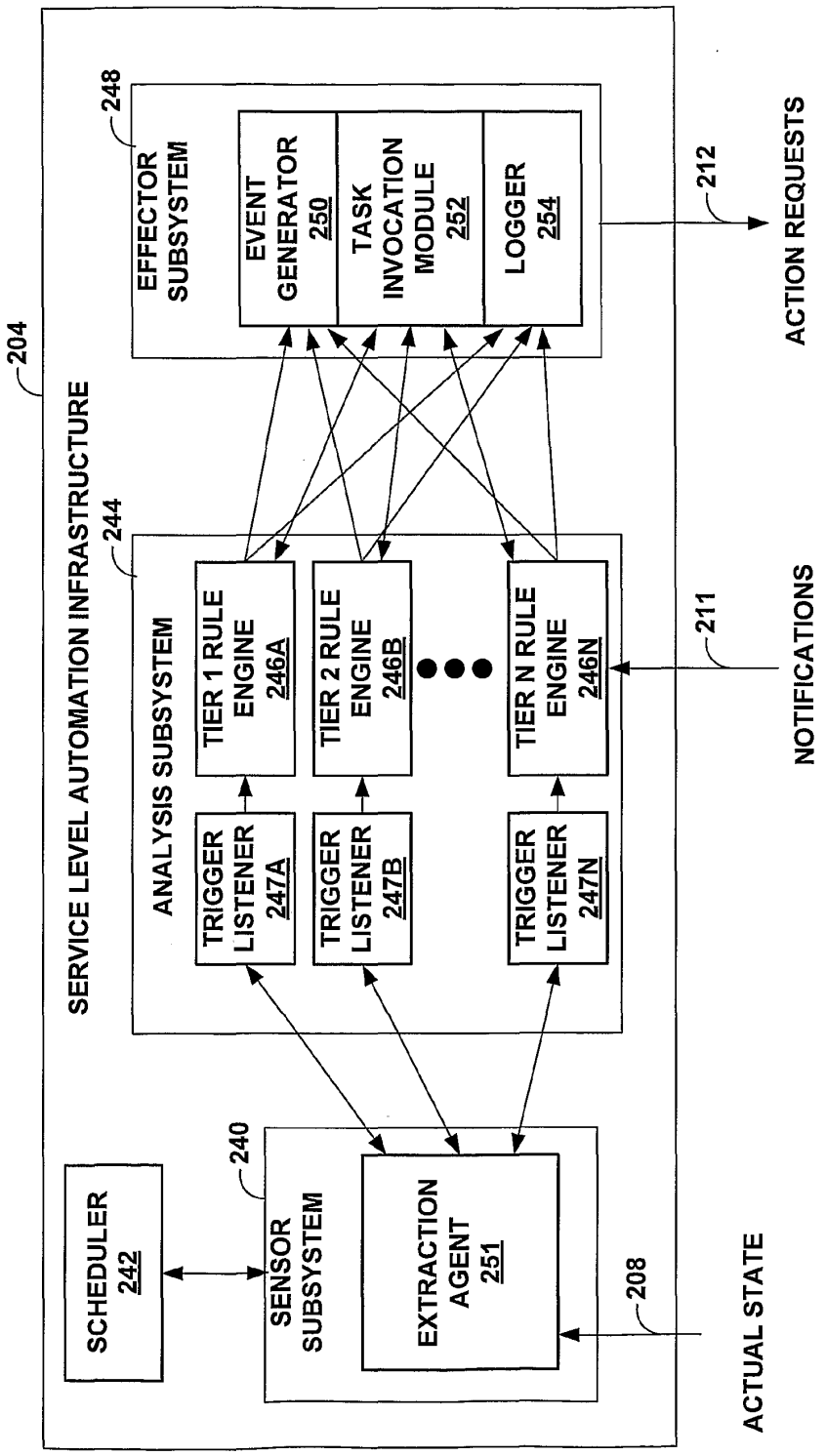

204
SERVICE LEVEL AUTOMATION INFRASTRUCTURE
248
EFFECTOR SUBSYSTEM
EVENT GENERATOR 250
TASK INVOCATION MODULE 252
LOGGER 254
212
ACTION REQUESTS
244
ANALYSIS SUBSYSTEM
TIER 1 RULE ENGINE 246A
TIER 2 RULE ENGINE 246B
TIER N RULE ENGINE 246N
TRIGGER LISTENER 247A
TRIGGER LISTENER 247B
TRIGGER LISTENER 247N
211
NOTIFICATIONS
240
SENSOR SUBSYSTEM
EXTRACTION AGENT 251
SCHEDULER 242
208
ACTUAL STATE

FIG. 17

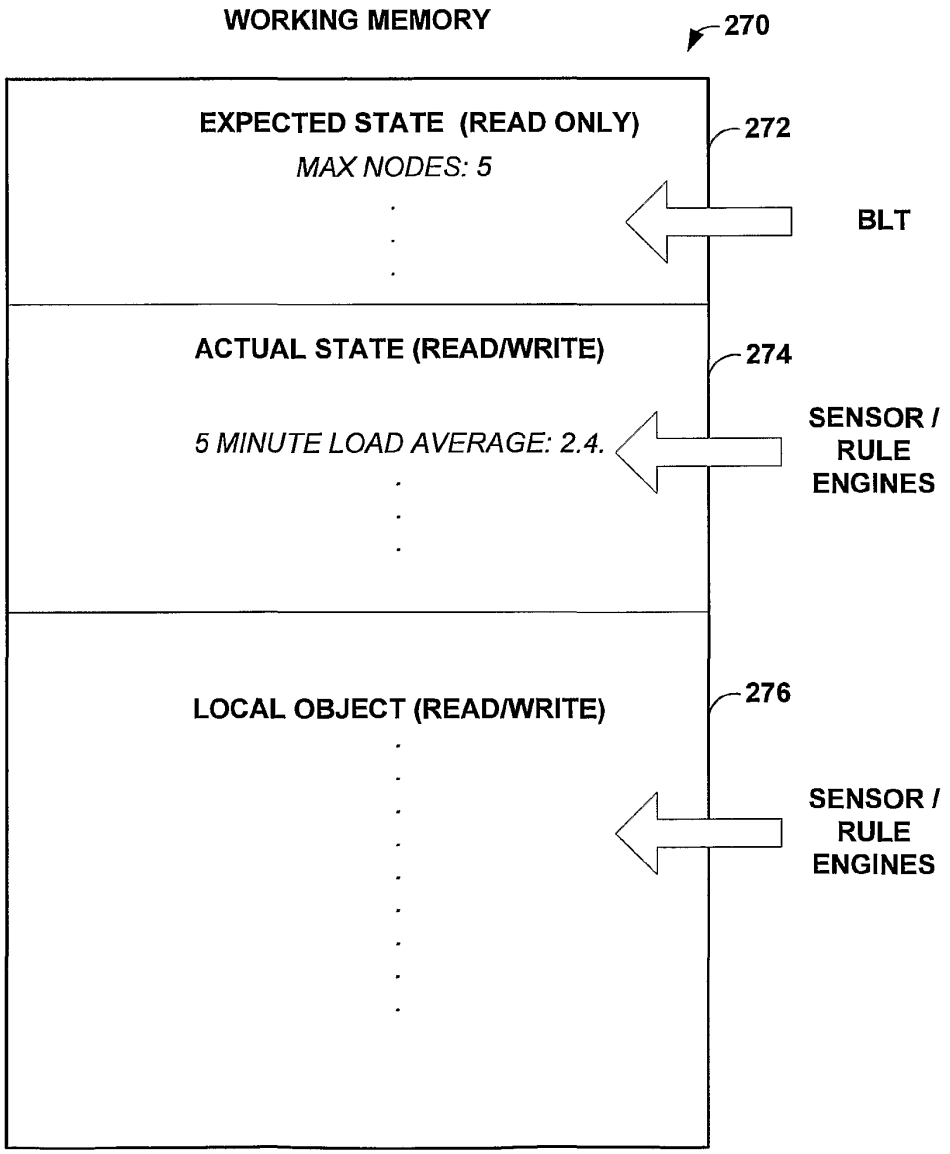

WORKING MEMORY
270
EXPECTED STATE (READ ONLY)
MAX NODES: 5
272
BLT
ACTUAL STATE (READ/WRITE)
274
5 MINUTE LOAD AVERAGE: 2.4.
SENSOR / RULE ENGINES
LOCAL OBJECT (READ/WRITE)
276
SENSOR / RULE ENGINES

FIG. 18

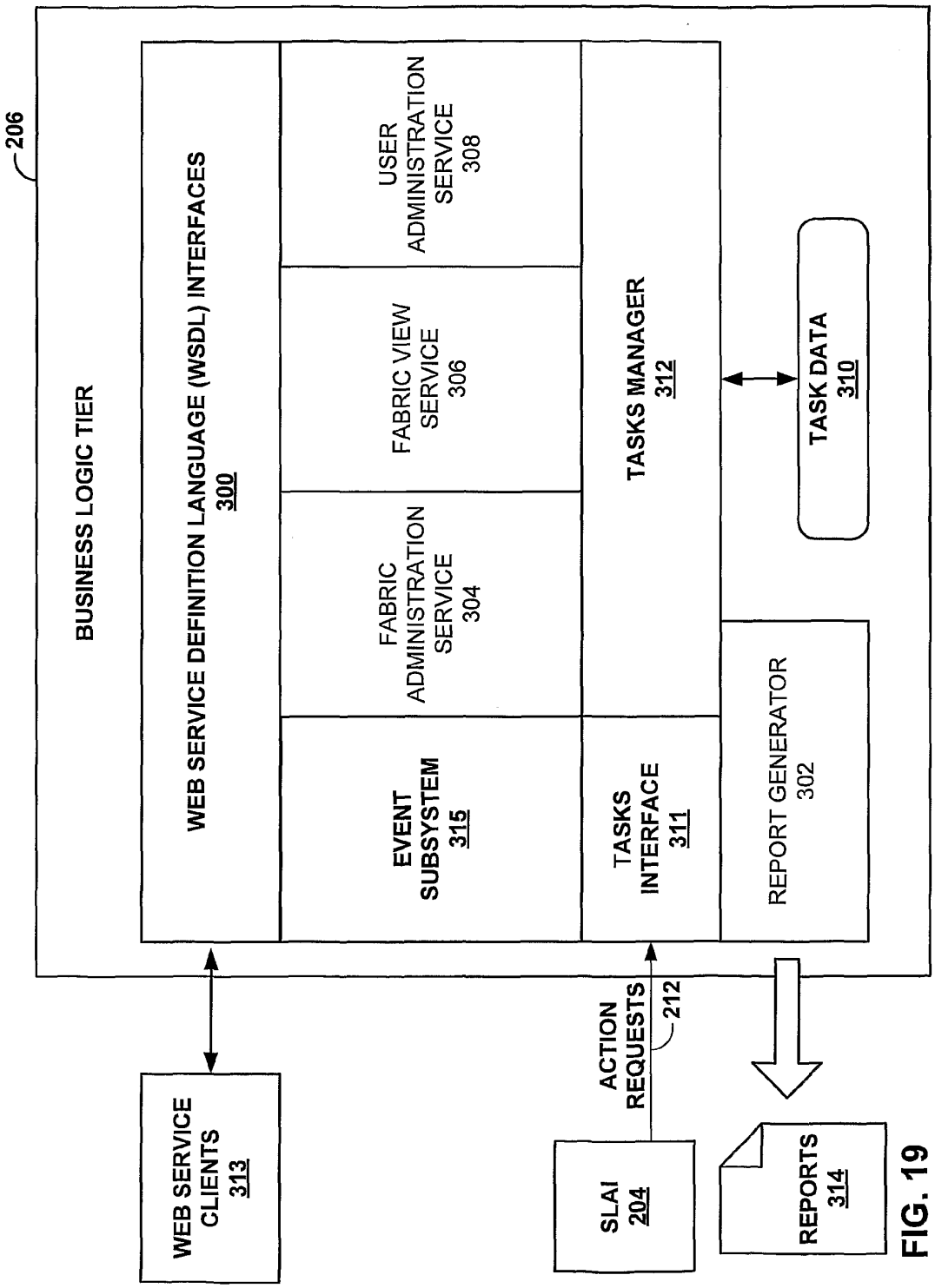

206
BUSINESS LOGIC TIER
WEB SERVICE DEFINITION LANGUAGE (WSDL) INTERFACES 300
EVENT SUBSYSTEM 315
FABRIC ADMINISTRATION SERVICE 304
FABRIC VIEW SERVICE 306
USER ADMINISTRATION SERVICE 308
TASKS INTERFACE 311
TASKS MANAGER 312
REPORT GENERATOR 302
TASK DATA 310
WEB SERVICE CLIENTS 313
SLAI 204
ACTION REQUESTS 212
REPORTS 314

FIG. 19

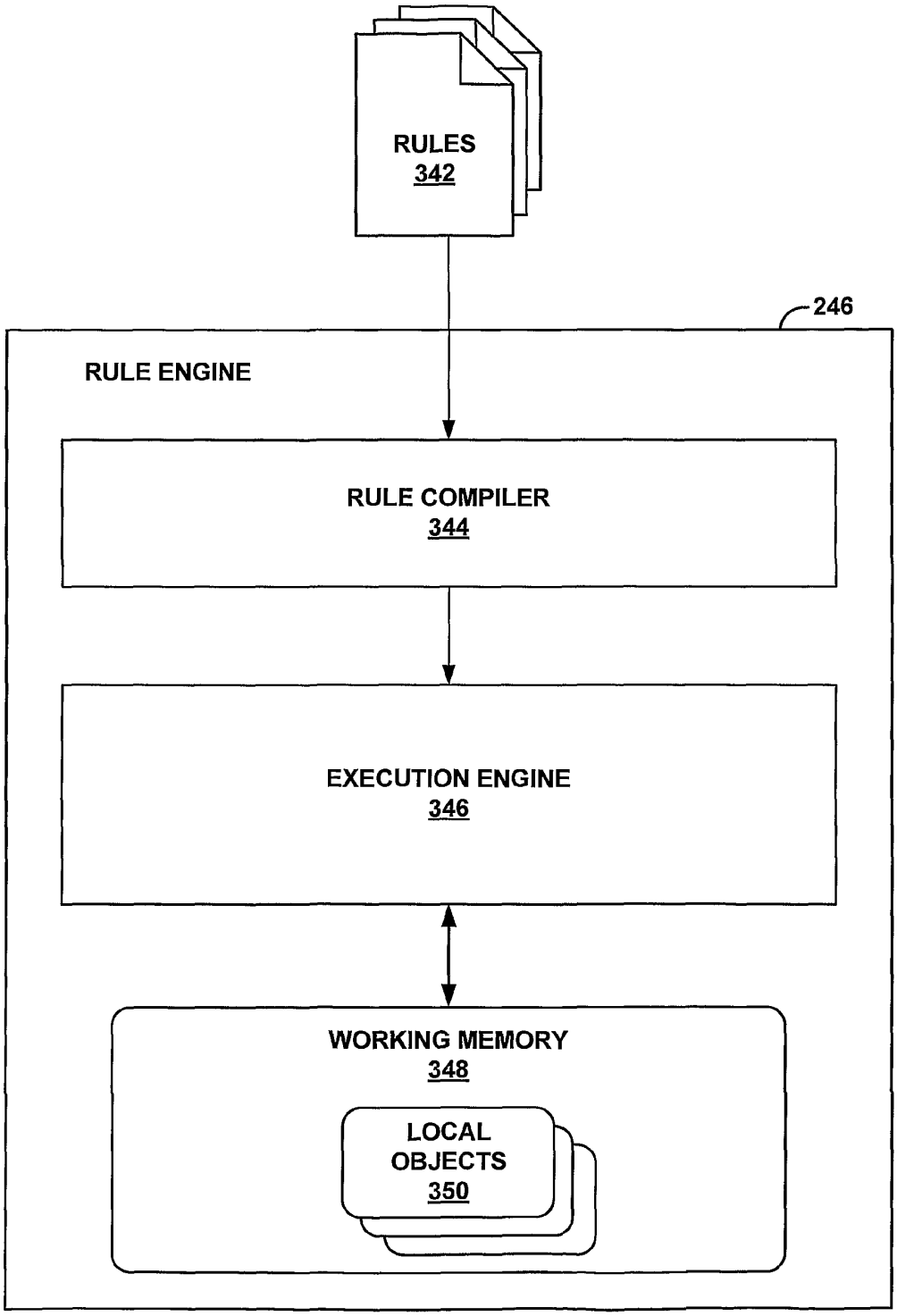

RULES
342
246
RULE ENGINE
RULE COMPILER
344
EXECUTION ENGINE
346
WORKING MEMORY
348
LOCAL
OBJECTS
350

FIG. 20

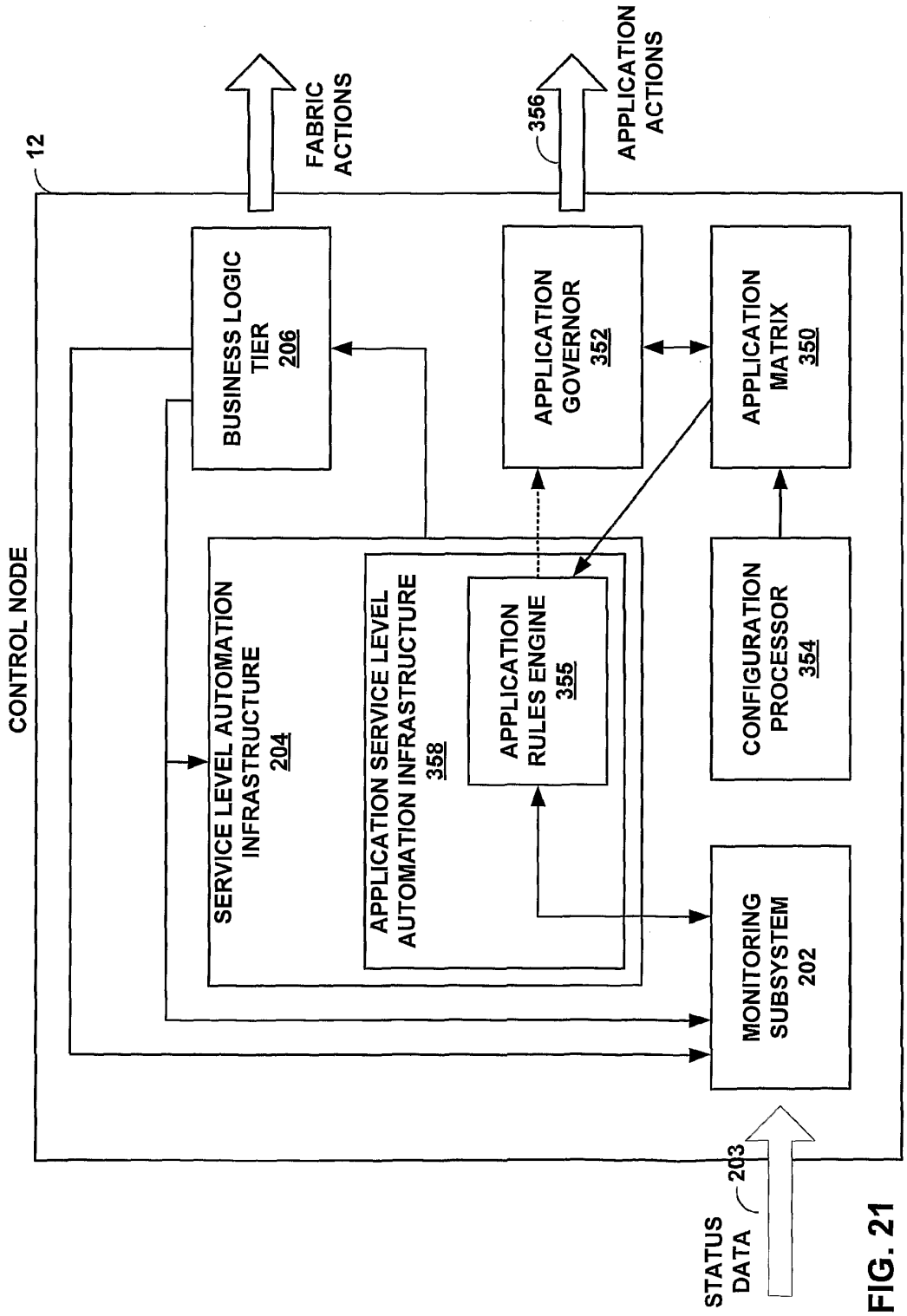

12
CONTROL NODE
FABRIC ACTIONS
356
APPLICATION ACTIONS
BUSINESS LOGIC TIER 206
APPLICATION GOVERNOR 352
APPLICATION MATRIX 350
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
APPLICATION SERVICE LEVEL AUTOMATION INFRASTRUCTURE 358
APPLICATION RULES ENGINE 355
CONFIGURATION PROCESSOR 354
MONITORING SUBSYSTEM 202
203
STATUS DATA

FIG. 21

APPLICATION (360)	APPLICATION NODES (362)	SERVICES (364)	NODE MONITORED VALUES (366)	SERVICE MONITORED ATTRIBUTES (368)	SERVICE LEVELS (370)	DEPLOYMENT CONSTRAINTS (372)
Name= DataDomain **adminIP=** "172.31.64.201" **adminPort=** "1100" **adminTier=** "Web Admin" **clusterName=** "PathCluster" **expectedStartupDelay=**120 **loadDelay=**120 **maxNodes=**2 **minNodes=**1	**IP=** 172.31.64.201 **Name=** managedServer_0 **IP=** 172.31.64.202 **Name=** managedServer_1	**name=**"AI Design-time" **path=**"/lib/ai-designtime.ear" **name=**"System EJBs" **path=**"/lib/ejbs.ear" **name=**"Worklist Worker User Interface" **path=**"/lib/worklist/worklist.ear" **name=**"DBMS_ADK" **path=**"/lib/DBMS_ADK.ear" **name=**"UserApp" **path=**"/user_projects/domains/userApp.ear"	**name=** Load5Average **name=** PercentFree-Memory	**name=** PendRequests **name=** ExecuteThread-IdleCount **name=** ExecuteThread-TotalCount	**attribute=** "BusyThread-Percentage" **expression=** "(ExecuteThreadTotalCount-ExecuteThreadIdleCount)*100/ExecuteThreadTotalCount" **frequency=**"15" **maxThreshold=**"20.0" **minThreshold=**"5.0" **period=**"30" **attribute=** "PendingRequests" **expression=** "PendingRequests" **frequency=**"15" **maxThreshold=**"20" **minThreshold=**"5" **period=**"30"	**attribute=** "LoadAverage" **expression=** "Load5Average" **frequency=**"15" **maxThreshold=**"4" **minThreshold=**"-1" **period=**"15" **attribute=** "FreeMemory" **expression=** "PercentMemoryFree *100" **frequency=**"15" **maxThreshold=**"-1" **minThreshold=**"1" **period=**"60"
name= Portal-Domain						

FIG. 22

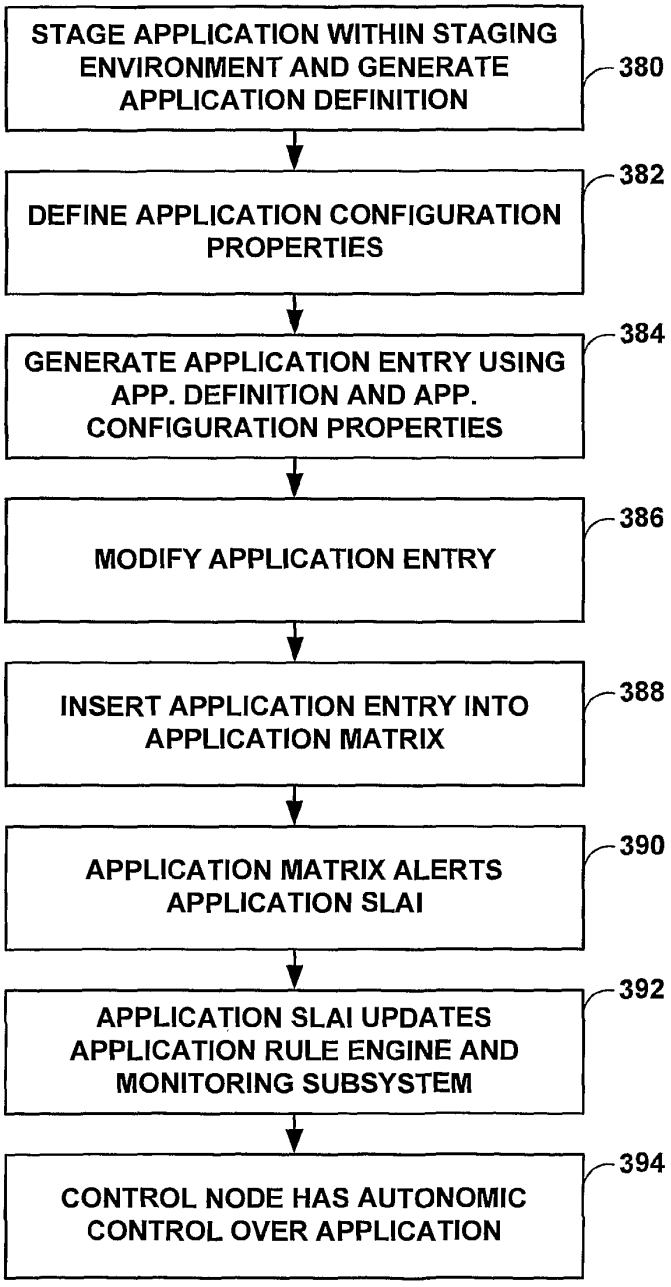

STAGE APPLICATION WITHIN STAGING ENVIRONMENT AND GENERATE APPLICATION DEFINITION
380
DEFINE APPLICATION CONFIGURATION PROPERTIES
382
GENERATE APPLICATION ENTRY USING APP. DEFINITION AND APP. CONFIGURATION PROPERTIES
384
MODIFY APPLICATION ENTRY
386
INSERT APPLICATION ENTRY INTO APPLICATION MATRIX
388
APPLICATION MATRIX ALERTS APPLICATION SLAI
390
APPLICATION SLAI UPDATES APPLICATION RULE ENGINE AND MONITORING SUBSYSTEM
392
CONTROL NODE HAS AUTONOMIC CONTROL OVER APPLICATION
394

FIG. 23

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003028

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 July 2006	18/07/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ross, C

Form PCT/ISA/210 (second sheet) (April 2005)

Netflix, Inc. - Ex. 1002, Page 000550
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HERNESS E N ET AL: "WebSphere Application Server: a foundation for on demand computing" IBM SYSTEMS JOURNAL IBM USA, vol. 43, no. 2, 2004, pages 213-237, XP002383791 ISSN: 0018-8670 Paragraph starting with "Multitiered distributed business computing": page 215, left-hand column page 220, left-hand column, line 26 - page 221, right-hand column, line 3 Paragraph starting with "Deployment automation": page 222, left-hand column - right-hand column, line 11 page 223, left-hand column, line 5 - line 23 page 224, right-hand column, line 11 - right-hand column, line 3 page 226, left-hand column, line 9 - page 229, left-hand column, line 2 page 229, right-hand column, line 1 - line 13 Paragraph starting with "Monitoring and application response measurement...": page 230, left-hand column	1-34
X	G LODI, F PANZIERI: "QoS-aware Clustering of Application Servers" PROCEEDINGS OF THE 1ST IEEE WORKSHOP ON QUALITY OF SERVICE FOR APPLICATION SERVERS, IN CONJUNCTION WITH THE 23RD INT'L SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS 2004), 17 October 2004 (2004-10-17), pages 1-6, XP002383792 Florianopolis, Brazil page 1, line 1 - line 19 page 2, left-hand column, line 10 - right-hand column, line 18 page 3, right-hand column, line 27 - line 45	1-34
X	US 2003/126265 A1 (AZIZ ASHAR ET AL) 3 July 2003 (2003-07-03) abstract; figure 1c paragraph [0049] paragraph [0052] paragraph [0088] - paragraph [0095] paragraph [0206]	1-34

-/--

Netflix, Inc. - Ex. 1002, Page 000551
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003028

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/177176 A1 (HIRSCHFELD ROBERT A ET AL) 18 September 2003 (2003-09-18) abstract column 7, line 4 - line 10 column 9, line 29 - line 52; figure 3 column 13, line 23 - line 53; figures 5a,5b	1-34

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

Netflix, Inc. - Ex. 1002, Page 000552
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/003028

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003126265 A1	03-07-2003	NONE	
US 2003177176 A1	18-09-2003	NONE	

Form PCT/ISA/210 (patent family annex) (April 2005)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 August 2006 (03.08.2006)

PCT

(10) International Publication Number
WO 2006/081503 A1

(51) International Patent Classification:
G06F 9/445 (2006.01)

(21) International Application Number:
PCT/US2006/003100

(22) International Filing Date: 27 January 2006 (27.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/046,133 28 January 2005 (28.01.2005) US

(71) Applicant *(for all designated States except US)*: **CASSATT CORPORATION** [US/US]; 1740 Technology Drive, San Jose, California 95110 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **HENSELER, David, A.** [US/US]; 928 Demont Avenue East, Maplewood, Minnesota 55109 (US). **GILBERTSON, Jeanne, C.** [US/US]; 7140 Derby Drive, Chanhassen, Minnesota 55317 (US). **CAHIR, Margaret** [US/US]; 1582 Mississippi River Blvd. S., St. Paul, Minnesota 55116 (US).

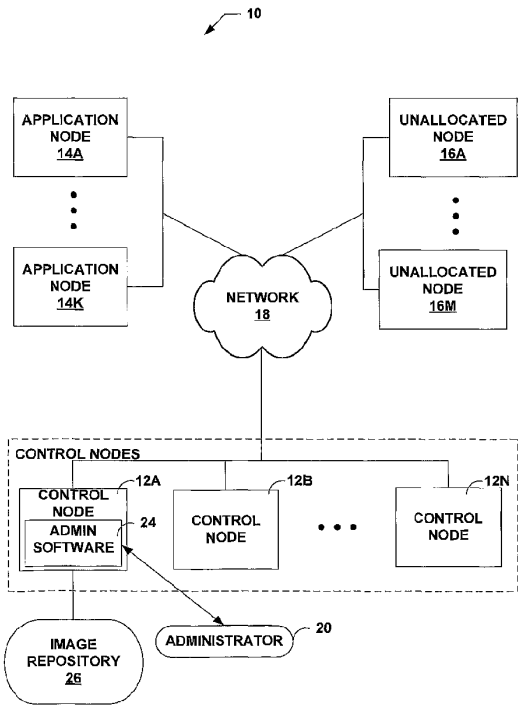
(74) Agent: **SIEFFERT, Kent, J.**; SHUMAKER & SIEFFERT, P.A., 8425 Seasons Parkway, Suite 105, St. Paul, Minnesota 55125 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT,

[Continued on next page]

(54) Title: MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM



(57) Abstract: In general, the invention is directed to software image management within a distributed computing system. The distributed computing system includes a plurality of application nodes interconnected via a communications network, and an image repository storing a plurality of pre-generated software image instances. A control node may pre-assign a network address to each of the image instances stored within the image repository, and assigns the network addresses of the image instances to the application nodes upon subsequent deployment of the software images to the application nodes.

WO 2006/081503 A1

RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM

TECHNICAL FIELD

[0001] The invention relates to computing environments and, more specifically, to management of computing nodes of a distributed processing systems.

BACKGROUND

[0002] Distributed computing systems are increasingly being utilized to support high-performance computing applications. Typically, distributed computing systems are constructed from a collection of computing nodes that combine to provide a set of processing services to implement the high performance computing applications. Each of the computing nodes in the distributed computing system is typically a separate, independent computing device interconnected with each of the other computing nodes via a communications medium, e.g., a network.

[0003] One of the challenges with distributed computing systems is the management of the software images associated with the computing nodes. The term "software image" generally refers to the complete set of software associated with an individual computing node, including the operating system and all boot code, middleware and application files.

[0004] One challenge arises when a new computing node is allocated to the distributed computing system. In this situation, the computing node must be loaded with a software image that is consistent with the current software images running on the related computing nodes. Similarly, it may be necessary to load a new software image on a computing node when the functions of computing node change within the distributed computing system.

[0005] Moreover, other challenges arise when installing new software, updating software versions or applying software patches to the software images associated with the computer nodes. For example, it is often necessary to shutdown and reboot a computing node when installing or updating software on the computing node. In some situations, it may even be necessary to take the entire distributed computing system offline when performing substantial software installations or updates. As a result, the computing performance of the distributed computing system may be severely impacted during the installation or update process.

Netflix, Inc. - Ex. 1002, Page 000556
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

SUMMARY

[0006] In general, the invention is directed to software image management within a distributed computing system. In particular, the invention provides techniques for the efficient allocation and management of software images within the distributed computing system. A control node, for example, generates a "golden image" for each type of software image that may be deployed on one or more of application nodes.

[0007] As described herein, the term "golden image" refers to a reference copy of a complete software stack that may be deployed on application node to achieve identical functionality. In other words, a golden image may be viewed as a "master copy" of the software image for a particular computing function. The control node may create a copy of a golden image, referred to as an "image instance," for each of possible application node that may be deployed within distributed computing system. In other words, the control node pre-generates a set of K image instances, where K represents the maximum number of application nodes for a particular computing function. Control node may create the set of K image instance regardless of whether the actual number of application nodes initially deployed to perform that function is less than K. Moreover, the control node creates a complete set of image instances for each computing function that may be deployed within the distributed computing system.

[0008] In one embodiment, a method comprises storing a master software image for a type of computing function to be deployed within a distributed computing system, and generating K software image instances from the master software image, wherein the distributed computing system is configured to deploy a maximum of K software image instances for the type of computing function. The method further comprises deploying the software image instances on M application nodes within the distributed computing system, where $M \leq K$.

[0009] In another embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network, and an image repository storing a plurality of K pre-generated software image instances, wherein K represents a maximum number of the software image instances to be deployed within the distributed computing system for a similar computing function.

[0010] In another embodiment, a computer-readable medium comprises instructions that cause a programmable process to present a first user interface to receive input identifying a master software image for a type of computing function to be deployed within a

Netflix, Inc. - Ex. 1002, Page 000557
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

distributed computing system, and present a second user interface to receive input identifying a maximum number of software image instances that can be deployed within the distributed computing system for the type of computing function. The instructions further cause the processor to generate a software image instance from the master software image for each of the image instances that can be deployed.

[0011] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0012] FIG. 1 is a block diagram illustrating a distributed computing system constructed from a collection of computing nodes.

[0013] FIG. 2 is an exemplary image management data structure maintained by a control node in associated with a software image repository.

[0014] FIG. 3 is a flow diagram illustrating exemplary operation of the administration computing node when managing software images during configuration of a distributed processing environment.

[0015] FIG. 4 is a flow diagram illustrating exemplary operation of the administration computing node when utilizing an unallocated computing node.

[0016] FIG. 5 is a flow diagram illustrating exemplary operation of the administration computing node when updating a software image.

[0017] FIG. 6 is a screen illustration of an exemplary user interface for viewing and managing software images.

[0018] FIG. 7 is a screen illustration of an exemplary user interface for allowing a system administrator to select computing nodes for a software image upgrade.

[0019] FIG. 8 is a screen illustration of an exemplary user interface for manually initiating software image updates.

DETAILED DESCRIPTION

[0020] FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. Distributed computing system 10 may

Netflix, Inc. - Ex. 1002, Page 000558
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

be viewed as a collection of computing nodes operating in cooperation with each other to provide high-performance processing.

[0021] In one embodiment, the collection of computing nodes forming distributed computing system 10 includes control nodes 12A-12N ("control nodes 12"), application nodes 14A-14K ("application nodes 14") and unallocated nodes 16A-16M ("unallocated nodes 16"). Control nodes 12, application nodes 14 and unallocated nodes 16 are interconnected via a communications network 18. Communications network 18 permits internode communications among the computing nodes as the nodes perform interrelated operations and functions. Communications network 18 may comprise, for example, direct connections between one or more of the computing nodes, one or more customer networks maintained by an enterprise, local area networks (LANs), wide area networks (WANs) or a combination thereof.

[0022] Within distributed computing system 10, a computing node represents an individual physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

[0023] As illustrated in FIG. 1, the computing nodes of distributed computing system 10 are divided into application nodes 14, control nodes 12, and unallocated nodes 16. In general, application nodes 14 provide computing environments for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. Application nodes 14 typically have more substantial I/O capabilities than control nodes 12, and are typically configured with more computing resources (e.g., processors and memory).

[0024] In contrast, control nodes 12 provide the system support functions for managing distributed computing system 10. More specifically, control nodes 12 manage the roles of each computing node within distributed computing system 10 and the installation and execution of software applications within the distributed computing system. In general, distributed computing system 10 includes at least one control node, such as control node 12A, and, optionally, additional control nodes 12B-12N.

[0025] Unallocated nodes 16 represent computing nodes that are available for use in distributed computing system 10, but that are not currently configured and deployed within the distributed computing system 10. Unallocated nodes 16 may, for example, be

Netflix, Inc. - Ex. 1002, Page 000559
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

used to replace failed application nodes or to add application nodes to distributed computing system 10 to increase processing capacity.

[0026] In the example of FIG. 1, control node 12A is responsible for software image management as described in detail herein. The term "software image" refers to a complete set of software associated with an individual computing node, including the operating system and all boot code, middleware and application files. Other control nodes 12 are optional and may be associated with a different subset of the computing nodes within distributed computing system 10. Moreover, control node 12A may be replicated to provide primary and backup administration functions, thereby allowing for graceful handling a failover in the event control node 12A fails.

[0027] In one embodiment, each of the computing nodes within distributed computing system 10 execute a common general-purpose operating system. One example of a general-purpose operating system is the Windows™ operating system provided by Microsoft Corporation. In some embodiments, the general-purpose operating system may be the Linux kernel.

[0028] In general, a system administrator 20 accesses administration software 24 ("admin software 24" in FIG. 1) executing on control node 12A to define the architecture of distributed computing system 10. For example, system administrator 20 may define the maximum number of application nodes 14 for distributed computing system 10, the software and hardware resources for each application node, the function of each application node (e.g., web server node), and the like. In some embodiments, control nodes 12 automatically detect the hardware resources for each of application nodes 14 and unallocated nodes 16.

[0029] System administrator 20 may further identify the particular types of software images to be installed on application nodes 14. Alternatively, administration software 24 executing on control node 12A will automatically identify the appropriate software images to be deployed to application nodes 14 based on the input received from the system administrator. For example, administration software 24 may determine the type of software image to associate with an application node 14 based on the functions assigned to the node by system administrator 20. Application nodes14 may be divided into a number of groups based on their assigned functionality. As one example, application nodes 14 may be divided into a first group to provide web server functions, a second group to provide business application functions and a third group to provide database

Netflix, Inc. - Ex. 1002, Page 000560
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

functions. The application nodes 14 of each group may be associated with different software images.

[0030] As described herein, a control node, such as control node 12A, provides for the efficient allocation and management of various software images within distributed computing system 10. In some embodiments, control node 12A generates a "golden image" for each type of software image that may be deployed on one or more of application nodes 14. As described herein, the term "golden image" refers to a reference copy of a complete software stack.

[0031] System administrator 20 may create a golden image by installing an operating system, middleware and software applications on a computing node and then making a complete copy of the installed software. In this manner, a golden image may be viewed as a "master copy" of the software image for a particular computing function. In the example described above, control node 12A may generate corresponding golden images for a web server function, a business application function and a database application function. Control node 12A maintains a software image repository 26 that stores the golden images associated with distributed computing system 10.

[0032] In accordance with the invention, control node 12A may create a copy of a golden image, referred to as an "image instance," for each possible image instance that may be deployed within distributed computing system 10 for a similar computing function. In other words, control node 12A pre-generates a set of K image instances, where K represents the maximum number of image instances for which distributed computing system 10 is configured for the particular type of computing function. For a given computing function, control node 12A creates the set of image instance even if not all of the image instances will be initially deployed. Control node 12A creates different sets of image instances for different computing functions, and each set may have a different number of image instances depending on the maximum number of image instances that may be deployed for each set. Control node 12A stores the image instances within software image repository 26. Each image instance represents a collection of bits that may be deployed on an application node.

[0033] For example, if administrator 20 configures distributed computing system 10 to have a maximum of five image instances for a particular type of computing function (i.e., K=5), then control node 12A creates five image instances from the corresponding golden image. In a similar fashion, control node 12A pre-generates the image instances for each functional grouping of application nodes 14. Continuing with the example, control node

Netflix, Inc. - Ex. 1002, Page 000561
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

12A may create software image instances including: (1) a first group having a maximum of five software image instances that may be deployed to support web server functions, (2) a second group having a maximum of three software image instances that may be deployed to support business application functions, and (3) a third group having a maximum of two software image instances that may be deployed to support database functions. Control node 12A utilizes the appropriate golden image to create the software image instances for each of the groups regardless of the number of software image instances that will initially be deployed for each of the functional groups.

[0034] For each image instance, control node 12A may pre-assign a network address, such as an Internet Protocol (IP) address and a host name, for use once the image instance is actually deployed on an application node. The network address may be assigned from a block of network addresses allocated to distributed computing system 10.

[0035] After creating and storing the image instances, control node 12A deploys all or a portion of the image instances. During this process, control node 12A selects the image instances to be deployed, possibly based on parameters provided by administrator 20, and associates the selected image instances with a respective one of application nodes 14 within distributed computing system 10. The remaining image instances are not assigned to specific application nodes at this time until actual deployment of the computing nodes. The computing nodes to which control node 12A assigns the image instances temporarily inherit the network address assigned to the image instance for as long as the image instance, or in some cases an associated patched image, is deployed on that particular application node. If for some reason the image instance is moved to a different application node 14, control node 12A moves the network address that new application node.

[0036] Different functional groups of application nodes 14 subsets are treated in a similar manner. Particularly, control node 12A creates image instances from a respective golden image for the maximum number computing nodes of the group, assigns network addresses to each of the pre-created image instances, stores the pre-created image instances within image repository 26, assigns one of the pre-created image instances to the each of application nodes 14 to be initially deployed for the group, and boots the assigned image instances on respective application nodes. Control node 12A maintains a mapping that associates each of the image instances with the assigned network address and tracks whether the image instance is booted on one of application nodes 14. Control

Netflix, Inc. - Ex. 1002, Page 000562
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node 12A may maintain the mapping within a relational database, such as SQL Server from Microsoft Corporation or PostgreSQL.

[0037] Pre-creation of image instances for future allocation may provide certain advantages. For example, pre-creation of image instances may reduce the amount of time required for deploying the image instances on unallocated nodes 16. Further, as explained in detail below, control node 12A allows the unassigned image instances within image repository 26 to be maintained and updated as if currently deployed. In this fashion, the image instances are in a current state and ready for assignment to an application node for deployment.

[0038] Control node 12A may also provide mechanisms for efficiently updating the image instances. For example, control node may provide efficient mechanisms for applying software patches to the image instances and for rollback of the image instance if desired. Control node 12A may provide a scheduled update process that periodically checks for and applies software updates, e.g., nightly or weekly. Alternatively, administrator 20 may manually invoke an image update process upon receiving a new patch for a particular software package or operating system.

[0039] When performing an update, control node 12A updates non-assigned image instances store within image repository 26 as well image instances deployed to application nodes 14. With respect to the deployed image instances, control node 12A may initiate and perform the image update process while maintaining the execution of as many of application nodes 14 as possible so that the disruption to distributed computing system 10 is reduced. For example, in some situations, control node 12A applies a rolling update across application nodes 14 so that only a single application node is offline at any time. Thus, control node 12A may update the software images of application nodes 14 in a relatively non-intrusive manner.

[0040] As described in further detail below, when applying a software update to a group of image instances, control node 12A first creates a clone of the golden image for the group of image instances to be updated. The clone of the golden image may be viewed as a bit for bit copy of the golden image. Control node 12A then temporarily boots the clone of the golden image on a computing node from the pool of unallocated computing nodes 16. Administrator 20 then applies the software update (e.g., patch) to the clone of the golden image, thus creating a patched golden image. Control node 12A computes a first set of differences (S_1) that specifies the differences between the original golden image and the patched golden image.

Netflix, Inc. - Ex. 1002, Page 000563
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0041] Control node 12A then proceeds to update each of the image instances of the group. For deployed image instances, control node 12A shuts down the application node 14 on which the image instance is deployed down and clones the image instance. Control node 12A then computes a second set of differences (S_2) that specifies the differences between the original golden image and the cloned image instance. Control node 12A propagates the computed set of differences (S_1) into the first cloned image instance, and reconciles those differences with the set of differences (S_2) between the original golden image and the cloned image instance. If there are no conflicts, control node 12A boots another application node 14 with the patched image instance. The image instances of the other application nodes 14 are sequentially cloned and updated in a similar fashion.

[0042] If any file conflicts are detected during the update of any of the image instances of the group, administrator 20 may select one of three actions. In particular, administrator 20 may direct control node 12A to: (1) use the file from the original image instance, (2) use the file from the updated golden image or (3) fail the update and rollback the updated image instance to the original image instance. For a rollback, control node 12A reboots application nodes 14 with the original image instances, i.e., the image instances prior to being cloned and patched.

[0043] FIG. 2 is an exemplary image management data structure 40 maintained by control node 12A in association with image repository 26. In this example, image management data structure 40 is represented as a table in which each row represents a unique software image. The software images represented in data structure 40 may be golden images (represented by the instance identification "GOLDEN") or specific image instances.

[0044] For each software image, image management data structure 40 includes an image type 42 identifying the functional type of the software image. In the example illustrated in FIG. 2, image management data structure 40 includes two different types of software images: web server images and application server images. In addition, image management data structure 40 includes an instance identification 44 that identifies an instance number and a version of golden image from which the instance was created. For example, the web server image with an instance ID of 2.1 indicates that the instance is the second image instance created from the version 1.0 web server golden image.

[0045] Image management data structure 40 also includes an IP or other network address 46 associated with each of the images. As described above, the IP address associated with the image instance may be assigned from a block of IP addresses reserved for distributed

Netflix, Inc. - Ex. 1002, Page 000564
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

computing system 10. As golden images are never deployed, IP addresses are not assigned. Other information may also be assigned, such as a host name.

[0046] Furthermore, image instance data structure 40 includes a status 48 of the image and the version of the image. The status of the image may either be "active" or "inactive." An image has a status of "active" if it is currently deployed on one of application nodes 14 of distributed computing system 10. Images that are not currently running on application nodes 14 are classified as "inactive." Golden images are always inactive, except when used during the update process. In the example illustrated in FIG. 2, of the four image instances created from the version 2.0 web server golden image only two are actively deployed on application nodes 14. Thus, as described in detail above, distributed computing system pre-creates image instances for future use on application nodes 14.

[0047] Control node 12A maintains entries for software images that have been updated. In the example illustrated in FIG. 2, control node 12A has updated web server image version 1.0 to version 2.0. Thus, in the example of FIG. 2, web server images having instance IDs of 1.2-4.2 are updated versions of web server images 1.1-4.1. Control node 12A maintains the image instances associated with version 1.0 in case a conflict is detected and the control node needs to rollback one or more of application nodes 14 to the original version of the image instances. As illustrated in image management data structure 40, control node 12A may assign the same IP address to an image instance and any corresponding updated instances. In some embodiments, however, the updated image instance may be assigned an IP address that is distinct from the parent image instance.

[0048] The data of FIG. 2 is illustrated for exemplary purposes, and may be subject to variation. For example, image management data structure 40 may further specify a type of operating system the image should be run on, a version of the operating system the image should be run on, an identifier (e.g., MAC address) for the application node to which an "active" image instance is assigned, and other data. Although image management data structure 40 is illustrated in FIG. 2 as a single image management data structure, image management data structure 40 may be divided into two or more data structures. For example, control node 12A may maintain a first image management data structure that maintains information associated with golden images and a second image management data structure that maintains information associated with image instances. Furthermore, image management data structure 40 may take forms other than a table,

Netflix, Inc. - Ex. 1002, Page 000565
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

such as one or more databases, link lists, radix trees, databases, flat files, or any other data structures.

[0049] FIG. 3 is a flow diagram illustrating exemplary operation of control node 12A when managing images during configuration of a distributed processing environment. Initially, control node 12A receives input defining an architecture of a distributed computing environment (50). Particularly, system administrator 20 accesses control node 12A and defines the architecture of distributed computing system 10. The system administrator may, for example, define the maximum number of application nodes 14 of system 10, an initial number of application nodes 14 to be deployed, the hardware requirements of each node, the type of each node (e.g., an application server node), and the like.

[0050] Control node 12A then generates and stores a golden image for each type of software image (i.e., computing function) to be deployed on application nodes 14 (52). As described above, the system administrator may create a golden image by temporarily installing and booting the drivers, operating system, middleware and applications on a computing node and then making a bit copy of the contents of the root partition, e.g., all files, the operating system, and applications.

[0051] Next, control node 12A creates image instances from the golden images and stores the image instances (54). Particularly, for each golden image, control node 12A creates the maximum number of image instances that may be deployed within distributed computing system 10 for the corresponding computing function. In this manner, control node 12A pre-creates image instances for subsequent allocation. Control node 12A may create the image instances by performing a bit-wise copy from the respective golden image. Control node 12A also assigns a network address to each of the image instances from a block of network addresses allocated to distributed computing system 10 (56). In one embodiment, each image instance consists of two parts: (1) a copy of those portions of the corresponding golden image that will be writable and, therefore, must be unique to each image instance, and (2) a read-only reference directly to the remaining portions of the golden image, thus allowing those portions to be shared by multiple image instances. This may allow for faster deployment.

[0052] Control node 12A selects the image instances that will be currently deployed, assigns the image instances to respective application nodes 14 (58), and deploys the image instances on their respectively nodes (60). In general, the term "deploy" refers to the process of configuring distributed computing system 10 to execute a particular

Netflix, Inc. - Ex. 1002, Page 000566
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

software image instance on a particular application node 14. When deploying an image instance, for example, control node 12A may configure distributed computing system 10 so that the image instance is utilized for remotely booting the application node 14 to which the image instance is assigned (i.e., a network boot). This process may involve configuring a "pre-boot execution environment" (PXE) including a kernel, RAM disk, network interfaces or other information. Deploying an image instance may also include: (1) configuring a dynamic host communication protocol (DHCP) environment, including a root file system location, a hostname and an IP address, (2) configuring a dynamic virtualization service (DVS) environment, including a hostname and IP address, (3) configuring a domain name server (DNS) environment, (4) configuring a Network File System (NFS) environment, (5) configuring a NVDD environment and (6) configuring a secure shell (SSH) environment.

[0053] The assigned and unassigned image instances remain stored in image repository 26, the unassigned image instances reserved possibly used for later allocation to unallocated nodes 16. For example, control node 12A may allocate image instances and deploy unallocated nodes 16 when distributed computing system 10 experiences high processing loads. Control node 12A maintains data describing the allocated and unallocated image instances (62). As described above in referenced to FIG. 3, the data describes each image instance and may map the allocated image instances to corresponding application node.

[0054] FIG. 4 is a flow diagram illustrating exemplary operation of control node 12A when utilizing one of unallocated nodes 16. Initially, administrator 20 or control node 12A identifies the need to increase computing bandwidth and add a new application node 14 to distributed computing system 10 (70). For example, control node 12A may automatically detect a high load on the currently deployed application nodes 14 and the need to bring another application node online. In another embodiment, system administrator 20 or other network monitor may identify the need to bring a new node online and input that need into control node 12A. As another example, control node 12A or administrator 20 may detect a failed application node and deploy a new application node with identical computing function.

[0055] Next, control node 12A or system administrator 20 identifies the particular type of node to allocate (72). As described above, for example, if application nodes executing web server software images are experiencing heaving loading, control node 12A may determine that a new web server application node needs to be brought online. Based on

Netflix, Inc. - Ex. 1002, Page 000567
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the determination, control node 12A selects an inactive (i.e., unassigned) image instance from image repository 26 to associate with the new application node (74). Continuing with the web server example, control node 12A selects an inactive image instance created from the web server golden image. For example, control node 12A may select an inactive software image created from a most current version of the web server golden image or the most recently updated image instance. With respect to FIG. 2, for example, control node 12A may select the web server image with instance identification number 2.3 or 2.4.

[0056] Control node 12A retrieves the identified image instance from image repository 26 (76), and boots one of unallocated nodes 16 with the retrieved image instance (78). Thus, the newly booted node is removed from the pool of unallocated nodes. As described above, the newly booted application node inherits the network address of the selected image instance for as long as the application node runs that particular image instance or an updated version of that particular image instance. Control node 12A updates the image repository 26 to reflect the assignment (79). For example, control node 12A may update image management data structure 40 to change the status of the image instance from "inactive" to "active," associates the image instance with a particular node, or the like.

[0057] FIG. 5 is a flow diagram illustrating exemplary operation of control node 12A when applying a software update. In the illustrated example, control node 12A performs image updating on a node-by-node basis while the other nodes continue operation so that the disruption to distributed computing system 10 is reduced.

[0058] Initially, control node 12A or administrator 20 detects an event identifying a need to update one or more image instances (80). Control node 12A may, for example, detect expiration of a timer in the case that image updates are performed periodically, detect receipt of a new software patch, or receives input from system administrator 20 in the case of a manually initiated update.

[0059] In any case, control node 12A clones (i.e.., makes a bit copy of) the golden image associated with the image instances to be updated (82). Control node 12A deploys and boots the clone of the golden image on one of unallocated nodes 16 (84). Next, control node 12A or system administrator 20 applies the software patch or software update to the installed clone of the golden image, thus creating a patched golden image (86). After the patch is complete, control node 12A shuts down the node used for the patch.

[0060] Control node 12A computes a set that identifies the differences between the original golden image and the patched golden image, referred to herein as the golden

Netflix, Inc. - Ex. 1002, Page 000568
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

image delta S_1 (88). In particular, control node 12A performs a file-by-file bit comparison (or optionally a checksum comparison) between the original golden image and the patched golden image to determine which files have been changed. Control node 12A may consider other information, such as a change in permissions or ownership associated with the files. Thus, the golden image delta S_1 includes a list of the files that have been changed in some manner by the patch. In addition, control node 12A may identify what specific changes have been made to the files.

[0061] Control node 12A then proceeds to update each of the image instances of the group being updated, i.e., each of the image instances created from the original golden image. For those image instances assigned to an active application node, control node 12A shuts down the application nodes (90) and clones the installed image instances (92). For those image instances that are not presently installed and assigned to active application nodes, control node 12A clones the image instances by performing a bit copy of the image instances.

[0062] Next, control node 12A computes a set that identifies the differences between the original golden image and the deployed image instance used to generate the clone. This set of differences is referred to herein as an image instance delta S_2 (94). As described briefly above, control node 12A may perform a file-by-file bit comparison (or optionally a checksum comparison) to identify the differences. The image instance delta includes a list of the files that have been changed in some manner during execution of the image instance on the particular computing node and, in some embodiments, the changes that have been made to the files.

[0063] Control node 12A propagates the changes identified by the golden image delta S_1 into the clone of the image instance and reconciles the changes with the delta S_2 (96). Specifically, control node 12A analyzes the intersection of the golden image delta S_1 with the image instance delta S_2. By doing so, control node 12A determines whether there are any overlapping changed files. For example, if the same system file was changed by installing and booting the image instance as well as by applying patch, control node 12A cannot reconcile the source of the change, thus resulting in a conflict.

[0064] If no conflicts are detected during the reconciliation, control node 12A stores the updated image instance and, if the instance is active, boots one of application nodes 14 of distributed computing system 10 with the updated image instance (98, 100). Control node 12A may, for example, boot one of unallocated nodes 16. In this case, the node on which the original image instance was executing may be placed in the pool of unallocated

Netflix, Inc. - Ex. 1002, Page 000569
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

nodes. Alternatively, control node 12A installs and boots the updated image instance on the original application node on which the original image instance was executing, i.e., the application node shutdown by control node 12A.

[0065] Control node 12A determines whether there are any other image instances created from the same golden image remain to be updated (102). If so, control node 12A selects one of the image instances and repeats the update process. Control node 12A repeats this process until all of the image instances associated with the golden image being patched have been updated.

[0066] In the event control node 12A identifies a conflict during reconciliation (i.e., the intersection of delta S_2 and delta S_2 is not an empty set) administrator 20 may select one of three actions. In particular, administrator 20 may direct control node 12A to: (1) use the file from the original image instance, (2) use the file from the updated golden image or (3) fail the update and rollback the updated image instance to the original image instance. For rollback, control node 12A boots a computing node with the original image instance that was running prior to the upgrade (106). This process is referred to as rollback in the sense that the computing node is running the version of the image instance without the updates from the patch. As described above, the computing node on which the rollback occurs may be an unallocated node 16 or the same node that was shutdown at the beginning of the upgrade/patch process. In addition, control node 12A may perform rollbacks on all of the previous image instances in the group of images to which the update or patch has been applied (108).

[0067] FIG. 6 is a screen illustration of an exemplary user interface 110 presented by a control node, such as control node 12A, for viewing and managing software images within distributed computing system 10. User interface 110 presents to a system administrator or other user a list of software images maintained by control node 12A. The list of images may include image instances, golden images or a combination thereof. Moreover, the list may include only the most current version of software images or all versions of the software images.

[0068] As illustrated in FIG. 6, user interface 110 presents the status of each software image (i.e., either active or inactive), the version of the image, the operating system and version on which the image should be run and a brief description of the software image.

[0069] The system administrator or other user may select one or more software image by selecting on input 112. Once selected, the user may perform an action on the selected images by selecting the action within menu 114. Actions that the system administrator

Netflix, Inc. - Ex. 1002, Page 000570
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

may perform on the selected images include deleting the images, updating the images, and assigning the images to application nodes. In some embodiments, user interface 110 may display other details about the software images, such as the node to which the active images are assigned, the IP addresses associated with the image instances and the like.

[0070] FIG. 7 is a screen illustration of an exemplary user interface 120 with which a system administrator interacts to perform operations on computing nodes, such as application nodes 14. In this example, user interface 120 presents computing nodes that belong to a particular group (referred to as a "tier" in FIG. 7). An exemplary tier may comprise, for example, all computing nodes that operate as web servers.

[0071] After the system administrator selects a tier, user interface 120 present the user with a list of computing nodes that belong to the selected tier. The system administrator may then select one or more of the computing nodes and perform a management operation on the selected computing nodes by interacting with menu 122. For example, the system administrator may initiate an image update for each of the selected nodes. When updating images, the system administrator will typically select all the computing nodes running image instances that were created from a common golden image. Other examples of actions include deploying the node, deallocating the node and moving the node to the free pool.

[0072] FIG. 8 is a screen illustration of an exemplary user interface 130 presented by a control node 12A once the user administrator has manually initiated an image update. In particular, user interface 130 to select an image version and a first node on which to install the image upgrade. The system administrator may, for example, select an image from a dropdown list 132 that is populated with the images illustrated in user interface 130 of FIG. 6. The system administrator selects the appropriate image and then clicks on upgrade button 134.

[0073] Upon receiving the information selected by the system administrator, control node 12A updates a first one of the identified computing nodes as described in detail above. In the event the updated software image works correctly, the remaining nodes are updated serially as described in detail above.

[0074] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.

Netflix, Inc. - Ex. 1002, Page 000571
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS:

1. A method comprising:

storing a master software image for a type of computing function to be deployed within a distributed computing system;

generating K software image instances from the master software image, wherein the distributed computing system is configured to deploy a maximum of K of the software image instances for the type of computing function; and

deploying the software image instances on M application nodes within the distributed computing system, where $M \leq K$.

2. The method of claim 1, further comprising storing the image instances within a software image repository.

3. The method of claim 2, further comprising:

designating the image instances deployed on the application nodes as active; and

designating the image instances that have not been deployed as inactive.

4. The method of claim 1, further comprising:

pre-assigning a network address to each of the image instances; and

when deploying the image instances, assigning the network addresses of the image instances to the application nodes.

5. The method of claim 4, wherein assigning a network address to each of the image instances comprises assigning an Internet Protocol (IP) address and a host name to each of the image instances.

6. The method of claim 1, further comprising generating the master software image by temporarily installing an operating system and one or more software applications on a computing node and copying the installed software.

Netflix, Inc. - Ex. 1002, Page 000572
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

7. The method of claim 1, further comprising maintaining a data structure that stores data defining each of the image instances including:

an instance identification that identifies an instance number and a version of the master software image from which the image instance was created,

a network address associated with the image instance, and

a status indicating whether the image instance is currently deployed on one of the application nodes.

8. The method of claim 1, further comprising:

detecting a need to deploy an additional one of the image instances;

selecting one of the image instances that has not been deployed; and

deploying the selected image instance on an additional application node.

9. The method of claim 8, wherein detecting a need comprises automatically detecting a high processing load on the application nodes.

10. The method of claim 8, wherein detecting a need comprises receiving input from a system administrator requesting deployment of the additional image instances.

11. A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network; and

an image repository storing a plurality of K pre-generated software image instances, wherein K represents a maximum number of the software image instances to be deployed within the distributed computing system for a similar computing function.

12. The distributed computing system of claim 11, further comprising a control node coupled to a communications network, wherein the control node deploys a subset of the K software image instances on M of the application nodes within the distributed computing network, where $M \leq K$.

13. The distributed computing system of claim 12, wherein the control node pre-assigns a network address to each of the image instances stored within the image repository, and assigns the network addresses of the image instances to the application nodes when deploying the image instances.

Netflix, Inc. - Ex. 1002, Page 000573
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

14. The distributed computing system of claim 12,

wherein the image repository stores a master software image for a type of computing function to be deployed within a distributed computing system, and

wherein the control nodes generates the software image instances from the master software image.

15. The distributed computing system of claim 14, wherein each of the image instances includes: (1) a copy of a first portion of the master image software that will be writable once deployed, and (2) a read-only reference to a remaining portion of the master image.

16. The distributed computing system of claim 12, further comprising:

wherein the control node presents a user interface to receiving a request to deploy an additional image instance, and

wherein the control node selects one of the image instances that has not been deployed and deploys the selected image instance on an additional application node.

17. The distributed computing system of claim 11, further comprising a database storing data that designates the image instances associated with the deployed application nodes as active, and designates the image instances that have not been associated with the deployed application as inactive.

18. The distributed computing system of claim 11, wherein the data includes:

an instance identification that identifies an instance number and a version of the master software image from which the image instance was created,

a network address associated with the image instance, and

a status indicating whether the image instance is currently deployed.

Netflix, Inc. - Ex. 1002, Page 000574
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

19. A computer-readable medium comprising instructions that cause a programmable process to:

present a first user interface to receive input identifying a master software image for a type of computing function to be deployed within a distributed computing system;

present a second user interface to receive input identifying a maximum number of software image instances that can be deployed within the distributed computing system for the type of computing function; and

generate a software image instance from the master software image for each of the image instances that can be deployed.

20. The computer-readable medium of claim 19, further comprising instructions to cause the processor to:

deploy a respective one of the software image instances on M of the application nodes within the distributed computing system, where M is less than the maximum number of software instances that can be deployed for the computing function.

Netflix, Inc. - Ex. 1002, Page 000575
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

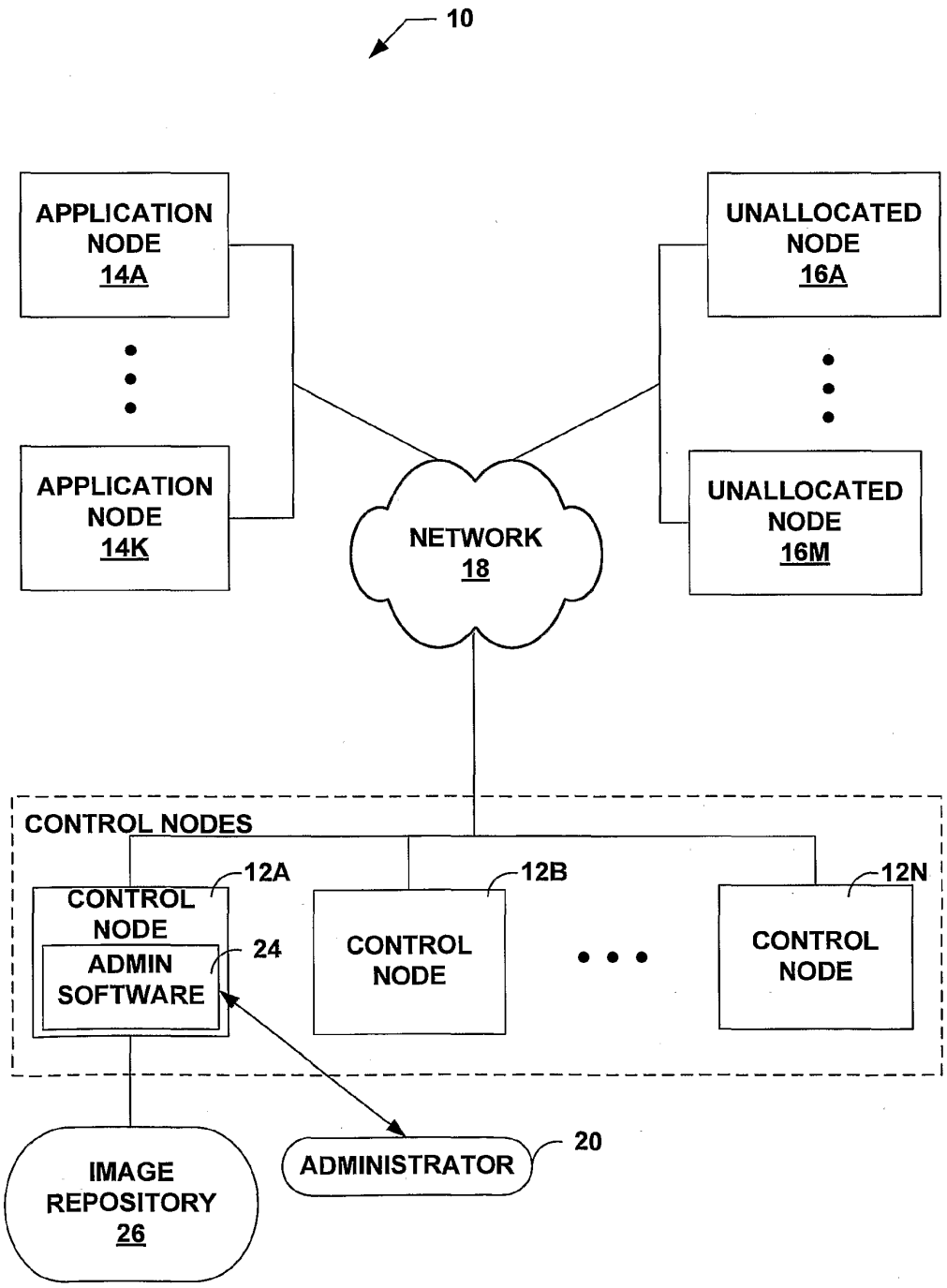

10
APPLICATION NODE 14A
APPLICATION NODE 14K
NETWORK 18
UNALLOCATED NODE 16A
UNALLOCATED NODE 16M
CONTROL NODES
12A
12B
12N
CONTROL NODE
ADMIN SOFTWARE
24
CONTROL NODE
CONTROL NODE
IMAGE REPOSITORY 26
ADMINISTRATOR
20

FIG. 1

40

42 IMAGE TYPE	44 INSTANCE ID	46 IP ADDRESS	48 STATUS	VERSION
WEB SERVER IMAGE	GOLDEN	---	INACTIVE	1.0
WEB SERVER IMAGE	1.1	12.1.56.2	INACTIVE	1.0
WEB SERVER IMAGE	2.1	12.1.56.3	INACTIVE	1.0
WEB SERVER IMAGE	3.1	12.1.56.4	INACTIVE	1.0
WEB SERVER IMAGE	4.1	12.1.56.5	INACTIVE	1.0
WEB SERVER IMAGE	GOLDEN	---	INACTIVE	2.0
WEB SERVER IMAGE	1.2	12.1.56.2	ACTIVE	2.0
WEB SERVER IMAGE	2.2	12.1.56.3	ACTIVE	2.0
WEB SERVER IMAGE	3.2	12.1.56.4	INACTIVE	2.0
WEB SERVER IMAGE	4.2	12.1.56.5	INACTIVE	2.0
APPLICATION SERVER IMAGE	GOLDEN	---	INACTIVE	1.0
APPLICATION SERVER IMAGE	1.1	12.1.56.19	ACTIVE	1.0
APPLICATION SERVER IMAGE	2.1	12.1.56.19	INACTIVE	1.0

FIG. 2

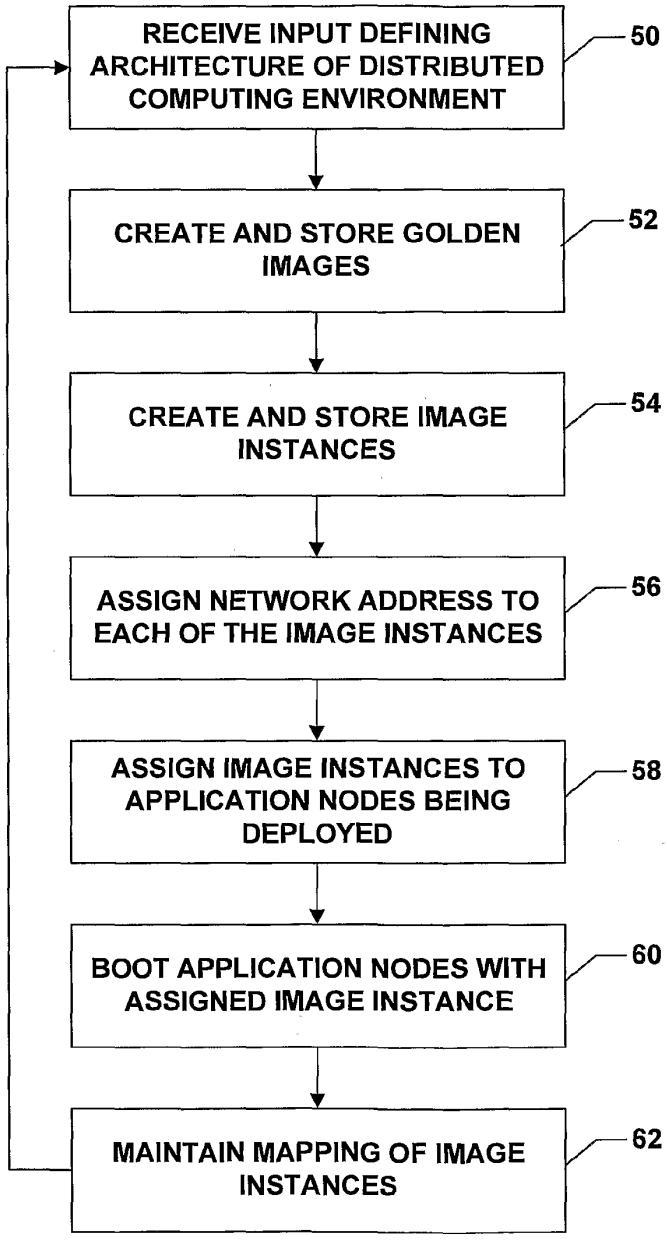

RECEIVE INPUT DEFINING ARCHITECTURE OF DISTRIBUTED COMPUTING ENVIRONMENT
50
CREATE AND STORE GOLDEN IMAGES
52
CREATE AND STORE IMAGE INSTANCES
54
ASSIGN NETWORK ADDRESS TO EACH OF THE IMAGE INSTANCES
56
ASSIGN IMAGE INSTANCES TO APPLICATION NODES BEING DEPLOYED
58
BOOT APPLICATION NODES WITH ASSIGNED IMAGE INSTANCE
60
MAINTAIN MAPPING OF IMAGE INSTANCES
62

FIG. 3

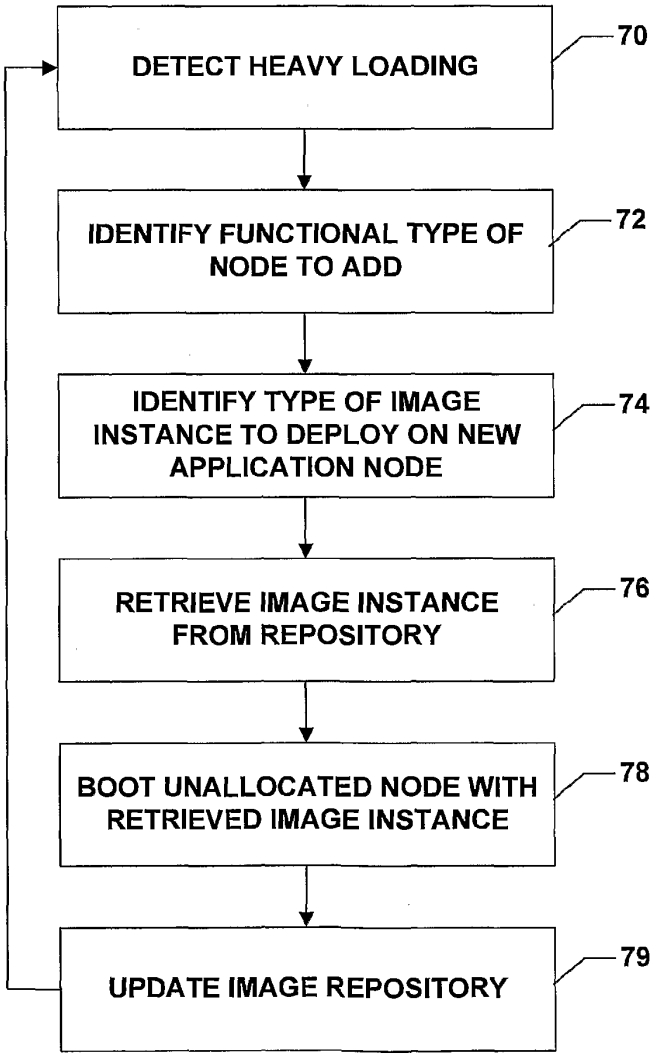

DETECT HEAVY LOADING
70
IDENTIFY FUNCTIONAL TYPE OF NODE TO ADD
72
IDENTIFY TYPE OF IMAGE INSTANCE TO DEPLOY ON NEW APPLICATION NODE
74
RETRIEVE IMAGE INSTANCE FROM REPOSITORY
76
BOOT UNALLOCATED NODE WITH RETRIEVED IMAGE INSTANCE
78
UPDATE IMAGE REPOSITORY
79

FIG. 4

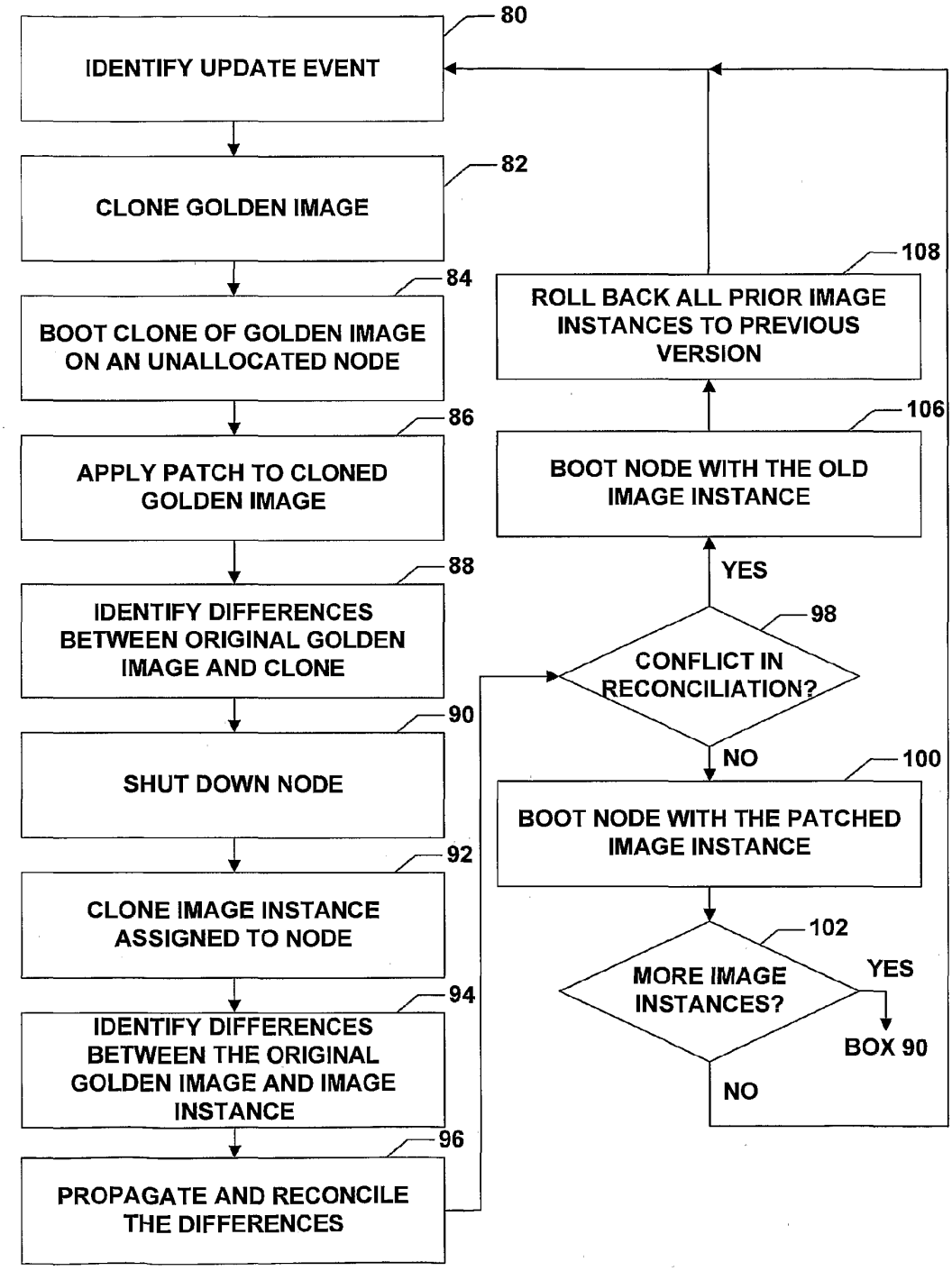

IDENTIFY UPDATE EVENT
80
CLONE GOLDEN IMAGE
82
BOOT CLONE OF GOLDEN IMAGE ON AN UNALLOCATED NODE
84
APPLY PATCH TO CLONED GOLDEN IMAGE
86
IDENTIFY DIFFERENCES BETWEEN ORIGINAL GOLDEN IMAGE AND CLONE
88
SHUT DOWN NODE
90
CLONE IMAGE INSTANCE ASSIGNED TO NODE
92
IDENTIFY DIFFERENCES BETWEEN THE ORIGINAL GOLDEN IMAGE AND IMAGE INSTANCE
94
PROPAGATE AND RECONCILE THE DIFFERENCES
96
CONFLICT IN RECONCILIATION?
98
YES
NO
BOOT NODE WITH THE OLD IMAGE INSTANCE
106
ROLL BACK ALL PRIOR IMAGE INSTANCES TO PREVIOUS VERSION
108
BOOT NODE WITH THE PATCHED IMAGE INSTANCE
100
MORE IMAGE INSTANCES?
102
YES
BOX 90
NO

FIG. 5

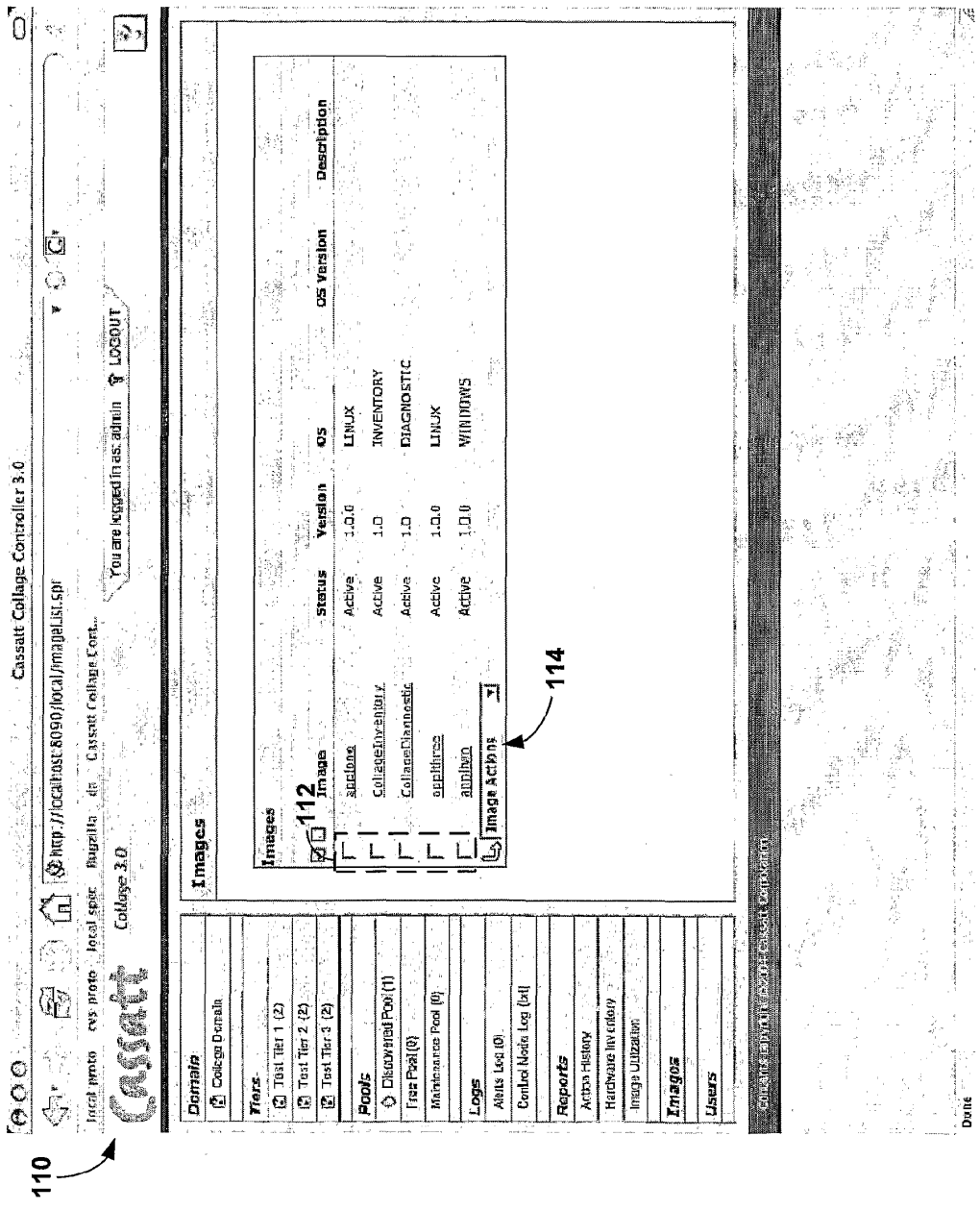

110
Cassatt Collage Controller 3.0
http://localhost:8090/local/imageList.spf
You are logged in as: admin
LOGOUT
Cassatt
Collage 3.0
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log [txt]
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Images
Images
112
Image
Status
Version
OS
OS Version
Description
appStore
Active
1.0.0
LINUX
CollageInventory
Active
1.0
INVENTORY
CollageDiagnostic
Active
1.0
DIAGNOSTIC
appThree
Active
1.0.0
LINUX
appTwo
Active
1.0.0
WINDOWS
Image Actions
114
Done

FIG. 6

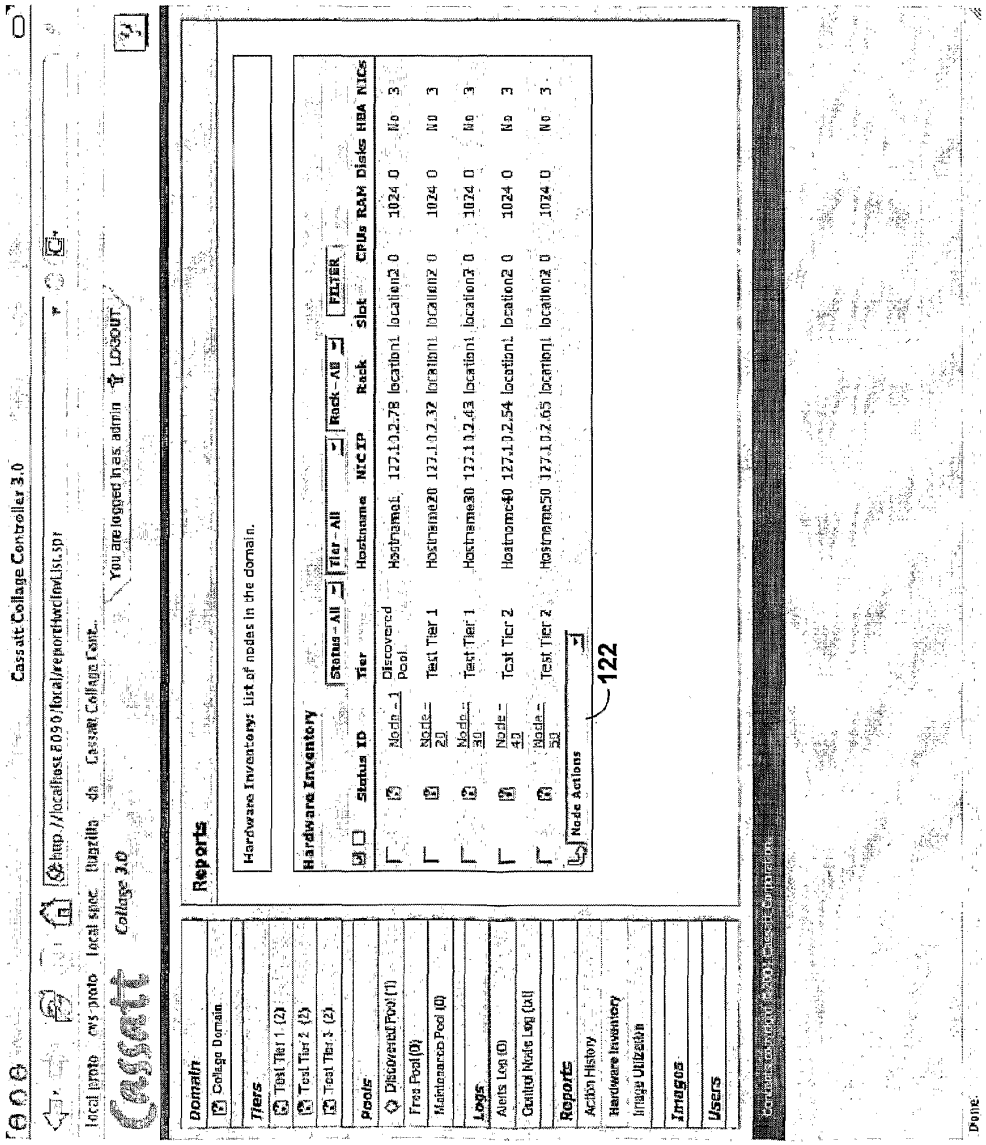

120
Cassatt Collage Controller 3.0
http://localhost:8090/local/reporthwcInvList.spr
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Cassatt Node Log (all
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Reports
Hardware Inventory: List of nodes in the domain.
Hardware Inventory
Status - All
Tier - All
Rack - All
FILTER
Status ID Tier Hostname NIC IP Rack Slot CPUs RAM Disks HBA NICs
Node - 1 Discovered Pool Hostname1 127.1.0.2.78 location1 location2 0 1024 0 No 3
Node - 20 Test Tier 1 Hostname20 127.1.0.2.32 location1 location2 0 1024 0 No 3
Node - 30 Test Tier 1 Hostname30 127.1.0.2.43 location1 location3 0 1024 0 No 3
Node - 40 Test Tier 2 Hostname40 127.1.0.2.54 location1 location2 0 1024 0 No 3
Node - 50 Test Tier 2 Hostname50 127.1.0.2.65 location1 location2 0 1024 0 No 3
Node Actions
122
Done

FIG. 7

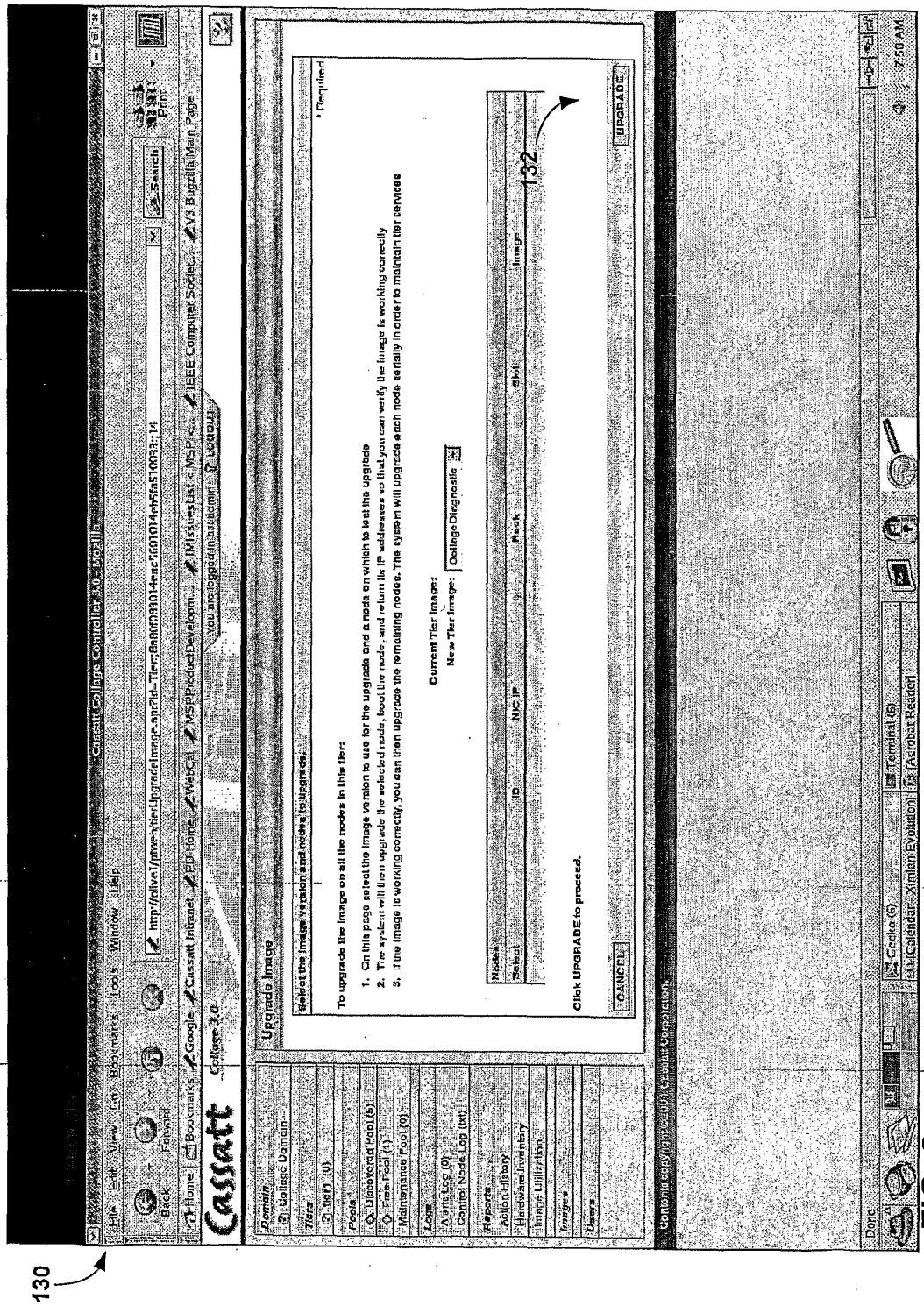

130
Cassatt Collage Controller - Mozilla
Upgrade Image
Select the image version and nodes to upgrade.
* Required
To upgrade the image on all the nodes in this tier:
1. On this page select the image version to use for the upgrade and a node on which to test the upgrade
2. The system will then upgrade the selected node, boot the node, and return its IP address so that you can verify the image is working correctly
3. If the image is working correctly, you can then upgrade the remaining nodes. The system will upgrade each node serially in order to maintain tier services
Current Tier Image:
New Tier Image: CollageDiagnostic
Nodes
Select ID NIC IP Rack Slot Image
132
Click UPGRADE to proceed.
CANCEL
UPGRADE
Console copyright 2005, Cassatt Corporation.

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003100

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/445

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/187104 A1 (SARDESAI SHANTANU ET AL) 23 September 2004 (2004-09-23) paragraph [0006] paragraph [0019] - paragraph [0041] paragraph [0062] - paragraph [0075]	1-20
X	US 2002/156877 A1 (LU JAMES C ET AL) 24 October 2002 (2002-10-24) paragraph [0017] - paragraph [0036] paragraph [0044] - paragraph [0091]	1-20
A	"Preinstalling Microsoft Windows XP by Using the OEM Preinstallation Kit, Part I" 4 April 2003 (2003-04-04), , XP002301441 the whole document	1,2,11, 12,19,20

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
8 June 2006	03/07/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kusnierczak, P

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003100

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	R. MARK KOAN, KELLY CAYE, STEVEN K. BRAWN: "It takes a village to build an image" PROCEEDINGS OF THE 31ST ANNUAL ACM SIGUCCS CONFERENCE ON USER SERVICES, 2003, pages 200-207, XP002384269 New York, NY, USA ISBN: 1-58113-665-X the whole document	1,2,11, 12,19,20

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/003100

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004187104 A1	23-09-2004	CN 1570868 A	26-01-2005
		EP 1465064 A2	06-10-2004
		JP 2004280839 A	07-10-2004
US 2002156877 A1	24-10-2002	NONE	

Form PCT/ISA/210 (patent family annex) (April 2005)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 October 2003 (16.10.2003)

PCT

(10) International Publication Number
WO 03/085526 A1

(51) International Patent Classification[7]: G06F 12/00, 12/02

(21) International Application Number: PCT/US03/10197

(22) International Filing Date: 2 April 2003 (02.04.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/370,100 3 April 2002 (03.04.2002) US

(71) Applicant *(for all designated States except US)*: **POWERQUEST CORPORATION** [US/US]; Building K, 1359 N. Research Way, Orem, UT 84097 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **KLEINSCHNITZ, Donald, Jr.** [US/US]; 3142 East Fawnwood Cove, Sandy, UT 84092 (US). **ARBON, Val, A.** [US/US]; 1276 East 620North, Orem, UT 84097 (US). **WHATCOTT, Roland, D.** [US/US]; 39 West 800 South, Salem, UT 84653 (US).

(74) Agent: **OGILVIE, John, W., L.**; Computer Law++, 1211 East Yale Avenue, Salt Lake City, UT 84105 (US).

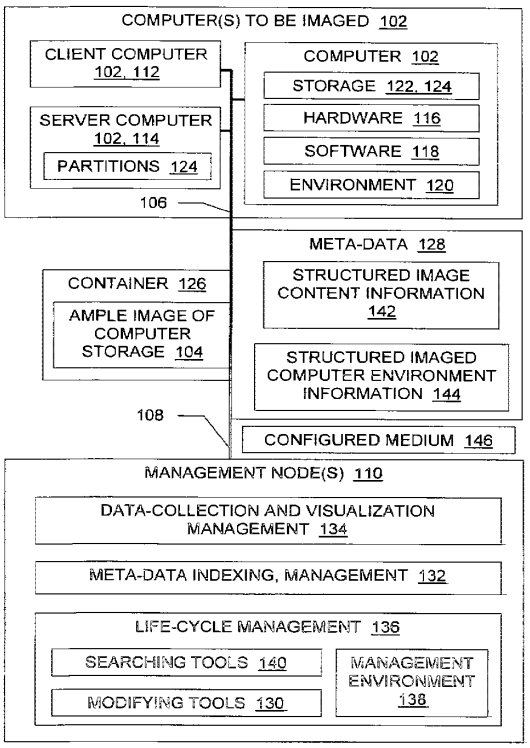
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: USING DISASSOCIATED IMAGES FOR COMPUTER AND STORAGE RESOURCE MANAGEMENT



(57) Abstract: The present invention provides tools and techniques for facilitating (314) management of storage (122), software (118), and other resources of a computer (102) at a distinct management computer (110) using a disassociated ample image (104) of a managed computer's storage. Ample images may be searched (310) to identify (414) infected files or illegal files, to extract (410) disk usage information, or for other reasons. Ample images may be modified (312) and then deployed (316) back to the original imaged computer and/or to other computers outside the management node. Modifications may change (502, 504, 506) application software, change (508, 510, 512) hardware drivers to match hardware changes on the target computer(s), manipulate (520) partitions, and/or perform other steps to optimize storage, software, or other resources.

WO 03/085526 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

USING DISASSOCIATED IMAGES FOR COMPUTER AND STORAGE RESOURCE MANAGEMENT

FIELD OF THE INVENTION

The present invention relates generally to computer storage and other computer resources. The invention provides tools and techniques used on one computer to browse or modify a copy of stored data from another computer, in order to help manage resources of that other computer. For classification purposes, it may also be helpful to note that computer operations discussed herein comprise taking inventory of computer state and components, communicating across a network, copying, indexing, loading, searching, and writing data, as well as data operations such as partition manipulation.

TECHNICAL BACKGROUND OF THE INVENTION

A wide variety of tools and techniques are known for making backup copies of data stored on a computer's disk(s), and for restoring that data to those disk(s) or to their replacement(s) on the computer. For instance, data may be copied from a permanently installed local hard disk onto a removable medium, such as a floppy, an Iomega ZIP brand drive, or a USB portable disk drive that is temporarily attached to the computer. Data may also be copied over a network or other communications link, e.g., to archive client data on a server, or to mirror data from one server to another server. Data may be copied a few files at a time, or it may be copied in larger amounts in a file-by-file image or a sector-by-sector image. Data may be copied in a single snapshot or other archive that matches the computer's storage contents at a given point in time, in which case the snapshot may be subsequently supplemented by incremental images, to update the snapshot so that it matches the state of the computer's storage at later times. Data in a remote copy may also be continuously updated by a mirroring process, so a secondary computer's storage is updated each time a sector of data is written to a corresponding primary computer's storage.

Often data is backed up simply as a precaution, so it can be restored if a hard drive fails or a virus corrupts it, for instance. If no disaster strikes, the backup copy is often not used. On occasion, data that has been backed up is also made available as a read-only copy. This can be done, for instance, using the V2i Protector product from PowerQuest Corporation, which provides a driver that lets users mount a sector-by-sector image file as a

Netflix, Inc. - Ex. 1002, Page 000589
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

drive on a computer running a file system under a Microsoft Windows brand operating system. (V2I and V2I PROTECTOR are marks of PowerQuest Corporation). But in general, the data produced on one computer and then stored elsewhere as a backup is often simply stored and then left alone. The backup is rarely read, except to be restored to the computer on which it was originally produced. It is rarely written, except to be updated to match a more current active state of the computer that first produced it.

By contrast, the data that is produced on a computer and then stored on that same computer is often used there in a wide variety of ways. Data is produced on a computer, and also manipulated there, through many familiar applications such as word processors, spreadsheets, and web browsers, and by the underlying operating system, file system, and networking software.

In particular, many tools and techniques have been developed for use on a computer to manage the storage and other resources of that managed computer. Examples include tools for installing, patching, upgrading, and/or uninstalling software on the managed computer; tools for optimizing storage by defragmenting files, changing partition cluster size, or performing other operations; tools for manipulating partitions or volume segments by resizing, moving, copying, or merging them; tools for recovering data despite loss or corruption of file system information; and tools for converting one file system type to another, such as converting between a FAT file system and a Microsoft NT brand file system.

PowerQuest Corporation provides many such resource management tools, as discussed at www.powerquest.com. The following United States Patents assigned to PowerQuest Corporation contain additional discussion of these and other tools and techniques for managing computer resources: 5675769, 5706472, 5930831, 5933647, 6088778, 6108697, 6108759, 6117188, 6173291, 6178487, 6178503, 6185575, 6185666, 6253300, 6330653, 6377958, 6453383, 6530077. Some of these patents discuss both modifying data to manage storage on a computer and sending data over a network to the computer, e.g., U.S. Patent No. 6253300 on "Computer partition manipulation during imaging" and U.S. Patent No. 5933647 on "System and method for software distribution and desktop management in a computer network environment". U.S. Patent No. 6330653 discusses manipulation of virtual and live computer storage device partitions, including remote partition manipulation over a network. Other remote control software is also available, to control execution of software on one computer through a command interface on another computer. PowerQuest also owns

Netflix, Inc. - Ex. 1002, Page 000590
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

United States Patent Application no. 09/532223 filed March 22, 2000, which is on "Storing a computer disk image within an imaged partition".

Other examples of tools and techniques used on a computer to manage the storage and other resources of that managed computer include antivirus software; and software for monitoring storage usage, blocking storage of files that exceed maximum allocated space quotas or belong to an illegal file type, determining billing for storage usage, and enforcing standardized storage locations for particular types of data. Such software is sometimes referred to as "storage resource management" or "SRM" software. Agents placed on a client computer gather information about the client computer's storage usage, hardware configuration, operating system configuration, software versions, and application software usage, for example, and send that information over a network to an administrative server. Commands are then issued from the administrative server back to the client agents, in response to administrator commands and/or policies, to be executed by the agents on the client computers.

Software for asset management or inventory control is also known. Such software gathers information about computing assets or resources, such as server and workstation hardware, system configuration, and software, including which applications are installed. Such asset management software may be considered an example of SRM software. It may have capabilities such as the conditional deployment of software upgrades or patches based on comparison of the actual configuration of a target computer and a desired configuration for that target; application usage tracking; and remote control of operations performed on a computer to manage the computer's resources.

In short, and with the benefit of hindsight gained from knowledge of the present invention, although many tools are used to manipulate data while that data is stored on the same computer that produced it, and many tools are used to make copies of data and send them to another computer, relatively little has been done to make use at a network management computer of information provided there in a backup copy of another computer's stored data. As discussed below, the present invention provides ways for disassociated backup information to be organized and put to many uses, instead of merely keeping such backups as a precaution against disaster or as a read-only copy shared by several individuals.

Netflix, Inc. - Ex. 1002, Page 000591
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

BRIEF SUMMARY OF THE INVENTION

The present invention provides tools and techniques which can be used on one computer to browse – or in some cases modify – a copy of stored backup data from another computer, in order to help manage resources of that other computer. Because the backup data copy is disassociated from the computer that produced it, the backup data copy can be used for management purposes without preventing continued use of the original data copy on the original computer. Depending on the management desired, the disassociated data may simply be "mined" for information and/or it may be modified and then installed back on the original computer and/or on other computers.

The present invention is defined by the claims, and to the extent this summary conflicts with the claims, the claims prevail. In some embodiments, the invention provides a method of computer resource management which comprises steps of ample image receiving; meta-data obtaining; indexing; using a management environment; and ample image searching. Some embodiments also provide at least one ample image modifying step and/or a modified image deploying step.

During the image receiving step, a management node which includes at least one management computer receives at least one ample image of storage of an imaged computer. The ample image is thus disassociated from the imaged computer, in that the management computer is distinct from the imaged computer.

During the meta-data obtaining step, the management node obtains meta-data that is structurally associated with the ample image. The meta-data may have been received with the ample image or received in a separate transmission. The meta-data includes structured image content information which describes content of the ample image, and/or structured imaged computer environment information which describes hardware that was installed on the imaged computer when the ample image was created.

During the indexing step, the ample image is indexed into a collection of ample images. One or more meta-data values are used as index keys, such that the ample image can be located by using at least part of the meta-data as a key. Database, hashing, and/or other indexing techniques can be used.

During the step of bringing the ample image into a management environment at the management node, the ample image is mounted for access by management computer system software, replicated and expanded into a virtual environment that simulates at least the

Netflix, Inc. - Ex. 1002, Page 000592
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

storage system portion of the imaged computer, or otherwise made accessible so that a user can at least read the files and their content, and in some embodiments, can also write them.

A resource management facilitating step uses the management environment and performs at least a step of searching of the ample image. It may also perform a step of modifying the ample image. In some cases the modified ample image may then be deployed to the imaged computer and/or to another computer which is distinct from the management node.

These are examples, not a comprehensive summary. The invention provides other methods, as well as products, systems, and configured storage media. Advantages and operation of the invention can be more fully appreciated by the description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

To illustrate the manner in which the advantages and features of the invention are obtained, a more particular description of the invention will be given with reference to the attached drawings. These drawings only illustrate selected aspects of the invention and thus do not fully determine the invention's scope.

Figure 1 is a data flow diagram generally illustrating the present invention.

Figure 2 is a data flow diagram further illustrating some embodiments of the present invention.

Figure 3 is a flowchart generally illustrating methods of the present invention, with emphasis on the perspective of a management computer that manages other computers' storage-related resources.

Figure 4 is a flowchart further illustrating ample image searching steps of the present invention.

Figure 5 is a flowchart further illustrating ample image modifying steps of the present invention.

Figure 6 is a flowchart further illustrating resource management steps of the present invention.

Figure 7 is a flowchart illustrating a resource management method of the present invention from the perspective of a managed computer.

Netflix, Inc. - Ex. 1002, Page 000593
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

DETAILED DESCRIPTION OF EMBODIMENTS

In describing the invention, the meaning of several important terms is clarified, so the claims must be read with careful attention to these clarifications. Specific examples are given to illustrate aspects of the invention, but those of skill in the relevant art(s) will understand that other examples may also fall within the meaning of the terms used, and hence within the scope of one or more claims. Important terms may be defined, either explicitly or implicitly, above, here in the Detailed Description, and/or elsewhere in the application file.

Some Definitions and Examples

An "ample image" is a copy of data stored at a computer, such as a backup copy, but not every backup copy is an ample image. An ample image contains at least file names, file contents, and sector allocation information. It may contain a boot sector. It may contain attributes of individual files and/or attributes of an entire file system, such as access permissions.

Unlike a mere directory tree, each ample image contains file contents in addition to file names. An ample image may be an image of an entire disk of an imaged computer, an image of all disks of the imaged computer, an image of selected partition(s) of the imaged computer, or any other image that contains at least file names, file contents and sector allocation information.

Unlike a file-by-file image, each ample image contains sector allocation information which defines the layout of sectors on disk relative to other sectors. Thus, disk-based optimizations such as file defragmentation are preserved when an ample image is restored into a target disk. For example, an ample image may include a File Allocation Table, a cluster allocation bitmap, a partition table, and/or other sector allocation information. A complete sector-by-sector image of a disk is one example of an ample image. A sector-by-sector image of one or more partitions, but not the entire disk, is also an ample image. A one-to-one mapping between sectors in the ample image and sectors on the disk is allowed but is not required; for instance, ample images may also note the presence of empty sectors in other ways, such as by using bitmaps or data structures that identify empty sector runs (e.g., "13 empty sectors starting at sector 47"). PowerQuest Drive Image® products produce ample images (DRIVE IMAGE is a registered mark of PowerQuest Corporation.)

Netflix, Inc. - Ex. 1002, Page 000594
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Because it contains sector allocation information in addition to mere file contents, an ample image, like a sector-by-sector disk image, may contain information not typically found in file-by-file images. Examples of such information include information stored in hidden partitions or hidden files, information stored on sectors outside any defined partition, and information stored inside file system structures but not in any user-defined file.

The ample image may be stored in a single file, or it may be stored in several files or blocks that are associated with one another in a recognized way through a file naming convention, by contiguity, by being identified as a group in a data structure, or by another recognizable association. File system limits on file size may require use of more than one file to hold a given ample image. Also, baseline ample images and their subsequent incremental ample images will typically be stored in different files.

The ample image may be created by conventional imaging tools such as PowerQuest's Drive Image® products (DRIVE IMAGE is a registered mark of PowerQuest Corporation), or by conventional data mirroring tools, so long as they capture more than mere file content, as discussed above. This is not to say that every ample image can be created using only conventional tools. Some ample images contain one or more meta-data files that are created according to the invention and are placed in the imaged computer's storage before the ample image is created, so that the ample image contains the meta-data file as well as other imaged data. Ample-imaged computers may be clients in a network, servers, laptops, stand-alone workstations, or other computing devices.

An ample image may be "full" in that it contains a copy of every bit and byte of the imaged computer's storage and the storage allocation information specifying how that data resided on disk(s). An ample image may be "partial" in the sense that it does not contain every such bit and byte of data. For instance, one or more partitions might not be imaged and/or the image might be an incremental image that only specifies recent changes to storage.

A "container" is one or more files, blocks, or other grouped data structures which contains at least an ample image and which is in a form permitting its transmission (over a network, or via physically transported media) toward a management computer. A container may contain meta-data in addition to an ample image.

A "disassociated" image is one that is stored offline, from the imaged computer's perspective. That is, the image is stored somewhere other than the computer that was imaged to create the image in question. It is a copy of what was in storage at some past time, and it is

Netflix, Inc. - Ex. 1002, Page 000595
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

not the storage currently being used by the imaged computer. Disassociated images are sent to management computers, where they can be searched, and in at least some embodiments, modified. Mere replacement is not modification; modification is done in place at the management node.

An "embodiment" of the invention may be a particular system, process product, method, and/or signal which configures a computer memory or other digital or analog computer-readable medium, as described herein to one of skill. "Method" and "process" are used interchangeably herein. Except as clearly indicated otherwise by operability constraints or the teachings herein, the description of one type of embodiment informs and illustrates the other types of embodiments. For example, systems of the invention may generally be understood to operate according to the methods described herein.

"Environment data" is data that modifies or describes a particular instance of a user's data or operating environment. For example, configuration files, INI type files, or registry files that describe the operating configuration or environment contain environment data. Environment data is a type of meta-data, which can be extracted from an ample image to provides a means of identification or classification of the data or characteristics of the data in the image file.

Environment meta-data may be captured on a client computer by running one or more programs on the client computer and saving the resulting output into one or more files in the file system before the ample image is created. Particular file name(s) are used to hold the meta-data. When the ample image is created, such meta-data files are imaged, and thus become part of the ample image. After the ample image is transported to a management node, the meta-data files are read from the ample image, and the meta-data is extracted from them. Environment meta-data may also be extracted from conventional system files, such as a Microsoft Windows brand registry file. Meta-data describes the hardware and/or software environment on the client computer. It is used to index the ample image file(s) in a collection of ample images.

A "management computer" is defined relative to the ample image in question. The management computer is any computer operating according to the invention to search and/or modify the ample image, which is not the imaged computer whose resources are being managed offline. One or more management computers working together are referred to herein as a "management node".

A "management environment" is an environment on a management computer which permits an administrator to at least browse (a.k.a. search or mine) an ample image of some

Netflix, Inc. - Ex. 1002, Page 000596
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

other computer's storage. It need not require execution on the management computer of code that is stored in the ample image. But it must provide at least the ability to perform at least one file system-type operation to read files and directories, and in embodiments that permit modification of the ample image at the management computer, the management environment must also permit at least one file system-type operation to write file and/or directory data. File system-type file and directory reads permit operations such as reading a directory to list all files, to list file attributes (size, creations date, permissions, etc.), to search a directory tree with wildcards, and to read file contents. File system-type file and directory writes permit operations such as creating, renaming, and moving files or file contents. Similar definitions pertain to volume, partition, and other storage structure reads and writes.

One suitable management environment is provided by a conventional file system that is running on a management node computer, in combination with a driver that supports mounting the ample image for at least read-access there by an administrator. One suitable read-only driver is provided by the V2i Protector product from PowerQuest Corporation. (V2I and V2I PROTECTOR are marks of PowerQuest Corporation). Read-write drivers are to be implemented and provided in some embodiments according to the invention. To the management computer operating system / file system, the mounted ample image looks like a standard local folder/volume/partition/disk, so that applications running on the management computer can access the ample image's data using standard operating system / file system calls.

Another suitable management environment is a virtual environment which contains data structures and code simulating at least storage system aspects of an imaged computer. Implementation of such a virtual environment may be informed by reference to the Virtual PC product of Connectix Corporation (see www.connectix.com) and/or by the discussion of a virtual engine environment, its components, their uses, and related information, in PowerQuest's United States Patent No. 6,330,653 at Figures 2-8 and columns 3-44. Virtual environments support storage operations (read/write) that can act on the content of an ample image without running file systems/operating system code that is stored in the ample image and that would have been used on the imaged computer to perform the operation(s) in question.

Some implementations may run, on the management computer, file systems / operating system code that is stored in the ample image, if that is necessary to perform a

Netflix, Inc. - Ex. 1002, Page 000597
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

hardware device driver or software application installation. This may be used to change the installed hardware support identified in an operating system registry, for example. But in general, only the storage system aspects of the imaged computer are needed at the management computer, not the full execution environment of the imaged computer.

"Meta-data" is structured data that describes the content and/or context of an ample image. Examples of meta-data include "structured image content information" and "structured imaged computer environment information". In each, the structure provides a computer-parseable structure, which distinguishes meta-data from free-form information or natural-language commentary like that found in readme files, press releases, email message bodies, and other texts intended primarily for human readers. For example, XML, data structure definitions, file formats, pointers and other addresses, indexes, and databases all provide computer-parseable structure.

Examples of structured image content information include: data specifying which applications were installed on the computer when the ample image was created, and may include version numbers; data specifying which operating system(s) were installed on the computer when the ample image was created, and may include version numbers, service release numbers, and patch history; file settings; other data which describes content of the ample image.

Examples of structured imaged computer environment information include: the time at which the ample image was created; the name of the user of the imaged computer; the name (e.g., machine address, processor ID, IP address) of the imaged computer; hardware that was installed on the imaged computer when the ample image was created.

Some meta-data, such as directory trees and Microsoft Windows brand registry entries, are implicitly stored in the ample image they describe. They are implicit in the ample image, in that they were created and placed in the ample image for purposes other than the methods of the present invention. The present invention provides new uses for such previously known data. Meta-data may also be explicitly stored in the ample image, by writing the meta-data to a file expressly for use by the present invention, and then creating an ample image which comprises that meta-data file. Meta-data for a given ample image may also be explicitly stored in a file that is stored outside the ample image but structurally associated with it. Network packets and other data structures that can be sent to a management computer are equivalent to files for the purpose of determining whether meta-

Netflix, Inc. - Ex. 1002, Page 000598
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

data is present, although some specific file or other data structures will provide better speed/reliability/etc. than others in a particular implementation.

Meta-data is "structurally associated" with an ample image by: using the meta-data as an index to locate the ample image in a collection of images; applying a file-naming convention so that the ample image file name corresponds to the name of at least one file containing meta-data for that ample image; placing meta-data in the same file(s) as the ample image to which that meta-data pertains; or otherwise providing a computer-parseable structural linkage between the ample image and meta-data for that ample image.

For instance, the invention provides an embedded meta-data file data management technique in which a file included in the ample image describes the contents of the ample image in computer-readable form. The file contains meta-data useful in describing the ample image inside which it is stored and the computer in which the ample image was taken. One can provide the ability to enable/disable the including of the file in the ample image. Examples of such meta-data include a file list, disk statistics, image creation date, computer name, other hardware configuration data, other software configuration data. This helps make ample images portable from computer to computer, helps provide quick identification of image contents without searching the entire image, and helps provide quick identification of the environment in which the image was taken. Like other features discussed herein, not every feature necessarily appears in every embodiment of the invention.

"Mining" refers to searching an ample image to extract data that facilitates computer resource management.

An Introduction

The invention provides tools and techniques for computer storage life-cycle management. As illustrated in Figure 1, one or more computers 102 are imaged, and then the image(s) 104 are used to help manage resources such as storage capacity, applications software, data files, and hardware drivers. The managed computers 102 may be stand-alone machines, or they may be connected to a network 106. If a network 106 is present, it may use known protocols or other tools such as HTTP, TCP/IP, XML. The network 106 can be owned and managed by a single entity, such as a corporation or a government agency, or by multiple entities. The network 106 or some other data transmission tool 108, such as shipped or carried DVDs or tapes, is used to transmit images 104 from the managed computer(s) 102 to one or more management computers 110.

Netflix, Inc. - Ex. 1002, Page 000599
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

The managed computers 102 may be network clients 112, network servers 114, or other computers. Each computer 102 has a hardware configuration 116, e.g., motherboard, processor, volatile storage, display, input devices, non-volatile storage, connection and connectivity devices. Each computer 102 also has a software configuration 118, e.g., one or more installed operating systems, corresponding license and computer naming information for the installed operating systems, application software, corresponding version and license information for that software, and user data. Each computer 102 operates in the context of a current environment 120, e.g., domain, workgroup, login validation information, access rights, settings, preferences, installed devices, and installed optional software.

Each computer 102 also has one or more local disks and/or other storage 122, on which the software and the user data 118 are stored. Each disk is organized into one or more partitions 124. Partitions are either formatted or unformatted. Unformatted partitions hold no data. Each formatted partition holds at least file system data, and also holds zero or more files. Each file typically contains data. Examples of files include operating system files such as registry files, configuration files, operating system code; application files such as initialization files, application code, sample or tutorial data; user files such as data files produced by users while using applications.

Because hardware, software, and context are often functionally linked in a computer, those of skill will appreciate that statements herein regarding any one of the components 116, 118, 120, 122 of a computer 102 may apply equally, for purposes of the present invention, to some or all of the other components. For instance, software 118 may be installed or removed, and so may hardware 116. Indeed, characterization of a given component as belonging to one of the categories 116, 118, 120, 122 may be more a matter of convenience or perspective than one of widespread consensus. For example, one may view a given component either as BIOS code 118 stored in a computer chip, or as a computer chip 116 that contains some BIOS code.

Regardless, the invention provides "offline" storage management, from the managed computer 102's point of view. That is, the invention provides storage management capabilities which use a copy of data from a computer's storage 122, in a place that is remote from the computer 102, to help manage the resources of the computer 102. In some cases the same computer 102 may also be managed online in a conventional manner, e.g., by using conventional SRM software which is running and controlled locally at the computer 102, or by using conventional SRM software that is controlled remotely, but which runs on the computer 102 and accesses the local storage 122 as its

Netflix, Inc. - Ex. 1002, Page 000600
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

primary data source. The invention is generally compatible with, but does not require, conventional online resource management.

One inventive approach includes taking a snapshot image 104 of a client 112, placing that snapshot image into a container 126 with meta-data 128, sending 108 the container 126 to another computer 110 such as an administrative server, modifying 130 the image there, sending 108 the modified image 104 back to the client 112, and installing the modified image on the client 112. Variations and alternatives are also discussed herein, including some in which the image 104 is not modified, and some in which an image 104 (modified or not, but at least indexed and searched) is installed on some computer other than the imaged computer 102 it originally came from.

Data collection and visualization management software and supporting hardware 134 allows administrators to perform tasks such as scheduling imaging, scheduling collection of meta-data, and managing the indexing 132 of images 104 into a collection organized for retrieval by specifying meta-data values. Meta-data may be structured image content information 142 such as directory trees or a list of installed applications, and/or structured imaged computer environment data 144 such as a list of installed hardware 116 or an IP address.

Life-cycle management software and supporting hardware 136 allows administrators to perform tasks such as bringing a specified image 104 into a management environment 138, searching the image 104, modifying the image 104, and (re)deploying the (modified) image 104 to one or more computers which may include the computer 102 on which the image 104 originated.

A Detailed Example

Figure 2 further illustrates a particular embodiment, which is more focused and hence narrower than the inventive data flow and architecture shown in Figure 1. For instance, Figure 2 calls for a virtual environment, which is a particular kind of management environment 138; Figure 2 assumes environment meta-data 128 is collected by an agent on a client computer 102; and Figure 2 provides for image 104 compression and an image store which are not required in Figure 1. Accordingly, limitations of Figure 2 are not necessarily found in every embodiment of the invention, but are presented instead to further illustrate some possible embodiments.

At the most general illustrated level of Figure 2, client-side components are both distinguishable from and connectable to three other groups of components by a computer network 106, which is one instance of a transmission means 108; other transmission means include, e.g., placing images 104 on DVD or tape and shipping the DVD or tape. The other three component

Netflix, Inc. - Ex. 1002, Page 000601
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

groups illustrated in Figure 2 are: data collection and visualization management components 134, shown to the left of a grouping line 202; meta-data indexing and management components 132, shown between grouping lines 202 and 204; and storage life-cycle management components 136, shown to the right of grouping line 204. The grouping lines 202, 204 are for clarity of illustration and discussion only, and they do not necessarily require any corresponding dividing or grouping structure in a given embodiment of the invention.

In the dataflow architecture shown in Figure 2, an administrator schedules 206 resource management for a particular user. In particular, this sends a triggering instruction 208 over the network 106 to a client computer 102 agent, which is part of the software 118 running on that client 102, 112. The agent collects 210 environment data 128, 144 reflecting the environment 120 and/or hardware 116 context, and takes 212 an ample image 104 or directs imaging software on the client 112 to take 212 an ample image 104. The agent may also collect other meta-data such as the user name, and the image 104 will typically contain content meta-data 142, 128 such as directory information and software 118 installation information in a registry. The agent bundles the image 104 and the meta-data 128 into a single container 126 for easier transport administration, and sends 214 the container 126 over the network 106 to a management node 110.

The management node 110 receives 216 the container 126, and separates/extracts the image 104 and the meta-data 128. The content meta-data 142 is indexed 218, and the environment meta-data 144 is indexed 220. Indexing may be done using conventional database indexing techniques and tools, familiar hash algorithms, and/or other approaches. The image 104 is stored 614 in a collection of images, which in this example is a database 222 of images; an indexed collection of files or storage blocks could also be used without a relational database so long as meta-data is indexed to permit image location 612. The environment meta-data 144 is stored in an inventory database 250. A joint database combining components 222, 250 could also be used.

The image database 222 is a storage location where the image 104 files are kept. One can specify characteristics of each image. For instance, it can be associated with a client computer so that the correct image can be later retrieved for restoration of the client computer. Images 104 may also be retrieved for further data extraction and/or image modification prior to restoration.

For example, one can locate 612 (and subsequently retrieve) a given image 104 in the image database 222 by specifying corresponding meta-data 128 values. One could locate all ample images 104 stored 614 in the database that were created within a specified time window, or all images for a given user (who might have used more than one machine 102), or all images for a given machine

Netflix, Inc. - Ex. 1002, Page 000602
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

102, or all images for machines that have a specified network card installed, or all images for machines 102 that lack a particular application program, or some Boolean / database query language combination of such meta-data key values.

One could also use the gathered meta-data 128 to build 252 an inventory summary of the hardware 116, software 118, storage devices and capacities 122, and/or other characteristics of the computers 102. The inventory database 250 would not always be current, since it is based on information collected from computers 102 when they are imaged, rather than being continuously updated as conventional SRM software may do. User environment information 220 may be displayed 254 to an administrator.

In some embodiments, the inventory database 250 is an indexed collection of data about the installed hardware 116 and software 118 on each client computer 112. This inventory database 250 is searchable to allow both reports and policies to be executed against selected conditions. For example, decisions to upgrade to a new operating system can be based upon the ability of existing client computers 112 to handle the minimum hardware requirements, and reports can be generated 256 identifying the upgrade or replacement hardware that is necessary to bring the lacking client computers 112 into compliance. This is also true for making decisions about upgrading software applications 118. The hardware inventory information is collected by agents or other programs running on the client computers 112. The results can be sent to the inventory database apart from the image 104, or they can be placed into a known file before imaging so that file will be in the image 104 that is sent up to the management node 110, where it is extracted from the image 104 and placed into the database 250.

In the illustrated architecture of Figure 2, extremely large amounts of image data may be managed. Accordingly, to reduce storage requirements, image clusters (blocks, sectors, etc.) may be compressed, to remove space-consuming redundancy within a given cluster and/or to identify identical clusters within more than one image 104. Compressed images are stored in an image store 228. Thus, the image store 228 is an optimized storage of multiple image 104 files where redundant pieces have been identified and replaced with references to fewer (ideally just one) stored copy of the imaged data. It is managed though an API 230 or other interface. An incremental image 104 may be merged with a prior complete snapshot image 104 to form a complete and more recent snapshot image which is placed in the image database 222 and/or the image store 228. A given image 104 in the image store 228 may also be the result of both image compression and merging in of incremental images.

Netflix, Inc. - Ex. 1002, Page 000603
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

The image compression 224 and the image store 228 provide an efficiency enhancement to the illustrated architecture, but aspects of them have been the subject of discussion outside PowerQuest more than a year prior to the priority date of the present application. For purposes of the present application, PowerQuest therefore does not intend to claim components 224, 226, 228, and/or 230 in and of themselves. PowerQuest will not object hereafter if components 224, 226, 228, and/or 230 are treated as prior art for examination purposes, regardless of whether a court of competent jurisdiction ultimately determines them to be prior art.

In some embodiments, the meta-data 128 identifies the user of the client computer; the imaging 212, meta-data 128 gathering, transmission 108, and indexing 218, 220 steps discussed above are repeated for multiple computers 102; user identifications are copied from the meta-data 128; and the image database 222 is built using the user identifications of multiple client computer containers 126 and images from those containers 126. The user identifications become 132 keys in the index 218/220 into the image database 222, so that one can locate 612 a particular image 104 by specifying the user identity. The time at which a container 126 was created, modified, or otherwise processed and time-stamped, can also be used as a meta-data key. Thus, one could locate all images 104 for a given user across time and/or across several machines 102, all images created within a given time window on one or more specified machines 102, or all images for a given user created in a given time window. A given embodiment may index 306 by one or more of the following meta-data 128 values: user name, computer name, image creation time, IP address, hardware installed, operating system type and version installed, application name and version installed, registry file content. Creation and use of indexes (e.g., database keys) is, in and of itself, well-understood technology, but embodiments that index 306 according to the present invention are believed to be new.

Turning now to the storage life-cycle management components 136, shown in Figure 2 to the right of grouping line 204, inputs such as user commands and policy specifications are received 232 through an interface. Resource management policy specifications in the inventive system are read from a computer-readable medium (volatile or not) in a computer-parseable form. Some inputs cause an embodiment to build 256 views showing the imaged computer's state at a specified time, and/or to build reports showing trends in storage usage, for example. Both local controls operating on the images 104 in the management node 110, and remote controls operating through the network 106 on imaged computers 102, may be

Netflix, Inc. - Ex. 1002, Page 000604
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

invoked in a command interpreter 234 to gather information for reports and views, or perform searching 310 and/or modifying 312 steps on disassociated ample images 104.

An administrator may schedule 236 storage life-cycle management tasks, such as upgrading images 104 with new drivers or other software 118 and deploying them, or tasks otherwise modifying images 104 and deploying them.

Ample images 104 are used in the illustrated system to build 238 a virtual environment, which can then be used as a management environment 138 to search 140 and/or modify 130, 240 the image 104 as it is embodied in that virtual environment on the management computer 110. Modified images 104 can be deployed 242 as new storage environments for one or more computers 102. Images 104 may also be deployed without modification, for familiar purposes of backup retrieval or disaster recovery. If one image 104 is to be deployed to multiple computers 102, it can be placed in a container 126 suitable for multicasting 244 over the network 106. The image 104 being deployed is received 246 at the target computer(s) 102, and imaged 248 into (partial ample image such as one that omits a partition 124) or onto (full ample image) each target computer's storage 122.

In some embodiments an image is recreated from the image store 228 or copied from the image database 222 and combined with the client's hardware and software configuration information from the inventory database 250, and is then used to build 238 a new virtual environment. Then the virtual environment is modified 240 as desired. A full image 104 in a virtual environment contains the necessary operating system software, application software and the required configuration files as well as the operating system and application data for use on a client computer 102, but with the invention the ample image 104 is configured without the need for the actual computer 102 to be used during the configuration process.

More About Modifying Ample Images

The ample image 104 may be modified and then redeployed back to the imaged computer. For instance, storage 122 may be reorganized and/or optimized. Files may be defragmented. New hardware drivers may be installed (driver software added if not already present, changes made to registry, etc.), to match hardware changes in the imaged computer that are/will be made after the ample image was created. Files may be rearranged in the ample image to make disk operations more efficient in other ways, e.g., by placing together on disk files that are frequently accessed together in time. Partition cluster size may be changed so that less storage space lies in the unused portion of clusters. Directory storage

Netflix, Inc. - Ex. 1002, Page 000605
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

space may be reclaimed. Storage may be divided between partitions 124, e.g., by placing user data in one partition and system data and software in another partition.

(Re)deployment of a modified ample image 104 may be one-to-one with the imaged computer, but it need not always be one-to-one. In a redeployment that is one-to-one with the imaged computer 102, an ample image of imaged computer A is taken, sent to management computer B, modified there, and then redeployed back to computer A only. The redeployed image replaces/overwrites the original data. In a first variation, an ample image of imaged computer A is taken, sent to management computer B, modified there, and then redeployed back to computer A and also deployed to one or more other computers C, D, E, and so on. In a second variation, an ample image of imaged computer A is taken, sent to management computer B, modified there, is not redeployed back to computer A, but is deployed to one or more other computers C, D, E, and so on.

In a third variation, an ample image of imaged computer A is taken, sent to management computer B, modified there, and then split between two or more computers on (re)deployment. For instance, suppose a single first server 114 is providing both web services and accounting services. Then a second server is purchased. To configure the second server, and reconfigure the first server, an ample image 104 of the first server is made and sent to a management computer 110. On the management computer, while the first server continues to run, the ample image is modified by splitting it into two ample images, one of which is suitable for a web server without accounting capabilities, and the other of which is suitable for an accounting server without web server capabilities. Then the two ample images are deployed. One overwrites the original server 114, and the other configures the new second server. As a result, the invention permits the functionality originally borne by a single server to be split between two (or more) servers, with relatively brief downtime, and thus puts less intense time-pressure on administrators during the reconfiguration process.

Some of the suitable modifications that may be made 240 to a virtual environment, or otherwise made to an ample image 104 according to the invention, include handling 530 viruses by detecting viruses and then quarantining/repairing/deleting infected files; taking 252 an inventory of the installed hardware and software present when the computer 102 was imaged; updating 504 applications with new releases and/or patches; checking 618 the integrity of a file system; defragmenting 522 one or more files; removing 532 one or more illegal files as defined 606 by corporate information systems policies (e.g., MP3 music files,

Netflix, Inc. - Ex. 1002, Page 000606
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

picture files, or other copyrighted files that lack clear authorization); checking 608 for security breaches by scanning for controlled or sensitive document or other information files; checking 608, 534 for remote mail folders that might constitute security policy violations; separating 620 data and systems volumes by identifying data folders and files, creating 520 a data volume, moving 622 the folders and files to the data volume and adjusting 516 the operating system and application configuration data to reflect the new logical location of the data (other separations could also be performed 622, e.g., migrating application software to one partition 124 and system software to another partition); sorting 624 files by type, function, or department, and combining 626 like files to provide a single master file on a server with clients having links defined to access the master copy; mapping 628 information location by enforcing standardized storage locations for the storage of particular types of data on all client computers; mapping shortcuts 630 by defining links and short cuts in the environment settings of all client computers to access certain data.

The modified virtual environment is then placed in a container file and sent over the network 106 back to the client computer 102. If multiple clients 102 should have the same modified storage environment, then the modified environment can be multicast 244 to those clients. An agent on each client then installs the modified image 248 onto the client, thereby making it the new storage environment of the client. The installation is accomplished by writing the container file content back to the appropriate locations on the hard disk(s) of the client computer and rebooting the client computer 102.

Thus, some embodiments of the invention include tools and techniques to read and temporarily store the image 104 of a specific user(s)/groups of users/computer(s) environment. The image 104 is indexed by user(s)/computer index and time of capture. At the management node 110, one can create a virtual environment or other management environment 138 that matches the current or historical state of the user(s)/groups of users/computer(s) environment, for the purpose of refreshing, or modifying the user(s)/groups of users/computer(s) online environment. Particular user(s) may need to have a display driver updated 510, for instance, because they all have the same display hardware. Likewise, a group of computers 102 may have been imaged just before a virus infected the enterprise, and so their active state can be rolled back using the saved images.

Embodiments may provide a language that allows the automation of methods discussed herein, so an administrator can customize the process by creating batch command

Netflix, Inc. - Ex. 1002, Page 000607
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

files. Such a language could allow the administrator to specify some or all aspects of assembling a specific operating environment or active state, including files and configuration information. Administrative server 110 applications 140 could search the saved images 104 and associated meta-data 128 for the purpose of reporting on, managing and subsequently modifying the operating environment based on the current and historical states of user(s)/computer environments.

More About Methods

Various embodiments of the present invention may be more fully understood by considering flowcharts illustrating methods of the invention. As noted elsewhere, system embodiments operate according to the methods discussed herein, as governed by the claims. The context, purposes, and results of method steps are likewise illustrated by the system and dataflow embodiments shown in Figures 1 and 2, and the discussion of those figures.

Figure 3 illustrates a method 300 of computer resource management according to the present invention. During a receiving step 302, the management node 110 (which includes at least one management computer) receives at least one ample image 104 (partial or full) of storage 122 of an imaged computer 102. The ample image 104 may be received as a single container 126 or as a single image file 104. Or it may be received in parts, as a sequence of image files which together comprise the ample image 104. Regardless, the ample image 104 is disassociated from the imaged computer 102 in that the ample image 104 is at the management computer 110 and the management computer 110 is distinct from the imaged computer 102.

During an obtaining step 304, the management node 110 obtains meta-data 128 that is structurally associated with the ample image 104. The meta-data 128 includes at least one of: structured image content information 142 which describes content of the ample image 104, and structured imaged computer environment information 144 which describes hardware 116 (or equivalently, hardware drivers / registry entries) that was installed on the imaged computer 102 when the ample image 104 was created 212. Meta-data may be embedded in the image 104, either in standard operating system and file system structures or in a file dedicated to meta-data. Meta-data 128 may also be outside the image 104 but inside the container 126. Finally, meta-data 128 may be transmitted separately from the image 104 and/or the image container 126.

During an indexing step 306, the ample image 104 is indexed into a collection of ample images, e.g., image database 222, such that the ample image can be located by using at least part of

Netflix, Inc. - Ex. 1002, Page 000608
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the meta-data 128 as a key. The meta-data 128 may be indexed 306, 218 into one or more databases or other data structures, to permit use of the meta-data as a key in locating the image, and optionally to permit use of the meta-data for other purposes, such as building an inventory database 250. In some embodiments, for example, the indexing step 306 indexes the ample image 104 into a collection of ample images such that the ample image can be located 612 by using at least part of the meta-data 128 as a key, where the image 104 is indexed by at least one of: a user name, a computer name, an image creation time, an IP address, a value identifying installed hardware, a value identifying installed operating system type, a value identifying an installed application, a value given in a registry file of the imaged computer.

During a bringing step 308, the ample image 104 is brought into a management environment 138 at the management node 110. This may require building 238 a virtual environment, or it may require use of a driver that lets an image 104 be mounted as a volume at the management node 110 so the mounted image is accessible to system and applications software running on the management computer 110.

Steps 310 and 312 are instances of a resource management facilitating step 314. Commands to perform steps 314 may be input interactively and/or in a script of batch-mode user commands, so the facilitating step is performed in response to interactive-mode user commands and/or in response to a script of batch-mode user commands. Commands may be triggered by a specified event or condition, or a combination of events and/or conditions. Commands may be subject to policies specified 606 by an administrator. Policies can trigger management actions such as running a script, creating a report, and/or sending 616 an alert to an administrator.

Either or both steps 310, 314 are performed at least once using the ample image 104 in the management environment 138. During image searching step 310, the directory tree, file contents, partition table, and/or other content of the ample image 104 is searched. Searching 310 involves looking inside the ample image 104, not merely looking at file system characteristics of the file(s) that contain the ample image 104. During image modifying step 312, the directory tree, file contents, partition table, sector allocation information, and/or other content of the ample image 104 is modified. Modifying 312 likewise involves changing information inside the ample image 104, not merely changing file system characteristics of the file(s) that contain the ample image 104.

During an optional deploying step 316, the ample image (possibly modified) is deployed over a network 106 or other transmission means 108 to the imaged computer 102 and/or to another

Netflix, Inc. - Ex. 1002, Page 000609
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

computer which is distinct from the management node 110. Familiar image deployment tools and techniques may be used.

Figure 4 further illustrates steps which may serve as image searching steps 310 in the management environment 138 of a given embodiment and/or in a given situation. These are illustrative only. The list shown is not necessarily comprehensive, and some of the steps may overlap, e.g., reading a file involves reading sectors. Not every listed step need be present in every embodiment. As with all flowcharts provided herein, steps may be grouped differently, reordered (except to the extent that one step requires results of another step), repeated, omitted, performed concurrently, and/or renamed in a given embodiment, to the extent that doing so still provides an operable embodiment consistent with the claims.

A file reading step 402 reads a file from a disassociated ample image 104, that is, from an image 104 in a management environment 138 at a management computer 110 that is distinct from the computer 102 whose image 104 is being read. A directory reading step 404 likewise reads file system directory information from a disassociated ample image 104; this may be done, for instance, to determine whether a particular file is present, to determine whether at least one file of a particular type is present, or to check for compliance with policies stating what types of files are stored in a given directory. A partition table reading step 406 reads partition table information from a disassociated ample image 104. A sector reading step 408 reads a disk sector from a disassociated ample image 104; it may also read a cluster or block containing multiple sectors. The sector read may be in a file, or it may lie outside any file. A disk usage information extracting step 410 performs similarly to conventional SRM disk usage information extraction but reads from a disassociated ample image 104 instead of reading from a local machine it runs on. Likewise, a billing determination step 412 performs similarly to conventional SRM billing determination but reads a disassociated ample image 104 to determine billing for the imaged computer 102 instead of the (management) computer 110 it is running on. A searching step 414 reads a disassociated ample image 104 to check for illegal files (such as MP3 files, or limited distribution files on an unauthorized machine 102) and/or to check files for virus infection.

Mining 310 an ample image 104 according to the present invention is different in several respects from mining storage 122 locally while that storage is in use. First, the ample image is not current, so the information it provides is aged. Second, by mining several ample images taken at different times, the present invention provides information about trends that is not limited to predetermined variables – any data in the ample images 104 can be examined

Netflix, Inc. - Ex. 1002, Page 000610
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

at any imaged point in time. Third, by mining several ample images taken from different machines 102 at substantially the same time, and doing this for sets of images at different times, the invention can help administrators identify the entry point/origin of suspect files (illegal files, viruses) and the propagation path(s) of suspect files. Travel across time and across the network with a focus on variables dynamically identified during the search is easier. Fourth, the present invention can provide detailed information about machines 102 that are not running, including machines that are now difficult or expensive to run because of changed circumstances. This may save administrative overhead and/or facilitate forensic investigations. Fifth, although agents can be used to gather environment meta-data, not every embodiment of the invention requires an agent running on the imaged computer 102. Environment meta-data may be gathered from a registry, or it may be omitted entirely.

Disassociated images 104 may be mined 310 according to the invention using software that does – on the mined image 104 – the same kind of operations that conventional SRM software does on the original storage 122 (either locally or via remote control). Like conventional SRM tools, some of the systems and software according to the present invention can be controlled partially or entirely by policies which are set 606 by an administrator. Such policies may set disk quotas, identify illegal file types, specify particular storage locations for particular types of data, require antivirus software to be current to within a specified time period, and/or require or prohibit certain hardware or software configurations, for instance. However, unlike conventional SRM operations, the methods of the present invention are not done in real-time on the actual system 102 whose resources are being managed, but can be done instead at the administrator's convenience using an ample image taken from the managed computer 102 to a management computer 110.

Figure 5 further illustrates steps which may serve as image modifying steps 312 in the management environment 138 of a given embodiment and/or in a given situation. It will be understood that these modifications take effect on managed computers 102 only if the modified image 104 is deployed 316 to the managed computer(s) 102. As with Figure 4, the steps in Figure 5 are illustrative only, not necessarily comprehensive or required, and may overlap. As illustrated, resource management methods of the invention may modify 312 the ample image 104 by performing at least one of the following steps on the disassociated ample image 104: installing 502 an application; updating 504 an application by patching code, adding new data files such as virus definitions, or otherwise; removing 506 an application (which may include regular uninstalls that

Netflix, Inc. - Ex. 1002, Page 000611
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

update the registry, as well as bare file system deletes that may not); installing 508 a hardware driver; updating 510 a hardware driver; removing 512 a hardware driver; installing 514 system software other than a hardware driver, such as an operating system or communications software; updating 516 system software other than a hardware driver; removing 518 system software other than a hardware driver.

Each of these steps 502 through 518 facilitates management 314 of software 118 as a resource. Note that removing a piece of software is done by uninstalling, overwriting, disabling, and/or deleting that software. Removing is directed specifically at that software rather than the entire image 104, since one does not normally delete an entire image from the management environment 138 merely to delete a particular program. Software installation, updating, and removal, in and of themselves, are well-known in other contexts, but their use on ample images 104 according to the present invention is believed to be new.

In addition to, or in place of, the foregoing image modifying steps, the invention supports modifying 312 an ample image 104 to manage storage capacity and/or to help manage 314 the organization of items within storage 122. For example, a partition manipulating step 520 is performed by performing at least one of the following steps on the ample image 104 in the management environment 138: creating a partition 124, resizing a partition, resizing clusters of a partition, moving a partition, moving data between partitions, canonizing a partition, deleting a partition, merging two partitions, manipulating volume segments. Partition manipulations, in and of themselves, are well-known in other contexts, but their use on ample images according to the present invention is believed to be new. Storage capacity may also be managed 314 by defragmenting 522 files in the ample image 104, by reclaiming 524 directory space in a partition 124 of the ample image 104, by recovering 526 data despite corrupted or lost file system information in the ample image 104, and/or by converting 528 one file system type to another file system type (e.g., converting between FAT and Microsoft NT brand file systems) in the ample image 104.

In some embodiments and/or situations, the invention provides a method that comprises both searching 310 the ample image 104 and modifying 312 the ample image 104, by performing at least one of the following pairs of steps on the ample image in the management environment: handling 530 a virus by detecting a virus and then quarantining or repairing an infected file; handling 530 a virus by detecting a virus and then removing an infected file; handling 532 an illegal file by identifying 404 an illegal file and then removing that illegal file; enforcing 534 policy by identifying a breach of a policy that defines

Netflix, Inc. - Ex. 1002, Page 000612
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

standardized storage locations for particular types of data and then moving or removing data to enforce that policy.

A modified ample image 312 may be viewed in claim terms as the product of a process. For instance, Figure 3 is stated from the perspective of the management computer, but as illustrated in Figure 7 the invention likewise provides a process that can be stated from the perspective of an imaged computer 102: send 702 an ample image 104 to a management node 110; send 704 meta-data to the management node 110; after both sending steps receive and install 706 a modified ample image on the computer 102. The modified ample image may differ from the image sent 702 to the management node 110 in one or more of the following ways: a change (installation, updating, patching, disabling, or removal) in application software; a change in driver software; a change in other software; a change in the partition table (e.g., new partition, partitions merged, partitions resized); a change in cluster size; a change in the folder in which given data is stored; a change in the partition in which data is stored; a change in status (e.g., quarantined or not, repaired or not, removed or not) of a file infected by a virus; a change in status (removed or not) in an illegal file.

Figure 6 further illustrates steps which may serve as resource management facilitating steps 314 in the management environment 138 of a given embodiment and/or in a given situation. The steps are illustrative only, not necessarily comprehensive or required, and may overlap. Groups of images 104 can be tracked, or used in tracking files. In one tracking step 602, multiple ample images 104 are searched to track propagation of a file, an infection, or another storage characteristic over time; the ample images used for this step 602 are imaged from a particular computer 102 at different points in time. For example, one could track 602 propagation of a virus through the files on a computer 102 to identify a virus' point of entry, to gain a better understanding of how the virus operates, and/or to determine which files may have been infected.

In another tracking step 604, multiple ample images 104 are searched to track propagation of a file, an infection, or another storage characteristic over nodes in a network, that is, over different computing devices attached to the network. In this case, the ample images used are imaged from multiple computers 102, and may also be imaged at different points in time. One could track 604 a virus across the network 106. One could also track 604 an illegal file, such as an MP3 file or a file containing highly confidential information, across

Netflix, Inc. - Ex. 1002, Page 000613
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the network 106 to identify users who have insecure machines 102, for example, or users who might be involved in unauthorized activities.

Other management steps 314 are discussed elsewhere, and shown in Figure 6 as a reminder. For instance, during a policy setting step 606, an administrator sets a policy, such as a policy of the type recognized in conventional SRM software, to be enforced 534 by appropriate modifications 312 of ample images 104 followed by deployment 316 of the modified images.

Also, if modification 312 of an image 104 makes the meta-data 128 associated with the image inaccurate, then the meta-data may be modified 610 to match the modified image. For example, if a hardware driver is installed 508, then the meta-data 144 describing the hardware 116 on the imaged computer may be modified 610 accordingly (regardless of whether the actual hardware has been swapped yet) so that the meta-data indexing 132, 218 operates to locate 612 the modified image 104 in the image database 222 when the hardware driver in question is specified as a key.

Some embodiments provide software similar to PowerQuest Drive Image® software, to build and write – in a recognized image file format – an image from the ample image 104 data in a management environment. The resulting collection of one or more image files can then be (re)deployed 316 using conventional image deployment software.

Configured Media Generally

The computers 102, 110 are capable of using floppy drives, tape drives, optical drives or other means to read a configured storage medium 146. A suitable configured medium includes a magnetic, optical, or other computer-readable storage device. Suitable storage devices include floppy disks, hard disks, tape, CD-ROMs, PROMs, RAM, and other computer system storage devices. The substrate configuration of the medium 146 represents data and instructions which cause the computer system to operate in a specific and predefined manner as described herein. For instance, the medium 146 tangibly embodies a program, functions, and/or instructions that are executable by the management computers 110 or imaged computers 102 to perform methods of the present invention substantially as described and claimed herein.

Netflix, Inc. - Ex. 1002, Page 000614
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Conclusion

Using the present invention, one can take a picture or snapshot of up to the entire functioning storage space of a client computer. Indeed, all computers in an enterprise including all servers, desktops, laptops, and mobile data devices could be imaged. After the image(s) are copied or moved to an alternate storage location at a management computer, one can perform various resource management tasks using the alternate copies of the image. An ample image contains persistent data from the client. In a full ample image, this is data that represents the exact state of both the operating system and all installed software as well as all the system and user data contained on the client computer. This is the "active state" of a fully imaged computer. Using the captured active state, one can mine this data, as in an SRM application, to learn any storage-evident aspect of the current operation of the operating system and applications, as well as anything else captured in the stored data.

As a further step, one can then make modifications to this active state data and send it, back to the same client and/or to another client computer, by rebuilding an image, taking that image to the target(s), and then (re)writing each target's storage with the image. When sending the image back to different client than the one on which it originated, one should make necessary modifications in the environment and configuration data used by the operating system to allow it to properly recognize and accept this change in its active state. For instance, hardware differences may require device driver changes, and disk capacity changes may require resizing or splitting partitions, or deleting unwanted data. Changes can be made interactively and/or by scripts, and they may be constrained or triggered by administrator-defined policies.

Embodiments of the invention may allow less disruptive management of a real-time computing environments than other tools, may reduce the number of copies of management software needed per node, may allow storage management to be centralized around one controlled set of images and meta-data, may allow storage management applications to view the historical conditions of a user(s)/computer and its environment, and/or may allow for the protection of computer 102 data by permitting environments to be rolled back to a specific point in time and the associated environment.

Suitable software to assist in implementing the invention is readily provided by those of skill in the pertinent art(s) using the teachings presented here and programming languages and tools such as C++, C, Java, Pascal, Perl, Python, XML, APIs, SDKs, assembly, firmware, microcode, and/or other languages and tools.

Netflix, Inc. - Ex. 1002, Page 000615
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Although particular embodiments of the present invention are expressly illustrated and described herein, it will be appreciated that discussion of one type of embodiment also generally extends to other embodiment types. For instance, the description of the methods illustrated in Figure 3 also helps describe systems like those shown in Figure 1 which can operate according to those methods. Embodiments such as the methods or systems illustrated may omit items/steps, repeat items/steps, group them differently, supplement them with familiar items/steps, or otherwise comprise variations on the given examples. All claims as filed are part of the specification and thus help describe the invention, and repeated claim language may be inserted outside the claims as needed. Any failure to expressly identify a thing as prior art in this document is not an assertion that it is not prior art; readers are assumed to be familiar with the state of the art.

As used herein, terms such as "a" and "the" and designations such as "ample image" and "searching", are inclusive of one or more of the indicated item or step. In particular, in the claims a reference to an item generally means at least one such item is present and a reference to a step means at least one instance of the step is performed.

The invention may be embodied in other specific forms without departing from its essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. Headings are for convenience only. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope to the full extent permitted by law.

What is claimed and desired to be secured by patent is:

Netflix, Inc. - Ex. 1002, Page 000616
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS

1. A method of computer resource management, comprising the steps of:

receiving, at a management node which includes at least one management computer, at least one ample image of storage of an imaged computer, the ample image thus disassociated from the imaged computer in that the management computer is distinct from the imaged computer;

obtaining at the management node meta-data that is structurally associated with the ample image, the meta-data including at least one of:

structured image content information which describes content of the ample image, and

structured imaged computer environment information which describes hardware that was installed on the imaged computer when the ample image was created;

indexing the ample image into a collection of ample images such that the ample image can be located by using at least part of the meta-data as a key;

bringing the ample image into a management environment at the management node; and

facilitating management of a resource by performing at least one of the following steps using the ample image in the management environment:

searching the ample image;

modifying the ample image.

2. The method of claim 1, wherein the method comprises modifying the ample image and further comprises the step of deploying the modified ample image to at least one of: the imaged computer, another computer which is distinct from the management node.

3. The method of claim 2, wherein the bringing step brings the ample image into a virtual environment, and the virtual environment is modified to produce the modified ample image that is deployed in the deploying step.

4. The method of claim 1, wherein the method comprises modifying the ample image by performing at least one of the following steps on the ample image in the

Netflix, Inc. - Ex. 1002, Page 000617
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

management environment: installing an application, updating an application, removing an application, installing a hardware driver, updating a hardware driver, removing a hardware driver, installing system software other than a hardware driver, updating system software other than a hardware driver, removing system software other than a hardware driver, wherein removing a piece of software is done by uninstalling, overwriting, disabling, and/or deleting that software and removing is directed specifically at that software rather than the entire image.

5. The method of claim 1, wherein the method comprises modifying the ample image by performing at least one of the following steps on the ample image in the management environment: creating a partition, resizing a partition, resizing clusters of a partition, moving a partition, moving data between partitions, canonizing a partition, deleting a partition, merging two partitions, manipulating a volume segment.

6. The method of claim 1, wherein the method comprises modifying the ample image by performing at least one of the following steps on the ample image in the management environment: defragmenting files, reclaiming directory space in a partition, recovering data despite corrupted or lost file system information, converting one file system type to another file system type.

7. The method of claim 1, wherein the method comprises modifying the ample image in the management environment, and further comprises modifying the meta-data to reflect a modification of the ample image.

8. The method of claim 1, wherein the method comprises searching the ample image and performing at least one of the following steps on the ample image in the management environment: extracting disk usage information, determining whether a particular file is present, determining whether at least one file of a particular type is present, determining billing for storage usage.

9. The method of claim 1, wherein the method comprises searching the ample image and modifying the ample image, by performing at least one of the following pairs of

Netflix, Inc. - Ex. 1002, Page 000618
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

steps on the ample image in the management environment: detecting a virus and then quarantining or repairing an infected file, detecting a virus and then removing an infected file, identifying an illegal file and then removing that illegal file, identifying a breach of a policy that defines standardized storage locations for particular types of data and then moving or removing data to enforce that policy.

10. The method of claim 1, wherein the method comprises searching multiple ample images to track file propagation by at least one of: tracking propagation of a file over time using ample images of a particular imaged computer that were created at different points in time, tracking propagation of a file over multiple computers using ample images of multiple imaged computers.

11. The method of claim 1, wherein the step of bringing the ample image into a management environment at the management node comprises mounting the ample image into a file system that is running on a management node computer.

12. The method of claim 1, wherein the step of receiving at a management node at least one ample image of storage of an imaged computer comprises receiving an image of an entire disk of the imaged computer.

13. The method of claim 1, wherein the step of indexing the ample image into a collection of ample images such that the ample image can be located by using at least part of the meta-data as a key comprises indexing by at least one of: a user name, a computer name, an image creation time, an IP address, a value identifying installed hardware, a value identifying installed operating system type, a value identifying an installed application, a value given in a registry file of the imaged computer.

14. The method of claim 1, wherein at least one facilitating step is performed in response to a script of batch-mode user commands.

15. The method of claim 1, wherein at least one facilitating step is performed in response to a policy specified on a computer-readable medium in a computer-parseable form.

Netflix, Inc. - Ex. 1002, Page 000619
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

16. A modified ample image produced by a process comprising the steps of:

sending, from an imaged computer to a management node which includes at least one management computer, at least one ample image of storage of the imaged computer, the ample image thus disassociated from the imaged computer in that the management node is distinct from the imaged computer;

sending, from the imaged computer to the management node, meta-data that is structurally associated with the ample image, the meta-data including at least one of:

structured image content information which describes content of the ample image, and

structured imaged computer environment information which describes hardware that was installed on the imaged computer when the ample image was created; and

receiving, at the imaged computer, a modified ample image which is a modification of the ample image sent to the management node.

17. The modified ample image of claim 16 in combination with the imaged computer, wherein the modified ample image has been installed in storage at the imaged computer.

18. The modified ample image of claim 16, wherein the modified ample image differs from the image sent to the management node in at least one of the following ways: a change in application software; a change in driver software; a change in other software; a change in a partition table; a change in cluster size; a change of given data's location from a folder in which the given data was stored; a change of given data's location from a partition in which the given data was stored; a change in status of a file infected by a virus; a change in status of an illegal file.

19. A computer-readable storage medium configured to cause at least one management node computer to perform a method of computer resource management, the method comprising the steps of:

Netflix, Inc. - Ex. 1002, Page 000620
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

receiving, at a management node computer, at least one ample image of storage of an imaged computer, the ample image thus disassociated from the imaged computer in that the management computer is distinct from the imaged computer;

obtaining at the management node meta-data that is structurally associated with the ample image;

indexing the ample image into a collection of ample images such that the ample image can be located by using at least part of the meta-data as a key;

modifying the ample image in the management environment; and

sending the modified ample image from the management node to be installed on at least one computer that is not in the management node.

20. The configured medium of claim 19, wherein the method comprises modifying the ample image by performing at least one of the following steps on the ample image: installing software, updating software, removing software, wherein removing a piece of software is done by uninstalling, overwriting, disabling, and/or deleting that software, and removing is directed specifically at that software rather than the entire image.

21. The configured medium of claim 19, wherein the method comprises modifying the ample image by performing at least one of the following steps on the ample image: modifying the partition table, resizing a partition, resizing clusters of a partition, moving data between partitions, canonizing a partition, deleting a partition, merging two partitions, manipulating a volume segment.

22. The configured medium of claim 19, wherein the method comprises modifying the ample image by performing at least one of the following steps on the ample image: defragmenting files, reclaiming directory space in a partition, recovering data despite corrupted or lost file system information, converting one file system type to another file system type, quarantining an infected file, repairing an infected file, removing an infected file, removing an illegal file.

Netflix, Inc. - Ex. 1002, Page 000621
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

23. The configured medium of claim 19, wherein the method further comprises modifying the meta-data to reflect a modification of the ample image.

24. The configured medium of claim 19, wherein the method further comprises searching the ample image.

25. A system for computer resource management, comprising:
a management node which includes at least one management computer;
an indexing component which indexes ample images into a collection of ample images such that a given ample image can be located by using meta-data as a key;
a management environment at the management node; and
a searching tool for searching an ample image within the management environment.

26. The computer resource management system of claim 25, further comprising a modifying tool for modifying an ample image within the management environment.

27. The computer resource management system of claim 26, further comprising code for modifying the modifying meta-data to reflect a modification of the ample image.

28. The computer resource management system of claim 25, further comprising a deploying tool for deploying an ample image from the management environment to at least one computer not in the management node.

29. The computer resource management system of claim 25, wherein the modifying tool is capable of modifying the ample image by performing at least one of the following steps on the ample image in the management environment: creating a file, installing a file, updating a file, removing a file, and wherein removing a file containing software is done by uninstalling, overwriting, disabling, and/or deleting that software, and removing a file is directed specifically at that file rather than the entire image.

Netflix, Inc. - Ex. 1002, Page 000622
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

30. The computer resource management system of claim 25, wherein the modifying tool is capable of modifying the ample image by performing at least one of the following steps on the ample image in the management environment: manipulating a partition, manipulating a volume segment, defragmenting files, reclaiming directory space, recovering data despite corrupted or lost file system information, converting one file system type to another file system type.

31. The computer resource management system of claim 25, wherein the searching tool is capable of searching the ample image by performing at least one of the following steps on the ample image in the management environment: extracting disk usage information, determining whether a particular file is present, determining whether at least one file of a particular type is present, determining billing for storage usage, detecting a virus-infected file, identifying an illegal file, identifying a breach of a policy that defines standardized storage locations for particular types of data, tracking propagation of a file over time using ample images of a particular imaged computer that were created at different points in time, tracking propagation of a file over multiple computers using ample images of multiple imaged computers.

32. The computer resource management system of claim 25, wherein the management environment comprises a driver capable of mounting the ample image into a file system that is running on a management node computer.

33. The computer resource management system of claim 25, wherein the management environment comprises a virtual environment that simulates storage-related characteristics of the imaged computer.

34. The computer resource management system of claim 25, further comprising a command interpreter which processes a script of batch-mode user commands.

Netflix, Inc. - Ex. 1002, Page 000623
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

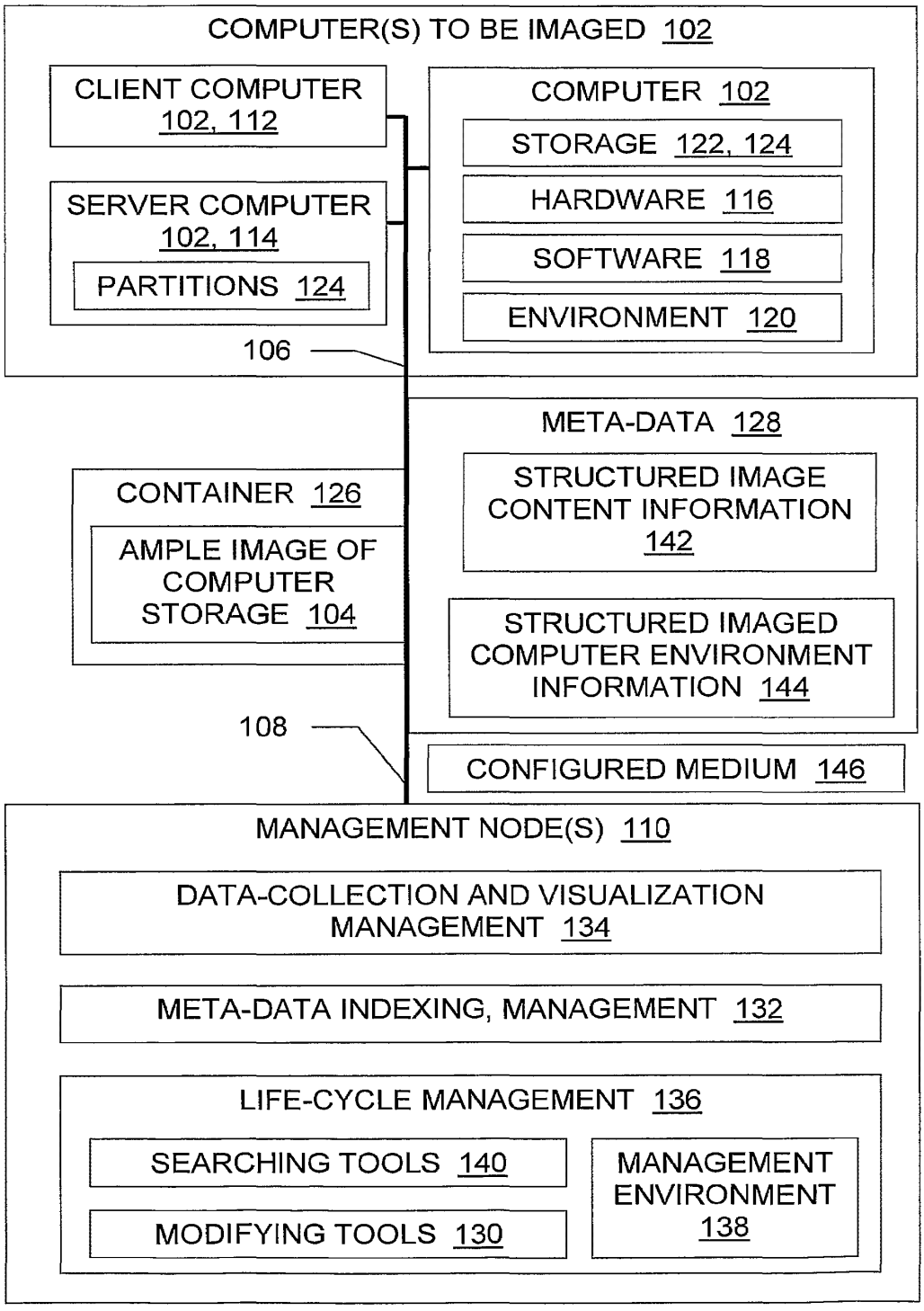

COMPUTER(S) TO BE IMAGED 102
CLIENT COMPUTER 102, 112
SERVER COMPUTER 102, 114
PARTITIONS 124
COMPUTER 102
STORAGE 122, 124
HARDWARE 116
SOFTWARE 118
ENVIRONMENT 120
106
META-DATA 128
CONTAINER 126
AMPLE IMAGE OF COMPUTER STORAGE 104
STRUCTURED IMAGE CONTENT INFORMATION 142
STRUCTURED IMAGED COMPUTER ENVIRONMENT INFORMATION 144
108
CONFIGURED MEDIUM 146
MANAGEMENT NODE(S) 110
DATA-COLLECTION AND VISUALIZATION MANAGEMENT 134
META-DATA INDEXING, MANAGEMENT 132
LIFE-CYCLE MANAGEMENT 136
SEARCHING TOOLS 140
MODIFYING TOOLS 130
MANAGEMENT ENVIRONMENT 138

Fig. 1

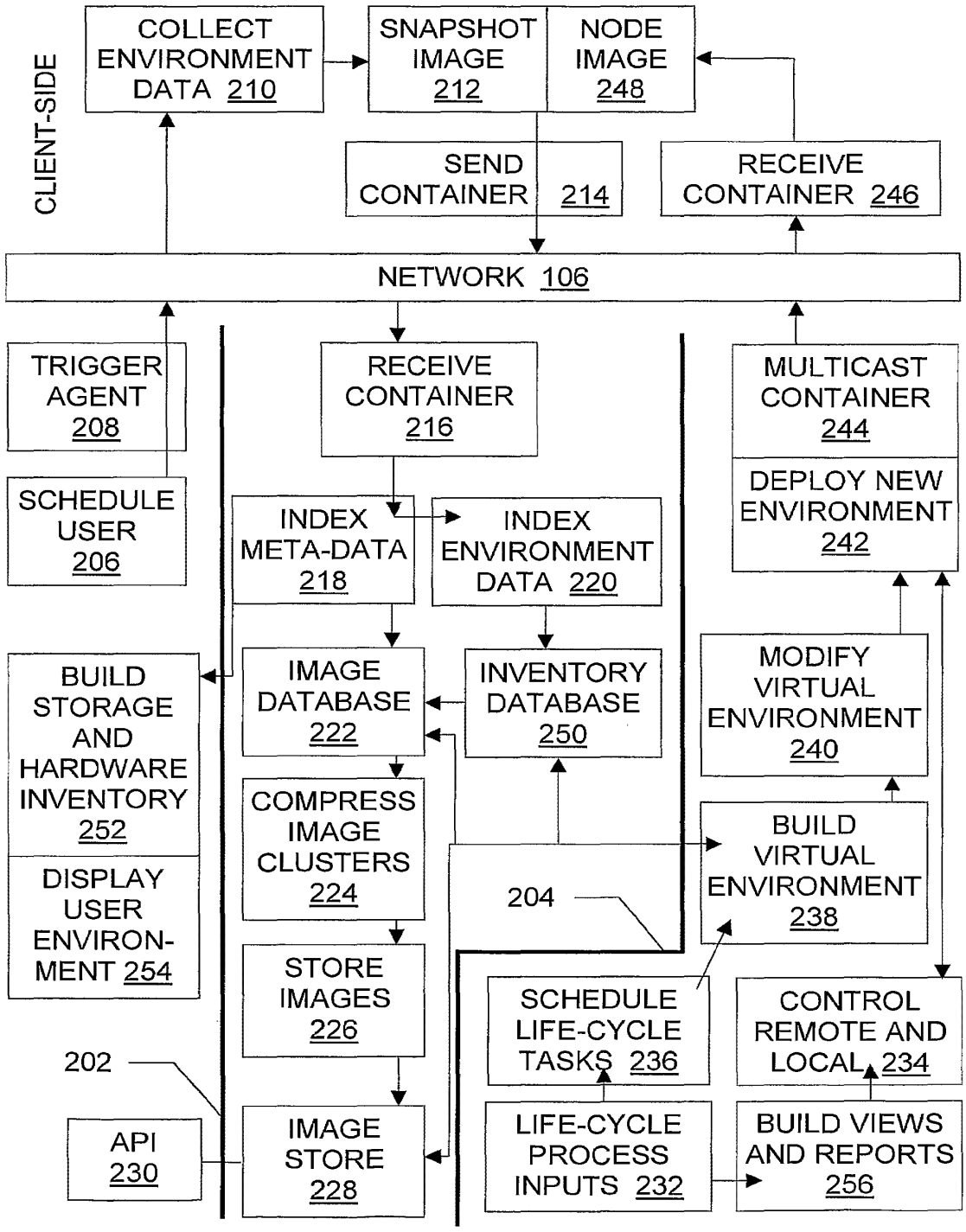

CLIENT-SIDE
COLLECT ENVIRONMENT DATA 210
SNAPSHOT IMAGE 212
NODE IMAGE 248
SEND CONTAINER 214
RECEIVE CONTAINER 246
NETWORK 106
TRIGGER AGENT 208
SCHEDULE USER 206
RECEIVE CONTAINER 216
INDEX META-DATA 218
INDEX ENVIRONMENT DATA 220
IMAGE DATABASE 222
INVENTORY DATABASE 250
COMPRESS IMAGE CLUSTERS 224
STORE IMAGES 226
IMAGE STORE 228
API 230
BUILD STORAGE AND HARDWARE INVENTORY 252
DISPLAY USER ENVIRON-MENT 254
202
204
MULTICAST CONTAINER 244
DEPLOY NEW ENVIRONMENT 242
MODIFY VIRTUAL ENVIRONMENT 240
BUILD VIRTUAL ENVIRONMENT 238
SCHEDULE LIFE-CYCLE TASKS 236
CONTROL REMOTE AND LOCAL 234
LIFE-CYCLE PROCESS INPUTS 232
BUILD VIEWS AND REPORTS 256

Fig. 2

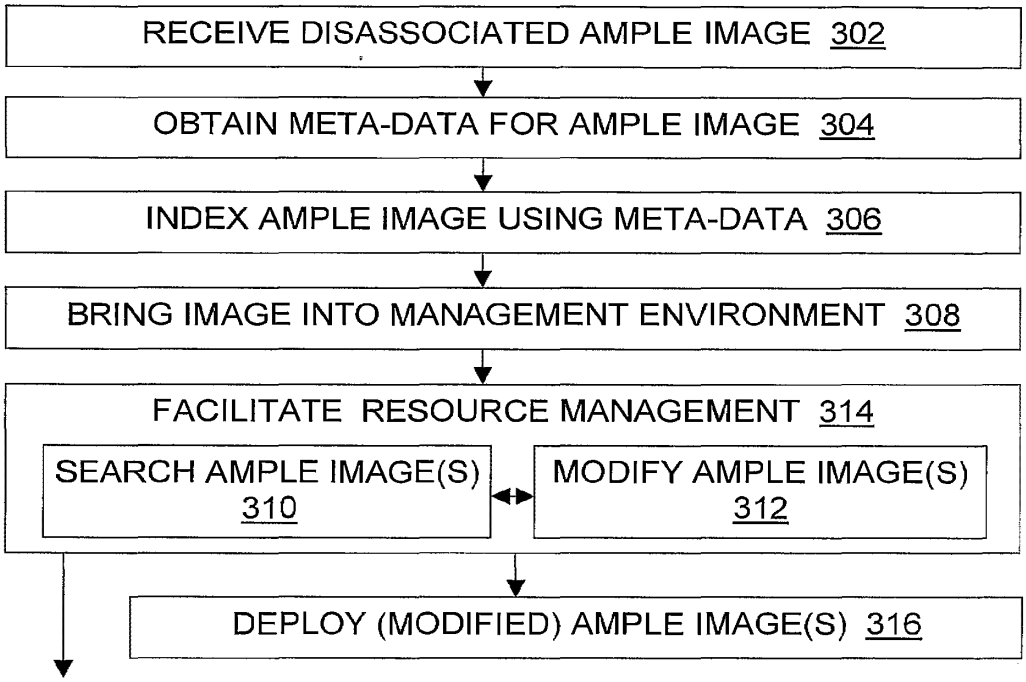

RECEIVE DISASSOCIATED AMPLE IMAGE 302
OBTAIN META-DATA FOR AMPLE IMAGE 304
INDEX AMPLE IMAGE USING META-DATA 306
BRING IMAGE INTO MANAGEMENT ENVIRONMENT 308
FACILITATE RESOURCE MANAGEMENT 314
SEARCH AMPLE IMAGE(S) 310
MODIFY AMPLE IMAGE(S) 312
DEPLOY (MODIFIED) AMPLE IMAGE(S) 316

Fig. 3

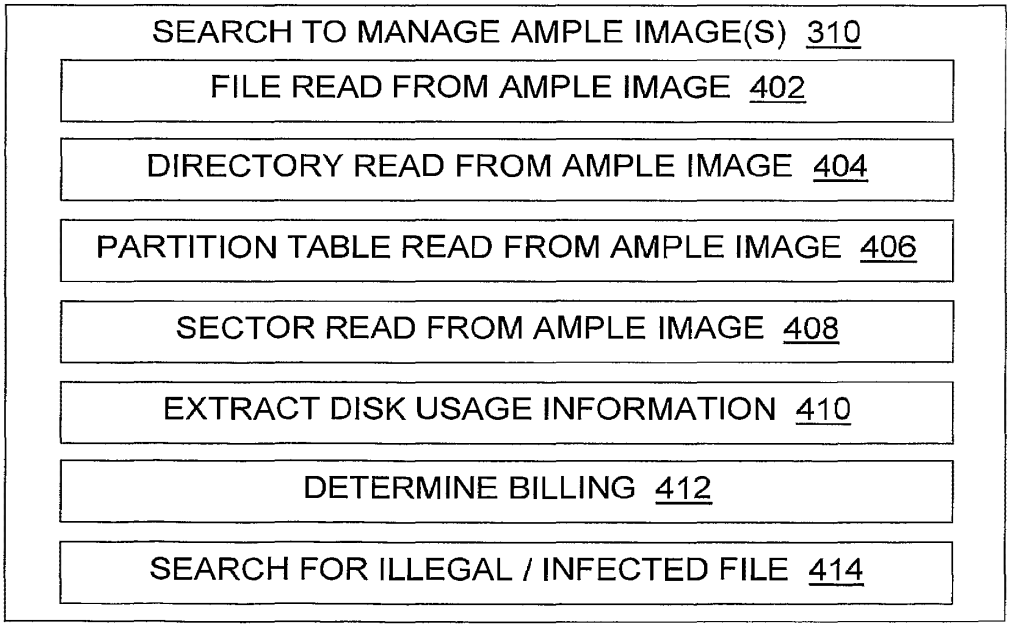

SEARCH TO MANAGE AMPLE IMAGE(S) 310
FILE READ FROM AMPLE IMAGE 402
DIRECTORY READ FROM AMPLE IMAGE 404
PARTITION TABLE READ FROM AMPLE IMAGE 406
SECTOR READ FROM AMPLE IMAGE 408
EXTRACT DISK USAGE INFORMATION 410
DETERMINE BILLING 412
SEARCH FOR ILLEGAL / INFECTED FILE 414

Fig. 4

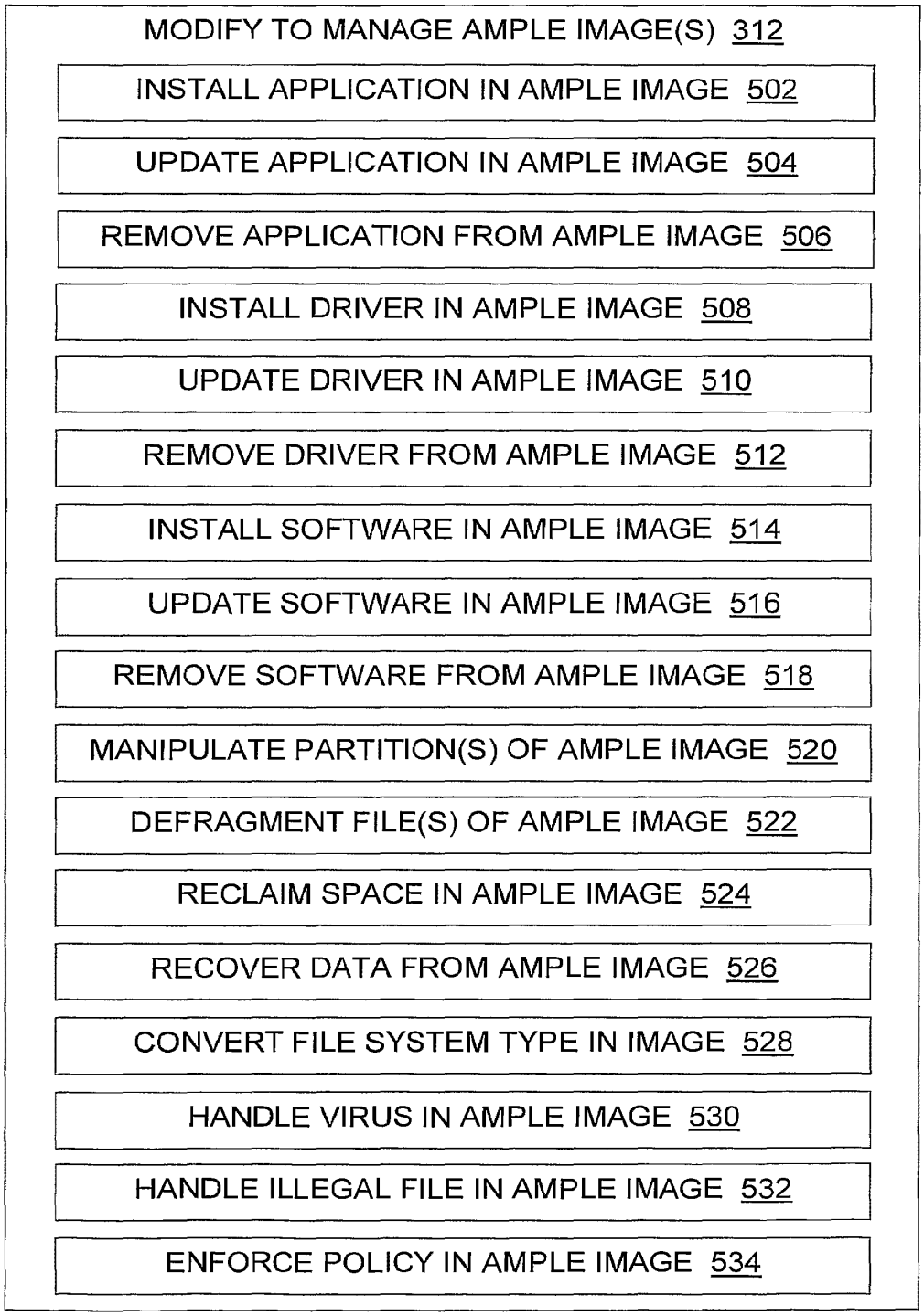

MODIFY TO MANAGE AMPLE IMAGE(S) 312
INSTALL APPLICATION IN AMPLE IMAGE 502
UPDATE APPLICATION IN AMPLE IMAGE 504
REMOVE APPLICATION FROM AMPLE IMAGE 506
INSTALL DRIVER IN AMPLE IMAGE 508
UPDATE DRIVER IN AMPLE IMAGE 510
REMOVE DRIVER FROM AMPLE IMAGE 512
INSTALL SOFTWARE IN AMPLE IMAGE 514
UPDATE SOFTWARE IN AMPLE IMAGE 516
REMOVE SOFTWARE FROM AMPLE IMAGE 518
MANIPULATE PARTITION(S) OF AMPLE IMAGE 520
DEFRAGMENT FILE(S) OF AMPLE IMAGE 522
RECLAIM SPACE IN AMPLE IMAGE 524
RECOVER DATA FROM AMPLE IMAGE 526
CONVERT FILE SYSTEM TYPE IN IMAGE 528
HANDLE VIRUS IN AMPLE IMAGE 530
HANDLE ILLEGAL FILE IN AMPLE IMAGE 532
ENFORCE POLICY IN AMPLE IMAGE 534

Fig. 5

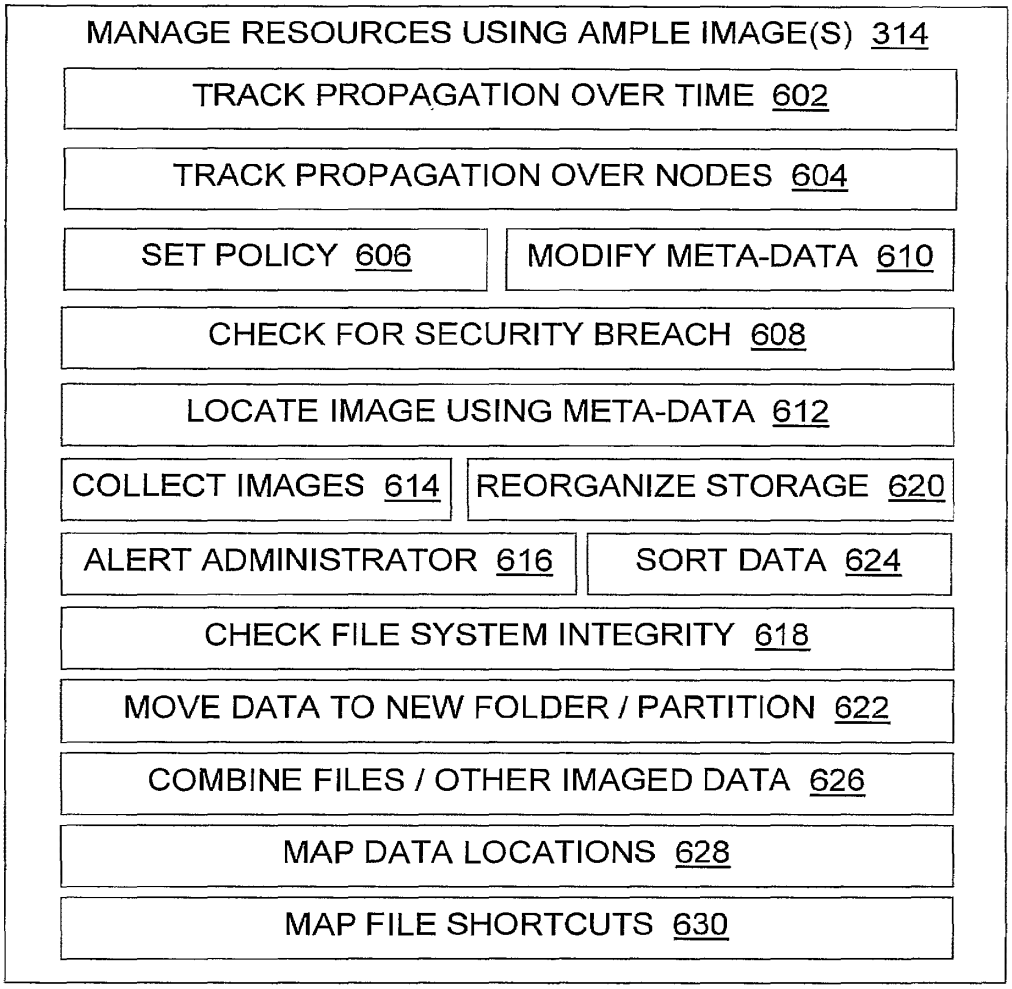

MANAGE RESOURCES USING AMPLE IMAGE(S) 314
TRACK PROPAGATION OVER TIME 602
TRACK PROPAGATION OVER NODES 604
SET POLICY 606
MODIFY META-DATA 610
CHECK FOR SECURITY BREACH 608
LOCATE IMAGE USING META-DATA 612
COLLECT IMAGES 614
REORGANIZE STORAGE 620
ALERT ADMINISTRATOR 616
SORT DATA 624
CHECK FILE SYSTEM INTEGRITY 618
MOVE DATA TO NEW FOLDER / PARTITION 622
COMBINE FILES / OTHER IMAGED DATA 626
MAP DATA LOCATIONS 628
MAP FILE SHORTCUTS 630

Fig. 6

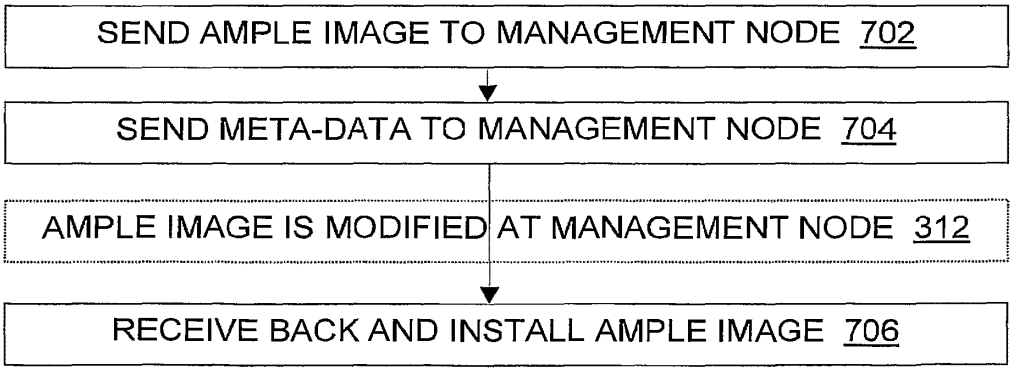

SEND AMPLE IMAGE TO MANAGEMENT NODE 702
SEND META-DATA TO MANAGEMENT NODE 704
AMPLE IMAGE IS MODIFIED AT MANAGEMENT NODE 312
RECEIVE BACK AND INSTALL AMPLE IMAGE 706

Fig. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/10197

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 12/00, 12/02
US CL : 711/112,173; 707/200; 709/218
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 711/112,173; 707/200; 709/218

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST, IEEE, Google

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,933,647 (ARONBERG et al) 03 August 1999 (03.08.1999), column 1, line 14 to column 9, line 43.	1-34
Y	US 6,067,410 (NACHENBERG) 23 May 2000 (23.05.2000), Abstract, column 1, line 14 to column 14, line 13).	1-34
Y	US 6,108,697 (RAYMOND et al) 22 August 2000 (22.08.2000), abstract, column 1, line 17 to column 16, line 16.	1-34
Y	US 6,253,300 B1 (LAWRENCE et al) 26 June 2001 (26.06.2001), column 1, line 21 to column 23, line15.	1-34
Y	US 6,330,653 B1 (MURRAY et al) 11 December 2001 (11.12.2001), abstract, column 1, line 35 to column 44, line 67.	1-34
Y	US 2001/0056425 A1 (RICHARD) 27 December 2001 (27.12.2001), [0002] - [0095].	1-34
Y	Symantec Enterprise Administration: Symantec Ghost 7.0, 2001. Entire Document.	1-34

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
25 June 2003 (25.06.2003)

Date of mailing of the international search report
15 JUL 2003

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
Facsimile No. (703)305-3230

Authorized officer
David A Wiley
Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

Electronic Acknowledgement Receipt

EFS ID:	1952576
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	28863
Filer:	Kent Joseph Sieffert/Nancy Burr
Filer Authorized By:	Kent Joseph Sieffert
Attorney Docket Number:	1072-021US01
Receipt Date:	09-JUL-2007
Filing Date:	30-MAR-2007
Time Stamp:	17:25:33
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes) /Message Digest	Multi Part /.zip	Pages (if appl.)
1	Foreign Reference	WO06083893.pdf	2271913 d2b1a54fc840608824afa6d2303918c09573a955	no	43
Warnings:					

Information:					
2	Information Disclosure Statement (IDS) Filed	IDS.pdf	88889 96412921e841dc258d2f788b6ef5124a2af59c9d	no	4
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Foreign Reference	WO06083894.pdf	3876040 28b7ac9d051fb90f248e5155ce5008aa65104a36	no	72
Warnings:					
Information:					
4	Foreign Reference	WO06083901.pdf	2804586 60fd4df65e29d65cde1f5fadb29f6e2aa94b7e8f	no	64
Warnings:					
Information:					
5	Foreign Reference	WO06083895.pdf	3097448 99971e25c0430c02e14b95d24a4f9b8092f12ac9	no	70
Warnings:					
Information:					
6	Foreign Reference	WO06083727.pdf	3595252 678a10af10f04c91e1acffdcc0d651f0a96c624f	no	79
Warnings:					
Information:					
7	Foreign Reference	WO06081503.pdf	2042042 729efbbb3a76a008a6c3d0183a911c540c29ff63	no	33
Warnings:					
Information:					
8	Foreign Reference	KleinschnitzW03085526.pdf	2523523 6be4a4e9ecc70967612bbb11bae21500ec3b0dcc	no	43
Warnings:					
Information:					
9	NPL Documents	Websphere.pdf	1009982 c9dad93d8183ff37be755b8ae30f3d1c03ae30c2	no	24
Warnings:					
Information:					

10	NPL Documents	Resource.pdf	664323 fe21cb9f01109b92d280df338b16c431b5546d29	no	17
Warnings:					
Information:					
11	NPL Documents	QoS-aware.pdf	265112 5d21cca3dbf5de8470c0dbbd1845fe97f07bde8d	no	6
Warnings:					
Information:					
12	NPL Documents	OEM.pdf	670875 b280165fda947e7f946c665cddf4e5ffd8016862	no	24
Warnings:					
Information:					
13	NPL Documents	Takes.pdf	313118 5fd7d68e1cf46b969c77e4c2e51ee30e5f041540	no	8
Warnings:					
Information:					
			Total Files Size (in bytes):	23223103	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 August 2006 (10.08.2006)

PCT

(10) International Publication Number
WO 2006/083893 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2006/003436

(22) International Filing Date: 31 January 2006 (31.01.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/047,468 31 January 2005 (31.01.2005) US

(71) Applicant *(for all designated States except US)*: **CASSATT CORPORATION** [US/US]; 1740 Technology Drive, San Jose, California 95110 (US).

(72) Inventors; and

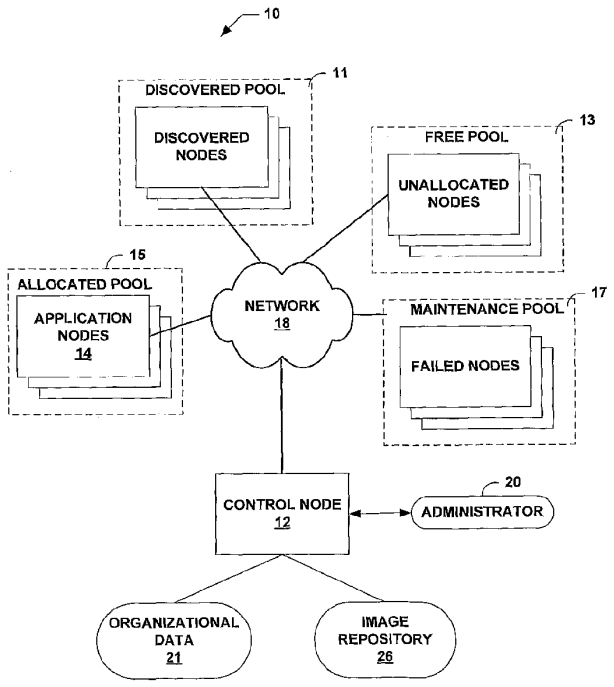
(75) Inventors/Applicants *(for US only)*: **ENGQUIST, James, D.** [US/US]; 7140 Oak Hills Lane, Colorado Springs, Colorado 80919 (US). **VOSBURGH, Craig, A.** [US/US]; 2255 Cape Pine Way, Colorado Springs, Colorado 80919 (US). **BERLINER, Brian** [US/US]; 754 The Alameda, Apt. 2320, Snaj Ose, California 95126 (US). **JACKSON, Jerry, R.** [US/US]; 11270 Wakely Road, Colorado Springs, Colorado 80908 (US). **LINDLEY, Craig, A.** [US/US]; 17925 Black Squirrel Road, Colorado Springs, Colorado 80908 (US). **COLLINS, Doreen, E.** [US/US]; 19175 Royal Troon, Monument, Colorado 80132 (US). **NORDBY, Jonathan, D.** [US/US]; 3010 Schilling Avenue, Monument, Colorado 80132 (US). **CHURCH, Dann, M.** [US/US]; 7119 Winthrop Ct., Castle Rock, Colorado 80104 (US). **RESCH, David, L.** [US/US]; 5310 Meadowgreen Drive, Colorado Springs, Colorado 80919 (US).

(74) Agent: **SIEFFERT, Kent, J.**; SHUMAKER & SIEFFERT, P.A., 8425 SEASONS PARKWAY, Suite 105, St. Paul, Minnesota 55125 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV,

[Continued on next page]

(54) Title: DISTRIBUTED COMPUTING SYSTEM HAVING HIERACHICAL ORGANIZATION



(57) Abstract: A distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model. The model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of distributed computing system. The control nodes are responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

WO 2006/083893 A1

Netflix, Inc. - Ex. 1002, Page 000633
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DISTRIBUTED COMPUTING SYSTEM HAVING HIERACHICAL ORGANIZATION

TECHNICAL FIELD

[0001] The invention relates to computing environments and, more specifically, to distributed computing systems.

BACKGROUND

[0002] Distributed computing systems are increasingly being utilized to support high-performance computing applications. Typically, distributed computing systems are constructed from a collection of computing nodes that combine to provide a set of processing services to implement the high performance computing applications. Each of the computing nodes in the distributed computing system is typically a separate, independent computing device interconnected with each of the other computing nodes via a communications medium, e.g., a network.

[0003] One challenge with distributed computing systems is the organization, deployment and administration of such a system within an enterprise environment. For example, it is often difficult to manage the allocation and deployment of enterprise computing functions within the distributed computing system. An enterprise, for example, often includes several business groups, and each group may have competing and variable computing requirements.

SUMMARY

[0004] In general, the invention is directed to a distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.

[0005] As described herein, the model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of distributed computing system. The control nodes are

Netflix, Inc. - Ex. 1002, Page 000635
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

[0006] In one embodiment, a method comprises receiving input that defines a model for a hierarchical organization of a distributed computing system having a plurality of computing nodes. The model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one node slot. The method further comprises automatically configuring the distributed computing system in accordance with the hierarchical organization defined by the model.

[0007] In another embodiment, a method comprises receiving input defining an organization model for a distributed computing system. The model includes a plurality of tiers that represent different computing functions, and defines a priority and computing node requirements for each of the tiers. The method further comprises assigning unallocated computing nodes to the tiers from a free pool based on the priority and computing requirements for each of the tiers.

[0008] In another embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network, and a database that stores a hierarchical model for the distributed computing system. The model specifies a fabric having one or more domains, and wherein each domain has at least one tier that includes one or more node slots. The distributed computing system further comprises a control node that automatically configures the application nodes in accordance with the hierarchical organization defined by the model.

[0009] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0010] FIG. 1 is a block diagram illustrating a distributed computing system constructed from a collection of computing nodes.

[0011] FIG. 2 is a schematic diagram illustrating an example of a model of an enterprise that logically defines an enterprise fabric.

[0012] FIG. 3 is a flow diagram that provides a high-level overview of the operation of A control node when configuring the distributed computing system.

Netflix, Inc. - Ex. 1002, Page 000636
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0013] FIG. 4 is a flow diagram illustrating exemplary operation of the control node when assigning computing nodes to node slots of tiers.

[0014] FIG. 5 is a flow diagram illustrating exemplary operation of a control node when adding an additional computing node to a tier to meet additional processing demands.

[0015] FIG. 6 is a flow diagram illustrating exemplary operation of a control node harvesting excess node capacity from one of the tiers and returning the harvested computing node to the free pool.

[0016] FIG. 7 is a screen illustration of an exemplary user interface for defining tiers in a particular domain.

[0017] FIG. 8 is a screen illustration of an exemplary user interface for defining properties of the tiers.

[0018] FIG. 9 is a screen illustration of an exemplary user interface for viewing and identify properties of a computing node.

[0019] FIG. 10 is a screen illustration of an exemplary user interface for viewing software images.

[0020] FIG. 11 is a screen illustration of an exemplary user interface for viewing a hardware inventory report.

[0021] FIG. 12 is a screen illustration of an exemplary user interface for viewing discovered nodes that are located in the free pool.

[0022] FIG. 13 is a screen illustration of an exemplary user interface for viewing users of distributed computing system.

[0023] FIG. 14 is a screen illustration of an exemplary user interface for viewing alerts for distributed computing system.

DETAILED DESCRIPTION

[0024] FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide high-performance processing.

[0025] In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated pool 15 and a maintenance pool 17. In addition, distributed computing system 10 includes at least one control node 12.

Netflix, Inc. - Ex. 1002, Page 000637
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0026] Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

[0027] Discovered pool 11 includes a set of discovered nodes that have been automatically "discovered" within distributed computing system 10 by control node 12. For example, control node 12 may sense dynamic host communication protocol (DHCP) requests or other communications to detect the connection of a discovered node to network 18. Once detected, control node 12 automatically inventories the attributes for the discovered node and reassigns the discovered node to free pool 13. The node attributes identified during the inventory process may include a CPU count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics or other computing resources. Control node 12 may also receive input identifying node attributes not detectable via the automatic inventory, such as whether the node includes I/O, such as HBA.

[0028] Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated pool 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to add an application node to allocated pool 15 to increase processing capacity of distributed computing system 10.

[0029] In general, allocated pool 15 includes application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory). Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated pool 15.

[0030] Control node 12 provides the system support functions for managing distributed computing system 10. More specifically, control node 12 manages the roles of each computing node within distributed computing system 10 and the execution of software

Netflix, Inc. - Ex. 1002, Page 000638
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

applications within the distributed computing system. In general, distributed computing system 10 includes at least one control node 12, but may utilize additional control nodes to assist with the management functions.

[0031] Other control nodes 12 (not shown in FIG. 1) are optional and may be associated with a different subset of the computing nodes within distributed computing system 10. Moreover, control node 12 may be replicated to provide primary and backup administration functions, thereby allowing for graceful handling a failover in the event control node 12 fails.

[0032] Network 18 provides a communications interconnect for control node 12 and application nodes 14, as well as discovered nodes, unallocated nodes and failed nodes. Communications network 18 permits internode communications among the computing nodes as the nodes perform interrelated operations and functions. Communications network 18 may comprise, for example, direct connections between one or more of the computing nodes, one or more customer networks maintained by an enterprise, local area networks (LANs), wide area networks (WANs) or a combination thereof. Communications network 18 may include a number of switches, routers, firewalls, load balancers, and the like.

[0033] In one embodiment, each of the computing nodes within distributed computing system 10 executes a common general-purpose operating system. One example of a general-purpose operating system is the Windows™ operating system provided by Microsoft Corporation. In some embodiments, the general-purpose operating system such as the Linux kernel.

[0034] In the example of FIG. 1, control node 12 is responsible for software image management. The term "software image" refers to a complete set of software loaded on an individual computing node, including the operating system and all boot code, middleware and application files. System administrator 20 may interact with control node 12 and identify the particular types of software images to be associated with application nodes 14. Alternatively, administration software executing on control node 12 may automatically identify the appropriate software images to be deployed to application nodes 14 based on the input received from system administrator 20. For example, control node 12 may determine the type of software image to load onto an application node 14 based on the functions assigned to the node by system administrator 20. Application nodes14 may be divided into a number of groups based on their assigned functionality. As one example, application nodes 14 may be divided into a first group to

Netflix, Inc. - Ex. 1002, Page 000639
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

provide web server functions, a second group to provide business application functions and a third group to provide database functions. The application nodes 14 of each group may be associated with different software images.

[0035] Control node 12 provides for the efficient allocation and management of the various software images within distributed computing system 10. In some embodiments, control node 12 generates a "golden image" for each type of software image that may be deployed on one or more of application nodes 14. As described herein, the term "golden image" refers to a reference copy of a complete software stack.

[0036] System administrator 20 may create a golden image by installing an operating system, middleware and software applications on a computing node and then making a complete copy of the installed software. In this manner, a golden image may be viewed as a "master copy" of the software image for a particular computing function. Control node 12 maintains a software image repository 26 that stores the golden images associated with distributed computing system 10.

[0037] Control node 12 may create a copy of a golden image, referred to as an "image instance," for each possible image instance that may be deployed within distributed computing system 10 for a similar computing function. In other words, control node 12 pre-generates a set of K image instances for a golden image, where K represents the maximum number of image instances for which distributed computing system 10 is configured for the particular type of computing function. For a given computing function, control node 12 may create the complete set of image instance even if not all of the image instances will be initially deployed. Control node 12 creates different sets of image instances for different computing functions, and each set may have a different number of image instances depending on the maximum number of image instances that may be deployed for each set. Control node 12 stores the image instances within software image repository 26. Each image instance represents a collection of bits that may be deployed on an application node.

[0038] Further details of software image management are described in co-pending U.S. patent application Ser. No. 11/046,133, Attorney Docket no. 1072-006US01, entitled "MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005 and co-pending U.S. patent application Ser. No. 11/046,152, Attorney Docket no. 1072-007US01, entitled "UPDATING SOFTWARE IMAGES ASSOCIATED WITH A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005.

Netflix, Inc. - Ex. 1002, Page 000640
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0039] In general, distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

[0040] As used herein, the "fabric" level generally refers to the logical constructs that allow for definition, deployment, partitioning and management of distinct enterprise applications. In other words, fabric refers to the integrated set of hardware, system software and application software that can be "knitted" together to form a complete enterprise system. In general, the fabric level consists of two elements: fabric components or fabric payload. Control node 12 provides fabric management and fabric services as described herein.

[0041] In contrast, a "domain" is a logical abstraction for containment and management within the fabric. The domain provides a logical unit of fabric allocation that enables the fabric to be partitioned amongst multiple uses, e.g. different business services.

[0042] Domains are comprised of tiers, such as a 4-tier application model (web server, application server, business logic, persistence layer) or a single tier monolithic application. Fabric domains contain the free pool of devices available for assignment to tiers.

[0043] A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. The tier structure within a domain is typically defined by a user, such as administrator 20. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

[0044] The lowest level, the node level, includes the physical components of the fabric. This includes computing nodes that, as described above, provide operating environments for system applications and enterprise software applications. In addition, the node level may include network devices (e.g., Ethernet switches, load balancers and firewalls) used in creating the infrastructure of network 18. The node level may further include network storage nodes that are network connected to the fabric.

[0045] System administrator 20 accesses administration software executing on control node 12 to logically define the hierarchical organization of distributed computing system

Netflix, Inc. - Ex. 1002, Page 000641
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

10. For example, system administrator 20 may provide organizational data 21 to develop a model for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.

[0046] More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation management (CRM) service. System administrator 20 further defines one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a web tier, an application tier and a database tier. In this manner, distributed computing system 10 may be configured to automatically provide web server functions, business application functions and database functions.

[0047] For each of the tiers, control node 12 creates a number of "node slot" equal to the maximum number of application nodes 14 that may be deployed. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot. The node slots may, for instance, identify a particular software image instance associated with an application node 14 as well as a network address associated with that particular image instance.

[0048] In this manner, each of the tiers include one or more node slots that references particular software images instances to boot on the application nodes 14 to which each software image instance is assigned. The application nodes 14 to which control node 12A assigns the image instances temporarily inherit the network address assigned to the image instance for as long as the image instance is deployed on that particular application node. If for some reason the image instance is moved to a different application node 14, control node 12A moves the network address that new application node.

[0049] System administrator 20 may further define specific node requirements for each tier of the fabric. For example, the node requirements specified by system administrator 20 may include a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics that may be detected on the individual computing nodes. System administrator 20 may also specify user-defined hardware attributes of the computing nodes, such as whether I/O (like HBA) is required. The user-defined hardware attributes are typically not capable of

Netflix, Inc. - Ex. 1002, Page 000642
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

detection during an automatic inventory. In this manner, system administrator 20 creates a list of attributes that the tier requires of its candidate computing nodes.

[0050] In addition to the node requirements described above, system administrator 20 may further define policies that are used when re-provisioning computing nodes within the fabric. System administrator 20 may define policies regarding tier characteristics, such as minimum number of nodes a tier requires, an indication of whether or not a failed node is dynamically replaced by a node from free pool 13, a priority for each tier relative to other tiers, an indication of whether or not a tier allows nodes to be re-provisioned to other tiers to satisfy processing requirements by other tiers of a higher priority or other policies. Control node 12 uses the policy information input by system administrator 20 to re-provision computing nodes to meet tier processing capacity demands.

[0051] After receiving input from system administrator 20 defining the architecture and policy of the enterprise fabric, control node 12 identifies unallocated nodes within free pool 13 that satisfy required node attributes. Control node 12 automatically assigns unallocated nodes from free pool 13 to respective tier node slots of a tier. As will be described in detail herein, in one embodiment, control node 12 may assign computing nodes to the tiers in a "best fit" fashion. Particularly, control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20. The assignment of the computing nodes may occur on a tier-by-tier basis beginning with a tier with the highest priority and ending with a tier with the lowest priority.

[0052] As will be described in detail below, control node 12 may automatically add unallocated nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to the free pool when the tier has excess capacity, transfer nodes from tier to tier to meet processing demands, or replace failed nodes with nodes from the free pool. Thus, computing resources, i.e., computing nodes, may be automatically shared between tiers and domains within the fabric based on user-defined policies to dynamically address high-processing demands, failures and other events.

[0053] FIG. 2 is a schematic diagram illustrating an example embodiment of organizational data 21 that defines a model logically representing an enterprise fabric in accordance with the invention. In the example illustrated in FIG. 2, control node 12 (FIG. 1) maintains organizational data 21 to define a simple e-commerce fabric 32.

Netflix, Inc. - Ex. 1002, Page 000643
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0054] In this example, e-commerce fabric 32 includes a storefront domain 34A and a financial planning domain 34B. Storefront domain 34A corresponds to the enterprise storefront domain and allows customers to find and purchase products over a network, such as the Internet. Financial planning domain 34B allows one or more employees to perform financial planning tasks for the enterprise.

[0055] Tier level 31C includes one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. For example, storefront domain 34A includes a web server tier (labeled "web tier") 36A, a business application tier (labeled "app tier") 36B, and a database tier (labeled "DB tier") 36C. Web server tier 36A, business application tier 36B and database tier 36C interact with one another to present a customer with an online storefront application and services. For example, the customer may interact with web server tier 36A via a web browser. When the customer searches for a product, web server tier 36A may interacts with business application tier 36B, which may in turn access a database tier 36C. Similarly, financial planning domain 34B includes a financial planning tier 36D that provides subcomponents of applications and services of the financial planning domain 34B. Thus, in this example, a domain may include a single tier.

[0056] Tier level 31D includes one or more logical node slots 38A-38H ("node slots 38") within each of the tiers. Each of node slots 38 include node specific information, such as software resources for an application node 14 that is assigned to a respective one of the node slots 38. Node slots 38 may, for instance, identify particular software image instances within image repository 26 and map the identified software image instances to respective application nodes 14. As an example, node slots 38A and 38B belonging to web server tier 36A may reference particular software images instance used to boot two application nodes 14 to provide web server functions. Similarly, the other node slots 38 may reference software image instances to provide business application functions, database functions, or financial application functions depending upon the tier to which the node slots are logically associated.

[0057] Although in the example of FIG. 2, there are two node slots 38 corresponding to each tier, the tiers may include any number of node slots depending on the processing capacity needed on the tier. Furthermore, not all of node slots 38 may be currently assigned to an application node 14. For example, node slot 28B may be associated with an inactive software image instance and, when needed, may be assigned to an application node 14 for deployment of the software image instance.

Netflix, Inc. - Ex. 1002, Page 000644
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0058] In this example, organizational data 21 associates free node pool 13 with the highest-level of the model, i.e., e-commerce fabric 32. As described above, control node 12 may automatically assign unallocated nodes from free node pool 13 to at least a portion of tier node slots 38 of tiers 36 as needed using the "best fit" algorithm described above or other algorithm. Additionally, control node 12 may also add nodes from free pool 13 to a tier when more processing capacity is needed within the tier, removes nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.

[0059] Although not illustrated, the model for the enterprise fabric may include multiple free node pools. For example, the model may associate free node pools with individual domains at the domain level or with individual tier levels. In this manner, administrator 20 may define policies for the model such that unallocated computing nodes of free node pools associated with domains or tiers may only be used within the domain or tier to which they are assigned. In this manner, a portion of the computing nodes may be shared between domains of the entire fabric while other computing nodes may be restricted to particular domains or tiers.

[0060] FIG. 3 is a flow diagram that provides a high-level overview of the operation of control node 12 when configuring distributed computing system 10. Initially, control node 12 receives input from a system administrator defining the hierarchical organization of distributed computing system 10 (50). In one example, control node 12 receives input that defines a model that specifies number of hierarchically arranged nodes as described in detail in FIG. 2. Particularly, the defined architecture of distributed computing system 10 includes an overall fabric having a number of hierarchically arranged domains, tiers and node slots.

[0061] During this process, control node 12 may receive input specifying node requirements of each of the tiers of the hierarchical model (52). As described above, administrator may specify a list of attributes, e.g., a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), or local disk characteristics, that the tiers require of their candidate computing nodes. In addition, control node 12 may further receive user-defined custom attributes, such as requiring the node to have I/O, such as HBA connectivity. The node requirements or attributes defined by system administrator 20 may each include a name used to identify the characteristic, a data type (e.g., an integer, long, float or string), and a weight to define the importance of the requirement.

Netflix, Inc. - Ex. 1002, Page 000645
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0062] Control node 12 identifies the attributes for all candidate computing nodes within free pool 13 (54). As described above, control node 12 may have already discovered the computing nodes and inventoried the candidate computing nodes to identify hardware characteristics of all candidate computing nodes. Additionally, control node 12 may receive input from system administrator 20 identifying specialized capabilities of one or more computing nodes that are not detectable by the inventory process.

[0063] Control node 12 dynamically assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a "best fit" algorithm.

[0064] FIG. 4 is a flow diagram illustrating exemplary operation of control node 12 when assigning computing nodes to node slots of tiers. Initially, control node 12 selects a tier to enable (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

[0065] Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database that having entries for each node slot, where the entries identify the node requirements for each of the tiers. Control node 12 retrieves the node requirements for the selected tier from the database.

[0066] In addition, control node 12 accesses the database and retrieves the computing node attributes of one of the unallocated computing nodes of free pool 13. Control node 12 compares the node requirements of the tier to the node attributes of the selected computing node (64).

[0067] Based on the comparison, control node 12 determines whether the node attributes of the computing node meets the minimum node requirements of the tier (66). If the node attributes of the selected computing node do not meet the minimum node requirements of the tier, then the computing node is removed from the list of candidate nodes for this particular tier (68). Control node 12 repeats the process by retrieving the node attributes

Netflix, Inc. - Ex. 1002, Page 000646
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

of another of the computing nodes of the free pool and compares the node requirements of the tier to the node attributes of the computing node.

[0068] If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the node requirements of the tier (70). If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).

[0069] Control node 12 then determines whether the node count for the tier is met (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20. When the node count for the tier is met, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority, and repeats the process until all defined tiers are enabled, i.e., populated with application nodes (60).

[0070] If the node attributes of the selected computing node and the node requirements of the tier are not a perfect match control node 12 calculates and records a "processing energy" of the node (76). As used herein, the term "processing energy" refers to a numerical representation of the difference between the node attributes of a selected node and the node requirements of the tier. A positive processing energy indicates the node attributes more than satisfy the node requirements of the tier. The magnitude of the processing energy represents the degree to which the node requirements exceed the tier requirements.

[0071] After computing and recording the processing energy of the nodes, control node 12 determines whether there are more candidate nodes in free pool 13 (78). If there are additional candidate nodes, control node 12 repeats the process by retrieving the computing node attributes of another one of the computing nodes of the free pool of computing nodes and comparing the node requirements of the tier to the node attributes of the computing node (64).

[0072] When all of the candidate computing nodes in the free pool have been examined, control node 12 selects the candidate computing node having the minimum positive processing energy and assigns the selected computing node to a node slot of the tier (80).

Netflix, Inc. - Ex. 1002, Page 000647
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Control node 12 determines whether the minimum node count for the tier is met (82). If the minimum node count for the tier has not been met, control node 12 assigns the computing node with the next lowest calculated processing energy to the tier (80). Control node 12 repeats this process until the node count is met. At this point, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority (60).

[0073] In the event there are an insufficient number of computing nodes in free pool 13, or an insufficient number of computing nodes that meet the tier requirements, control node 12 notifies system administrator 20. System administrator 20 may add more nodes to free pool 13, add more capable nodes to the free pool, reduce the node requirements of the tier so more of the unallocated nodes meet the requirements, or reduce the configured minimum node counts for the tiers.

[0074] FIG. 5 is a flow diagram illustrating exemplary operation of control node 12 when adding an additional computing node to a tier to meet increased processing demands. Initially, control node 12 or system administrator 20 identifies a need for additional processing capacity on one of the tiers (90). Control node 12 may, for example, identify a high processing load on the tier or receive input from a system administrator identifying the need for additional processing capacity on the tier.

[0075] Control node 12 then determines whether there are any computing nodes in the free pool of nodes that meet the minimum node requirements of the tier (92). When there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and assigns the node to the tier (95). As described in detail with respect to FIG. 4, control node 12 may determine whether there are any nodes that have node attributes that are an exact match to the node requirements of the tier. If an exact match is found, the corresponding computing node is assigned to a node slot of the tier. If no exact match is found, control node 12 computes the processing energy for each node and assigns the computing node with the minimum processing energy to the tier. Control node 12 remotely powers on the assigned node and remotely boots the node with the image instance associated with the node slot. Additionally, the booted computing node inherits the network address associated with the node slot.

[0076] If there are no adequate computing nodes in the free pool, i.e., no nodes at all or no nodes that match the minimal node requirements of the tier, control node 12 identifies the tiers with a lower priority than the tier needing more processing capacity (96).

Netflix, Inc. - Ex. 1002, Page 000648
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0077] Control node 12 determines which of the nodes of the lower priority tiers meet the minimum requirements of the tier in need of processing capacity (98). Control node 12 may, for example, compare the attributes of each of the nodes assigned to node slots of the lower priority tiers to the node requirements of the tier in need of processing capacity. Lower priority nodes that have the minimum number of computing nodes may be removed from possible tiers from which to harvest an application node. If, however, all the lower priority tiers have the minimum number of computing nodes defined for the respective tier, the lowest priority tier is selected from which to harvest the one or more nodes.

[0078] Control node 12 calculates the processing energy of each of the nodes of the lower priority tiers that meet the minimum requirements (100). The energies of the nodes are calculated using the differences between the node attributes and the node requirements of the tier needing additional capacity. Control node 12 selects the computing node with the lowest processing energy that meets the minimum requirements, and assigns the selected computing node to the tier in need of processing capacity (102, 95).

[0079] FIG. 6 is a flow diagram illustrating exemplary operation of control node 12 when harvesting excess node capacity from one of the tiers and returning the harvested computing node to free pool 13. Initially, control node 12 identifies a tier having excess node capacity (110). Control node 12 may, for example, periodically check the node capacity of the tiers to identify any tiers having excess node capacity. Performing a periodic check and removal of excess nodes increases the likelihood that a capable computing node will be in free pool 13 in the event one of the tiers needs additional node capacity.

[0080] When harvesting a node, control node 12 calculates the processing energy of all the nodes in the tier as described above with reference to FIG. 4 (112). Control node 12 identifies the node within the tier with the highest processing energy and returns the identified node to the free pool of nodes (114, 116). As described above, the node with the highest processing energy corresponds to the node whose node attributes are the most in excess the node requirements of the tier.

[0081] Returning the node to the free pool may involve remotely powering off the computing node and updating the database to associate the harvested node with free pool 13. In addition, control node 12 updates the database to disassociate the returned node with the node slot to which it was assigned. At this point, the node no longer uses the network address associated with the image instance mapped to the node slot. Control

Netflix, Inc. - Ex. 1002, Page 000649
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node 12 may, therefore, assign a temporary network address to the node while the node is assigned to free pool 13.

[0082] FIG. 7 is a screen illustration of an exemplary user interface 120 presented by control node 12 with which administrator 20 interacts to define tiers for a particular domain. In the example illustrated in FIG. 7, system administrator 20 has selected the "Collage Domain." User interface 120 presents the tiers that are currently in the selected domain. In the example illustrated, the Collage Domain includes three tiers, "test tier 1," "test tier 2," and "test tier 3." As shown in FIG. 7, in this example, each of the tiers includes two nodes. In addition, user interface 120 lists the type of software image currently deployed to application nodes for each of the tiers. In the example illustrated, image "applone (1.0.0)" is deployed to the nodes of test tier 1 and image "appltwo (1.0.0)" is deployed to the nodes of test tier 2. System administrator 20 may add one or more tiers to the domain by clicking on new tier button 122.

[0083] FIG. 8 is a screen illustration of an exemplary user interface 130 for defining properties of the tiers. In particular, user interface 130 allows system administrator 20 to input a name for the tier, a description of the tier, and an image associated with the tier. The image associated with the tier refers to a master image from which image instances are generated and deployed to the nodes assigned to the tier.

[0084] When configuring a tier, system administrator 20 may elect to activate email alerts. For example, system administrator 20 may activate the email alerts feature in order to receive email alerts providing system administrator 20 with critical and/or non-critical tier information, such as a notification that a tier has been upgraded, a node of the tier has failed or the like. Furthermore, system administrator 20 may input various policies, such node failure rules. For example, system administrator 20 may identify whether control node 12 should reboot a node in case of failure or whether the failed node should automatically be moved to maintenance pool 17. Similarly, system administrator 20 may identify whether nodes assigned to the tier may be harvested by other tiers.

[0085] User interface 130 may also allow system administrator 20 to input node requirements of a tier. In order to input node requirements of a tier, system administrator 20 may click on the "Requirements" tab 132, causing user interface 130 to present an input area to particular node requirements of the tier.

[0086] FIG. 9 is a screen illustration of an exemplary user interface 140 for viewing and identify properties of a computing node. User interface 140 allows system administrator 20 to define a name, description, and location (including a rack and slot) of a computing

Netflix, Inc. - Ex. 1002, Page 000650
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node. In addition the user may specify user-defined properties of a node, such as whether the computing node has I/O HBA capabilities.

[0087] User interface 140 also displays properties that control node 12 has identified during the computing node inventory process. In this example, user interface 140 presents system administrator 20 with the a CPU node count, a CPU speed, the amount of RAM, the disk size and other characteristics that are identifiable during the automated node inventory. User interface 140 additionally presents interface information to system administrator 20. Specifically, user interface 140 provides system administrator 20 with a list of components and their associated IP and MAC addresses.

[0088] User interface 140 also allows system administrator 20 to define other custom requirements. For example, system administrator 20 may define one or more attributes and add those attributes to the list of node attributes presented to system administrator 20.

[0089] FIG. 10 is a screen illustration of an exemplary user interface 150 for viewing software images. User interface 150 presents to a system administrator or other user a list of images maintained by control node 12 within image repository 26. The image list further includes the status of each image (i.e., either active or inactive), the version of the image, the operating system on which the image should be run, the operating system version on which the image should be run and a brief description of the image.

[0090] System administrator 20 or other user may select an image by clicking on the box in front of the image identifier/name and perform one or more actions on the image. Actions that system administrator 20 may perform on an image include deleting the image, updating the image, and the like. System administrator 20 may select one of the image actions via dropdown menu 152. In some embodiments, user interface 150 may further display other details about the images such as the node to which the images are assigned (if the node status is "active"), the network address associated with the images and the like.

[0091] FIG. 11 is a screen illustration of an exemplary user interface 160 for viewing a hardware inventory report. User interface 160 presents to system administrator 20 or other user a list of the nodes that are currently assigned to a domain. System administrator 20 may elect to view the nodes for the entire domain, for a single tier within the domain or for a single rack within a tier.

[0092] For each node, user interface 160 presents a node ID, a status of the node, the tier to which the node belongs, a hostname associated with the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the

Netflix, Inc. - Ex. 1002, Page 000651
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

node, number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0093] System administrator 20 or other user may select a node by clicking on the box in front of the node identifier/name and perform one or more actions on the node. Actions that system administrator 20 may perform on the node include deleting the node, updating the node attributes or other properties of the node, and the like. System administrator 20 may select one of the node actions via dropdown menu 162.

[0094] FIG. 12 is a screen illustration of an exemplary user interface 170 for viewing discovered nodes that are located in free pool 13. For each node, user interface 170 presents a node ID, a state of the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0095] FIG. 13 is a screen illustration of an exemplary user interface 180 for viewing users of distributed computing system 10. User interface 180 presents a list of users as well as the role assigned to each of the users and the status of each of the users. Thus, system administrator 20 may define different roles to each of the users. For example, a user may be either an operator (i.e., general user) or an administrator. System administrator 20 may add a new user to the list of users by clicking on the "New User" button 182.

[0096] FIG. 14 is a screen illustration of an exemplary user interface 190 for viewing alerts for distributed computing system 10. For each of the alerts, user interface 190 identifies the severity of the alert, whether the alert has been acknowledged, an object associated with the alert, and event associated with the alert, a state of the alert, a user associated with the alert and a date associated with the alert.

[0097] System administrator 20 or other user may select an alert by clicking on the box in front of the logged alert and perform one or more actions on the logged alert. Actions that system administrator 20 may perform include deleting the alert, changing the status of the alert, or the like. System administrator 20 may specify the log actions via dropdown menu 192.

[0098] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.

Netflix, Inc. - Ex. 1002, Page 000652
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS:

1. A method comprising:

receiving input that defines a model for a hierarchical organization of a distributed computing system having a plurality of computing nodes, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one node slot; and

automatically configuring the distributed computing system in accordance with the hierarchical organization defined by the model.

2. The method of claim 1,

wherein the model defines a different computing function for each of the tiers, and

wherein automatically configuring the distributed computing system comprises:

mapping software images associated with the computing functions to the node slots in accordance with model;

assigning the computing nodes to the node slots of the tiers; and

deploying the software images on assigned computing nodes.

3. The method of claim 1, further comprising:

receiving input that defines specific node requirements of each of the tiers of the fabric; and

assigning the computing nodes to the node slots in accordance with the node requirements.

4. The method of claim 3, wherein the node requirements include at least one of a central processing unit (CPU) count, a CPU speed, an amount of memory or local disk characteristics.

5. The method of claim 3, further comprising assigning the computing nodes having attributes that most closely match the node requirements defined for the tiers.

Netflix, Inc. - Ex. 1002, Page 000653
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

6. The method of claim 3, wherein assigning the computing nodes comprises:

selecting one of the tiers;

identifying the node requirements of the selected tier;

retrieving from a database computing node attributes associated with an unallocated set of the computing nodes; and

comparing the node attributes of the unallocated set of computing nodes to the node requirements of the selected tier.

7. The method of claim 6, further comprising:

determining whether the node attributes for each of the unallocated computing nodes meet a set of minimum node requirements of the selected tier; and

removing the unallocated computing nodes from a list of candidate computing nodes for the selected tier when the node attributes do not meet the minimum node requirements of the selected tier.

8. The method of claim 6, further comprising assigning the unallocated computing nodes to the node slots of the tier when the node attributes of the unallocated computing nodes and the node requirements of the selected tier are an exact match.

9. The method of claim 6, further comprising:

calculating a processing energy for each of the unallocated computing nodes based on differences between the node attributes of the unallocated computing nodes and the node requirements of the selected tier; and

recording the processing energy of the unallocated computing node.

10. The method of claim 9, further comprising:

calculating the processing energy of other unallocated computing nodes;

selecting the unallocated computing node having a lowest positive processing energy; and

assigning the selected unallocated computing node to a node slot of the selected tier.

Netflix, Inc. - Ex. 1002, Page 000654
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

11. The method of claim 1, further comprising:

identifying a need for additional processing capacity on one of the tiers; and

assigning an additional computing node to the tier.

12. The method of claim 11, wherein assigning an additional computing node comprises:

receiving input that defines specific node requirements of each of the tiers of the fabric;

selecting an unallocated computing node from a free pool based the node requirements associated with the tier in need of the additional processing capacity; and

assigning the selected unallocated computing node to the tier in need of processing capacity.

13. The method of claim 1, further comprising:

receiving input that defines a re-provisioning policy for one or more of the tiers of the fabric;

identifying a need for additional processing capacity on one of the tiers; and

transferring a node from one of the other tiers to the tier in need of additional processing capacity according to the re-provisioning policy.

14. The method of claim 13, wherein the tiers are assigned priority values, and wherein transferring a node comprises:

identifying the tiers with a priority lower than the tier needing the additional processing capacity;

determining which of the computing nodes associated with the lower priority tiers satisfy requirements of the tier in need of processing capacity;

calculating a processing energy of each of the nodes of the lower priority tiers that satisfy the requirements; and

assigning the computing node with the highest processing energy to the tier in need of processing capacity

15. The method of claim 1, wherein each of the domains corresponds to one of an enterprise application or an enterprise service.

Netflix, Inc. - Ex. 1002, Page 000655
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

16. The method of claim 1, wherein the node slots represent a data set that describes specific information for a corresponding node.

17. A method comprising:

receiving input defining an organization model for a distributed computing system, wherein the model includes a plurality of tiers that represent different computing functions, and wherein the model defines a priority and computing node requirements for each of the tiers; and

assigning unallocated computing nodes to the tiers from a free pool based on the priority and computing requirements for each of the tiers.

18. The method of claim 17, wherein assigning unallocated computing nodes further comprises:

selecting one of the tiers based on the priorities of the tiers;

identifying the node requirements associated with the selected tier;

selecting one of the unallocated computing nodes from the free pool;

retrieving computing node attributes associated with the selected unallocated computing node;

comparing the node attributes of the selected unallocated computing node to the node requirements associated with the selected tier; and

assigning the selecting unallocated computing node to the selected tier based on the comparison.

19. The method of claim 18, further comprising:

determining whether the node attributes of the selected unallocated computing node meets a minimum node requirements of the selected tier; and

removing the selected unallocated node from a list of candidate nodes when the node attributes do not meet the minimum node requirements of the selected tier.

20. The method of claim 18, further comprising assigning the selected unallocated computing node to a node slot of the tier when the node attributes of the selected unallocated computing node match the node requirements of the selected tier.

Netflix, Inc. - Ex. 1002, Page 000656
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

21. The method of claim 17, wherein assigning unallocated computing nodes:

calculating a processing energy for each of the unallocated computing node based on differences between node attributes of the unallocated computing nodes and the node requirements of the selected tier; and

assigning the unallocated computing nodes to tier based on the calculated processing energy for each of the unallocated computing nodes.

22. A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network;

a database that stores a hierarchical model for the distributed computing system, wherein the model specifies a fabric having one or more domains, and wherein each domain has at least one tier that includes one or more node slots; and

a control node that automatically configures the application nodes in accordance with the hierarchical organization defined by the model.

23. The distributed computing system of claim 22, wherein the model defines policies that are used by the control node to assign the application nodes to the node slots of the tiers and to re-provision the application nodes between the tiers.

24. The distributed computing system of claim 22, further comprising an image repository storing master software images and software image instances generated from the master software images,

wherein the control node deploys the software images instances to the application nodes by assigning the software images to the node slots in accordance with the model.

25. The distributed computing system of claim 24,

wherein each tier represents a different computing function and is associated with a different master software image, and

wherein each node slot identifies one of the software image instance associated and a network address to be assigned to the respective application node.

Netflix, Inc. - Ex. 1002, Page 000657
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

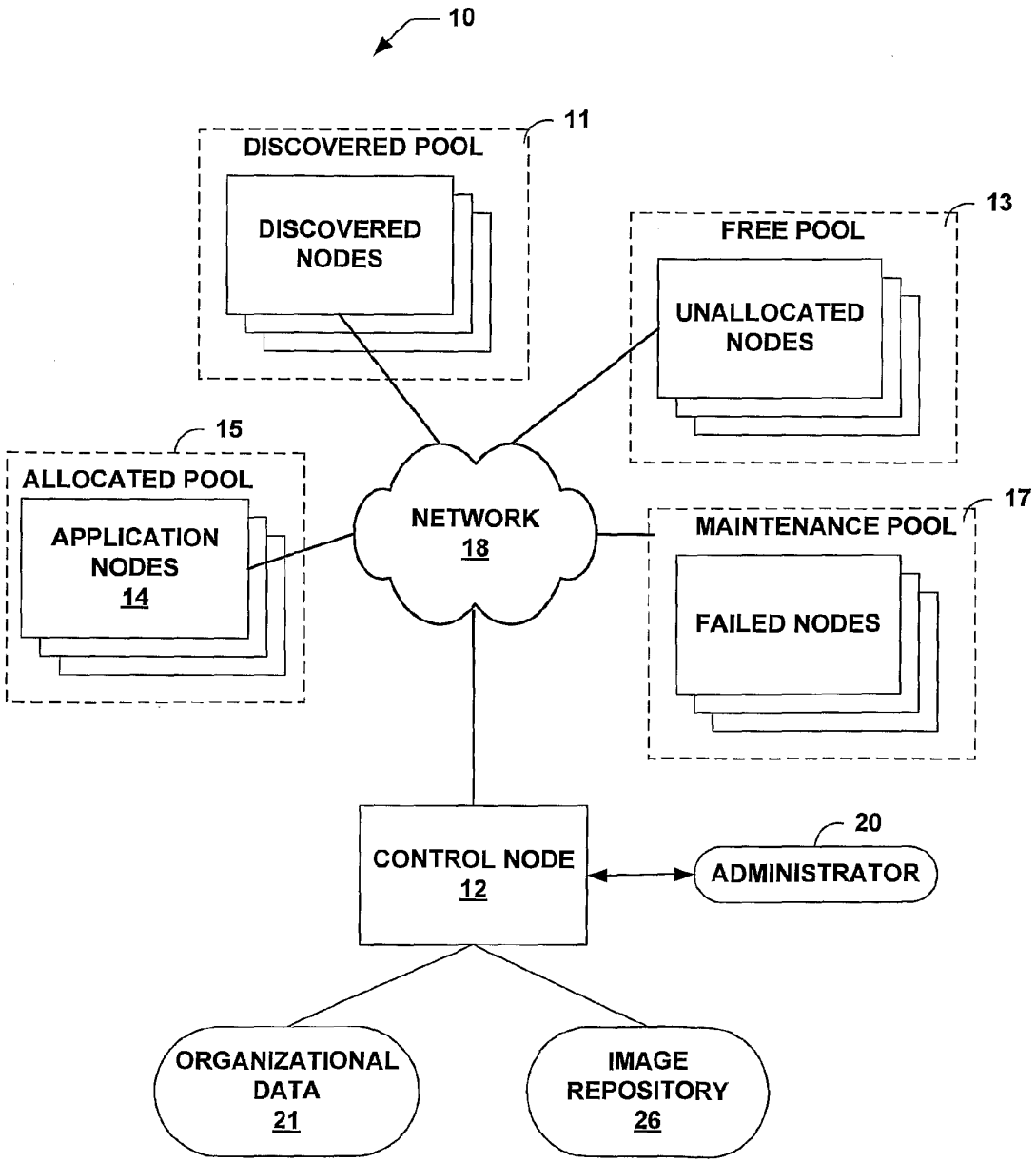

10
11
DISCOVERED POOL
DISCOVERED NODES
13
FREE POOL
UNALLOCATED NODES
15
ALLOCATED POOL
APPLICATION NODES 14
NETWORK 18
17
MAINTENANCE POOL
FAILED NODES
CONTROL NODE 12
20
ADMINISTRATOR
ORGANIZATIONAL DATA 21
IMAGE REPOSITORY 26

FIG. 1

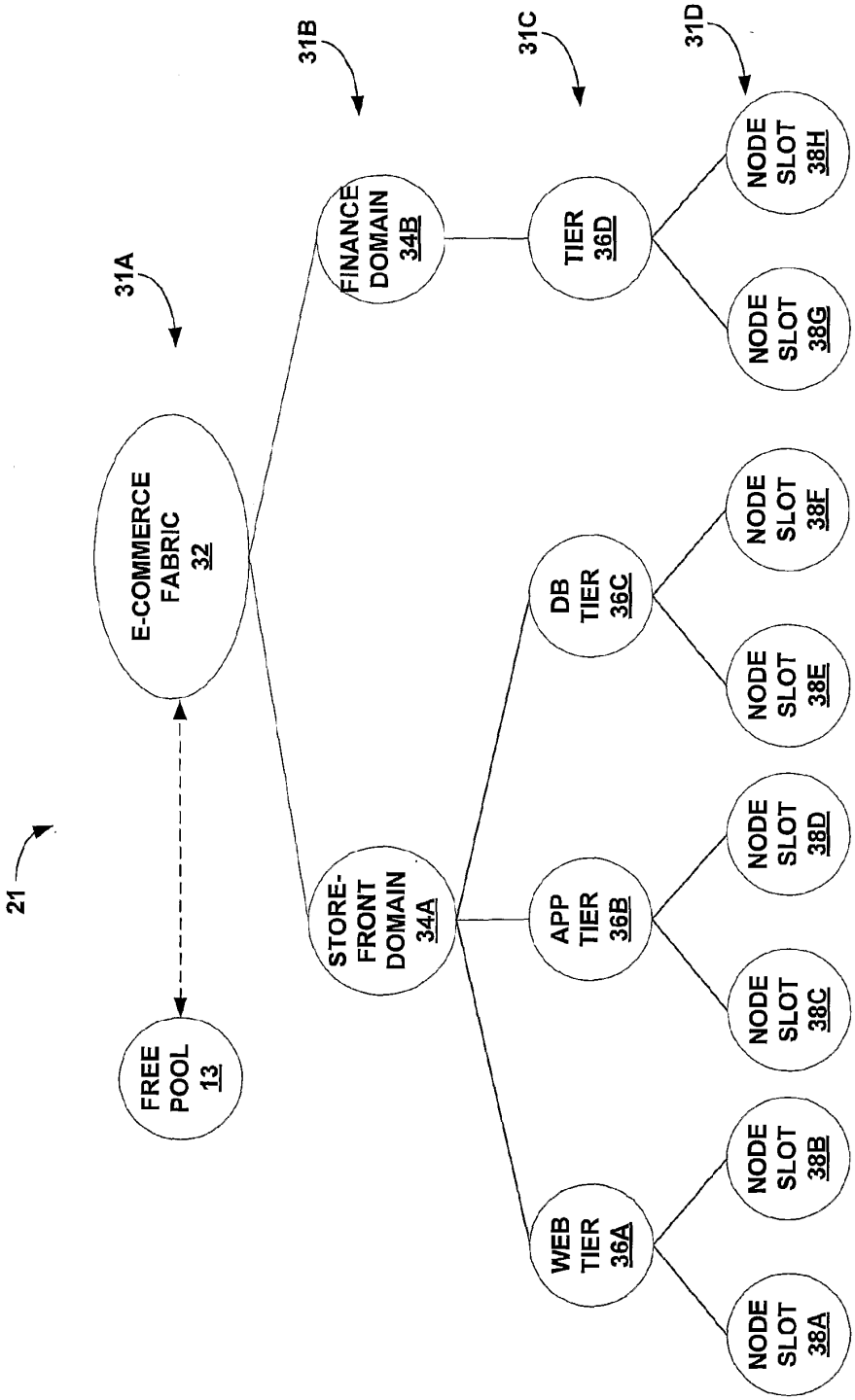

21
E-COMMERCE FABRIC 32
31A
FREE POOL 13
STORE-FRONT DOMAIN 34A
FINANCE DOMAIN 34B
31B
WEB TIER 36A
APP TIER 36B
DB TIER 36C
TIER 36D
31C
31D
NODE SLOT 38A
NODE SLOT 38B
NODE SLOT 38C
NODE SLOT 38D
NODE SLOT 38E
NODE SLOT 38F
NODE SLOT 38G
NODE SLOT 38H

FIG. 2

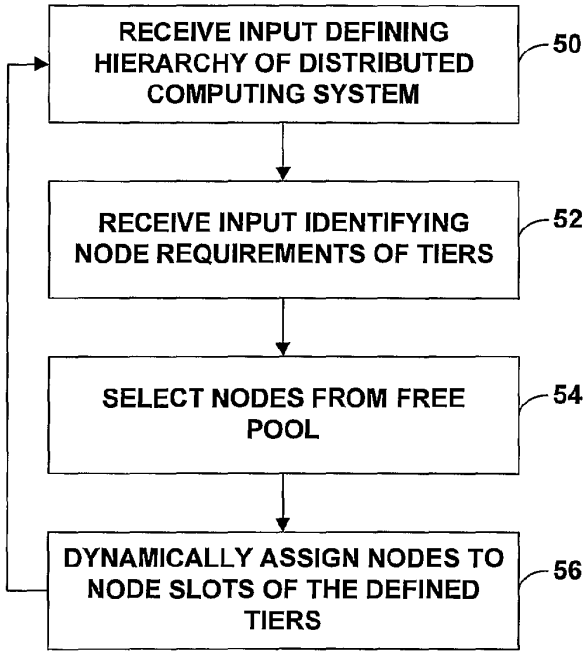

RECEIVE INPUT DEFINING HIERARCHY OF DISTRIBUTED COMPUTING SYSTEM
50
RECEIVE INPUT IDENTIFYING NODE REQUIREMENTS OF TIERS
52
SELECT NODES FROM FREE POOL
54
DYNAMICALLY ASSIGN NODES TO NODE SLOTS OF THE DEFINED TIERS
56

FIG. 3

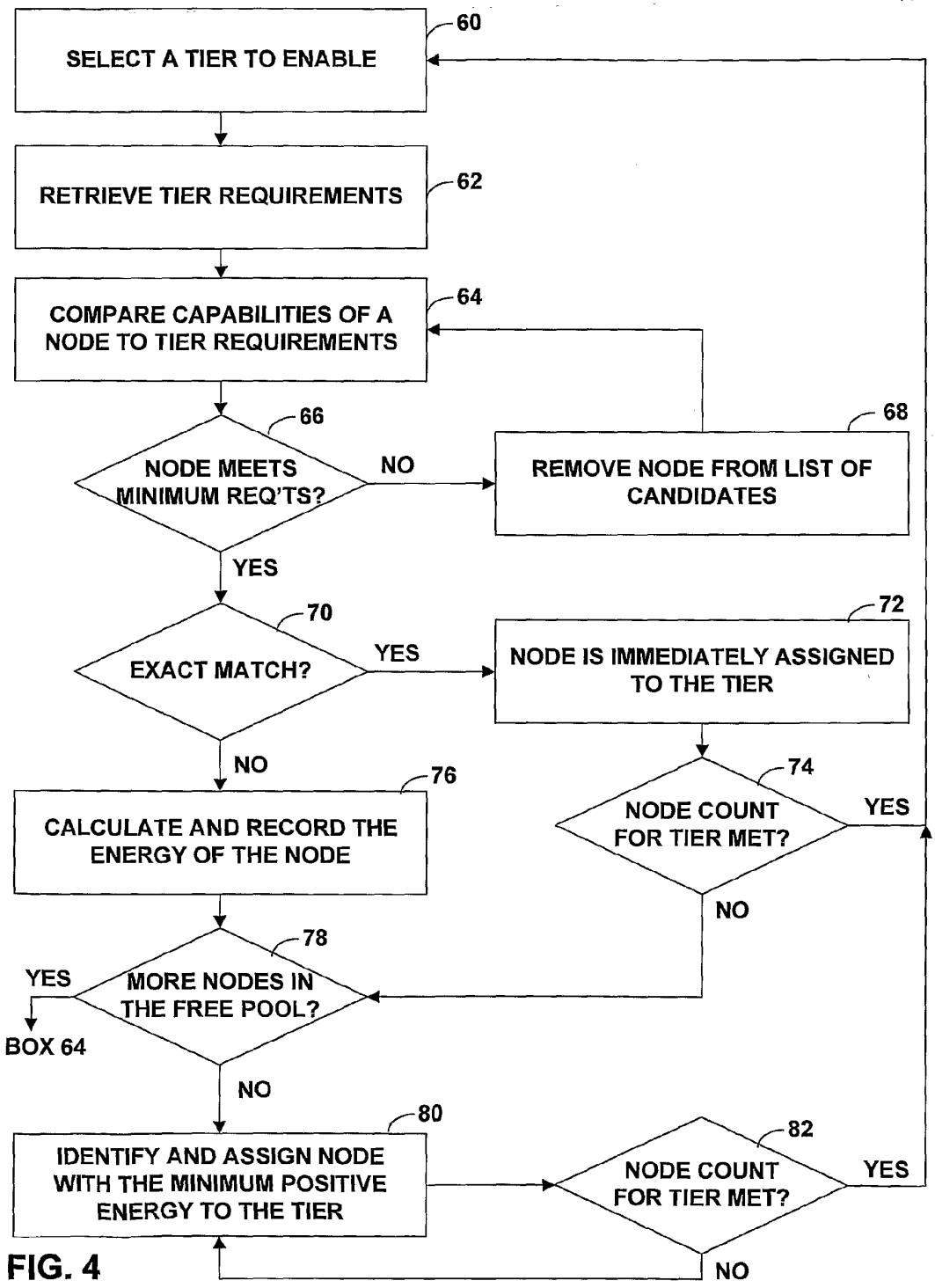

SELECT A TIER TO ENABLE
60
RETRIEVE TIER REQUIREMENTS
62
COMPARE CAPABILITIES OF A NODE TO TIER REQUIREMENTS
64
NODE MEETS MINIMUM REQ'TS?
66
NO
REMOVE NODE FROM LIST OF CANDIDATES
68
YES
EXACT MATCH?
70
YES
NODE IS IMMEDIATELY ASSIGNED TO THE TIER
72
NO
CALCULATE AND RECORD THE ENERGY OF THE NODE
76
NODE COUNT FOR TIER MET?
74
YES
NO
MORE NODES IN THE FREE POOL?
78
YES
BOX 64
NO
IDENTIFY AND ASSIGN NODE WITH THE MINIMUM POSITIVE ENERGY TO THE TIER
80
NODE COUNT FOR TIER MET?
82
YES
NO

FIG. 4

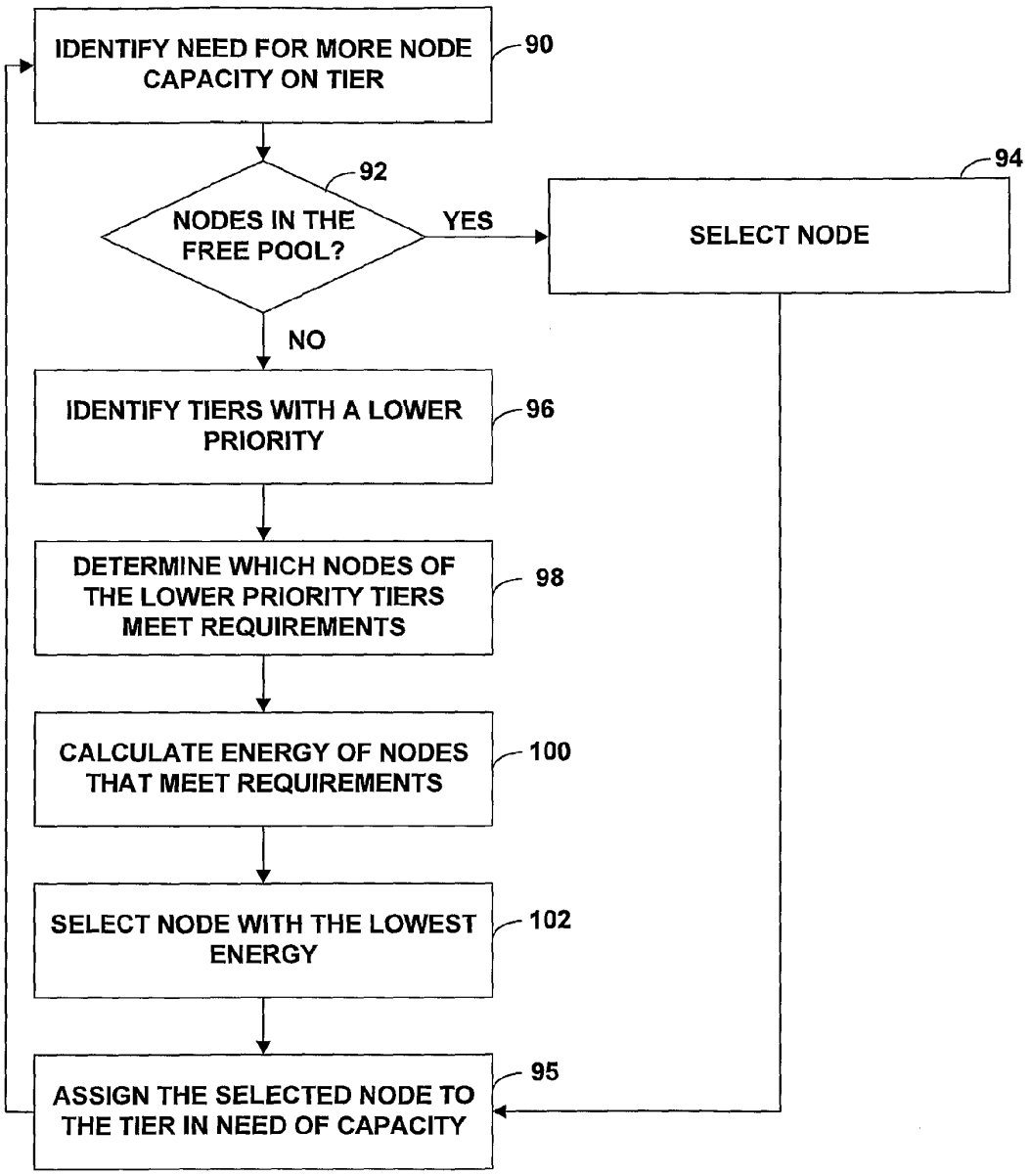

IDENTIFY NEED FOR MORE NODE CAPACITY ON TIER
90
NODES IN THE FREE POOL?
92
YES
SELECT NODE
94
NO
IDENTIFY TIERS WITH A LOWER PRIORITY
96
DETERMINE WHICH NODES OF THE LOWER PRIORITY TIERS MEET REQUIREMENTS
98
CALCULATE ENERGY OF NODES THAT MEET REQUIREMENTS
100
SELECT NODE WITH THE LOWEST ENERGY
102
ASSIGN THE SELECTED NODE TO THE TIER IN NEED OF CAPACITY
95

FIG. 5

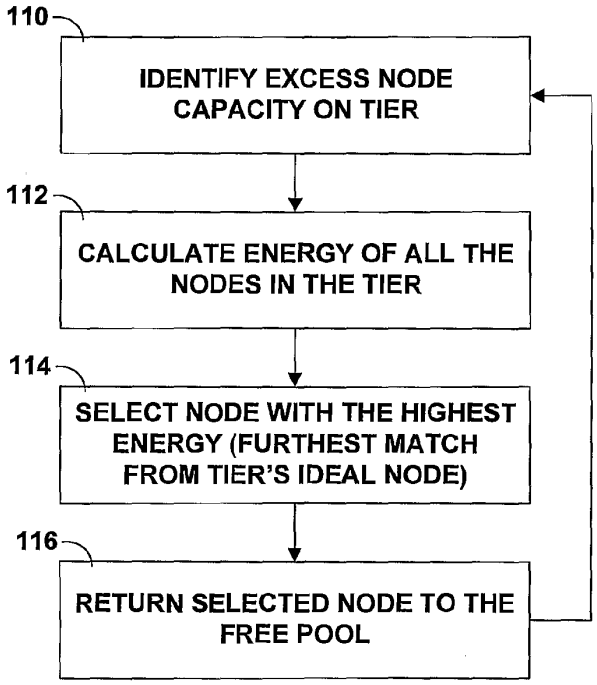


FIG. 6

Netflix, Inc. - Ex. 1002, Page 000663
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

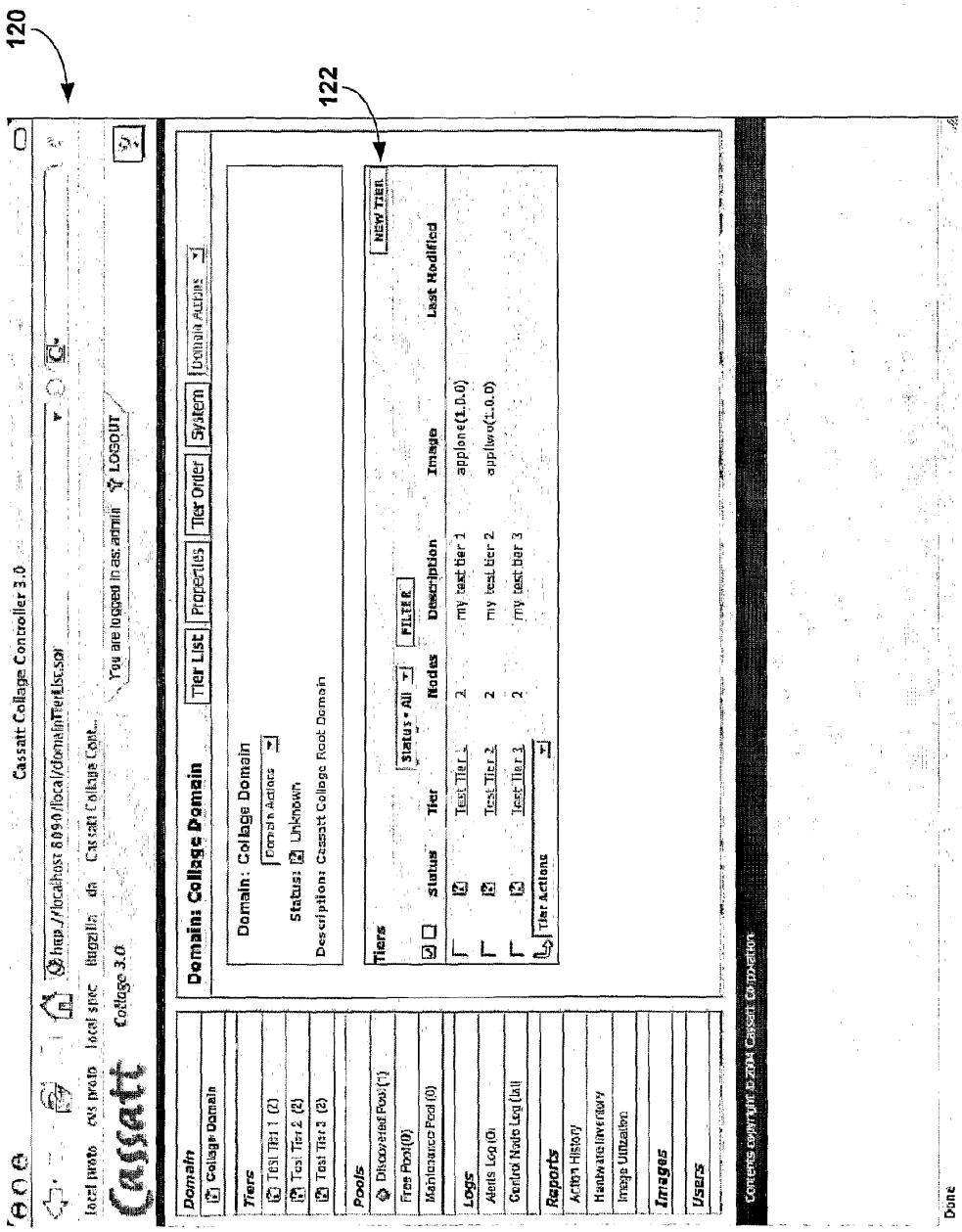

120
122
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (all
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Domain: Collage Domain
Tier List
Properties
Tier Order
System
Domain Actions
Domain: Collage Domain
Domain Actions
Status: Unknown
Description: Cassatt College Root Domain
Tiers
Status = All
FILTER
NEW TIER
Status
Tier
Nodes
Description
Image
Last Modified
Test Tier 1
2
my test tier 1
appone(1.0.0)
Test Tier 2
2
my test tier 2
apptwo(1.0.0)
Test Tier 3
2
my test tier 3
Tier Actions
Done

FIG. 7

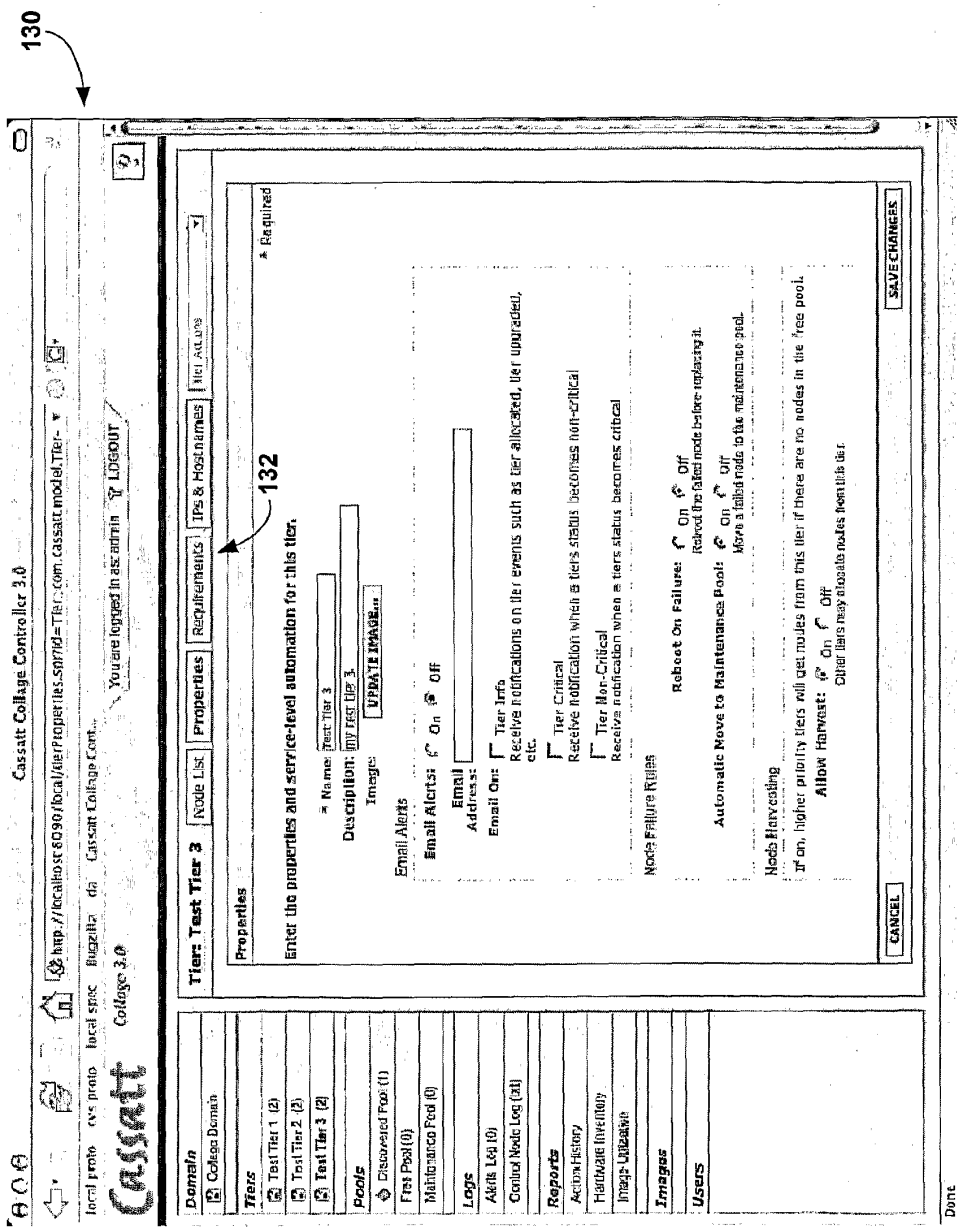

130
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
Tier: Test Tier 3
Node List
Properties
Requirements
IPs & Hostnames
Hot Actions
Properties
* Required
Enter the properties and service-level automation for this tier.
132
* Name:
Description:
Image:
Email Alerts
Email Alerts:
Email Addresses:
Email On:
Tier Info
Receive notifications on tier events such as tier allocated, tier upgraded, etc.
Tier Critical
Receive notification when a tiers status becomes non-critical
Tier Non-Critical
Receive notification when a tiers status becomes critical
Node Failure Rules
Reboot On Failure:
Reboot the failed node before replacing it.
Automatic Move to Maintenance Pool:
Move a failed node to the maintenance pool.
Node Harvesting
If on, higher priority tiers will get nodes from this tier if there are no nodes in the free pool.
Allow Harvest:
Other tiers may acquire nodes from this tier.
CANCEL
SAVE CHANGES
Domain
Tiers
Pools
Logs
Reports
Images
Users
Done

FIG. 8

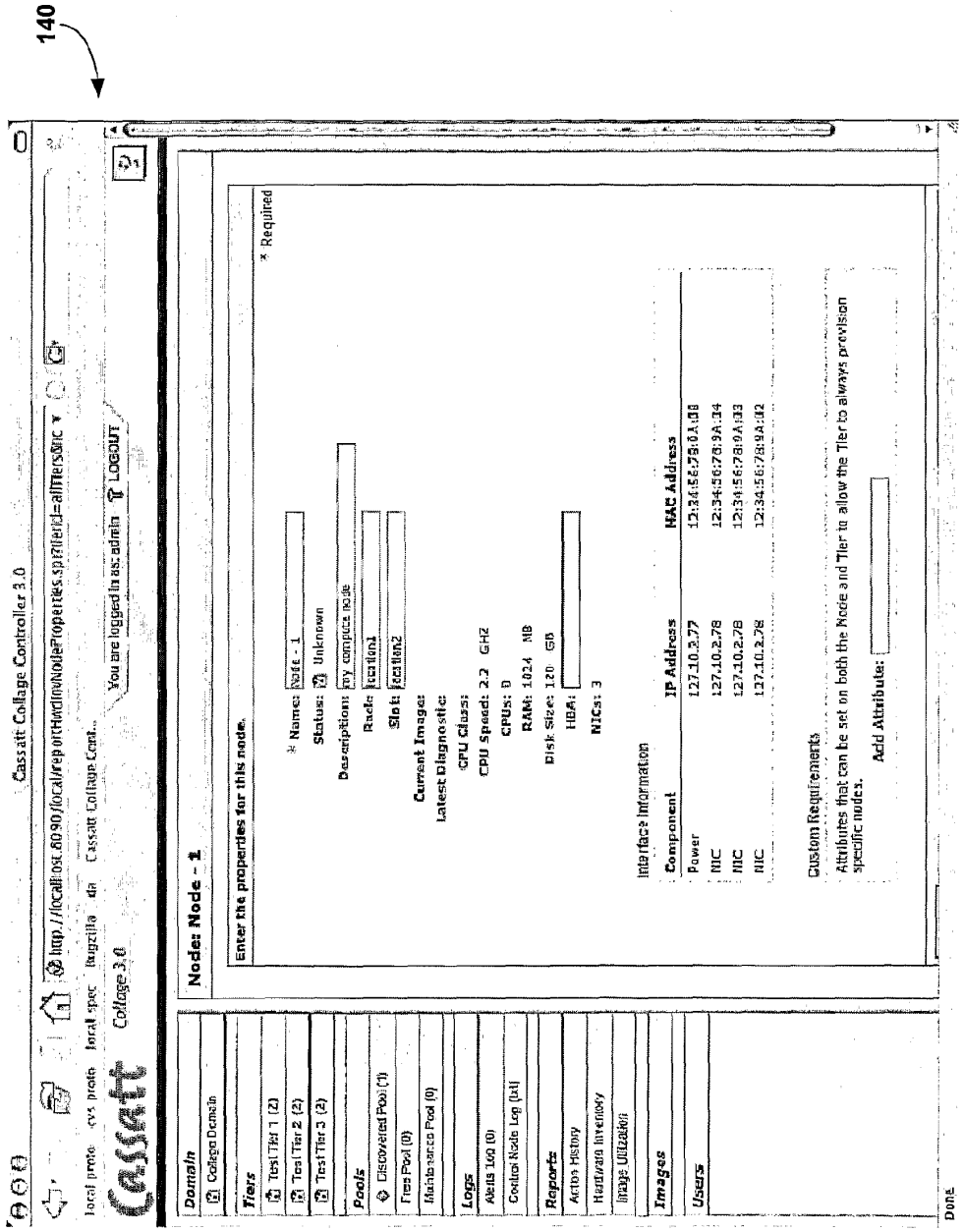

140
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (all)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Node: Node - 1
Enter the properties for this node.
* Required
* Name: Node - 1
Status: Unknown
Description: my compute node
Rack: location1
Slot: location2
Current Images
Latest Diagnostic
CPU Class:
CPU Speed: 2.2 GHz
CPUs: 0
RAM: 1024 MB
Disk Size: 120 GB
HBAs:
NICs: 3
Interface Information
Component
IP Address
MAC Address
Power 127.10.2.77 12:34:56:78:9A:00
NIC 127.10.2.78 12:34:56:78:9A:01
NIC 127.10.2.78 12:34:56:78:9A:02
NIC 127.10.2.78 12:34:56:78:9A:03
Custom Requirements
Attributes that can be set on both the Node and Tier to allow the Tier to always provision specific nodes.
Add Attribute:
Done

FIG. 9

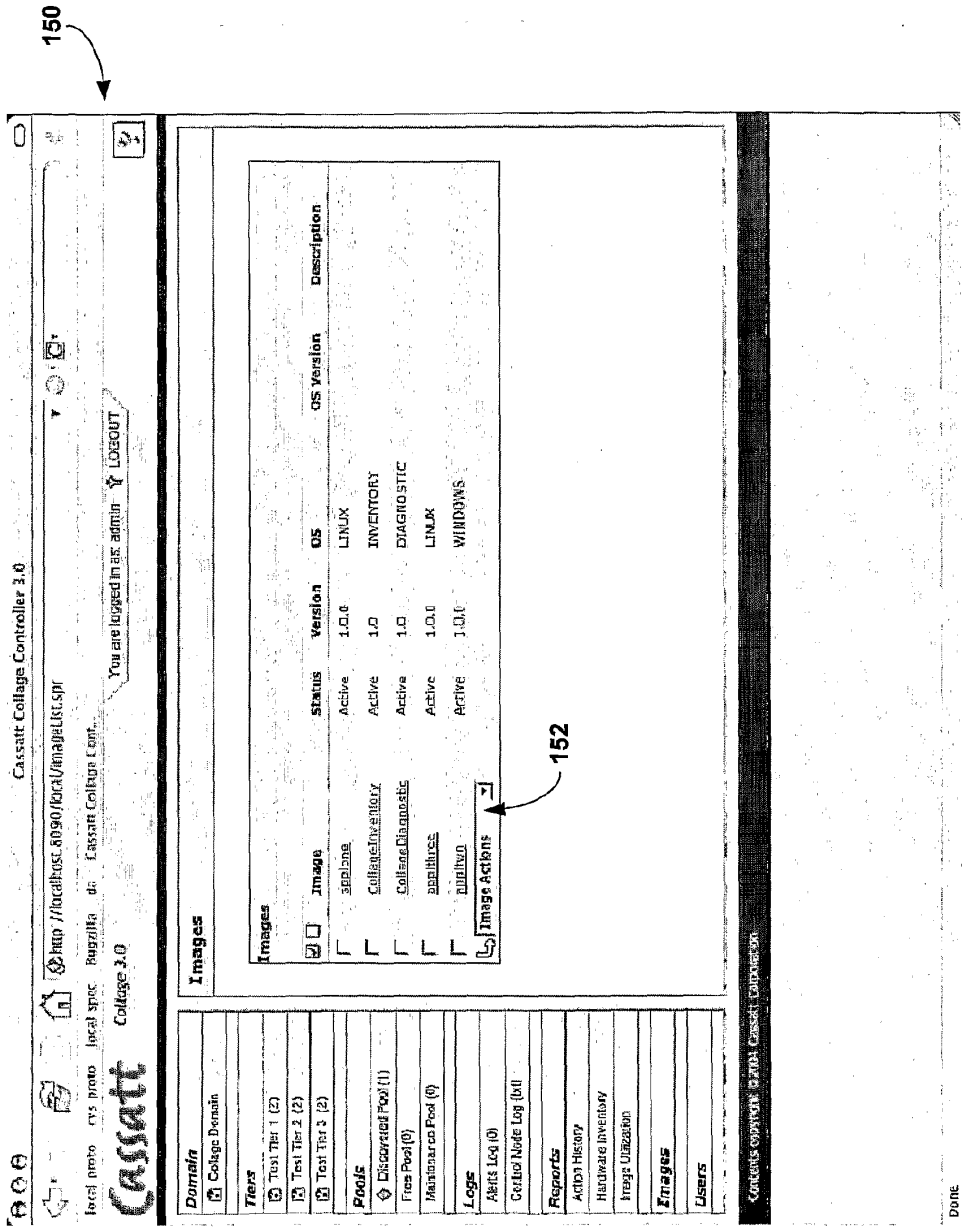

150
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (bll)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Images
Images
Image
Status
Version
OS
OS Version
Description
appliance
Active
1.0.0
LINUX
CollageInventory
Active
1.0
INVENTORY
CollageDiagnostic
Active
1.0
DIAGNOSTIC
peaktree
Active
1.0.0
LINUX
pipeliner
Active
1.0.0
WINDOWS
Image Actions
152
Contents copyright ©2004 Cassatt Corporation
Done

FIG. 10

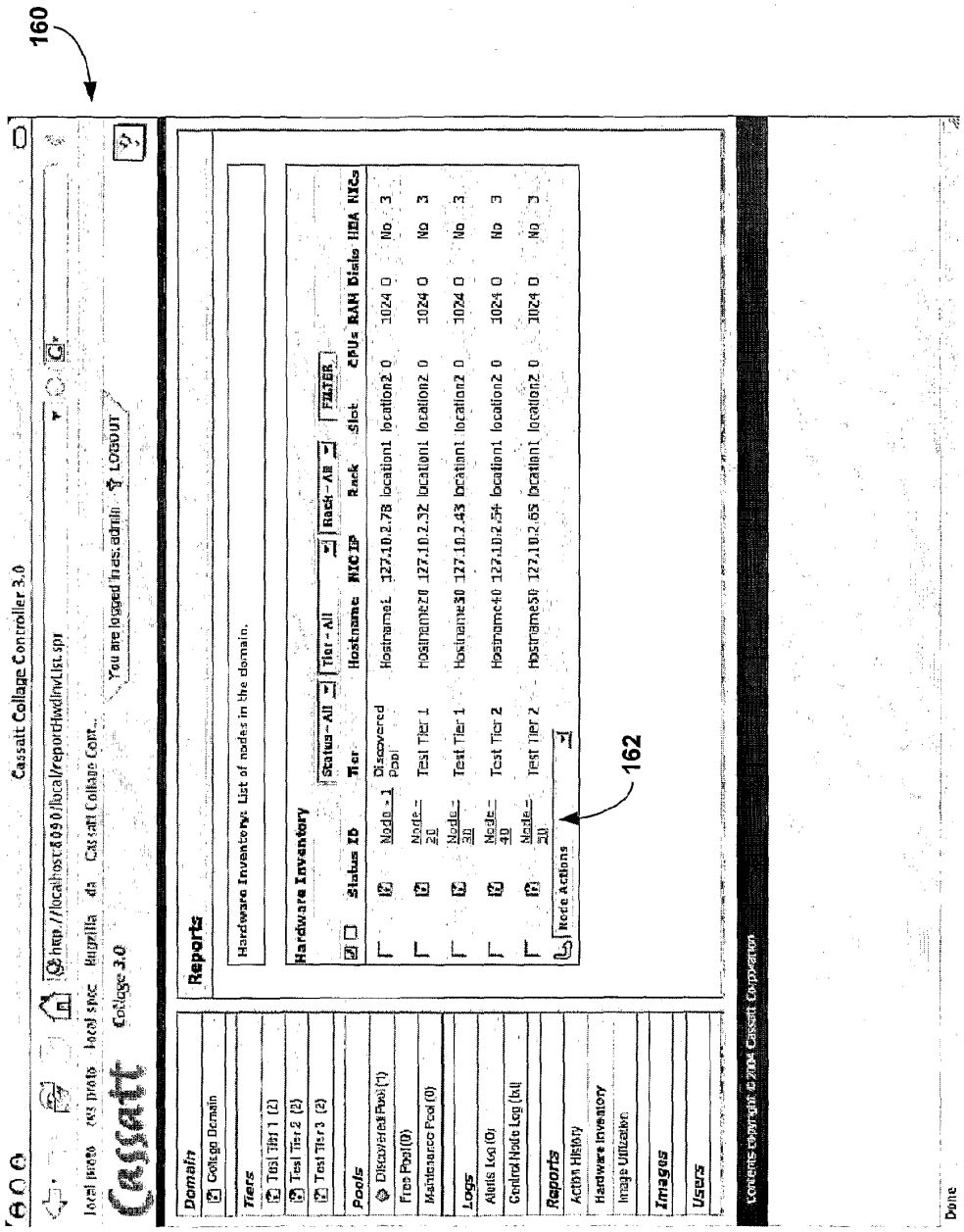

160
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (full
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Reports
Hardware Inventory: List of nodes in the domain.
Hardware Inventory
Status - All
Tier - All
Rack - All
FILTER
Status ID
Tier
Hostname
NIC IP
Rack
Slot
CPUs
RAM
Disks
HBA
NICs
Node - 1
Discovered Pool
Hostname1
127.10.2.78
location1
location2
0
1024
0
No
3
Node - 20
Test Tier 1
Hostname20
127.10.2.32
location1
location2
0
1024
0
No
3
Node - 30
Test Tier 1
Hostname30
127.10.2.43
location1
location2
0
1024
0
No
3
Node - 40
Test Tier 2
Hostname40
127.10.2.54
location1
location2
0
1024
0
No
3
Node - 50
Test Tier 2
Hostname50
127.10.2.65
location1
location2
0
1024
0
No
3
Node Actions
162
Contents Copyright © 2004 Cassatt Corporation
Done

FIG. 11

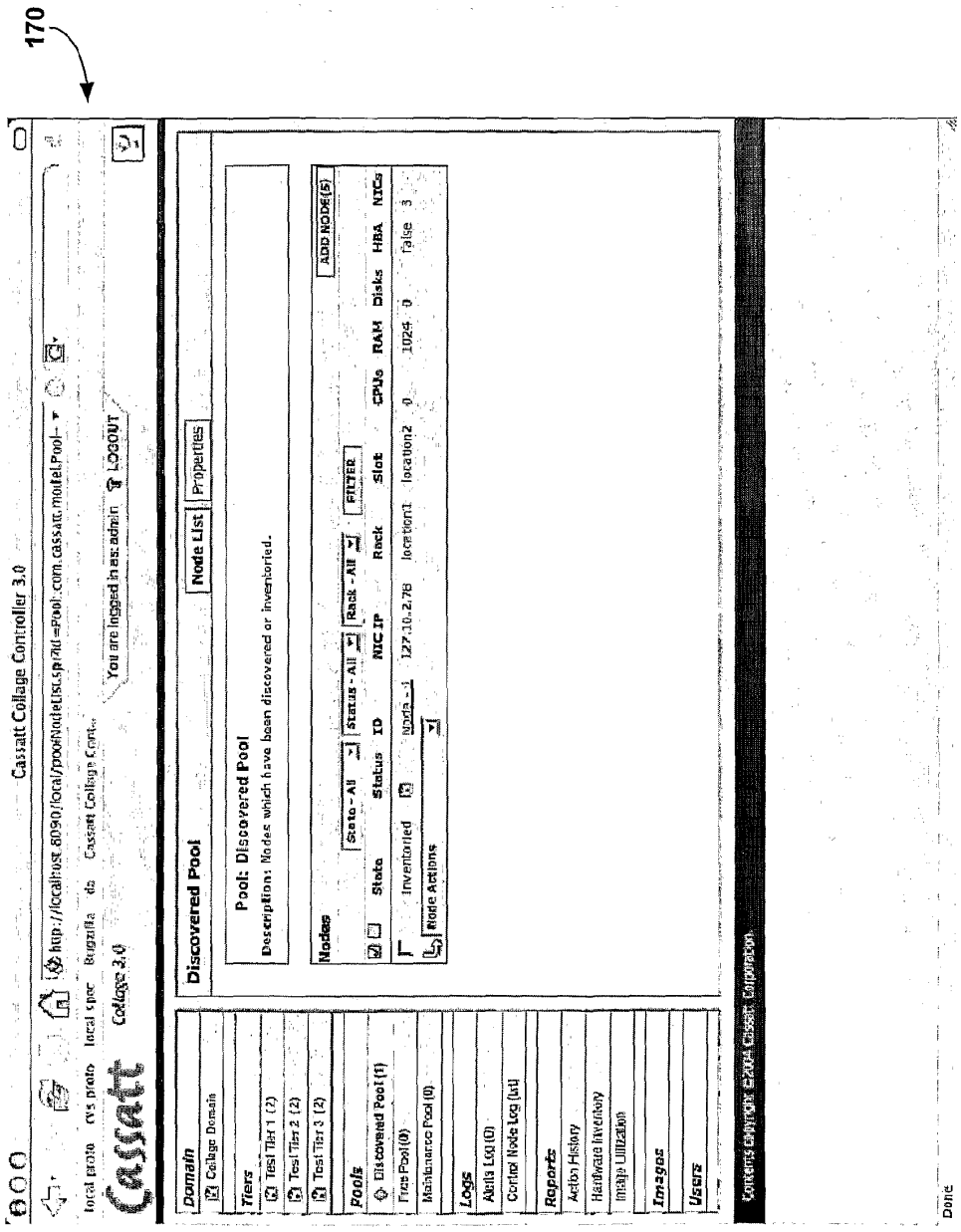

170
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (all)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Discovered Pool
Node List
Properties
Pool: Discovered Pool
Description: Nodes which have been discovered or inventoried.
Nodes
State - All
Status - All
Rack - All
FILTER
ADD NODE(S)
State
Status
ID
NIC IP
Rack
Slot
CPUs
RAM
Disks
HBA
NICs
Inventoried
node-1
127.10.2.78
location1
location2
0
1024
0
false
3
Node Actions
Contents copyright ©2004 Cassatt Corporation
Done

FIG. 12

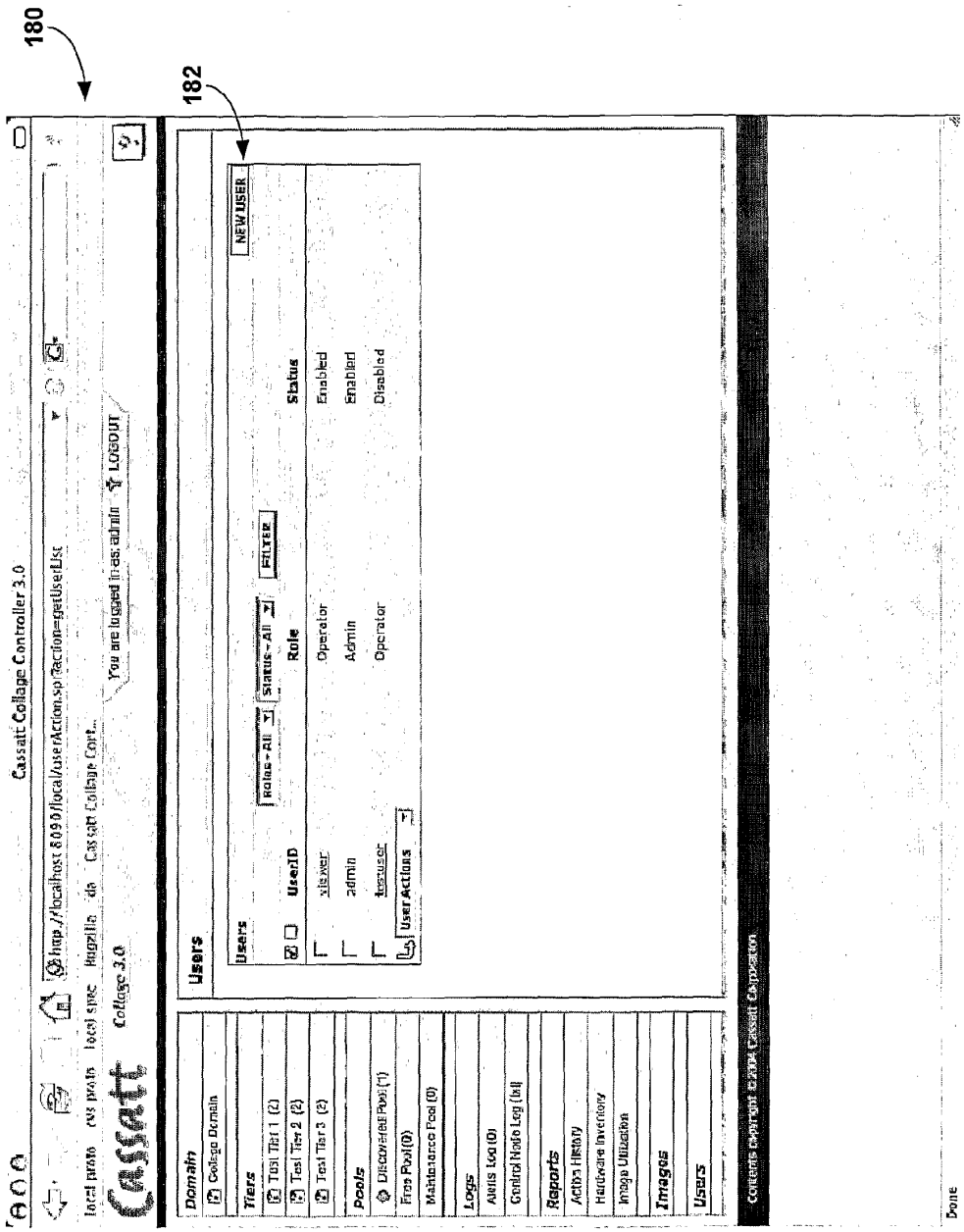

180
182
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
LOGOUT
Domain
Tiers
Pools
Logs
Reports
Images
Users
NEW USER
FILTER
UserID
Role
Status
Operator
Admin
Enabled
Disabled
User Actions
Done

FIG. 13

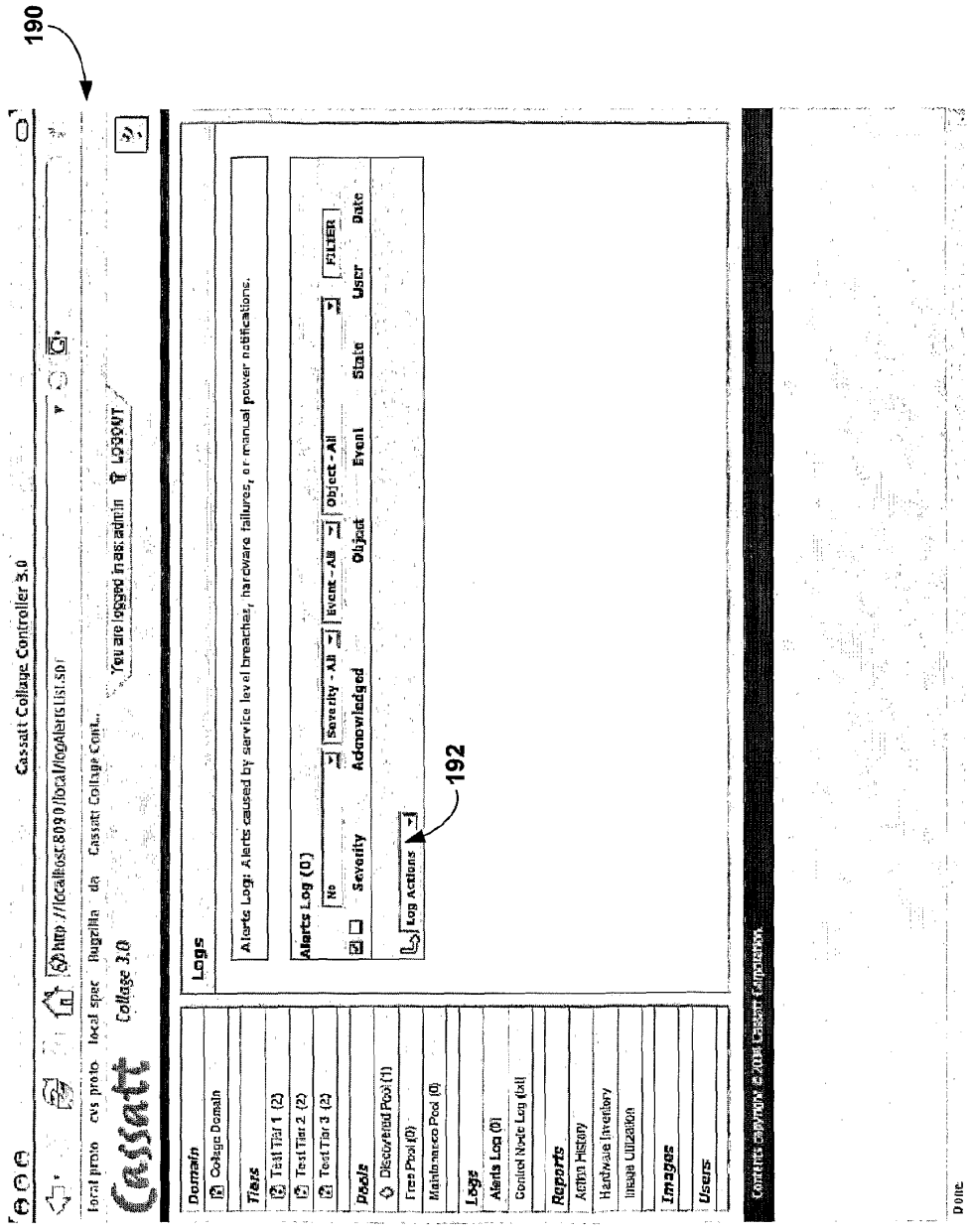

190
Cassatt Collage Controller 3.0
http://localhost:8090/local/logAlertsList.sbf
local proto cvs proto local spec Bugzilla da Cassatt Collage Cont...
Cassatt
Collage 3.0
You are logged in as: admin
LOGOUT
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (all)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Logs
Alerts Log: Alerts caused by service level breaches, hardware failures, or manual power notifications.
Alerts Log (0)
No
Severity - All
Event - All
Object - All
FILTER
Severity
Acknowledged
Object
Event
State
User
Date
Log Actions
192
Done

FIG. 14

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003436

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
5 July 2006	18/07/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Ross, C

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003436

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HERNESS E N ET AL: "WebSphere Application Server: a foundation for on demand computing" IBM SYSTEMS JOURNAL IBM USA, vol. 43, no. 2, 2004, pages 213-237, XP002383791 ISSN: 0018-8670 Paragraph starting with "Multitiered distributed business computing": page 215, left-hand column page 220, left-hand column, line 26 - page 221, right-hand column, line 3 Paragraph starting with "Deployment automation": page 222, left-hand column - right-hand column, line 11 page 223, left-hand column, line 5 - line 23 page 224, right-hand column, line 11 - right-hand column, line 3 page 226, left-hand column, line 9 - page 229, left-hand column, line 2 page 229, right-hand column, line 1 - line 13 Paragraph starting with "Monitoring and application response measurement...": page 230, left-hand column	1-25
X	G LODI, F PANZIERI: "QoS-aware Clustering of Application Servers" PROCEEDINGS OF THE 1ST IEEE WORKSHOP ON QUALITY OF SERVICE FOR APPLICATION SERVERS, IN CONJUNCTION WITH THE 23RD INT'L SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS 2004), 17 October 2004 (2004-10-17), pages 1-6, XP002383792 Florianopolis, Brazil page 1, line 1 - line 19 page 2, left-hand column, line 10 - right-hand column, line 18 page 3, right-hand column, line 27 - line 45	1-25
X	US 2003/126265 A1 (AZIZ ASHAR ET AL) 3 July 2003 (2003-07-03) abstract; figure 1c paragraph [0049] paragraph [0052] paragraph [0088] - paragraph [0095] paragraph [0206]	1-25

-/--

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/003436

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/177176 A1 (HIRSCHFELD ROBERT A ET AL) 18 September 2003 (2003-09-18) abstract column 7, line 4 - line 10 column 9, line 29 - line 52; figure 3 column 13, line 23 - line 53; figures 5a,5b	1-25

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/003436

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003126265 A1	03-07-2003	NONE	
US 2003177176 A1	18-09-2003	NONE	

Form PCT/ISA/210 (patent family annex) (April 2005)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
11/694,483	03/30/2007	2161	690	1072-021US01	28	3

CONFIRMATION NO. 1040

UPDATED FILING RECEIPT

28863
SHUMAKER & SIEFFERT, P. A.
1625 RADIO DRIVE
SUITE 300
WOODBURY, MN55125

Date Mailed: 06/22/2007

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please mail to the Commissioner for Patents P.O. Box 1450 Alexandria Va 22313-1450. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

Applicant(s)

Jagane Sundar, Saratoga, CA;
Sanjay Radia, Fremont, CA;
David A. Henseler, Maplewood, MN;

Assignment For Published Patent Application

Cassatt Corporation, San Jose, CA

Power of Attorney: The patent practitioners associated with Customer Number 28863

Domestic Priority data as claimed by applicant

This appln claims benefit of 60/787,280 03/30/2006

Foreign Applications

If Required, Foreign Filing License Granted: 04/10/2007

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US11/694,483**

Projected Publication Date: 10/04/2007

Non-Publication Request: No

Early Publication Request: No

** SMALL ENTITY **

Title

DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

Preliminary Class

707

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date

thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

RECEIVED
CENTRAL FAX CENTER
JUN 11 2007

SHUMAKER & SIEFFERT, P.A.
1625 RADIO DRIVE, SUITE 300
WOODBURY, MINNESOTA 55125
TEL 651.735-1100
FAX 651.735-1102
WWW.SSIPLAW.COM

FACSIMILE SUBMISSION UNDER 37 CFR 1.8

TO:	FROM:
Mail Stop Missing Parts	Albert W. Vredeveld
COMPANY:	DATE:
US Patent and Trademark Office	JUNE 11, 2007
FAX NUMBER:	TOTAL NO. OF PAGES INCLUDING COVER:
571-273-8300	12
PHONE NUMBER:	SENDER'S REFERENCE NUMBER:
	1072-021US01
RE:	APPLICATION SERIAL NUMBER:
Notice to File Missing Parts of Non-provisional Application	11/694,483

This facsimile message is intended for the sole use of the intended recipient(s) and may contain information that is confidential, privileged and/or attorneys' work product. Any review or distribution by any other person is prohibited. If you are not an intended recipient, please immediately contact the sender and delete all copies.

PAGE 1/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

RECEIVED Page 1 of 2
CENTRAL FAX CENTER
JUN 11 2007

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/694,483	03/30/2007	Jagane Sundar	1072-021US01

CONFIRMATION NO. 1040

28863
SHUMAKER & SIEFFERT, P. A.
1625 RADIO DRIVE
SUITE 300
WOODBURY, MN 55125

FORMALITIES LETTER

Date Mailed: 04/11/2007

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing. *A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.*
  Note: If a petition under 37 CFR 1.47 is being filed, an oath or declaration in compliance with 37 CFR 1.63 signed by all available joint inventors, or if no inventor is available by a party with sufficient proprietary interest, is required.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is $65 for a small entity

- $65 Surcharge.

Replies should be mailed to: Mail Stop Missing Parts

06/13/2007 SFELEKE1 00000061 501778 11694483
01 FC:2051 65.00 DA

Netflix, Inc. - Ex. 1002, Page 000680
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Registered users of EFS-Web may alternatively submit their reply to this notice via EFS-Web.
https://sportal.uspto.gov/authenticate/AuthenticateUserLocalEPF.html

For more information about EFS-Web please call the USPTO Electronic Business Center at **1-866-217-9197** or visit our website at http://www.uspto.gov/ebc.

If you are not using EFS-Web to submit your reply, you must include a copy of this notice.

Jovey Wyatt

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199
PART 2 - COPY TO BE RETURNED WITH RESPONSE

Netflix, Inc. - Ex. 1002, Page 000681
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

RECEIVED
CENTRAL FAX CENTER
JUN 11 2007

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Jagane Sundar; Sanjay Radia; David A. Henseler	Confirmation No.	1040
Serial No.:	11/694,483		
Filed:	March 30, 2007	Customer No.:	28863
Examiner:	Unknown	Group Art Unit:	2161
Docket No.:	1072-021US01		
Title:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES		

CERTIFICATE UNDER 37 CFR 1.8 I hereby certify that this correspondence is being transmitted via facsimile to the United States Patent and Trademark Office on June 11, 2007.

By: Nancy J. Burr
Name: Nancy J. Burr

MAIL STOP MISSING PARTS
Commissioner for Patents
Alexandria, VA 22313-1450

Sir:

We are transmitting herewith the attached correspondence relating to this application:

- ☒ Transmittal sheet containing Certificate of Mailing
- ☒ Signed Combined Declaration and Power of Attorney (8 pgs.)
- ☒ Part 2 - Notice to File Missing Parts of Nonprovisional Application (2 pgs.)
- ☒ Please charge Deposit Account 50-1778 in the amount of $65.00 for missing parts surcharge

Please apply any charges not covered, or any credits, to Deposit Account No. 50-1778.

Date: June 11, 2007

By: _____
Name: Albert W. Vredeveld
Reg. No.: 60,315

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Netflix, Inc. - Ex. 1002, Page 000682
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

RECEIVED
CENTRAL FAX CENTER
JUN 11 2007

Attorney Docket No. 1072-021US01

SHUMAKER & SIEFFERT, P.A.

United States Patent Application

COMBINED DECLARATION AND POWER OF ATTORNEY

As a below named inventor I hereby declare that my residence, post office address and citizenship are as stated below next to my name, and that I believe I am an original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled: DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

The specification of which
a. ☐ is attached hereto
b. ☒ was filed on March 30, 2007 as application serial no. 11/694,483 and was amended on (if applicable) or
c. ☐ was (in the case of a PCT-filed application) described and claimed in international no. filed and as amended on (if any), which I have reviewed and for which I solicit a United States patent.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, § 1.56 (attached hereto).

I hereby claim foreign priority benefits under Title 35, United States Code, § 119/365 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on the basis of which priority is claimed:

a. ☒ no such applications have been filed.
b. ☐ such applications have been filed as follows:

FOREIGN APPLICATION(S), IF ANY, CLAIMING PRIORITY UNDER 35 USC § 119			
COUNTRY	APPLICATION NUMBER	DATE OF FILING (day, month, year)	DATE OF ISSUE (day, month, year)
ALL FOREIGN APPLICATION(S), IF ANY, FILED BEFORE THE PRIORITY APPLICATION(S)			
COUNTRY	APPLICATION NUMBER	DATE OF FILING (day, month, year)	DATE OF ISSUE (day, month, year)

I hereby claim the benefit of any United States and PCT international application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application.

U.S. APPLICATION NUMBER	DATE OF FILING (day, month, year)	STATUS
60/787,280	March 30, 2006	

I hereby appoint Practitioners at Customer Number 28863

28863
PATENT TRADEMARK OFFICE

PAGE 3/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000683
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Steven J. Shumaker	Reg. No. 36,275	Kelly P. Fitzgerald	Reg. No. 46,326
Kent J. Sieffert	Reg. No. 41,312	H. Sanders Gwin, Jr.	Reg. No. 33,242
Jason D. Kelly	Reg. No. 54,213	Kari H. Bartingale	Reg. No. 35,183
Darcy L. Grunwald	Reg. No. 56,902	Daniel T. Lund	Reg. No. 58,614
Michael J. Ostrom	Reg. No. 58,730	Jennifer M.K. Rogers	Reg. No. 58,695
Jessica H. Kwak	Reg. No. 58,975		

as my/our attorney(s) and/or patent agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

I hereby authorize them to act and rely on instructions from and communicate directly with the person/assignee/attorney/firm/ organization who/which first sends/sent this case to them and by whom/which I hereby declare that I have consented after full disclosure to be represented unless/until I instruct Shumaker & Sieffert, P.A. to the contrary.

Please direct all correspondence in this case to:

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

	Full Name Of Inventor	Family Name Sundar	First Given Name Jagane	Second Given Name
	Residence & Citizenship	City Saratoga	State or Foreign Country California	Country of Citizenship India
	Post Office Address	Post Office Address 19050 Brookview Drive	City Saratoga	State & Zip Code/Country California 95070/USA
Signature of Inventor:			Date:	

	Full Name Of Inventor	Family Name Radia	First Given Name Sanjay	Second Given Name
	Residence & Citizenship	City Fremont	State or Foreign Country California	Country of Citizenship United States of America
	Post Office Address	Post Office Address 883 Boar Circle	City Fremont	State & Zip Code/Country California 94539
Signature of Inventor:			Date:	

PAGE 4/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000684
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

	Full Name Of Inventor	Family Name Henseler	First Given Name David	Second Given Name A.
	Residence & Citizenship	City Maplewood	State or Foreign Country Minnesota	Country of Citizenship United States of America
	Post Office Address	Post Office Address 928 Dermont Ave. E.	City Maplewood	State & Zip Code/Country Minnesota 55109/USA
Signature of Inventor:			Date: 4/20/07	

PAGE 5/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000685
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

§ 1.56 Duty to disclose information material to patentability.

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is canceled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is canceled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§ 1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) the closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made of record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim;

or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application:

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.

PAGE 6/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000686
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Attorney Docket No. 1072-021US01

SHUMAKER & SIEFFERT, P.A.

United States Patent Application

COMBINED DECLARATION AND POWER OF ATTORNEY

As a below named inventor I hereby declare that my residence, post office address and citizenship are as stated below next to my name, and that I believe I am an original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled: DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

The specification of which
a. ☐ is attached hereto
b. ☒ was filed on March 30, 2007 as application serial no. 11/694,483 and was amended on (if applicable) or
c. ☐ was (in the case of a PCT-filed application) described and claimed in international no. filed and as amended on (if any), which I have reviewed and for which I solicit a United States patent.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, § 1.56 (attached hereto).

I hereby claim foreign priority benefits under Title 35, United States Code, § 119/365 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on the basis of which priority is claimed:

a. ☒ no such applications have been filed.
b. ☐ such applications have been filed as follows:

FOREIGN APPLICATION(S), IF ANY, CLAIMING PRIORITY UNDER 35 USC § 119			
COUNTRY	APPLICATION NUMBER	DATE OF FILING (day, month, year)	DATE OF ISSUE (day, month, year)
ALL FOREIGN APPLICATION(S), IF ANY, FILED BEFORE THE PRIORITY APPLICATION(S)			
COUNTRY	APPLICATION NUMBER	DATE OF FILING (day, month, year)	DATE OF ISSUE (day, month, year)

I hereby claim the benefit of any United States and PCT international application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application.

U.S. APPLICATION NUMBER	DATE OF FILING (day, month, year)	STATUS
60/787,280	March 30, 2006	

I hereby appoint Practitioners at Customer Number 28863

28863
PATENT TRADEMARK OFFICE

PAGE 7/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000687
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Steven J. Shumaker	Reg. No. 36,275	Kelly P. Fitzgerald	Reg. No. 46,326
Kent J. Sieffert	Reg. No. 41,312	H. Sanders Gwin, Jr.	Reg. No. 33,242
Jason D. Kelly	Reg. No. 54,213	Kari H. Bartingale	Reg. No. 35,183
Darcy L. Grunwald	Reg. No. 56,902	Daniel T. Lund	Reg. No. 58,614
Michael J. Ostrom	Reg. No. 58,730	Jennifer M.K. Rogers	Reg. No. 58,695
Jessica H. Kwak	Reg. No. 58,975		

as my/our attorney(s) and/or patent agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

I hereby authorize them to act and rely on instructions from and communicate directly with the person/assignee/attorney/firm/ organization who/which first sends/sent this case to them and by whom/which I hereby declare that I have consented after full disclosure to be represented unless/until I instruct Shumaker & Sieffert, P.A. to the contrary.

Please direct all correspondence in this case to:

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

	Full Name Of Inventor	Family Name Sundar	First Given Name Jagane	Second Given Name
	Residence & Citizenship	City Saratoga	State or Foreign Country California	Country of Citizenship India
	Post Office Address	Post Office Address 19050 Brookview Drive	City Saratoga	State & Zip Code/Country California 95070/USA
Signature of Inventor:				Date: 5 June 2007

	Full Name Of Inventor	Family Name Radia	First Given Name Sanjay	Second Given Name
	Residence & Citizenship	City Fremont	State or Foreign Country California	Country of Citizenship United States of America
	Post Office Address	Post Office Address 883 Boar Circle	City Fremont	State & Zip Code/Country California 94539
Signature of Inventor:				Date: 4 June 2007

PAGE 8/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000688
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

	Full Name Of Inventor	Family Name Henseler	First Given Name David	Second Given Name A.
	Residence & Citizenship	City Maplewood	State or Foreign Country Minnesota	Country of Citizenship United States of America
	Post Office Address	Post Office Address 928 Demont Ave. E.	City Maplewood	State & Zip Code/Country Minnesota 55109/USA
Signature of Inventor:			Date:	

PAGE 9/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000689
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

§ 1.56 Duty to disclose information material to patentability.

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is canceled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is canceled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§ 1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) the closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made of record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim;

or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.

PAGE 10/12 * RCVD AT 6/11/2007 4:34:17 PM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/15 * DNIS:2738300 * CSID:6517351102 * DURATION (mm-ss):02-40

Netflix, Inc. - Ex. 1002, Page 000690
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
11/694,483	03/30/2007	2161	625	1072-021US01	28	3

CONFIRMATION NO. 1040

28863
SHUMAKER & SIEFFERT, P. A.
1625 RADIO DRIVE
SUITE 300
WOODBURY, MN55125

FILING RECEIPT

Date Mailed: 04/11/2007

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please mail to the Commissioner for Patents P.O. Box 1450 Alexandria Va 22313-1450. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

Applicant(s)

Jagane Sundar, Saratoga, CA;
Sanjay Radia, Fremont, CA;
David A. Henseler, Maplewood, MN;

Assignment For Published Patent Application

Cassatt Corporation, San Jose, CA

Power of Attorney: None

Domestic Priority data as claimed by applicant

This appln claims benefit of 60/787,280 03/30/2006

Foreign Applications

If Required, Foreign Filing License Granted: 04/10/2007

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US11/694,483**

Projected Publication Date: To Be Determined - pending completion of Missing Parts

Non-Publication Request: No

Early Publication Request: No

** SMALL ENTITY **

Title

DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

Preliminary Class

707

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/694,483	03/30/2007	Jagane Sundar	1072-021US01

CONFIRMATION NO. 1040

28863
SHUMAKER & SIEFFERT, P. A.
1625 RADIO DRIVE
SUITE 300
WOODBURY, MN 55125

FORMALITIES LETTER

Date Mailed: 04/11/2007

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The oath or declaration is missing. *A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.*
  Note: If a petition under 37 CFR 1.47 is being filed, an oath or declaration in compliance with 37 CFR 1.63 signed by all available joint inventors, or if no inventor is available by a party with sufficient proprietary interest, is required.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

• To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $65 for a small entity in compliance with 37 CFR 1.27, must be submitted with the missing items identified in this letter.

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **$65** for a small entity

- **$65** Surcharge.

Replies should be mailed to: Mail Stop Missing Parts

Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Registered users of EFS-Web may alternatively submit their reply to this notice via EFS-Web.
https://sportal.uspto.gov/authenticate/AuthenticateUserLocalEPF.html

For more information about EFS-Web please call the USPTO Electronic Business Center at **1-866-217-9197** or visit our website at http://www.uspto.gov/ebc.

If you are not using EFS-Web to submit your reply, you must include a copy of this notice.

Torrey Wyatt

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199
PART 3 - OFFICE COPY

DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

[0001] This application claims the benefit of U.S. provisional Application Serial No. 60/787,280, filed March 30, 2006, the entire content of which is incorporated herein by reference.

TECHNICAL FIELD

[0002] The invention relates to computing environments and, more specifically, to distributed computing systems.

BACKGROUND

[0003] Distributed computing systems are increasingly being utilized to support business as well as technical applications. Typically, distributed computing systems are constructed from a collection of computing nodes that combine to provide a set of processing services to implement the distributed computing applications. Each of the computing nodes in the distributed computing system is typically a separate, independent computing device interconnected with each of the other computing nodes via a communications medium, e.g., a network.

[0004] One challenge with distributed computing systems is the organization, deployment and administration of such a system within an enterprise environment. For example, it is often difficult to manage the allocation and deployment of enterprise computing functions within the distributed computing system. An enterprise, for example, often includes several business groups, and each group may have competing and variable computing requirements.

SUMMARY

[0005] In general, the invention is directed to a distributed computing system that conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for

Netflix, Inc. - Ex. 1002, Page 000696
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model.

[0006] As described herein, the model includes four distinct levels: fabric, domains, tiers and nodes that provide for the logical abstraction and containment of the physical components as well as system and service application software of the enterprise. A user, such as a system administrator, interacts with the control nodes to logically define the hierarchical organization of the distributed computing system. The control nodes are responsible for all levels of management in accordance with the model, including fabric management, domain creation, tier creation and node allocation and deployment.

[0007] In one embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network and a control node. The control node comprises a set of one or more applications to be executed on the application nodes, an application matrix that includes parameters for controlling the deployment of the applications within the distributed computing system, and an automation unit having one or more rule engines that provide autonomic control of the application nodes and the applications in accordance with a set of one or more rules and the application matrix.

[0008] In another embodiment, a method comprises generating an application matrix that specifies data for controlling the deployment of a set of applications within a distributed computing system, and performing operations to provide autonomic control over the deployment of the applications within the distributed computing system in accordance with parameters of the application matrix.

[0009] In another embodiment, a computer-readable medium comprises instructions. The instructions cause the processor to generate an application matrix that specifies data for controlling the deployment of a set of applications within a distributed computing system, and perform operations to provide autonomic control over the deployment of the applications within the distributed computing system in accordance with parameters of the application matrix.

[0010] In another embodiment, a distributed computing system comprises a plurality of application nodes interconnected via a communications network. In addition, the distributed computing system includes a software image repository storing: (i) one or more image instances of a virtual machine manager that is executable on the application nodes, wherein

Netflix, Inc. - Ex. 1002, Page 000697
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, and (ii) one or more image instances of one or more software applications that are executable on the virtual machines. The distributed computing system also includes a control node that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the image instances of the software applications on the virtual machines.

[0011] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0012] FIG. 1 is a block diagram illustrating a distributed computing system constructed from a collection of computing nodes.

[0013] FIG. 2 is a schematic diagram illustrating an example of a model of an enterprise that logically defines an enterprise fabric.

[0014] FIG. 3 is a flow diagram that provides a high-level overview of the operation of a control node when configuring the distributed computing system.

[0015] FIG. 4 is a flow diagram illustrating exemplary operation of the control node when assigning computing nodes to node slots of tiers.

[0016] FIG. 5 is a flow diagram illustrating exemplary operation of a control node when adding an additional computing node to a tier to meet additional processing demands.

[0017] FIG. 6 is a flow diagram illustrating exemplary operation of a control node harvesting excess node capacity from one of the tiers and returning the harvested computing node to the free pool.

[0018] FIG. 7 is a screen illustration of an exemplary user interface for defining tiers in a particular domain.

[0019] FIG. 8 is a screen illustration of an exemplary user interface for defining properties of the tiers.

[0020] FIG. 9 is a screen illustration of an exemplary user interface for viewing and identify properties of a computing node.

Netflix, Inc. - Ex. 1002, Page 000698
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0021] FIG. 10 is a screen illustration of an exemplary user interface for viewing software images.

[0022] FIG. 11 is a screen illustration of an exemplary user interface for viewing a hardware inventory report.

[0023] FIG. 12 is a screen illustration of an exemplary user interface for viewing discovered nodes that are located in the free pool.

[0024] FIG. 13 is a screen illustration of an exemplary user interface for viewing users of a distributed computing system.

[0025] FIG. 14 is a screen illustration of an exemplary user interface for viewing alerts for the distributed computing system.

[0026] FIG. 15 is a block diagram illustrating one embodiment of control node that includes a monitoring subsystem, a service level automation infrastructure (SLAI), and a business logic tier (BLT).

[0027] FIG. 16 is a block diagram illustrating one embodiment of the monitoring subsystem.

[0028] FIG. 17 is a block diagram illustrating one embodiment of the SLAI in further detail.

[0029] FIG. 18 is a block diagram of an example working memory associated with rule engines of the SLAI.

[0030] FIG. 19 is a block diagram illustrating an example embodiment for the BLT of the control node.

[0031] FIG. 20 is a block diagram illustrating one embodiment of a rule engine in further detail.

[0032] FIG. 21 is a block diagram illustrating another example embodiment of the control node.

[0033] FIG. 22 is a flowchart illustrating an exemplary mode of initiating a control node utilizing an application matrix.

[0034] FIG. 23 is a block diagram illustrating an exemplary application matrix.

[0035] FIG. 24 is a block diagram illustrating an exemplary configuration of a physical node to which a virtual machine manager and one or more virtual machines are deployed.

[0036] FIG. 25 is a flowchart illustrating an exemplary procedure for creating and capturing an image of a virtual machine manager for autonomic deployment within the distributed computing system.

Netflix, Inc. - Ex. 1002, Page 000699
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0037] FIG. 26 is a flowchart illustrating an exemplary procedure for creating and capturing a virtual disk image file for autonomic deployment within the distributed computing system.

[0038] FIG. 27A is a flowchart illustrating an exemplary procedure for deploying a virtual machine manager to a node within the distributed computing system.

[0039] FIG. 27B is a flowchart illustrating an exemplary procedure for deploying an operating system and application image to a node that is executing a virtual machine manager within the distributed computing system.

[0040] FIG. 28 is a screen image illustrating an exemplary tier control interface for the embodiment described in FIGS. 24 through 27.

[0041] FIG. 29 is a screen image illustrating an exemplary node control interface for the embodiment described in FIGS. 24 through 27.

DETAILED DESCRIPTION

[0042] FIG. 1 is a block diagram illustrating a distributed computing system 10 constructed from a collection of computing nodes. Distributed computing system 10 may be viewed as a collection of computing nodes operating in cooperation with each other to provide distributed processing.

[0043] In the illustrated example, the collection of computing nodes forming distributed computing system 10 are logically grouped within a discovered pool 11, a free pool 13, an allocated tiers 15 and a maintenance pool 17. In addition, distributed computing system 10 includes at least one control node 12.

[0044] Within distributed computing system 10, a computing node refers to the physical computing device. The number of computing nodes needed within distributed computing system 10 is dependent on the processing requirements. For example, distributed computing system 10 may include 8 to 512 computing nodes or more. Each computing node includes one or more programmable processors for executing software instructions stored on one or more computer-readable media.

[0045] Discovered pool 11 includes a set of discovered nodes that have been automatically "discovered" within distributed computing system 10 by control node 12. For example, control node 12 may monitor dynamic host communication protocol (DHCP) leases to discover the connection of a node to network 18. Once detected, control node 12

Netflix, Inc. - Ex. 1002, Page 000700
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

automatically inventories the attributes for the discovered node and reassigns the discovered node to free pool 13. The node attributes identified during the inventory process may include a CPU count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics or other computing resources. Control node 12 may also receive input identifying node attributes not detectable via the automatic inventory, such as whether the node includes I/O, such as HBA. Further details with respect to the automated discovery and inventory processes are described in U.S. Patent Application No. 11/070,851, having attorney docket no. 1072-009US01, entitled "AUTOMATED DISCOVERY AND INVENTORY OF NODES WITHIN AN AUTONOMIC DISTRIBUTED COMPUTING SYSTEM," filed March 2, 2005, the entire content of which is hereby incorporated by reference.

[0046] Free pool 13 includes a set of unallocated nodes that are available for use within distributed computing system 10. Control node 12 may dynamically reallocate an unallocated node from free pool 13 to allocated tiers 15 as an application node 14. For example, control node 12 may use unallocated nodes from free pool 13 to replace a failed application node 14 or to add an application node to allocated tiers 15 to increase processing capacity of distributed computing system 10.

[0047] In general, allocated tiers 15 include one or more tiers of application nodes 14 that are currently providing a computing environment for execution of user software applications. In addition, although not illustrated separately, application nodes 14 may include one or more input/output (I/O) nodes. Application nodes 14 typically have more substantial I/O capabilities than control node 12, and are typically configured with more computing resources (e.g., processors and memory). Maintenance pool 17 includes a set of nodes that either could not be inventoried or that failed and have been taken out of service from allocated tiers 15.

[0048] Control node 12 provides the system support functions for managing distributed computing system 10. More specifically, control node 12 manages the roles of each computing node within distributed computing system 10 and the execution of software applications within the distributed computing system. In general, distributed computing system 10 includes at least one control node 12, but may utilize additional control nodes to assist with the management functions.

Netflix, Inc. - Ex. 1002, Page 000701
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0049] Other control nodes 12 (not shown in FIG. 1) are optional and may be associated with a different subset of the computing nodes within distributed computing system 10. Moreover, control node 12 may be replicated to provide primary and backup administration functions, thereby allowing for graceful handling a failover in the event control node 12 fails.

[0050] Network 18 provides a communications interconnect for control node 12 and application nodes 14, as well as discovered nodes, unallocated nodes and failed nodes. Communications network 18 permits internode communications among the computing nodes as the nodes perform interrelated operations and functions. Communications network 18 may comprise, for example, direct connections between one or more of the computing nodes, one or more customer networks maintained by an enterprise, local area networks (LANs), wide area networks (WANs) or a combination thereof. Communications network 18 may include a number of switches, routers, firewalls, load balancers, and the like.

[0051] In one embodiment, each of the computing nodes within distributed computing system 10 executes a common general-purpose operating system. One example of a general-purpose operating system is the Windows™ operating system provided by Microsoft Corporation. In some embodiments, the general-purpose operating system such as the Linux kernel may be used.

[0052] In the example of FIG. 1, control node 12 is responsible for software image management. The term "software image" refers to a complete set of software loaded on an individual computing node to provide an execution environment for one or more applications. The software image including the operating system and all boot code, middleware files, and may include application files. As described below embodiments of the invention provide application-level autonomic control over the deployment, execution and monitoring of applications onto software images associated with application nodes 14.

[0053] System administrator 20 may interact with control node 12 and identify the particular types of software images to be associated with application nodes 14. Alternatively, administration software executing on control node 12 may automatically identify the appropriate software images to be deployed to application nodes 14 based on the input received from system administrator 20. For example, control node 12 may determine the type of software image to load onto an application node 14 based on the functions assigned to the node by system administrator 20. Application nodes14 may be divided into a number of

Netflix, Inc. - Ex. 1002, Page 000702
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

groups based on their assigned functionality. As one example, application nodes 14 may be divided into a first group to provide web server functions, a second group to provide business application functions and a third group to provide database functions. The application nodes 14 of each group may be associated with different software images.

[0054] Control node 12 provides for the efficient allocation and management of the various software images within distributed computing system 10. In some embodiments, control node 12 generates a "golden image" for each type of software image that may be deployed on one or more of application nodes 14. As described herein, the term "golden image" refers to a reference copy of a complete software stack for providing an execution environment for applications.

[0055] System administrator 20 may create a golden image by installing an operating system, middleware and software applications on a computing node and then making a complete copy of the installed software. In this manner, a golden image may be viewed as a "master copy" of the software image for a particular computing function. Control node 12 maintains a software image repository 26 that stores the golden images associated with distributed computing system 10.

[0056] Control node 12 may create a copy of a golden image, referred to as an "image instance," for each possible image instance that may be deployed within distributed computing system 10 for a similar computing function. In other words, control node 12 pre-generates a set of K image instances for a golden image, where K represents the maximum number of image instances for which distributed computing system 10 is configured for the particular type of computing function. For a given computing function, control node 12 may create the complete set of image instance even if not all of the image instances will be initially deployed. Control node 12 creates different sets of image instances for different computing functions, and each set may have a different number of image instances depending on the maximum number of image instances that may be deployed for each set. Control node 12 stores the image instances within software image repository 26. Each image instance represents a collection of bits that may be deployed on an application node.

[0057] Further details of software image management are described in co-pending U.S. patent application Ser. No. 11/046,133, entitled "MANAGEMENT OF SOFTWARE IMAGES FOR COMPUTING NODES OF A DISTRIBUTED COMPUTING SYSTEM,"

Netflix, Inc. - Ex. 1002, Page 000703
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

filed January 28, 2005 and co-pending U.S. patent application Ser. No. 11/046,152, entitled "UPDATING SOFTWARE IMAGES ASSOCIATED WITH A DISTRIBUTED COMPUTING SYSTEM," filed January 28, 2005, each of which is incorporated herein by reference in its entirety.

[0058] In general, distributed computing system 10 conforms to a multi-level, hierarchical organizational model that includes four distinct levels: fabric, domains, tiers and nodes. Control node 12 is responsible for all levels of management, including fabric management, domain creation, tier creation and node allocation and deployment.

[0059] As used herein, the "fabric" level generally refers to the logical constructs that allow for definition, deployment, partitioning and management of distinct enterprise applications. In other words, fabric refers to the integrated set of hardware, system software and application software that can be "knitted" together to form a complete enterprise system. In general, the fabric level consists of two elements: fabric components or fabric payload. Control node 12 provides fabric management and fabric services as described herein.

[0060] In contrast, a "domain" is a logical abstraction for containment and management within the fabric. The domain provides a logical unit of fabric allocation that enables the fabric to be partitioned amongst multiple uses, e.g. different business services.

[0061] Domains are comprised of tiers, such as a 4-tier application model (web server, application server, business logic, persistence layer) or a single tier monolithic application. Fabric domains contain the free pool of devices available for assignment to tiers.

[0062] A tier is a logically associated group of fabric components within a domain that share a set of attributes: usage, availability model or business service mission. Tiers are used to define structure within a domain e.g. N-tier application, and each tier represents a different computing function. A user, such as administrator 20, typically defines the tier structure within a domain. The hierarchical architecture may provide a high degree of flexibility in mapping customer applications to logical models which run within the fabric environment. The tier is one construct in this modeling process and is the logical container of application resources.

[0063] The lowest level, the node level, includes the physical components of the fabric. This includes computing nodes that, as described above, provide operating environments for system applications and enterprise software applications. In addition, the node level may

Netflix, Inc. - Ex. 1002, Page 000704
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

include network devices (e.g., Ethernet switches, load balancers and firewalls) used in creating the infrastructure of network 18. The node level may further include network storage nodes that are network connected to the fabric.

[0064] System administrator 20 accesses administration software executing on control node 12 to logically define the hierarchical organization of distributed computing system 10. For example, system administrator 20 may provide organizational data 21 to develop a model for the enterprise and logically define the enterprise fabric. System administrator 20 may, for instance, develop a model for the enterprise that includes a number of domains, tiers, and node slots hierarchically arranged within a single enterprise fabric.

[0065] More specifically, system administrator 20 defines one or more domains that each correspond to a single enterprise application or service, such as a customer relation management (CRM) service. System administrator 20 further defines one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. As an example, system administrator 20 may define a storefront domain within the enterprise fabric that includes a web tier, an application tier and a database tier. In this manner, distributed computing system 10 may be configured to automatically provide web server functions, business application functions and database functions.

[0066] For each of the tiers, control node 12 creates a number of "node slots" equal to the maximum number of application nodes 14 that may be deployed. In general, each node slot represents a data set that describes specific information for a corresponding node, such as software resources for a physical node that is assigned to the node slot. The node slots may, for instance, identify a particular software image instance associated with an application node 14 as well as a network address associated with that particular image instance.

[0067] In this manner, each of the tiers include one or more node slots that reference particular software image instances to boot on the application nodes 14 to which each software image instance is assigned. The application nodes 14 to which control node 12 assigns the image instances temporarily inherit the network address assigned to the image instance for as long as the image instance is deployed on that particular application node. If for some reason the image instance is moved to a different application node 14, control node 12A moves the network address to that new application node.

Netflix, Inc. - Ex. 1002, Page 000705
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0068] System administrator 20 may further define specific node requirements for each tier of the fabric. For example, the node requirements specified by system administrator 20 may include a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), local disk characteristics and other hardware characteristics that may be detected on the individual computing nodes. System administrator 20 may also specify user-defined hardware attributes of the computing nodes, such as whether I/O (like HBA) is required. The user-defined hardware attributes are typically not capable of detection during an automatic inventory. In this manner, system administrator 20 creates a list of attributes that the tier requires of its candidate computing nodes. In addition, particular node requirements may be defined for software image instances.

[0069] In addition to the node requirements described above, system administrator 20 may further define policies that are used when re-provisioning computing nodes within the fabric. System administrator 20 may define policies regarding tier characteristics, such as a minimum number of nodes a tier requires, an indication of whether or not a failed node is dynamically replaced by a node from free pool 13, a priority for each tier relative to other tiers, an indication of whether or not a tier allows nodes to be re-provisioned to other tiers to satisfy processing requirements by other tiers of a higher priority or other policies. Control node 12 uses the policy information input by system administrator 20 to re-provision computing nodes to meet tier processing capacity demands.

[0070] After receiving input from system administrator 20 defining the architecture and policy of the enterprise fabric, control node 12 identifies unallocated nodes within free pool 13 that satisfy required node attributes. Control node 12 automatically assigns unallocated nodes from free pool 13 to respective tier node slots of a tier. As will be described in detail herein, in one embodiment, control node 12 may assign computing nodes to the tiers in a "best fit" fashion. Particularly, control node 12 assigns computing nodes to the tier whose node attributes most closely match the node requirements of the tier as defined by administrator 20. The assignment of the computing nodes may occur on a tier-by-tier basis beginning with a tier with the highest priority and ending with a tier with the lowest priority. Alternatively, or in addition, assignment of computing nodes may be based on dependencies defined between tiers.

Netflix, Inc. - Ex. 1002, Page 000706
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0071] As will be described in detail below, control node 12 may automatically add unallocated nodes from free pool 13 to a tier when more processing capacity is needed within the tier, remove nodes from a tier to the free pool when the tier has excess capacity, transfer nodes from tier to tier to meet processing demands, or replace failed nodes with nodes from the free pool. Thus, computing resources, i.e., computing nodes, may be automatically shared between tiers and domains within the fabric based on user-defined policies to dynamically address high-processing demands, failures and other events.

[0072] Distributed computing system 10 may provide one or more advantages. For example, distributed computing system 10 may deliver application services in a manner that is independent of the computational infrastructure which generates the application services. For example, the computation infrastructure that provides an application service to a user of distributed computing system 10 may be irrelevant to the user, so long as distributed computing system provides the application service. Distributed computing system 10 may deliver application services in a manner that is independent of the computational infrastructure by utilizing an infrastructure management facility (IMF) that may guarantee application service delivery using "service level automation."

[0073] This concept is itself based upon two concepts: The Service Delivery Model and Service Level Automation.

[0074] 1. Service Delivery Model: The concept of computing as a service is based upon the ability of the consumer to be guaranteed availability of the quantity and quality of an application service required at the time and location necessary, independent of the type of service provided (e.g. audio, video, application programs, etc). This may be achieved if the application services are provided in a manner that is independent of the location and condition of the physical and logical infrastructure which generates the application service, limited only by total capability and capacity available. This disclosure may refer to this concept as the "Service Delivery Model."

[0075] Physical and logical infrastructure facilities that provide an application service may comprise a "computing infrastructure." A "computing infrastructure" may include computing, storage and communication hardware and related software infrastructure; the application software and application containers; and other enabling facilities and services that are part of the environment necessary to generate and deliver application services.

Netflix, Inc. - Ex. 1002, Page 000707
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0076] 2. Service Level Automation: In order to implement the service delivery model, the IMF may provide service level automation. The IMF may implement the service delivery model for single and/or multiple application service flows. For example, the IMF may automatically and dynamically adjust application service levels of these application service flows based on a policy automation facility. The policy automation facility may automate delivery of application service based upon dynamically extensible policies. These dynamically extensible policies may include but are not limited to: customer, location, priority, duration, date and time of day, service type, business demands and conditions, quantity and quality of the application service. The policy automation facility may also enable independent control of the application service by a service provider and by a consumer. Furthermore, the policy automation facility may allow the consumer provide the application service or a service based on the application service to another consumer. In addition, the IMF may measure both efficiency and cost as reflected in the behavior of the product. For example, the IMF may provide the ability to make decisions that account for the best uses of the resources on a dynamic basis in order to choose between functionally equivalent alternatives that make the best usage of the economics of the infrastructure at the time. This disclosure may refer to this concept as "Service Level Automation" (SLA).

[0077] Service-level automation may enable the separation of some or all aspects of the implementation of the computing infrastructure from the delivery of application services. Separating the implementation of the computing infrastructure from the delivery of application services may allow a consumer of the application services to decrease risk and cost while increasing business agility. Implementations of service-level automation may be based upon a new abstraction which operates independent of existing software categories (e.g. applications, middleware, data base, operating systems and system management) which for the purposes of this disclosure shall be referred to as "Metaware."

[0078] The IMF may comprise Metaware that regulates the delivery of application services through service-level automation. The IMF may use service-level automation to regulate the delivery of application services in a manner that is independent of the computing infrastructure that provides the application services. Because application services are the result of applications running within distributed computing system 10, the IMF may automate application service levels and may make independent of all aspects of the

Netflix, Inc. - Ex. 1002, Page 000708
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

computing infrastructure within the limits of the capability and capacity available for delivery to the consumer. Service-level automation may provide extensible, dynamic policy-driven control of service delivery for both service providers and service consumers. The IMF may include the following six functional design capabilities:

[0079] 1. Application Service Independence: The basic concept which enables the IMF to deliver application services may be the ability of the IMF to dynamically manage service-level automation in a manner that is independent of the computing infrastructure. A key element of this is that application services themselves are independent and unmodified. The IMF may be able to accomplish this by abstracting the physical and logical computing infrastructure away from the quantity and quality of application services generated by the computing infrastructure.

[0080] 2. Scale Independence: The incremental overhead of the IMF required by any server included in distributed computing system 10 may be independent of the number of nodes in distributed computing system 10.

[0081] 3. Container Capacity Metering: Distributed computing system 10 may achieve application service independence by dynamically metering higher-level software facilities of the computing infrastructure. The higher-level software facilities may generate the application services, service flow and flux. This disclosure may refer to these facilities as "container services." Examples of container services may include, but are not limited to, virtual machine managers, operating systems, and web application servers. Furthermore, some container services may contain other container services that the IMF manages. The system may conceptualize the application service as being independent of the type of computing infrastructure (e.g., middleware and hardware nodes) that generate these services.

[0082] 4. Policy Automation: The IMF may deliver the required service-level automation through its extensible policy management facility. The policy management facility may dynamically adjust application service levels in order to implement user-defined policy. The ability to adjust application service levels may be limited only by the capacity and capability of the available computing infrastructure. The policy management facility may be capable of providing guaranteed service levels for single and/or multiple application service flows which the IMF automatically and dynamically adjust based on user-defined and extensible policies in response to changes in the environment. Examples of policies may

Netflix, Inc. - Ex. 1002, Page 000709
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

include, but are not limited to: customer, location, service type, cost and/or pricing, quantity and quality of service, priority, duration, date and time of day, business demands and conditions, and other policies. The policy management facility may also allow the service provider to manage delivery and meter the capacity that the service provider delivers to individual customers. The policy management facility may allow the service provider to manage delivery and meter capacity independent of the ability of the customer to set and manage their own policies of how their capacity is utilized. This may be effectively done by leveraging application service independence.

[0083] 5. Dynamic Provisioning: The IMF described herein may support the ability to dynamically provision at boot and run time all software in order to support installation, capacity management, service levels and other policies, and software upgrades. Offering a computing infrastructure architecture that is application service independent may create flexibility in deployment of a single instance of a service.

[0084] 6. Data Capture Capability: The IMF described herein may acquire the data necessary to enable the service provider to implement metering and auditing facilities necessary to control, audit, bill and account for services provided in complete detail. The customer may also use the ability to capture this data as a means to educate the IMF about the performance characteristics of combinations of resources in order to allow the IMF to improve the quality of its scheduling decisions over time. Improvement of the quality of scheduling decisions may enhance the ability of the IMF to provide the requisite services with high efficiency.

[0085] The described system may allow customers to conveniently acquire the guaranteed quantity and quality of application services required at competitive market-driven rates with the ability to directly control the SLA capability of the IMF in order to maximize their operations efficiency and effectiveness without having to tradeoff operation risk, cost and business agility. Potential advantages may include, but are not limited to:

[0086] 1. Efficiency and Effectiveness: The customer may not need to manage the technology, and can focus on the automation of the application service to improve their efficiency and effectiveness without having to tradeoff operation risk, cost and business agility which the IMF may enable through the service-level automation capability of the IMF.

Netflix, Inc. - Ex. 1002, Page 000710
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0087] 2. Capabilities & Convenience: By leveraging the Internet as a generic, global delivery facility and separating the computing infrastructure from the application service delivery, the IMF may simplify the consumer's life in several ways:

i. *Complexity:* The IMF may eliminate the complexity of purchasing, installing, configuring, managing and administering the computing infrastructure.

ii. *Access:* The IMF may provide ubiquitous access to a full range of computing services wherever and whenever desired from any client device capable of accessing the Internet.

iii. *Service Provider Independence:* Consumers may be free to change providers by simply transferring their software images to a new service provider.

iv. *Service Management:* Service management may enable a consumer to provide services to customers of the consumer. For example, a corporation may be able to regulate and bill their individual business units for the capacity consumed.

[0088] 3. Cost: By logically abstracting the application services from the computing infrastructure, the IMF may enable a competitive service provider industry that includes:

i. *Capacity Utilization:* A service provider using the IMF may be able to charge customers for capacity actually utilized by the customers. Presently, customers typically pay for the maximum capacity they would need at peak load.

ii. *Choice of Suppliers:* As the basic computing infrastructure facilities may be generic, the customer may have a choice of suppliers without the level of lock-in current vendors may enjoy. This may be enabled through the image and policy management implementation of the IMF. The image and policy management implementation of the IMF may allow portability across service providers through by enabling the customer to simply transfer the files to the new provider and initiating it with the IMF Metaware. The ability to easily migrate from a first service provider to a second service provider may drive the pricing competition. In this way, the ability of the IMF to provide application services independent of the computing infrastructure may enable customers to choose a service provider more effectively.

iii. *Capital Expense Leverage:* The customer may not need to own the capital equipment or software. Rather the service provider may be motivated to use the most

Netflix, Inc. - Ex. 1002, Page 000711
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

cost effective components and amortize costs of these components across a customer base of the service provider. As a result, the cost to the consumer may be lower.

iv. *Operations Expense Leverage:* As the IMF automates most of the information technology operations functions, these costs may drop dramatically.

[0089] The IMF may enable the implementation of a service delivery model based on service-level automation by service providers:

[0090] 1. A service provider may use the IMF to implement a service delivery model for application services independent of the computing infrastructure or of the service types provided. This capability may enable the service provider to deliver service-level automation on a global basis utilizing generic computing infrastructure for all types of digital services. Such digital service may include traditional applications and service flow processing, digital audio and digital video services independently to any consumer type (e.g. individual, organization or corporation) as long as the consumer has access to sufficient capacity of broadband communications for the services levels require, and other types of digital services. This capability of the IMF may also allow a consumer of an application service to, in turn, be a service provider as well.

[0091] 2. The IMF described herein may enable a service provider to meter application services that distributed computing system 10 delivers. For example, the service provider may meter application services based on the capacity of computing infrastructure consumed by customer. The service provider may meter application services in order to control, audit, bill and account for the application services provided to individual consumers of the application services. This capability may represent a uniform model of service-level automation that is independent of the capacity of computing infrastructure for some or all types of application services. As a result, a pricing model of a service provider may reflect the goal of utility computing by enabling the service provider to bill based upon a combination of application service delivery parameters. These application service delivery parameters may include, but not limited to: customer, type of service, duration, date and/or time of day, location, point of delivery, quantity and quality of service and excess or reserved capacity, and other application service delivery parameters.

[0092] 3. The IMF may also enable the service provider to leverage the economics of the service delivery model in several ways:

Netflix, Inc. - Ex. 1002, Page 000712
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

i. *Utilization:* By managing the computing infrastructure with the IMF, a service provider may be able to maintain much higher average utilization levels of the computing infrastructure. The maintenance of higher average utilization levels may dramatically reduce the amount of hardware purchased, software licensed and operations expenses.

ii. *Global Service Demand Levels:* Demand for an application service by an organization may vary by local time of day. By leveraging the global reach of the Internet, a service provider may be able to effectively manage the utilization of the computing infrastructure necessary to meet demand for the application service on a global "follow the sun" basis.

iii. *Service Digitization:* As more services become digital, service providers may deliver these services using a generic computing infrastructure architecture.

iv. *Value-Added Services:* A service provider may be free to provide value-added services beyond basic application services. A few examples of value-added services may include: a bundle of personal computer services that includes applications, storage, security, administration and related services; software as a service; compliance verification and certification, and other types of value-added services.

v. *Hardware Commoditization:* Hardware commoditization may enable a service provider to implement a computing infrastructure using generic low cost servers, storage facilities, and network switches. These hardware elements are undergoing dramatic commoditization which may allows for much lower capital expenses then in the past. This commoditization may continue for many years to come making capital expenses a much smaller part of the fixed costs for the service provider.

vi. *Automation of Operations:* One of the primary economic values of the IMF is that IMF may automate most of the runtime operations costs of the computing infrastructure. Operations expenses may constitute the majority of the costs of today. Because the IMF automates the runtime operations, the IMF may reduce runtime operations costs. Thus by leveraging the ability of the IMF to dramatically reduce the

Netflix, Inc. - Ex. 1002, Page 000713
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

capital and operations expenses, the service provider may be able to implement an service delivery model that may justify the consumers switching cost.

vii. Software Commoditization: Software pricing is being commoditized based upon several factors which may continue to accelerate. These factors may include:

1. Service-Oriented Architecture: Software systems that utilize a service-oriented architecture may accelerate innovation, allow for incremental independent innovation, eliminate down stream lock in by software vendors, and may make it possible to use commodity servers to run applications that previously required large expensive servers. Service-oriented architecture may also reduce complex applications to their most fundamental independently divisible functional services which will be the basis for the complete commoditization of software further leveraging the open source movement.

2. Open Source Software: The open source software paradigm may serve as an accelerator for the commoditization of many types of software by both compromising the pricing power of software and serving as a reusable model of software services which may further compromise the value of horizontal software services.

3. Good Enough: Application services are becoming more and more generic, thus accelerating their commoditization further in combination with service-oriented architectures and Open Source.

[0093] FIG. 2 is a schematic diagram illustrating an example embodiment of organizational data 21 that defines a model logically representing an enterprise fabric in accordance with the invention. In the example illustrated in FIG. 2, control node 12 (FIG. 1) maintains organizational data 21 to define a simple e-commerce fabric 32.

[0094] In this example, e-commerce fabric 32 includes a storefront domain 34A and a financial planning domain 34B. Storefront domain 34A corresponds to the enterprise storefront domain and allows customers to find and purchase products over a network, such as the Internet. Financial planning domain 34B allows one or more employees to perform financial planning tasks for the enterprise.

[0095] Tier level 31C includes one or more tiers within each domain that represent the functional subcomponents of applications and services provided by the domain. For

Netflix, Inc. - Ex. 1002, Page 000714
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

example, storefront domain 34A includes a web server tier (labeled "web tier") 36A, a business application tier (labeled "app tier") 36B, and a database tier (labeled "DB tier") 36C. Web server tier 36A, business application tier 36B and database tier 36C interact with one another to present a customer with an online storefront application and services. For example, the customer may interact with web server tier 36A via a web browser. When the customer searches for a product, web server tier 36A may interacts with business application tier 36B, which may in turn access a database tier 36C. Similarly, financial planning domain 34B includes a financial planning tier 36D that provides subcomponents of applications and services of the financial planning domain 34B. Thus, in this example, a domain may include a single tier.

[0096] Tier level 31D includes one or more logical node slots 38A-38H ("node slots 38") within each of the tiers. Each of node slots 38 include node specific information, such as software resources for an application node 14 that is assigned to a respective one of the node slots 38. Node slots 38 may, for instance, identify particular software image instances within image repository 26 and map the identified software image instances to respective application nodes 14. As an example, node slots 38A and 38B belonging to web server tier 36A may reference particular software image instances used to boot two application nodes 14 to provide web server functions. Similarly, the other node slots 38 may reference software image instances to provide business application functions, database functions, or financial application functions depending upon the tier to which the node slots are logically associated.

[0097] Although in the example of FIG. 2, there are two node slots 38 corresponding to each tier, the tiers may include any number of node slots depending on the processing capacity needed on the tier. Furthermore, not all of node slots 38 may be currently assigned to an application node 14. For example, node slot 28B may be associated with an inactive software image instance and, when needed, may be assigned to an application node 14 for deployment of the software image instance.

[0098] In this example, organizational data 21 associates free node pool 13 with the highest-level of the model, i.e., e-commerce fabric 32. As described above, control node 12 may automatically assign unallocated nodes from free node pool 13 to at least a portion of tier node slots 38 of tiers 36 as needed using the "best fit" algorithm described above or another algorithm. Additionally, control node 12 may also add nodes from free pool 13 to a tier

Netflix, Inc. - Ex. 1002, Page 000715
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

when more processing capacity is needed within the tier, remove nodes from a tier to free pool 13 when a tier has excess capacity, transfer nodes from tier to tier to meet processing demands, and replace failed nodes with nodes from the free tier.

[0099] Although not illustrated, the model for the enterprise fabric may include multiple free node pools. For example, the model may associate free node pools with individual domains at the domain level or with individual tier levels. In this manner, administrator 20 may define policies for the model such that unallocated computing nodes of free node pools associated with domains or tiers may only be used within the domain or tier to which they are assigned. In this manner, a portion of the computing nodes may be shared between domains of the entire fabric while other computing nodes may be restricted to particular domains or tiers.

[0100] FIG. 3 is a flow diagram that provides a high-level overview of the operation of control node 12 when configuring distributed computing system 10. Initially, control node 12 receives input from a system administrator defining the hierarchical organization of distributed computing system 10 (50). In one example, control node 12 receives input that defines a model that specifies a number of hierarchically arranged nodes as described in detail in FIG. 2. Particularly, the defined architecture of distributed computing system 10 includes an overall fabric having a number of hierarchically arranged domains, tiers and node slots.

[0101] During this process, control node 12 may receive input specifying node requirements of each of the tiers of the hierarchical model (52). As described above, administrator 20 may specify a list of attributes, e.g., a central processing unit (CPU) count, a CPU speed, an amount of memory (e.g., RAM), or local disk characteristics, that the tiers require of their candidate computing nodes. In addition, control node 12 may further receive user-defined custom attributes, such as requiring the node to have I/O, such as HBA connectivity. The node requirements or attributes defined by system administrator 20 may each include a name used to identify the characteristic, a data type (e.g., integer, long, float or string), and a weight to define the importance of the requirement.

[0102] Control node 12 identifies the attributes for all candidate computing nodes within free pool 13 or a lower priority tier (54). As described above, control node 12 may have already discovered the computing nodes and inventoried the candidate computing nodes to identify

Netflix, Inc. - Ex. 1002, Page 000716
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

hardware characteristics of all candidate computing nodes. Additionally, control node 12 may receive input from system administrator 20 identifying specialized capabilities of one or more computing nodes that are not detectable by the inventory process.

[0103] Control node 12 dynamically assigns computing nodes to the node slots of each tier based on the node requirements specified for the tiers and the identified node attributes (56). Population of the node slots of the tier may be performed on a tier-by-tier basis beginning with the tier with the highest priority, i.e., the tier with the highest weight assigned to it. As will be described in detail, in one embodiment, control node 12 may populate the node slots of the tiers with the computing nodes that have attributes that most closely match the node requirements of the particular tiers. Thus, the computing nodes may be assigned using a "best fit" algorithm.

[0104] FIG. 4 is a flow diagram illustrating exemplary operation of control node 12 when assigning computing nodes to node slots of tiers. Initially, control node 12 selects a tier to enable (60). As described above, control node 12 may select the tier based on a weight or priority assigned to the tier by administrator 20. Control node 12 may, for example, initially select the tier with the highest priority and successively enable the tiers based on priority.

[0105] Next, control node 12 retrieves the node requirements associated with the selected tier (62). Control node 12 may, for example, maintain a database having entries for each node slot, where the entries identify the node requirements for each of the tiers. Control node 12 retrieves the node requirements for the selected tier from the database.

[0106] In addition, control node 12 accesses the database and retrieves the computing node attributes of one of the unallocated computing nodes of free pool 13. Control node 12 compares the node requirements of the tier to the node attributes of the selected computing node (64).

[0107] Based on the comparison, control node 12 determines whether the node attributes of the computing node meets the minimum node requirements of the tier (66). If the node attributes of the selected computing node do not meet the minimum node requirements of the tier, then the computing node is removed from the list of candidate nodes for this particular tier (68). Control node 12 repeats the process by retrieving the node attributes of another of the computing nodes of the free pool and compares the node requirements of the tier to the node attributes of the computing node.

Netflix, Inc. - Ex. 1002, Page 000717
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0108] If the node attributes of the selected computing node meet the minimum node requirements of the tier (YES of 66), control node 12 determines whether the node attributes are an exact match to the node requirements of the tier (70). If the node attributes of the selected computing node and the node requirements of the tier are a perfect match (YES of 70), the computing node is immediately assigned from the free pool to a node slot of the tier and the image instance for the slot is associated with the computing node for deployment (72).

[0109] Control node 12 then determines whether the node count for the tier is met (74). Control node 12 may, for example, determine whether the tier is assigned the minimum number of nodes necessary to provide adequate processing capabilities. In another example, control node 12 may determine whether the tier is assigned the ideal number of nodes defined by system administrator 20. When the node count for the tier is met, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority, and repeats the process until all defined tiers are enabled, i.e., populated with application nodes (60).

[0110] If the node attributes of the selected computing node and the node requirements of the tier are not a perfect match control node 12 calculates and records a "processing energy" of the node (76). As used herein, the term "processing energy" refers to a numerical representation of the difference between the node attributes of a selected node and the node requirements of the tier. A positive processing energy indicates the node attributes more than satisfy the node requirements of the tier. The magnitude of the processing energy represents the degree to which the node requirements exceed the tier requirements.

[0111] After computing and recording the processing energy of the nodes, control node 12 determines whether there are more candidate nodes in free pool 13 (78). If there are additional candidate nodes, control node 12 repeats the process by retrieving the computing node attributes of another one of the computing nodes of the free pool of computing nodes and comparing the node requirements of the tier to the node attributes of the computing node (64).

[0112] When all of the candidate computing nodes in the free pool have been examined, control node 12 selects the candidate computing node having the minimum positive processing energy and assigns the selected computing node to a node slot of the tier (80). Control node 12 determines whether the minimum node count for the tier is met (82). If the

Netflix, Inc. - Ex. 1002, Page 000718
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

minimum node count for the tier has not been met, control node 12 assigns the computing node with the next lowest calculated processing energy to the tier (80). Control node 12 repeats this process until the node count is met. At this point, control node 12 selects the next tier to enable, e.g., the tier with the next largest priority (60).

[0113] In the event there are an insufficient number of computing nodes in free pool 13, or an insufficient number of computing nodes that meet the tier requirements, control node 12 notifies system administrator 20. System administrator 20 may add more nodes to free pool 13, add more capable nodes to the free pool, reduce the node requirements of the tier so more of the unallocated nodes meet the requirements, or reduce the configured minimum node counts for the tiers.

[0114] FIG. 5 is a flow diagram illustrating exemplary operation of control node 12 when adding an additional computing node to a tier to meet increased processing demands. Initially, control node 12 or system administrator 20 identifies a need for additional processing capacity on one of the tiers (90). Control node 12 may, for example, identify a high processing load on the tier or receive input from a system administrator identifying the need for additional processing capacity on the tier.

[0115] Control node 12 then determines whether there are any computing nodes in the free pool of nodes that meet the minimum node requirements of the tier (92). When there are one or more nodes that meet the minimum node requirements of the tier, control node 12 selects the node from the free pool based the node requirements of the tier, as described above, (94) and assigns the node to the tier (95). As described in detail with respect to FIG. 4, control node 12 may determine whether there are any nodes that have node attributes that are an exact match to the node requirements of the tier. If an exact match is found, the corresponding computing node is assigned to a node slot of the tier. If no exact match is found, control node 12 computes the processing energy for each node and assigns the computing node with the minimum processing energy to the tier. Control node 12 remotely powers on the assigned node and remotely boots the node with the image instance associated with the node slot. Additionally, the booted computing node inherits the network address associated with the node slot.

Netflix, Inc. - Ex. 1002, Page 000719
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0116] If there are no adequate computing nodes in the free pool, i.e., no nodes at all or no nodes that match the minimal node requirements of the tier, control node 12 identifies the tiers with a lower priority than the tier needing more processing capacity (96).

[0117] Control node 12 determines which of the nodes of the lower priority tiers meet the minimum requirements of the tier in need of processing capacity (98). Control node 12 may, for example, compare the attributes of each of the nodes assigned to node slots of the lower priority tiers to the node requirements of the tier in need of processing capacity. Lower priority tiers that have the minimum number of computing nodes may be removed from possible tiers from which to harvest an application node. If, however, all the lower priority tiers have the minimum number of computing nodes defined for the respective tier, the lowest priority tier is selected from which to harvest the one or more nodes.

[0118] Control node 12 calculates the processing energy of each of the nodes of the lower priority tiers that meet the minimum requirements (100). The energies of the nodes are calculated using the differences between the node attributes and the node requirements of the tier needing additional capacity. Control node 12 selects the computing node with the lowest processing energy that meets the minimum requirements, and assigns the selected computing node to the tier in need of processing capacity (102, 95).

[0119] FIG. 6 is a flow diagram illustrating exemplary operation of control node 12 when harvesting excess node capacity from one of the tiers and returning the harvested computing node to free pool 13. Initially, control node 12 identifies a tier having excess node capacity (110). Control node 12 may, for example, periodically check the node capacity of the tiers to identify any tiers having excess node capacity. Performing a periodic check and removal of excess nodes increases the likelihood that a capable computing node will be in free pool 13 in the event one of the tiers needs additional node capacity.

[0120] When harvesting a node, control node 12 calculates the processing energy of all the nodes in the tier as described above with reference to FIG. 4 (112). Control node 12 identifies the node within the tier with the highest processing energy and returns the identified node to the free pool of nodes (114, 116). As described above, the node with the highest processing energy corresponds to the node whose node attributes are the most in excess of the node requirements of the tier.

Netflix, Inc. - Ex. 1002, Page 000720
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0121] Returning the node to the free pool may involve remotely powering off the computing node and updating the database to associate the harvested node with free pool 13. In addition, control node 12 updates the database to disassociate the returned node with the node slot to which it was assigned. At this point, the node no longer uses the network address associated with the image instance mapped to the node slot. Control node 12 may, therefore, assign a temporary network address to the node while the node is assigned to free pool 13.

[0122] FIG. 7 is a screen illustration of an exemplary user interface 120 presented by control node 12 with which administrator 20 interacts to define tiers for a particular domain. In the example illustrated in FIG. 7, system administrator 20 has selected the "Collage Domain." User interface 120 presents the tiers that are currently in the selected domain. In the example illustrated, the Collage Domain includes three tiers, "test tier 1," "test tier 2," and "test tier 3." As shown in FIG. 7, in this example, each of the tiers includes two nodes. In addition, user interface 120 lists the type of software image currently deployed to application nodes for each of the tiers. In the example illustrated, image "applone (1.0.0)" is deployed to the nodes of test tier 1 and image "appltwo (1.0.0)" is deployed to the nodes of test tier 2. System administrator 20 may add one or more tiers to the domain by clicking on new tier button 122.

[0123] FIG. 8 is a screen illustration of an exemplary user interface 130 for defining properties of the tiers. In particular, user interface 130 allows system administrator 20 to input a name for the tier, a description of the tier, and an image associated with the tier. The image associated with the tier refers to a golden image from which image instances are generated and deployed to the nodes assigned to the tier.

[0124] When configuring a tier, system administrator 20 may elect to activate email alerts. For example, system administrator 20 may activate the email alerts feature in order to receive email alerts providing system administrator 20 with critical and/or non-critical tier information, such as a notification that a tier has been upgraded, a node of the tier has failed or the like. Furthermore, system administrator 20 may input various policies, such node failure rules. For example, system administrator 20 may identify whether control node 12 should reboot a node in case of failure or whether the failed node should automatically be moved to maintenance pool 17. Similarly, system administrator 20 may identify whether nodes assigned to the tier may be harvested by other tiers.

Netflix, Inc. - Ex. 1002, Page 000721
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0125] User interface 130 may also allow system administrator 20 to input node requirements of a tier. In order to input node requirements of a tier, system administrator 20 may click on the "Requirements" tab 132, causing user interface 130 to present an input area to particular node requirements of the tier.

[0126] FIG. 9 is a screen illustration of an exemplary user interface 140 for viewing and identifying properties of a computing node. User interface 140 allows system administrator 20 to define a name, description, and location (including a rack and slot) of a computing node. In addition user interface 140 may specify user-defined properties of a node, such as whether the computing node has I/O HBA capabilities.

[0127] User interface 140 also displays properties that control node 12 has identified during the computing node inventory process. In this example, user interface 140 presents system administrator 20 with the a CPU node count, a CPU speed, the amount of RAM, the disk size and other characteristics that are identifiable during the automated node inventory. User interface 140 additionally presents interface information to system administrator 20. Specifically, user interface 140 provides system administrator 20 with a list of components and their associated IP and MAC addresses.

[0128] User interface 140 also allows system administrator 20 to define other custom requirements. For example, system administrator 20 may define one or more attributes and add those attributes to the list of node attributes presented to system administrator 20.

[0129] FIG. 10 is a screen illustration of an exemplary user interface 150 for viewing software images. User interface 150 presents to a system administrator or another user a list of images maintained by control node 12 within image repository 26. The image list further includes the status of each image (i.e., either active or inactive), the version of the image, the operating system on which the image should be run, the operating system version on which the image should be run and a brief description of the image.

[0130] System administrator 20 or another user may select an image by clicking on the box in front of the image identifier/name and perform one or more actions on the image. Actions that system administrator 20 may perform on an image include deleting the image, updating the image, and the like. System administrator 20 may select one of the image actions via dropdown menu 152. In some embodiments, user interface 150 may further display other

Netflix, Inc. - Ex. 1002, Page 000722
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

details about the images such as the node to which the images are assigned (if the node status is "active"), the network address associated with the images and the like.

[0131] FIG. 11 is a screen illustration of an exemplary user interface 160 for viewing a hardware inventory report. User interface 160 presents to system administrator 20 or another user a list of the nodes that are currently assigned to a domain. System administrator 20 may elect to view the nodes for the entire domain, for a single tier within the domain or for a single rack within a tier.

[0132] For each node, user interface 160 presents a node ID, a status of the node, the tier to which the node belongs, a hostname associated with the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0133] System administrator 20 or other user may select a node by clicking on the box in front of the node identifier/name and perform one or more actions on the node. Actions that system administrator 20 may perform on the node include deleting the node, updating the node attributes or other properties of the node, and the like. System administrator 20 may select one of the node actions via dropdown menu 162.

[0134] FIG. 12 is a screen illustration of an exemplary user interface 170 for viewing discovered nodes that are located in discovered pool 11. For each node, user interface 170 presents a node ID, a state of the node, a NIC IP address, a rack location, a slot location, the number of CPU's of the node, the amount of RAM on the node, the number of disks on the node, whether the node has I/O HBA, and the number of NICs of the node.

[0135] FIG. 13 is a screen illustration of an exemplary user interface 180 for viewing users of distributed computing system 10. User interface 180 presents a list of users as well as the role assigned to each of the users and the status of each of the users. Thus, system administrator 20 may define different roles to each of the users. For example, a user may be either an operator (i.e., general user) or an administrator. System administrator 20 may add a new user to the list of users by clicking on the "New User" button 182.

[0136] FIG. 14 is a screen illustration of an exemplary user interface 190 for viewing alerts for distributed computing system 10. For each of the alerts, user interface 190 identifies the severity of the alert, whether the alert has been acknowledged, an object associated with the

Netflix, Inc. - Ex. 1002, Page 000723
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

alert, an event associated with the alert, a state of the alert, a user associated with the alert and a date associated with the alert.

[0137] System administrator 20 or other user may select an alert by clicking on the box in front of the logged alert and perform one or more actions on the logged alert. Actions that system administrator 20 may perform include deleting the alert, changing the status of the alert, or the like. System administrator 20 may specify the log actions via dropdown menu 192.

[0138] FIG. 15 is a block diagram illustrating one embodiment of control node 12 in further detail. In the illustrated example, control node 12 includes a monitoring subsystem 202, a service level automation infrastructure (SLAI) 204, and a business logic tier (BLT) 206.

[0139] Monitoring subsystem 202 provides real-time monitoring of the distributed computing system 10. In particular, monitoring subsystem 202 dynamically collects status data 203 from the hardware and software operating within distributed computing system 10, and feeds the status data in the form of monitor inputs 208 to SLAI 204. Monitoring inputs 208 may be viewed as representing the actual state of the fabric defined for the organizational model implemented by distributed computing system 10. Monitoring subsystem 202 may utilize well defined interfaces, e.g., the Simple Network Management Protocol (SNMP) and the Java Management Extensions (JMX), to collect and export real-time monitoring information to SLAI 204.

[0140] SLAI 204 may be viewed as an automation subsystem that provides support for autonomic computing and acts as a central nervous system for the controlled fabric. In general, SLAI 204 receives monitoring inputs 208 from monitoring subsystem 202, analyzes the inputs and outputs appropriate action requests 212 to BLT 206. In one embodiment, SLAI 204 is a cybernetic system that controls the defined fabric via feedback loops. More specifically, administrator 20 may interact with BLT 206 to define an expected state 210 for the fabric. BLT 206 communicates expected state 210 to SLAI 204. SLAI 204 receives the monitoring inputs from monitoring subsystem 202 and applies rules to determine the most effective way of reducing the differences between the expected and actual states for the fabric.

[0141] For example, SLAI 204 may apply a rule to determine that a node within a high priority tier has failed and that the node should be replaced by harvesting a node from a

Netflix, Inc. - Ex. 1002, Page 000724
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

lower priority tier. In this example, SLAI 204 outputs an action request 212 to invoke BLT 206 to move a node from one tier to the other.

[0142] In general, BLT 206 implements high-level business operations on fabrics, domains and tiers. SLAI 204 invokes BLT 206 to bring the actual state of the fabric into accordance with the expected state. In particular, BLT 206 outputs fabric actions 207 to perform the physical fabric changes. In addition, BLT 206 outputs an initial expected state 210 to SLAI 204 and initial monitoring information 214 to SLAI 204 and monitoring subsystem 202, respectively. In addition, BLT 206 outputs notifications 211 to SLAI 204 and monitoring subsystem 202 to indicate the state and monitoring changes to distributed computing system 10. As one example, BLT 206 may provide control operations that can be used to replace failed nodes. For example, BLT 206 may output an action request indicating that a node having address 10.10.10.10 has been removed from tier ABC and a node having address 10.10.10.11 has been added to tier XYZ. In response, monitoring subsystem 202 stops attempting to collect status data 203 from node 10.10.10.10 and starts monitoring for status data from node 10.10.10.11. In addition, SLAI 204 updates an internal model to automatically associate monitoring inputs from node 10.10.10.11 with tier XYZ.

[0143] FIG. 16 is a block diagram illustrating one embodiment of monitoring subsystem 202. In general, monitoring subsystem 202 dynamically detects and monitors a variety of hardware and software components within the fabric. For example, monitoring subsystem 202 identifies, in a timely and efficient manner, any computing nodes that have failed, i.e., any node that does not respond to a request to a known service. More generally, monitoring subsystem 202 provides a concise, consistent and constantly updating view of the components of the fabric.

[0144] As described further below, monitoring subsystem 202 employs a modular architecture that allows new detection and monitoring collectors 224 to be "plugged-in" for existing and new protocols and for existing and new hardware and software. As illustrated in FIG. 16, monitoring subsystem 202 provides a plug-in architecture that allows different information collectors 224 to be installed. In general, collectors 224 are responsible for protocol-specific collection of monitoring information. The plug-in architecture allows for new protocols to be added by simply adhering to a collector plug-in signature. In this example, monitoring subsystem 202 includes collectors 224A and 224B for collecting

Netflix, Inc. - Ex. 1002, Page 000725
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

information from operating systems and applications executing on nodes within tier A and tier B, respectively.

[0145] In one embodiment, collectors 224 are loaded at startup of control node 12 and are configured with information retrieved from BLT 206. Monitoring engine 222 receives collection requests from SLAI 204, sorts and prioritizes the requests, and invokes the appropriate one of collectors 224 based on the protocol specified in the collection requests. The invoked collector is responsible for collecting the required status data and returning the status data to monitoring engine 222. If the collector is unable to collect the requested status data, the collector returns an error code.

[0146] In one embodiment, collectors 224 are Java code compiled into a jar file and loaded with a class loader at run time. Each of collectors 224 has an associated configuration file written in a data description language, such as the extensible markup language (XML). In addition, a user may interact with BLT 206 to add run-time configuration to dynamically configure collectors 224 for specific computing environments. Each of collectors 224 expose an application programming interface (API) to monitoring engine 222 for communication and data exchange.

[0147] A user, such as a system administrator, specifies the protocol or protocols to be used for monitoring a software image when the image is created. In addition, the users may specify the protocols to be used for monitoring the nodes and each service executing on the nodes. Example protocols supported by the collectors 224 include Secure Shell (SSH), Simple Network Management Protocol (SNMP), Internet Control Message Protocol (ICMP) ping, Java Management Extensions (JMX) and the Hypertext Transfer Protocol (HTTP).

[0148] Some protocols require special privileges, e.g., root privileges, to perform the required data collection. In this case, the corresponding collectors 224 communicate with a separate process that executes as the root. Moreover, some protocols may require deployment and/or configuration of data providers within the fabric. Software agents may, for example, be installed and configured on nodes and configured on other hardware. If needed, custom in-fabric components may be deployed.

[0149] In this example, the modular architecture of monitoring subsystem 202 also supports one or more plug-in interfaces 220 for data collection from a wide range of third-party

Netflix, Inc. - Ex. 1002, Page 000726
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

monitoring systems 228. Third-party monitoring systems 228 monitor portions of the fabric and may be vendor-specific.

[0150] FIG. 17 is a block diagram illustrating one embodiment of SLAI 204 in further detail. In the illustrated embodiment, SLAI 204 is composed of three subsystems: a sensor subsystem 240, an analysis subsystem 244 and an effector subsystem 248.

[0151] In general, sensor subsystem 240 receives actual state data from monitoring subsystem 202 in the form of monitoring inputs 208 and supplies ongoing, dynamic input data to analysis subsystem 244. For example, sensor subsystem 240 is notified of physical changes to distributed computing system 10 by monitoring subsystem 202. Sensor subsystem 240 uses the state data received from monitoring subsystem 202 to maintain ongoing, calculated values that can be sent to analysis subsystem 244 in accordance with scheduler 242.

[0152] In one embodiment, sensor subsystem 240 performs time-based hierarchical data aggregation of the actual state data in accordance with the defined organization model. Sensor subsystem 240 maintains organizational data in a tree-like structure that reflects the current configuration of the hierarchical organization model. Sensor subsystem 240 uses the organizational data to perform the real-time data aggregation and map tiers and domains to specific nodes. Sensor subsystem 240 maintains the organizational data based on notifications 211 received from BLT 206.

[0153] Sensor subsystem 240 sends inputs to analysis subsystem 244 to communicate the aggregated data on a periodic or event-driven basis. Analysis subsystem 244 may register an interest in a particular aggregated data value with sensor subsystem 240 and request updates at a specified frequency. In response, sensor subsystem 240 interacts with monitoring subsystem 202 and scheduler 242 to generate the aggregated data required by analysis subsystem 244.

[0154] Sensor subsystem 240 performs arbitrary data aggregations via instances of plug-in classes (referred to as "triggers") that define the aggregations. Each trigger is registered under a compound name based on the entity being monitored and the type of data being gathered. For example, a trigger may be defined to aggregate and compute an average computing load for a tier every five minutes. Analysis subsystem 244 requests the

Netflix, Inc. - Ex. 1002, Page 000727
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

aggregated data based on the registered names. In some embodiments, analysis subsystem 244 may define calculations directly and pass them to sensor subsystem 240 dynamically.

[0155] Analysis subsystem 244 is composed of a plurality of forward chaining rule engines 246A-246N. In general, rule engines 246 match patterns in a combination of configuration data and monitoring data, which is presented by extraction agent 251 in the form of events. Events contain the aggregated data values that are sent to rule engines 246 in accordance with scheduler 242.

[0156] Sensor subsystem 240 may interact with analysis subsystem 244 via trigger listeners 247 that receives updates from a trigger within sensor subsystem 240 when specified events occur. An event may be based on system state (e.g., a node transitioning to an up or down state) or may be time based.

[0157] Analysis subsystem 244 allows rule sets to be loaded in source form and compiled at load time into discrimination networks. Each rule set specifies trigger-delivered attributes. Upon loading the rule sets, analysis subsystem 244 establishes trigger listeners 247 to receive sensor notifications and update respective working memories of rule engines 246. As illustrated in FIG. 17, each of rule engines 246 may serve a different tier defined within the fabric. Alternatively, multiple rule engines 246 may serve a single tier or a single rule engine may serve multiple tiers.

[0158] Rule engines 246 process the events and invoke action requests via calls to effector subsystem 248. In addition, rule engines 246 provide a call-back interface so that effector subsystem 248 can inform a rule engine when an action has completed. Rule engines 246 prevent a particular rule from re-firing as long as any action invoked by the rule has not finished. In general, rules contain notification calls and service invocations though either may be disabled by configuration of effector subsystem 248. BLT 206 supplies initial system configuration descriptions to seed each of rule engines 246.

[0159] In general, rule engines 246 analyze the events and discover discrepancies between an expected state of the fabric and an actual state. Each of rule engines 246 may be viewed as software that performs logical reasoning using knowledge encoded in high-level condition-action rules. Each of rule engines 246 applies automated reasoning that works forward from preconditions to goals defined by system administrator 20. For example, rule engines 246 may apply modus ponens inferences rules.

Netflix, Inc. - Ex. 1002, Page 000728
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0160] Rule engines 246 output requests to effector subsystem 248 which produce actions requests 212 for BLT 206 to resolve the discrepancies. Effector subsystem 248 performs all operations on behalf of analysis subsystem 244. For example, event generator 250, task invocation module 252 and logger 254 of effector subsystem 248 perform event generation, BLT action invocation and rule logging, respectively. More specifically, task invocation module 252 invokes asynchronous operations within BLT 206. In response, BLT 206 creates a new thread of control for each task which is tracked by a unique task identifier (task id). Rules engine 246 uses the task id to determine when a task completes and, if needed, to re-fire any rules that were pended until completion of the task. These tasks may take arbitrary amounts of time, and rules engine 246 tracks the progress of individual task via change notifications 211 produced by BLT 206.

[0161] Event generator 250 creates persistent event records of the state of processing of SLAI 204 and stores the event records within a database. Clients uses these event records to track progress and determine the current state of the SLAI 204.

[0162] Logger 254 generates detailed trace information about system activities for use in rule development and debugging. The logging level can be raised or lowered as needed without changing operation of SLAI 204.

[0163] FIG. 18 is a block diagram of an example working memory 270 associated with rule engines 246. In this example, working memory 270 includes a read-only first data region 272 that stores the expected state received from BLT 206. Data region 272 is read-only in the sense that it cannot be modified in response to a trigger from sensor subsystem 240 or by rule engines 246 without notification from BLT 206.

[0164] In addition, working memory 270 includes a second data region 274 that is modifiable (i.e., read/write) and may be updated by monitoring subsystem 202 or used internally by rule engines 246. In general, data region 274 stores aggregated data representing the actual state of the fabric and can be updated by sensor subsystem 240 or by rule engines 246. The actual state may consist of a set of property annotations that can be attached to objects received from BLT 206 or to objects locally defined within a rule engine, such as local object 276.

[0165] FIG. 19 is a block diagram illustrating an example embodiment for BLT 206. In this example, BLT 206 includes a set of one or more web service definition language (WSDL)

Netflix, Inc. - Ex. 1002, Page 000729
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

interfaces 300, a report generator 302, a fabric administration interface service 304, a fabric view service 306, a user administration service 308, a task interface 311, a task manager 312 and an event subsystem 315.

[0166] As described, BLT 206 provides the facilities necessary to create and administer the organizational model (e.g., fabric, domains, tiers and nodes) implemented by distributed computing system 10. In general, BLT 206 abstracts access to the persisted configuration state of the fabric, and controls the interactions with interfaces to fabric hardware services. As such, BLT 206 provides fabric management capabilities, such as the ability to create a tier and replace a failed node. WSDL interfaces 300 provide web service interfaces to the functionality of BLT 206 that may be invoked by web service clients 313. Many of WSDL interfaces 300 offered by BLT 206 allow administrator 20 to define goals, such as specifying a goal of the expected state of the fabric. As further described below, rule engines 246 within SLAI 204, in turn, invoke task manger 312 to initiate one or more BLT tasks to achieve the specified goal. In general, web service clients 313 may be presentation layer applications, command line applications, or other clients.

[0167] BLT 206 abstracts all interaction with physical hardware for web service clients 313. BLT 206 is an enabling component for autonomic management behavior, but does not respond to real-time events that either prevent a goal from being achieved or produce a set of deviations between the expected state and the actual state of the system. In contrast, BLT 206 originates goals for autonomic reactions to changing configuration and state. SLAI 204 analyzes and acts upon these goals along with real-time state changes. BLT 206 sets the goals to which SLAI 204 strives to achieve, and provides functionality used by the SLAI in order to achieve the goals.

[0168] In general, BLT 206 does not dictate the steps taken in pursuit of a goal since these are likely to change based on the current state of distributed computing system 10 and changes to configurable policy. SLAI 204 makes these decisions based on the configured rule sets for the fabric and by evaluating monitoring data received from monitoring subsystem 202.

[0169] Fabric administration service 304 implements a set of methods for managing all aspects of the fabric. Example methods include methods for adding, viewing, updating and removing domains, tiers, nodes, notifications, assets, applications, software images,

Netflix, Inc. - Ex. 1002, Page 000730
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

connectors, and monitors. Other example methods include controlling power at a node, and cloning, capturing, importing, exporting or upgrading software images. Rule engines 246 of SLAI 204 may, for example, invoke these methods by issuing action requests 212.

[0170] Task manager 312 receives action requests 212 via task interface 311. In general, task interface 311 provides an interface for receiving action requests 212 from SLAI 204 or other internal subsystem. In response, task manager 312 manages asynchronous and long running actions that are invoked by SLAI 204 to satisfy a goal or perform an action requested by a client.

[0171] Task manager 312 generates task data 310 that represents identification and status for each task. Task manager 312 returns a task identifier to the calling web service clients 313 or the internal subsystem, e.g., SLAI 204, that initiated the task. Rule engines 246 and web service clients 313 use the task identifiers to track progress and retrieve output, results, and errors associated with achieving the goal.

[0172] In one embodiment, there are no WSDL interfaces 300 for initiating specific tasks. Rather, administrator 20 interacts with BLT 206 though goal interfaces presented by WSDL interfaces 300 to define the goals for the fabric. In contrast, the term task is used to refer to internal system constructs that require no user interaction. Tasks are distinct, low-level units of work that affect the state of the fabric. SLAI 204 may combine tasks to achieve or maintain a goal state.

[0173] For example, administrator 20 can request configuration changes by either adding new goals to an object or by modifying the attributes on existing goals. Scheduled goals apply a configuration at a designated time. For example, the goals for a particular tier may specify the minimum, maximum, and target node counts for that tier. As a result, the tier can increase or decrease current node capacity by scheduling goals with different configuration values.

[0174] This may be useful, for example, in scheduling a software image upgrade. As another example, entire domains may transition online and offline per a defined grid schedule. Administrator 20 may mix and match goals on a component to achieve configurations specific to the application and environment. For example, a tier that does not support autonomic node replacement would not be configured with a harvesting goal.

Netflix, Inc. - Ex. 1002, Page 000731
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0175] In some embodiments, goals are either "in force" or "out of force." SLAI 204 only works to achieve and maintain those goals that are currently in force. SLAI 204 may applies a concept of "gravity" as the goals transition from in force to out of force. For example, SLAI 204 may transition a tier offline when an online goal is marked out of force. Some goal types may have prerequisite goals. For example, an image upgrade goal may require as a prerequisite that a tier be transitioned to offline before the image upgrade can be performed. In other embodiments, goals are always in force until modified.

[0176] SLAI 204 may automatically formulate dependencies between goals or may allow a user to specify the dependencies. For example, a user may request that a newly created tier come online. As a result of this goal, SLAI 204 may automatically direct task manager 312 to generate a task of harvesting a target number of nodes to enable the tier. Generally, all goals remain in-force by SLAI 204 until modified by BLT 206. In one embodiment, each goal remains in-force in one of three states: Satisfied, Warning, or Critical depending on how successful SLAI 204 was in achieving the goal at the time the event record was generated and stored.

[0177] In this manner, SLAI 204 controls the life cycle of a goal (i.e., the creation, scheduling, update, deletion of the goal), and provides a common implementation of these and other services such as timeout, event writing, goal conflicts, management of intra-goal dependencies, and tracking tasks to achieving the goals.

[0178] Progress toward a goal is tracked though event subsystem 315. In particular, event subsystem 315 tracks the progress of each in force goal based on the goal identifiers. Tasks executed to achieve a particular goal produce events to communicate result or errors. The events provide a convenient time-based view of all actions and behaviors.

[0179] Examples of goal types that may be defined by administrator 20 include software image management goals, node allocation goals, harvest goals, tier capacity goals, asset requirement goals, tier online / offline goals, and data gathering goals.

[0180] In one embodiment, BLT 206 presents a task interface to SLAI 204 for the creation and management of specific tasks in order to achieve the currently in force goals. In particular, rule engines 246 invoke the task interface based on evaluation of the defined rule sets in view of the expected state and actual state for the fabric. Example task interfaces include interfaces to: reserve node resources; query resources for a node slot; associate or

Netflix, Inc. - Ex. 1002, Page 000732
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

disassociate an image with a node in a tier node slot; allocate, de-allocate, startup or shutdown a node; move a node to a tier; apply, remove or cycle power of a node; create a golden image; create or delete an image instance; and delete an activity, node or tier.

[0181] Report generator 302 provides an extensible mechanism for generating reports 314. Typical reports include image utilization reports that contain information with respect to the number of nodes running each software image, inventory reports detailing both the logical and physical aspects of the fabric, and system event reports showing all events that have occurred within the fabric. Report generator 302 gathers, localizes, formats and displays data into report form for presentation to the user. Report generator 302 may include one or more data gathering modules (not shown) that gather events in accordance with a schedule and update an events table to record the events. The data gathering modules may write the events in XML format.

[0182] FIG. 20 is a block diagram illustrating one embodiment of a rule engine 246 (FIG. 17). In the illustrated embodiment, rule engine 246 includes a rule compiler 344 and an execution engine 346. Each of rules 342 represents a unit of code that conforms to a rule language and expresses a set of triggering conditions and a set of implied actions. When the conditions are met, the actions are eligible to occur. The following is one example of a configuration rule:

```
rule checkTierLoad {
      Tier t where status != "overloaded";
      LoadParameter p where app == t.app && maxload < t.load;
} -> {
      modify t {
            status: "overloaded";
      };
}
```

When translated, this example rule marks a tier as overloaded if an application is implemented by the tier and the maximum specified load for the application has been exceeded. Another example rule for outputting a notification that a tier is overloaded and automatically invoking a task within BLT 206 to add a node is:

```
rule tierOverloadNotify {
      Tier t where status == "overloaded";
} -> {
```


Netflix, Inc. - Ex. 1002, Page 000733
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
        notify "Tier: " + t + "is overloaded.";
        BLT.addNode(f);
    }
```

[0183] Rule compiler 344 compiles each of rules 344 and translates match conditions of the rules into a discrimination network that avoids redundant tests during rule execution. Execution engine 346 handles rule administration, object insertion and retrieval, rule invocation and execution of rule actions. In general, execution engine 346 first matches a current set of rules 342 against a current state of working memory 348 and local objects 350. Execution engine 346 then collects all rules that match as well as the matched objects and selects a particular rule instantiation to fire. Next, execution engine 346 fires (executes) the instantiated rule and propagates any changes to working memory 348. Execution engine 346 repeats the process until no more matching rule instantiations can be found.

[0184] Firing of a rule typically produces a very small number of changes to working memory 348. This allows sophisticated rule engines to scale by retaining match state between cycles. Only the rules and rule instantiations affected by changes are updated, thereby avoiding the bulk of the matching process. One exemplary algorithm that may be used by execution engine 346 to handle the matching process includes the RETE algorithm that creates a decision tree that combines the patterns in all the rules and is intended to improve the speed of forward-chained rule system by limiting the effort required to re-compute a conflict set after a rule is fired. One example of a RETE algorithm is described in Forgy, C.L.: 1982, 'RETE: a fast algorithm for the many pattern/many object pattern match problem', Artificial Intelligence 19, 1737, hereby incorporated by reference. Other alternatives include the TREAT algorithms, and LEAPS algorithm, as described by Miranker, D.P.: 'TREAT: A New and Efficient Match Algorithm for AI Production Systems'. ISBN 0934613710 Daniel P. Miranker, David A. Brant, Bernie Lofaso, David Gadbois: On the Performance of Lazy Matching in Production Systems. AAAI 1990: 685692, each of which is hereby incorporated by reference.

[0185] FIG. 21 is a block diagram illustrating an alternative embodiment of control unit 12 (FIG. 15). In this embodiment, control unit 12 operates substantially as described above, but includes an application matrix 350, an application governor 352, a configuration processor 354, and an application service level automation infrastructure ("application SLAI") 358. As described below, application matrix 350, application governor 352, configuration processor

Netflix, Inc. - Ex. 1002, Page 000734
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

354, and application SLAI 358 provide a framework that allows control unit 12 to autonomically control the deployment, execution and monitoring of applications across application nodes 14 of distributed computing system 10.

[0186] Application matrix 350 contains all the information needed by control unit 12 to interact with one or more applications or application servers and provide autonomic control over a set of applications. Specifically, application matrix 350 provides a logical definition for deploying and controlling the set of applications to one or more tiers within distributed computing system 10. In one embodiment, application matrix 350 is an electronic document that conforms to a data description language, e.g., the extensible markup language (XML). Application SLAI 358 includes an application rules engine 355 dedicated to processing application-level rules, i.e., forward-chaining rules to provide autonomic control over the applications defined within application matrix 350. Like rules engine 246, application rules engine 355 contains a rule compiler, an execution engine, and a working memory. In order to give effect to changes in application matrix 350, application SLAI 358 automatically updates application rules engine 355 and monitoring subsystem 202. In particular, application matrix 350 sends an alert whenever application matrix 350 changes. In response to this alert, application SLAI 358 captures application-specific attributes from application matrix 350. Specifically, application SLAI 358 captures configuration attributes and rule attributes contained in application matrix 350. Application SLAI 358 transfers any new rule attributes to the working memory of application rules engine 355 to provide autonomic control over the deployment, monitoring and the execution of the applications defined within application matrix 350. In addition, application SLAI 358 updates monitoring subsystem 202 to collect information required to control the applications In this manner, administrator 20 may continue to add new application definitions and configurations to application matrix 350 after distributed control node 12 has started.

[0187] As described in further detail below, configuration processor 354 is a software module that generates an application matrix entry based on an application definition and application configuration properties of a "staged" application. A staged application is an application that has been deployed in a staging environment and customized for subsequent deployment within distributed computing system 10. After creating the application matrix entry, administrator 20 may insert the application matrix entry into application matrix 350.

Netflix, Inc. - Ex. 1002, Page 000735
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0188] Configuration processor 354 is "pluggable." That is, administrator 20 can "plug in" different implementations of configuration processor 354 as needed. For example, administrator 20 may need to "plug in" a different implementation of configuration processor 354 to handle applications that do not use an application server.

[0189] Application governor 352 is a software engine that performs application-level actions based on requests received from application rules engine 355. In this manner, BLT 206 effects fabric-level actions (e.g., deployment and monitoring of nodes and images) based on request from fabric-level rules engines 246 (FIG. 17), while matrix governor 352 performs application-level actions (e.g., deployment and monitoring of applications) based on requests from application rules engine 355.

[0190] Application governor 352 uses application matrix 350 as a source of parameters when carrying out the application-level operations requested by application rules engine 355. For example, application rules engine 355 may detect that a node to which the application is not deployed is ready for use by the application. As a result, application rules engine 355 directs application governor 352 to handle the details of deploying the application to the node. In turn, application governor 352 accesses application matrix 350 to retrieve application-specific parameters necessary to deploy the application. Storing application-specific parameters in application matrix 350 allows the application-specific parameters to change without having to recompile the rules within working memory of application rules engine 355.

[0191] Application governor 352 performs a similar procedure to undeploy an application. That is, application rules engine 355 may detect that a second application needs to use a node more than a first application that is currently deployed to the node. In this situation, application rules engine 355 sends an instruction to application governor 352 to undeploy the first application and deploy the second application. To carry out this instruction, application governor 352 accesses application matrix 350 to discover configuration parameters of both applications. Application governor 352 then uses the discovered configuration parameters to communicate with the applications.

[0192] Like configuration processor 354, application governor 352 is also "pluggable." That is, administrator 20 can easily install or remove implementations of application governor 352 depending on the circumstances. Because application governor 352, and configuration

Netflix, Inc. - Ex. 1002, Page 000736
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

processor 354 represent interchangeable, plug-in modules, the other parts of control unit 12 and system 10, including application rules engine 355, can remain generic and application neutral while providing autonomic control over distributed computing system 10.

[0193] FIG. 22 provides a conceptual view of an exemplary application matrix 350. Although application matrix 350 is typically represented in an encoded, electronic document (e.g., an XML document), FIG. 22 provides a conceptual view for ease of illustration.

[0194] In this example, application matrix 350 contains seven columns and two rows. Each row represents a different application entry for deployment within distributed computing system 10. In FIG. 22, only the first entry is shown in detail.

[0195] Each of the columns represents a different category of elements that make up an application's logical definition. In this example, the columns include:

(1) an application column 360 that includes elements generally related to the deployment of the application,

(2) an application nodes column 362 that contains elements related to the tier node slots to which the application may be assigned,

(3) a services column 364 that contains elements related to the executable services launched when the application is deployed,

(4) a node monitor values column 366 that contains elements related to attributes of the nodes that are to be monitored after the application is deployed to that node,

(5) a service monitored attributes column 368 that contains elements related to attributes of the services that are to be monitored after the application is deployed,

(6) a service levels column 370 that contains elements related to attributes for use when constructing rules to monitor execution of the services, and

(7) a deployment constraints column 372 that contains elements related to attributes for use when constructing rules to control deployment of the application.

Different types of applications may have different elements in each column, and different numbers of columns.

[0196] In the example of FIG. 22, application matrix 350 contains two applications 360 "DataDomain" 374 and "PortalDomain" 376. DataDomain application 374 has eleven attributes that define how control node 12 accesses and launches the application. For instance, the "adminIP" and "adminPort" attributes instruct governor 352 as to which the

Netflix, Inc. - Ex. 1002, Page 000737
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

server and port hosts the administrative part of the logically defined application. Other attributes like "maxNodes" and "minNodes" instruct application rules engine 355 to run the application no less than minNodes and no more than maxNodes. Applications other than application 374 may have different numbers or types of attributes. In XML format, the attributes of application 374 may appear as follows:

```
<WebServerDomain
      name="DataDomain"
      adminIP="172.31.64.201"
      adminPort="1100"
      adminTier="Web Admin"
      clusterName="PathCluster"
      expectedStartupDelay="120"
      loadDelay="120"
      maxNodes="2"
      minNodes="1"
      nodeManagerPort="5811"
      nodeTier="Web App"
>
```

[0197] In addition to these attributes, application 374 contains a series of elements (columns 362 – 372). In general, application nodes 362 contain a list of all of the available tier-node slots to which control node 12 may deploy application 374. In this example, two tier-node slots are specified. In XML, managed servers 362 appears as:

```
<ManagedServers IP="172.31.64.201" name="managedServer_0"
state="STOPPED" />
<ManagedServers IP="172.31.64.201" name="managedServer_1"
state="STOPPED" />
```

[0198] Each services 364 element identifies a service that control node 12 launches when deploying application 374 on a given application node. A service element comprises a service name and a path to a file containing the executable service. Governor 352 uses the path to locate and launch the service. For example, the following XML code indicates that governor 352 must access the file at "/lib/worklistApp/worklistApp.ear".

```
<services name="Worklist User Interface"
path="/lib/worklistApp/worklistApp.ear" />
```

[0199] Node Monitored Values 366 elements represent characteristics for use in constructing rules for monitoring nodes to which the applications are deployed. Similarly, Service Monitored Values 368 elements represent characteristics for use in constructing rules for monitoring services that are launched once the application is deployed. In this example, a "nodeMonitoredValues" element defines characteristics of a particular node that are to be

Netflix, Inc. - Ex. 1002, Page 000738
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

monitored. For instance, the amount of free memory in a node is one example of a characteristic listed as a "nodeMonitoredValues" element. On the other hand, a "serviceMonitoredValues" element is specific attribute of a service that is to be monitored. For example, the number of pending operating system-level requests for the service, the number of idle threads, etc., could be service monitored values. In a XML rendition of application matrix 350, node monitored values and service monitored values could appear as follows:

```
<nodeMonitoredValues name="Load5Average" />
<nodeMonitoredValues name="PercentMemoryFree" />
<serviceMonitoredValues name="PendingRequests" />
<serviceMonitoredValues name="ExecuteThreadIdleCount" />
<serviceMonitoredValues name="ExecuteThreadTotalCount" />
```

[0200] Deployment constraint elements 372 specify characteristics of a node under which application rules engine 355 should (or should not) deploy a service to the node. In this example, a deployment constraint element has five attributes: "attribute", "expression", "frequency", "maxThreshold", "minThreshold", and "period." The "attribute" attribute names the deployment constraint. The "expression" attribute specifies an arithmetic expression manipulating a monitored value. For example, the expression could be "PercentMemoryFree * 100", meaning monitor the value of the "PercentMemoryFree" node monitored value multiplied by 100. The "expression" attribute may specify one or more node monitored values. The "frequency" attribute informs application rules engine 355 how frequently to check the monitored value. The "maxThreshold" attribute tells application rules engine 355 to invoke a rule when the value of the expression exceeds the value specified by the "maxThreshold" attribute. Similarly, the "minThreshold" attribute tells application rules engine 355 to invoke the rule when the value of the expression drops below the value specified by the "minThreshold" attribute. Finally, the "period" attribute informs application rules engine 355 of the period over which to collect monitored value. For example, a deployment constraint element may specify that application rules engine 355 should monitor the PercentMemoryFree attribute of a node every 15 seconds for 60 seconds. If the value of PercentMemoryFree * 100 should drop below 1.0 (i.e. 1% of memory free) for 60 seconds, then application rules engine 355 should not deploy the application to that node. In XML, this rule would be represented as:

Netflix, Inc. - Ex. 1002, Page 000739
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
<deploymentConstraints attribute="FreeMemory"
expression="PercentMemoryFree*100" frequency="15" maxThreshold="-
1.0" minThreshold="1.0" period="60" />
```

[0201] Service level elements 370 have the same five attributes as deployment constraints elements: "attribute", "expression", "frequency", "maxThreshold", "minThreshold", and "period". However, the "expression" attribute deals with service monitored attributes rather than node monitored attributes. For example, a service level element may specify that application rules engine 355 check the "pendingRequest" service-monitored attribute every 15 seconds for 30 seconds. Then, if there are more than 20 pending requests for more than 30 seconds, application rules engine 355 should take the action of starting a new application. On the other hand, if there are fewer than 5 pending requests for 30 seconds, application rules engine 355 enables the action to remove the application to free up space on a node. Such a service level element could be represented as:

```
<serviceLevels attribute="PendingRequests"
expression="PendingRequests" frequency="15" maxThreshold="20.0"
minThreshold="5.0" period="30" />
```

[0202] Put together, an XML representation of an application matrix logically defining a single application for deployment within autonomically controlled distributed computing system 10 may appear as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<appMatrixRegister>
<WebServerDomain adminIP="172.31.64.201" adminPort="1100" adminTier="Web
Admin" clusterName="PathCluster" expectedStartupDelay="120"
loadDelay="120" maxNodes="2" minNodes="1" name="DataDomain"
nodeManagerPort="5811" nodeTier="Web App">
    <ManagedServers IP="172.31.64.201" name="managedServer_0"
    state="STOPPED" />

    <ManagedServers IP="172.31.64.201" name="managedServer_1"
    state="STOPPED" />
    <applications name="AI Design-time" path="/lib/ai-designtime.ear" />

    <applications name="System EJBs" path="/lib/ejbs.ear" />
```


Netflix, Inc. - Ex. 1002, Page 000740
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

```
    <applications name="Worklist Worker User Interface"
    path="/lib/worklist/worklist.ear" />

    <applications name="DBMS_ADK" path="/lib/DBMS_ADK.ear" />

    <applications name="UserApp" path="/user_projects/domains/userApp.ear"
    />

    <nodeMonitoredValues name="Load5Average" />

    <nodeMonitoredValues name="PercentMemoryFree" />

    <serviceMonitoredValues name="PendingRequests" />

    <serviceMonitoredValues name="ExecuteThreadIdleCount" />

    <serviceMonitoredValues name="ExecuteThreadTotalCount" />

    <deploymentConstraints attribute="LoadAverage"
    expression="Load5Average" frequency="15" maxThreshold="4.0"
    minThreshold="-1.0" period="15" />

    <deploymentConstraints attribute="FreeMemory"
    expression="PercentMemoryFree*100" frequency="15" maxThreshold="-1.0"
    minThreshold="1.0" period="60" />

    <serviceLevels attribute="BusyThreadPercentage"
    expression="(ExecuteThreadTotalCount-
    ExecuteThreadIdleCount)*100/ExecuteThreadTotalCount" frequency="15"
    maxThreshold="20.0" minThreshold="5.0" period="30" />

    <serviceLevels attribute="PendingRequests"
    expression="PendingRequests" frequency="15" maxThreshold="20.0"
    minThreshold="5.0" period="30" />
</WebserverDomain>
</appMatrixRegister>
```

[0203] FIG. 23 is a flowchart illustrating an exemplary series of general steps that are performed to add an application to application matrix 350 in accordance with the principles

Netflix, Inc. - Ex. 1002, Page 000741
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

of this invention. Administrator 20 first stages the application within a staging environment including deploying the application on a desired image, creating a domain, specifying any services and external system connectivity (e.g., connections to databases, queues) (380).

[0204] During this process, administrator 20 directs the application to generate an application definition file that represents an application-specific configuration based on the environment specified by the administrator. The application definition file contains information specific to an installation instance of the application. For example, the application definition file typically specifies the locations (paths) for all software components of the application as staged within the staging environment, a list of files to execute, connectivity information for the application, and other information that may be recorded by the configured application.

[0205] After staging the application, administrator 20 defines a set of application configuration properties that specify desired behavior of the application within distributed computing system 10 (382). Examples of such application configuration properties include a minimum number of nodes, minimum resource requirements for a node, deployment timing information, network addresses of tier node slots that are able to execute the application and other properties.

[0206] Next, administrator 20 directs configuration processor 354 to generate an application entry for the application using the application definition and the application configuration properties (384). The application entry contains configuration attributes used by application governor 352 to interact with the application. In addition, the application entry contains rule attributes used by application rules engine 355 that define how control node 12 monitors the deployment and execution of the application within distributed computing environment 10.

[0207] After configuration processor 354 creates the application entry, administrator 20 may modify the application entry (386). In particular, administrator 20 may update start scripts or shell scripts for the application due to path changes between the staging environment and distributed computing system 10.

[0208] Once administrator 20 has finished modifying the application entry, administrator 20 inserts the application entry into application matrix 350 (388). Because application matrix 350 detects that it has changed, application matrix 350 sends an alert to application SLAI 358 (390).

Netflix, Inc. - Ex. 1002, Page 000742
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0209] In response to the alert, application SLAI 358 may update application rules engine 355 and monitoring subsystem 202 (392). In particular, application SLAI 358 automatically scans application matrix 350. If application SLAI 358 detects new rule attributes, application SLAI 358 creates local objects reflecting the new rule attributes in the working memory of application rules engine 355. In addition, if application SLAI 358 detects new monitored values, application SLAI 358 updates monitoring subsystem 202 to add new monitoring collectors 224 (FIG. 16).

[0210] After application SLAI 358 updates application rules engine 355 and monitoring subsystem 202, control node 12 has autonomic control over the deployment of applications to tiers based on the configuration of application matrix 350 (394). For example, control node 12 deploys images, deploys applications, monitors the state of nodes and the execution of the applications, and applies tier-level rules as well as application-specific rules. Application SLAI 358 continues to listen for alerts from application matrix 350.

[0211] FIGS. 24 through 29 illustrate exemplary embodiments of distributed computing system 10 that provides autonomic control and deployment of virtual machines and virtual disk image files. For example, FIG. 24 is a block diagram illustrating an exemplary configuration of a physical node 400. Node 400 may be any node within distributed computing system 10 (FIG. 1). For instance, node 400 may be any one of application nodes 14.

[0212] As illustrated in FIG. 24, a virtual machine manager 402 is executing on node 400. As used herein, a virtual machine manager is a software program that is capable of managing one or more virtual machines. For example, virtual machine manager 402 may be an instance of VMWare ESX software produced by VMWare, Inc. of Palo Alto, California.

[0213] When deployed on physical node 200, virtual machine manager 402 hosts virtual machines 404A through 404N (collectively, virtual machines 404). The term virtual machine is used herein to refer to software that creates an environment that emulates a computer platform. For example, virtual machines 40 provide environments on which guest operating systems (OS) execute as through the operating systems were operating directly on a physical computing platform. In this manner, multiple operating systems and software applications 406A through 406N (collectively, operating systems and applications 406) may execute on virtual machines 404A through 404N, respectively, and the virtual machines 404 may

Netflix, Inc. - Ex. 1002, Page 000743
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

execute on a single physical node 400 or multiple physical nodes. For example, if virtual machine 404A emulates an x86 machine, an instance of the Microsoft Windows operating system (e.g., OS 406A) may execute on virtual machine 404A as through the instance of Windows were running on a physical x86 machine. Other software applications may be installed and configured on the guest operating system running on virtual machine 404. For example, an instance of Microsoft Internet Information Server may execute on an instance of Microsoft Windows that is executing on a virtual machine that emulates an x86 machine. In this manner, virtual machine manager 402 and virtual machines 404 provide a level of independence from the particular hardware environment provided by physical node 400.

[0214] Virtual machine manager 402 utilizes operating system data and application data for execution of each of virtual machines 404. The operating system data and application data, as well as instance-specific configuration data for the underlying virtual machines 404, is stored as respective virtual disk image files 408. From the perspective of the guest operating system, each of virtual disk image files 408 appears as a bootable hard disk of the host computer. In other words, the virtual disk image file provides a bootable software image of the operating system and applications, and includes all necessary configuration data and file structures. As illustrated in FIG. 24, virtual machine disk image file 408A provides a bootable software image for operating system/application information 406A that is capable of being executed on virtual machine 404A. For instance, if virtual machine manager 402 is an instance of VMWare ESX, virtual machine disk image files 408 may be specially captured .vmdk files.

[0215] A two-phase capture process may be used to provide autonomic control and deployment of the virtual machine disk image files 408 as well as virtual machine manager 402 to physical nodes within distributed computing system 10. In particular, control node 12 may first capture an image of virtual machine manager 402 installed on physical node 400 prior to installation and configuration of any guest operating system and applications 406. Control node 12 may utilize this captured image of virtual machine manager 402 as a "golden image" from which instance images can be created for this type virtual machine. The instance images may be stored within application matrix 350 for autonomic deployment as other software images (FIG. 21).

Netflix, Inc. - Ex. 1002, Page 000744
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

[0216] After the first capture is complete, an administrator may install one or more virtual machines 404 to virtual machine manager 402. After the user installs and configures guest operating systems and any necessary applications 406 on the virtual machines 404, control node 12 may capture software images for storage as virtual disk image files 408. Specifically, control node 12 may utilize the captured images as a golden images for the particular operating systems, applications and specific configuration of the virtual machine. The golden image may then be used to produce image instances, i.e., virtual disk images in this case, that are bootable on the virtual machines. The image instances may likewise be stored in application matrix 350 for autonomic deployement.

[0217] After generation of image instances for virtual machine manager 402 and virtual disk image files 408, control node 12 may deploy the image instance of virtual machine manager 402 to one or more nodes in free pool 13. Once control node 12 deploys the image instance of virtual machine manager 402 to a node, the node appears to be one or more new, unallocated nodes. For instance, if the image instance of virtual machine manager 402 is configured to support five virtual machines, control node 12 represents the newly deployed image instance as though five new, unallocated nodes have been added to free pool 13. Subsequently, control node 12 may deploy an image instance of virtual disk image files 408 to one of the new unallocated nodes.

[0218] In one embodiment, during deployment, administrator 20 may specify whether data in virtual disk image files is to be treated as persistent or non-persistent. If administrator 20 specifies that the virtual disk image files are persistent, changes made to a virtual disk image file persist when node is restarted. To ensure the persistence of the data, administrator 20 may configure virtual machine manager 402 to store virtual disk image files in a storage area network (SAN). Otherwise, if administrator 20 specifies that virtual disk image files are non-persistent, the original virtual disk image file is reloaded from control node 12. Non-persistence may be valuable to minimize risks (e.g., potential corruption due to viruses) associated with prolonged use of an operating system. Further, corrupt or invalid data in the virtual disk image file may be corrected by use of non-persistent image instances for the virtual disk image files.

[0219] In this way, distributed computing system 10 may autonomically utilize application nodes, e.g, node 400, to flexibly support a wide variety of operating systems and

Netflix, Inc. - Ex. 1002, Page 000745
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

applications, and provide virtualized execution environment for those operating systems and applications. For instance, node 400 may autonomically configured to support a Macintosh Operating System designed for a PowerPC processor, and a Microsoft Windows Operating System designed for an x86 processor based on the particular goals of the autonomic system. Thus, control node 12 may not need to distinguish between hardware processor architectures when deploying operating systems and applications. However, this system may also allow administrator 20 to specify that control node 12 operate an image instance directly on a node without the intervening virtual machine and virtual machine manager. Furthermore, by executing multiple operating systems on a single node, distributed computing system 10 may be able to use and deploy the resources of each node more efficiently.

[0220] FIG. 25 is a flowchart illustrating an exemplary procedure for creating and capturing a golden image for a particular virtual machine manager 402 (FIG. 24). Initially, administrator 20 sets aside a node (e.g., node 400) as an image host (410). Administrator 20 may then install a particular type of virtual machine manager 402 on node 400 (412). After administrator 20 installs virtual machine manager 402 on node 400, administrator 20 may configure the virtual machine manager to support a desired number of virtual machine instances (414). For example, administrator 20 could configure virtual machine manager 402 to support one or more virtual machines. Administrator 20 may then perform other network or Storage Area Network (SAN) configuration on virtual machine manager 402 (416). After administrator 20 performs all of the necessary configuration, administrator 20 instructs control node 12 to capture the image of virtual machine manager 402 with the configurations (418). Control node 12 may store the captured image as a golden image in application matrix 350 for use in subsequent autonomic deployment of image instances.

[0221] FIG. 26 is a flowchart illustrating an exemplary procedure for creating and capturing a virtual disk image file, i.e., a golden image of a guest operating system and applications for execution on a virtual machine. Initially, administrator 20 sets aside a node (e.g., node 400) that is running a virtual machine manager (e.g., virtual machine manager 402) (420). Administrator 20 may then configure the virtual machine manager to create a virtual machine (e.g., virtual machine 404) on node 400 (422). After creating the virtual machine, administrator 20 installs the particular operating system on virtual machine 404 (424). Administrator 20 may then install one or more applications on the operating system (426).

Netflix, Inc. - Ex. 1002, Page 000746
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

After installation, administrator 20 may configure the operating system and applications as needed (428). In some embodiments, virtual machine manager 402 stores definition data for the particular virtual machine, operating system, and applications in a virtual disk image file. Once configuration is complete, administrator 20 takes a "snapshot" of the data in the virtual disk image file (430). Administrator 20 may then capture this "snapshot" to control node 12 as a golden image(432). Control node 12 may store the "snapshot" in application matrix 350 as a golden image for use in autonomic deployment of one or more image instances to virtual machines hosting the particular operating system and applications.

[0222] FIG. 27A is a flowchart illustrating an exemplary procedure for autonomic deploying a virtual machine manager to a node. Initially, administrator 20 may instruct control node 12 to create a new tier (440). Subsequently, administrator 20 specifies one or more captured virtual machine manager image instances for assignment to the slots of the new tier (442). Administrator 20 may also specify whether the image instances of the tier run in persistent or non-persistent mode. Administrator 20 may then activate the new tier (444). After these physical nodes boot with the virtual machine manager image instance, each of the nodes creates a set of virtual machines as specified by the image instances. Control node 12 may then discover each of these virtual machines and place them in free pool 13 as if they were independent nodes. In this manner, each of the virtual machines appear to be physical nodes available for use as application nodes 14 when needed. However, control node 12 maintains tags or other metadata that distinguish between virtual machines and actual physical nodes. Finally, based on the current state and goals of distributed computing system 10, control node 12 may allocate one or more of the virtual machines to the slots within the tier, similar to the process by which nodes are deployed from the free pool 13 to a tier as needed.

[0223] FIG. 27B is a flowchart illustrating an exemplary procedure for deploying an operating system and application image instance to a node that is executing a virtual machine manager. To start the procedure, administrator 20 may instruct control node 12 to create a new tier, including defining the number of slots for the tier, or select an existing tier (446). Administrator 20 may then specify a captured operating system and application image for the tier (448). In addition, administrator 20 may set minimum, maximum, and target values for the image instances to be deployed to the slots new tier (450). Administrator 20 may then activate the tier (452). After administrator 20 activates the tier, based on the current state and

Netflix, Inc. - Ex. 1002, Page 000747
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

goals of distributed computing system 10, control node 12 autonomically selects a virtual machine from free pool 13, associates the virtual machine with a slot in the tier, identifies the virtual disk image file (image instance) associated with the particular slot, copies the virtual disk image file to the local hard disk of the physical node on which the virtual node resides or the an associated SAN, and finally boots the virtual machine from the virtual disk image file.

[0224] FIG. 28 is a screen image illustrating an exemplary tier control interface 460 for the embodiment described in FIGS. 24 through 27 above. In this example, control interface 460 displays a set of three tiers "elas3u6_2", "elas3u6_vm", and "esx17_8". As illustrated in FIG. 28, tier "esx17_8", labeled 462, contains eight physical nodes. The image "esx_17" is operating on each node in tier 462. If the "esx_17" image is a virtual machine manager image that operates seventeen virtual machines, the eight nodes of tier 462 operate as 136 virtual machines.

[0225] FIG. 29 is a screen image illustrating an exemplary node control interface 470 for the embodiment described in FIGs. 24 through 27 above. As illustrated in FIG. 29, a single physical node "collage0104" hosts seventeen virtual machines "collage0104_vnode1" through "collage0104_vnode17". Each of these virtual machines is loaded with the "w2k3_iis_sql (1.0)" image. For instance, the "w2k3_iis_sql (1.0)" image may be an image containing instances of Microsoft Windows 2003 Internet Information Server and Microsoft SQL Server.

[0226] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.

Netflix, Inc. - Ex. 1002, Page 000748
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

CLAIMS:

1. A distributed computing system comprising:

a plurality of application nodes interconnected via a communications network;

a software image repository storing: (i) one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, and (ii) one or more image instances of one or more software applications that are executable on the virtual machines; and

a control node that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the image instances of the software applications on the virtual machines.

2. The distributed computing system of claim 1, wherein the image instances of the one or more software applications comprise virtual disk image files.

3. The distributed computing system of claim 1, wherein when the automation infrastructure deploys the image of the one or more applications, the automation infrastructure copies a first one of the image instances to a first one of the application nodes.

4. The distributed computing system of claim 1, wherein an application image in the set of application images comprises an operating system.

5. The distributed computing system of claim 1, wherein an application image in the set of application images comprises a server application.

6. The distributed computing system of claim 1, wherein a first one of the virtual machines provides an environment that emulates an x86 platform.

Netflix, Inc. - Ex. 1002, Page 000749
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

7. The distributed computing system of claim 1, wherein the automation infrastructure captures the image of the virtual machine manager from an application node.

8. The distributed computing system of claim 7, wherein the automation infrastructure captures the image of the application from an application node executing the virtual machine manager.

9. The distributed computing system of claim 1, wherein the control node maintains the image instances as persistent or non-persistent based on a configuration.

10. The distributed computing system of claim 1, wherein the control node further comprises one or more rule engines that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules.

11. The distributed computing system of claim 10, wherein the rule engines are forward-chaining rule engines.

12. The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to autonomically control the deployment of the applications to the application nodes in accordance with a current state of the application matrix.

13. The distributed computing system of claim 1, wherein the control node further comprises an application matrix that includes parameters for controlling the deployment, undeployment, and execution of the software applications to the virtual machines within the distributed computing system.

14. The distributed computing system of claim 13, further comprising an application governor that accesses the application matrix to communicate instructions from the automation infrastructure to the software applications.

Netflix, Inc. - Ex. 1002, Page 000750
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

15. The distributed computing system of claim 10, wherein the automation infrastructure automatically updates the rules engines to monitor the execution of the applications when deployed to the application nodes in accordance with a current state of the application matrix.

16. The distributed computing system of claim 1, wherein the automation subsystem includes a rule compiler that compiles the rules into one or more discrimination networks.

17. The distributed computing system of claim 1, wherein the control node further comprises:

a monitoring subsystem that collects the status data from the application nodes and communicates the status data to the automation subsystem, wherein the status data represents an actual state for the application nodes; and

a business logic tier that provides expected state data to the automation subsystem, wherein the expected state data represents an expected state for the application nodes,

wherein the rule engines analyze the status data from the monitoring subsystem and apply the set of rules to produce action requests to the business logic tier to control the application nodes to reduce any difference between the actual state and the expected state.

18. The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein each of the information collectors collects the status data from the application nodes of different tiers.

19. The distributed computing system of claim 1, further comprising:

a database storing data that defines a model for a hierarchical organization of the distributed computing system, wherein the model specifies a fabric having a one or more domains, and wherein each domain has at least one tier that includes at least one of the application nodes,

wherein the rule engines automatically produce action requests to control the deployment of the applications in accordance with a hierarchical organization of the distributed computing system defined by the model.

Netflix, Inc. - Ex. 1002, Page 000751
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

20. A method comprising:

storing one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines;

storing one or more image instances of one or more software applications that are executable on the virtual machines;

executing a rules engine to perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

executing the rules engine to perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

21. The method of claim 20, wherein storing the image instances of the one or more software applications comprise storing virtual disk image files.

22. The method of claim 20, wherein performing operations to autonomically deploy the image instances comprises copying a first one of the image instances to a first one of the application nodes.

23. The method of claim 20, wherein the software applications comprise an operating system.

24. The method of claim 20,

further comprising generating an application matrix that specifies data for controlling the deployment of the software applications within a distributed computing system; and

wherein performing operations to provide autonomic control comprises provide autonomic control over the deployment, undeployment, and execution of the software applications on the virtual machines within the distributed computing system in accordance with parameters of the application matrix.

Netflix, Inc. - Ex. 1002, Page 000752
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

25. The method of claim 20, wherein performing operations comprises:

receiving status data for the distributed computing system having the plurality of application nodes interconnected via a communications network, wherein the status data represents an actual state of the application nodes;

processing the status data with rules in a set of rule engines to determine operations for reducing any difference between an expected state and the actual state of the distributed computing system; and

performing the operations to provide autonomic control over the deployment of the software applications on the virtual machines within the distributed computing system in accordance with the rules.

26. The method of claim 20, further comprising accessing parameters of the application matrix and generating one or more rules for use in autonomically controlling the deployment of the software applications on the virtual machines.

27. The method of claim 26, further comprising processing the rules with a plurality of forward-chaining rule engines that compile the rules into one or more discrimination networks.

28. A computer-readable medium comprising instructions that cause a programmable processor to:

store one or more image instances of a virtual machine manager that is executable on application nodes in a distributed computing system, wherein the image instances of the virtual machine manager provide one or more virtual machines;

store one or more image instances of one or more software applications that are executable on the virtual machines;

perform operations to autonomically deploy the image instances of the virtual machine manager on application nodes within a distributed computing system; and

perform operations to autonomically deploy the image instances of the software applications on the virtual machines.

Netflix, Inc. - Ex. 1002, Page 000753
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES

ABSTRACT

A distributed computing system conforms to a multi-level, hierarchical organizational model. One or more control nodes provide for the efficient and automated allocation and management of computing functions and resources within the distributed computing system in accordance with the organization model. For instance, a distributed computing system comprises a plurality of application nodes interconnected via a communications network. In addition, the distributed computing system includes a software image repository storing: (i) one or more image instances of a virtual machine manager that is executable on the application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide one or more virtual machines, and (ii) one or more image instances of one or more software applications that are executable on the virtual machines. The distributed computing system also includes a control node that comprises an automation infrastructure to provide autonomic deployment of the image instances of the virtual machine manager on the application nodes and to provide autonomic deployment of the image instances of the software applications on the virtual machines.

Netflix, Inc. - Ex. 1002, Page 000754
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

Application Data Sheet

Application Information

Application Number::	Unknown
Filing Date::	Herewith
Application Type::	Regular
Subject Matter::	Utility
Title::	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
Attorney Docket Number::	1072-021US01
Total Drawing Sheets::	29
Small Entity::	Yes

Applicant Information

Applicant Authority Type::	Inventor
Primary Citizenship Country::	India
Status::	Full Capacity
Given Name::	Jagane
Middle Name::	
Family Name::	Sundar
City of Residence::	Saratoga
State of Residence::	California
Street of mailing address::	19050
City of mailing address::	Saratoga
Country of mailing address::	United States of America
Postal or Zip Code of mailing address::	95070

Applicant Authority Type::	Inventor
Primary Citizenship Country::	
Status::	Full Capacity
Given Name::	Sanjay
Middle Name::	
Family Name::	Radia

Netflix, Inc. - Ex. 1002, Page 000755
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

City of Residence:: Fremont
State of Residence:: California
Street of mailing address:: 883 Boar Circle
City of mailing address:: Fremont
Country of mailing address:: United States of America
Postal or Zip Code of mailing address:: 94539

Applicant Authority Type:: Inventor
Primary Citizenship Country:: United States
Status:: Full Capacity
Given Name:: David
Middle Name:: A.
Family Name:: Henseler
City of Residence:: Maplewood
State of Residence:: Minnesota
Street of mailing address:: 928 Demont Avenue S.
City of mailing address:: Maplewood
Country of mailing address:: United States
Postal or Zip Code of mailing address:: 55109

Correspondence Information
Correspondence Customer Number:: 28863

Representative Information
Representative Customer Number:: 28863

Domestic Priority Information

Application::	**Continuity Type::**	**Parent Application::**	**Parent Filing Date::**
This Application	**An Application claiming the benefit under 35 U.S.C.**	60/787,280	March 30, 2007

Netflix, Inc. - Ex. 1002, Page 000756
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

	119(e)		

Foreign Priority Information

Country:	**Application number:**	**Filing Date:**	**Priority Claimed:**

Assignment Information

Assignee name:: Cassatt Corporation
Street of mailing address:: 1740 Technology Drive
City of mailing address:: San Jose
Country of mailing address:: United States of America
Postal or Zip Code of mailing address:: 95110

Netflix, Inc. - Ex. 1002, Page 000757
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Jagane Sundar; Sanjay Radia; David A. Henseler		
Filed:	Herewith	Customer No.:	28863
Docket No.:	1072-021US01		
Title:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES		

REQUEST FOR PATENT APPLICATION UNDER 37 C.F.R. § 1.53(b)

Commissioner for Patents
Alexandria, VA 22313-1450

Sir:

This application claims priority from U.S. Provisional Application Serial No. 60/787,280, filed March 30, 2006.

We are transmitting herewith the attached correspondence relating to this application:

☒ Transmittal sheet
☒ Utility Patent Application: Spec.53 pgs; Claims 5 pgs; Abstract 1 pg.
The fee has been calculated as shown below in the "Claims as Filed" table.
☒ 29 sheets of formal drawings.
☒ Application Data Sheet
☒ Applicant Claims Small Entity Status

CLAIMS AS FILED

Number of Claims Filed		In Excess of:		Extra		Rate		Fee
Basic Filing Fee								$75.00
Total Claims								
28		20		8	x	$25.00	=	$200.00
Independent Claims								
3		3		0	x	$100.00	=	$
MULTIPLE DEPENDENT CLAIM FEE								
EXAMINATION FEE								$100.00
SEARCH FEE								$250.00
APPLICATION SIZE FEE								$
TOTAL FILING FEE								$625.00

Please charge Deposit Account No. 50-1778 the amount of $625.00 for the filing fee.

Please apply any charges not covered, or any credits, to Deposit Account No. 50-1778.

If this application is found to be incomplete, or if a telephone conference would otherwise be helpful, please call the undersigned at (651) 735-1100.

Please send all correspondence to Practitioners at Customer Number 28863.



28863
PATENT TRADEMARK OFFICE

Date: March 30, 2007

By: Kent J. Sieffert

Name: Kent J. Sieffert
Reg. No.: 41,312

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Netflix, Inc. - Ex. 1002, Page 000759
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Jagane Sundar; Sanjay Radia; David A. Henseler		
Filed:	Herewith	Customer No.:	28863
Docket No.:	1072-021US01		
Title:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES		

REQUEST FOR PATENT APPLICATION UNDER 37 C.F.R. § 1.53(b)

Commissioner for Patents
Alexandria, VA 22313-1450

Sir:

This application claims priority from U.S. Provisional Application Serial No. 60/787,280, filed March 30, 2006.

We are transmitting herewith the attached correspondence relating to this application:

☒ Transmittal sheet
☒ Utility Patent Application: Spec.53 pgs; Claims 5 pgs; Abstract 1 pg.
The fee has been calculated as shown below in the "Claims as Filed" table.
☒ 29 sheets of formal drawings.
☒ Application Data Sheet
☒ Applicant Claims Small Entity Status

CLAIMS AS FILED

Number of Claims Filed		In Excess of:		Extra		Rate		Fee
Basic Filing Fee								$75.00
Total Claims								
28		20		8	x	$25.00	=	$200.00
Independent Claims								
3		3		0	x	$100.00	=	$
MULTIPLE DEPENDENT CLAIM FEE								
EXAMINATION FEE								$100.00
SEARCH FEE								$250.00
APPLICATION SIZE FEE								$
TOTAL FILING FEE								$625.00

Please charge Deposit Account No. 50-1778 the amount of $625.00 for the filing fee.

Please apply any charges not covered, or any credits, to Deposit Account No. 50-1778.

If this application is found to be incomplete, or if a telephone conference would otherwise be helpful, please call the undersigned at (651) 735-1100.

Please send all correspondence to Practitioners at Customer Number 28863.



28863
PATENT TRADEMARK OFFICE

Date: March 30, 2007

By: Kent J. Sieffert

Name: Kent J. Sieffert
Reg. No.: 41,312

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Netflix, Inc. - Ex. 1002, Page 000761
IPR2020-01582 (Netflix, Inc. v. Avago Technologies International Sales PTE. Limited)

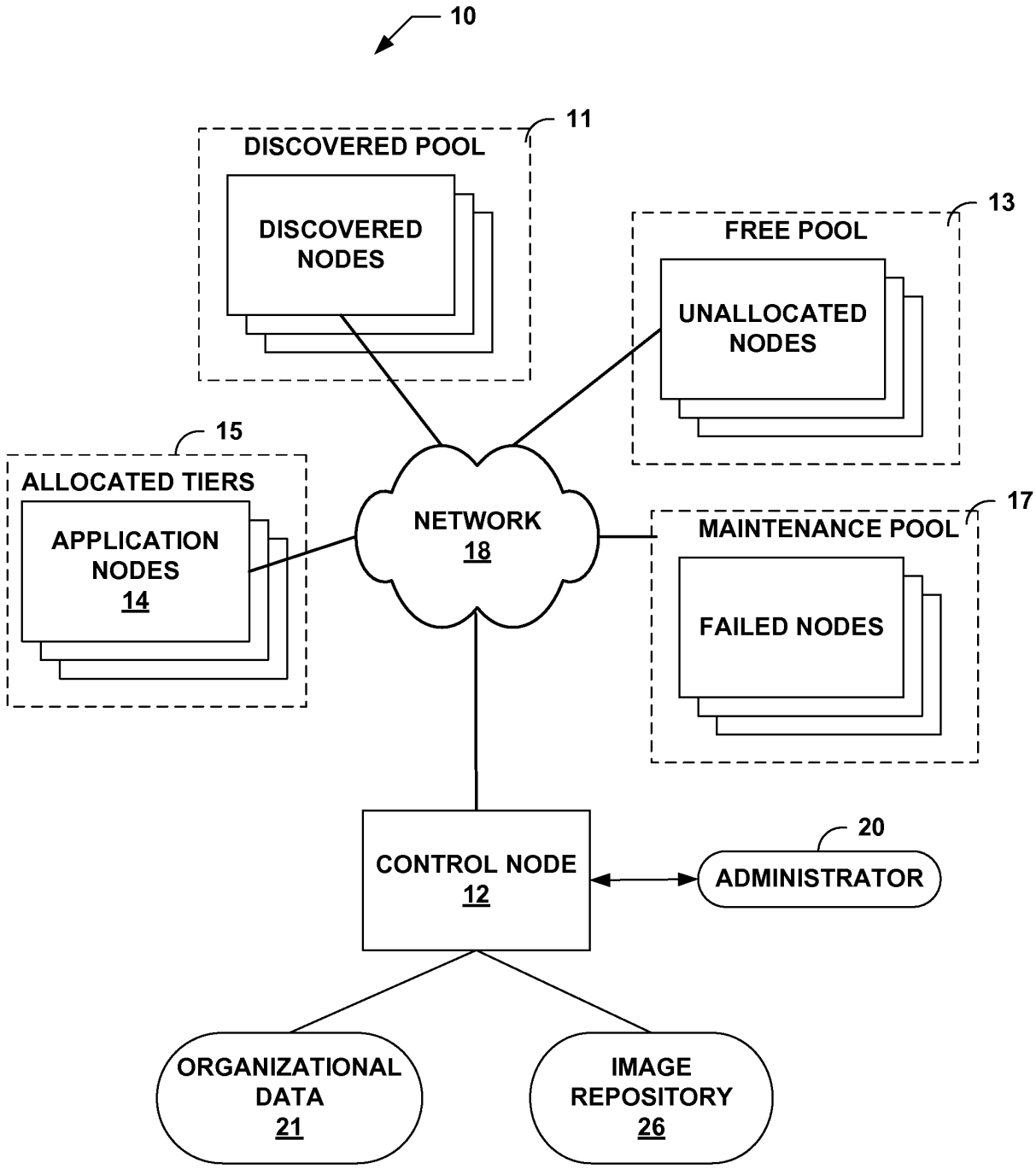

10
11
DISCOVERED POOL
DISCOVERED NODES
13
FREE POOL
UNALLOCATED NODES
15
ALLOCATED TIERS
APPLICATION NODES 14
NETWORK 18
17
MAINTENANCE POOL
FAILED NODES
CONTROL NODE 12
20
ADMINISTRATOR
ORGANIZATIONAL DATA 21
IMAGE REPOSITORY 26

FIG. 1

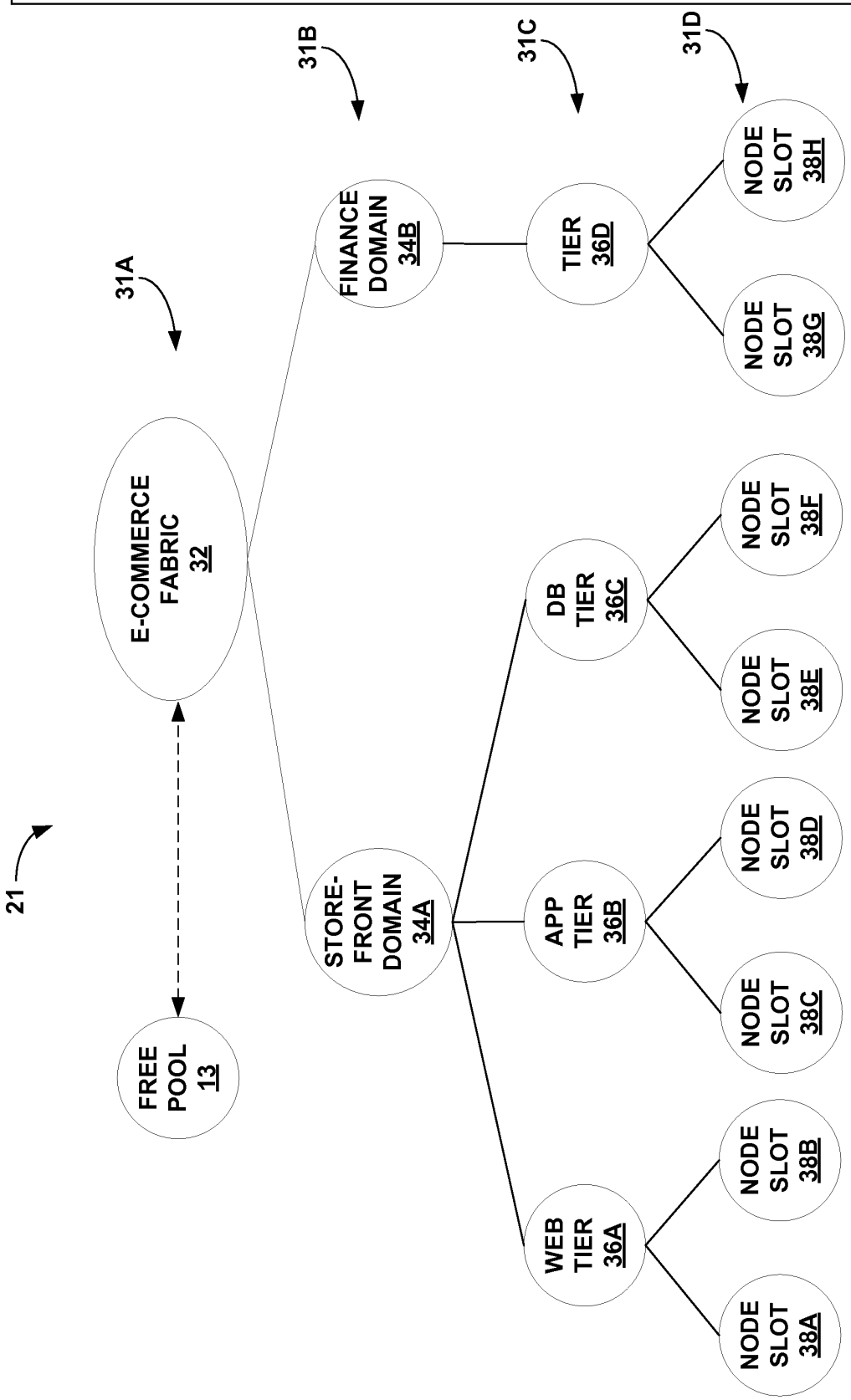

21
E-COMMERCE FABRIC 32
FREE POOL 13
31A
31B
31C
31D
FINANCE DOMAIN 34B
STORE-FRONT DOMAIN 34A
TIER 36D
DB TIER 36C
APP TIER 36B
WEB TIER 36A
NODE SLOT 38H
NODE SLOT 38G
NODE SLOT 38F
NODE SLOT 38E
NODE SLOT 38D
NODE SLOT 38C
NODE SLOT 38B
NODE SLOT 38A

FIG. 2

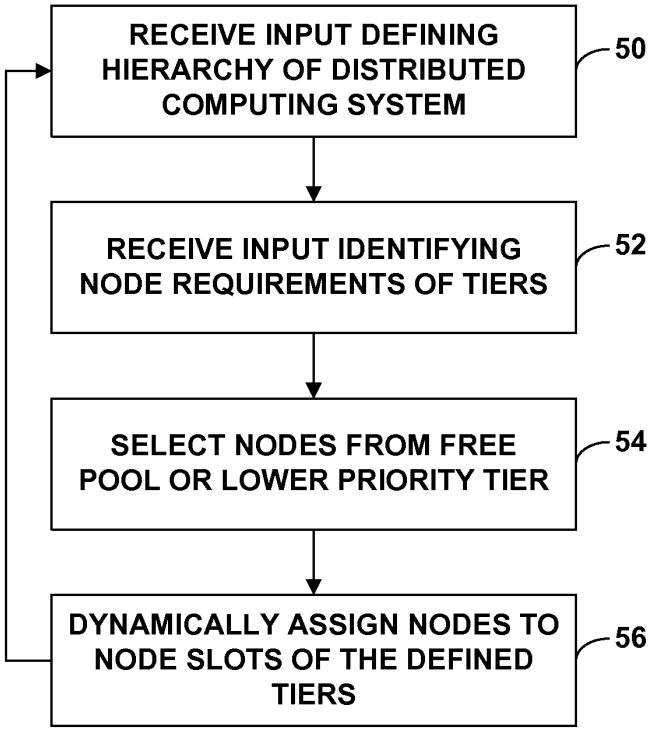

RECEIVE INPUT DEFINING HIERARCHY OF DISTRIBUTED COMPUTING SYSTEM
50
RECEIVE INPUT IDENTIFYING NODE REQUIREMENTS OF TIERS
52
SELECT NODES FROM FREE POOL OR LOWER PRIORITY TIER
54
DYNAMICALLY ASSIGN NODES TO NODE SLOTS OF THE DEFINED TIERS
56

FIG. 3

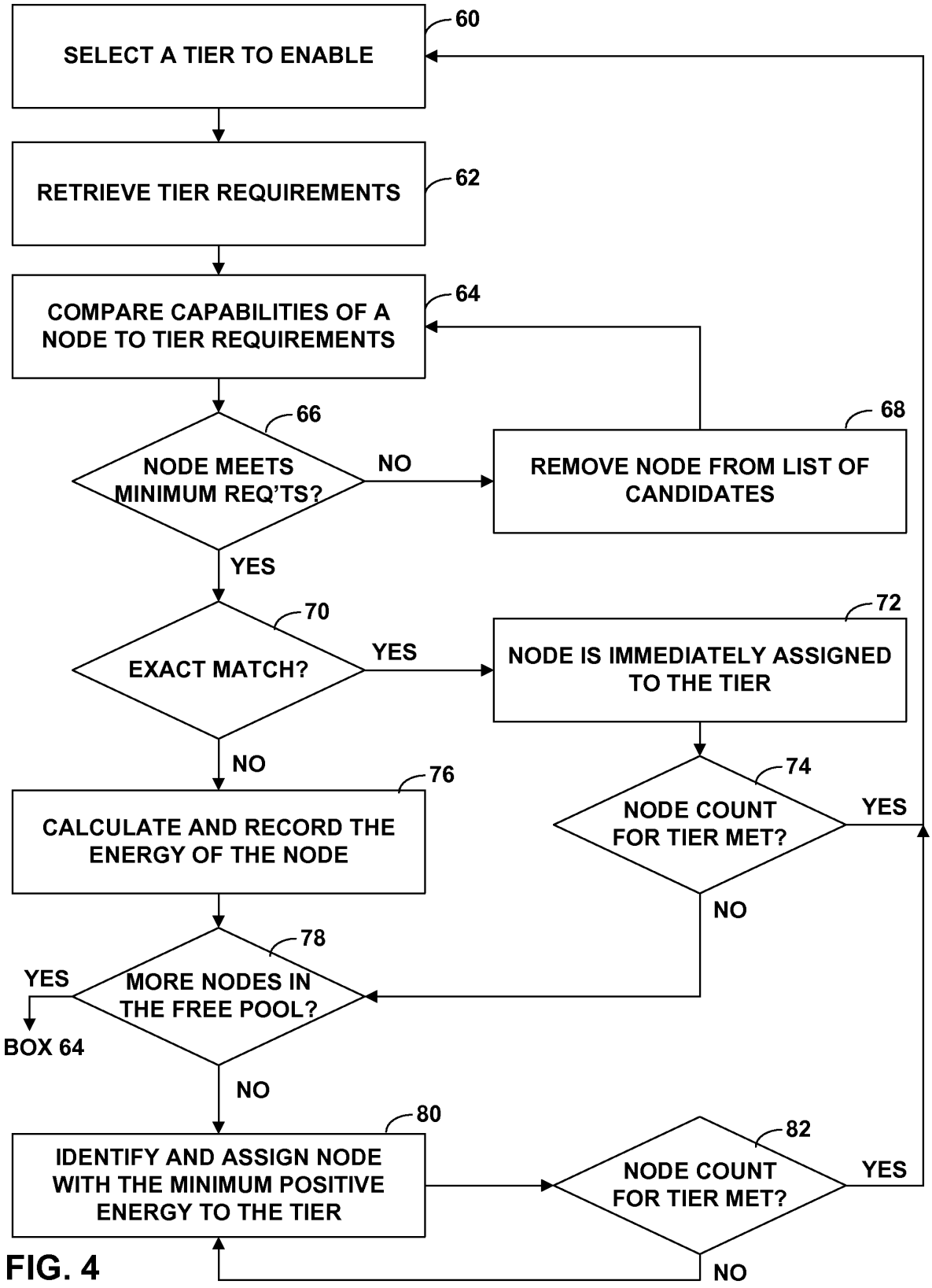

SELECT A TIER TO ENABLE
60
RETRIEVE TIER REQUIREMENTS
62
COMPARE CAPABILITIES OF A NODE TO TIER REQUIREMENTS
64
NODE MEETS MINIMUM REQ'TS?
66
NO
REMOVE NODE FROM LIST OF CANDIDATES
68
YES
EXACT MATCH?
70
YES
NODE IS IMMEDIATELY ASSIGNED TO THE TIER
72
NO
CALCULATE AND RECORD THE ENERGY OF THE NODE
76
NODE COUNT FOR TIER MET?
74
YES
NO
MORE NODES IN THE FREE POOL?
78
YES
BOX 64
NO
IDENTIFY AND ASSIGN NODE WITH THE MINIMUM POSITIVE ENERGY TO THE TIER
80
NODE COUNT FOR TIER MET?
82
YES
NO

FIG. 4

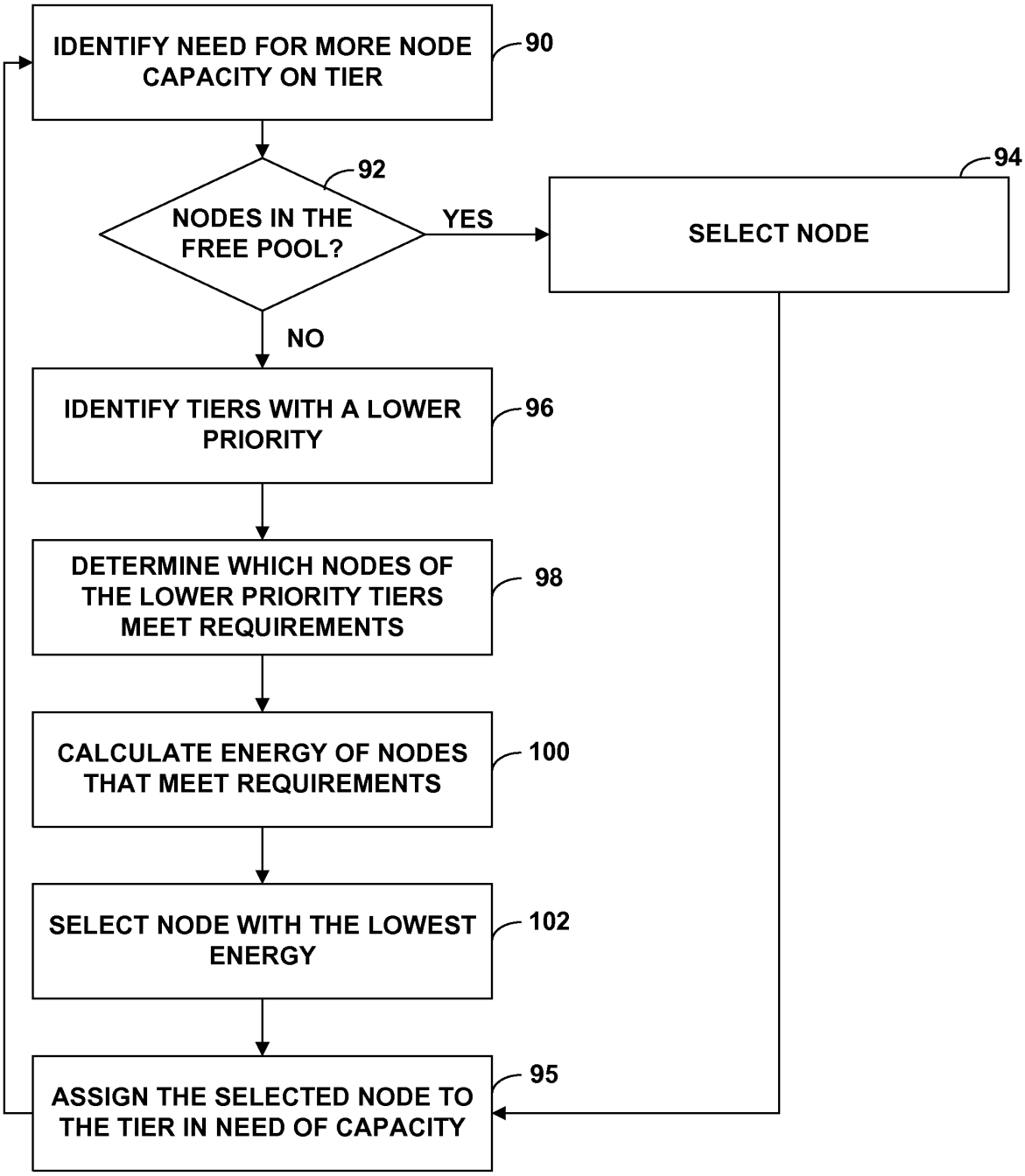

IDENTIFY NEED FOR MORE NODE CAPACITY ON TIER
90
92
NODES IN THE FREE POOL?
YES
94
SELECT NODE
NO
IDENTIFY TIERS WITH A LOWER PRIORITY
96
DETERMINE WHICH NODES OF THE LOWER PRIORITY TIERS MEET REQUIREMENTS
98
CALCULATE ENERGY OF NODES THAT MEET REQUIREMENTS
100
SELECT NODE WITH THE LOWEST ENERGY
102
ASSIGN THE SELECTED NODE TO THE TIER IN NEED OF CAPACITY
95

FIG. 5

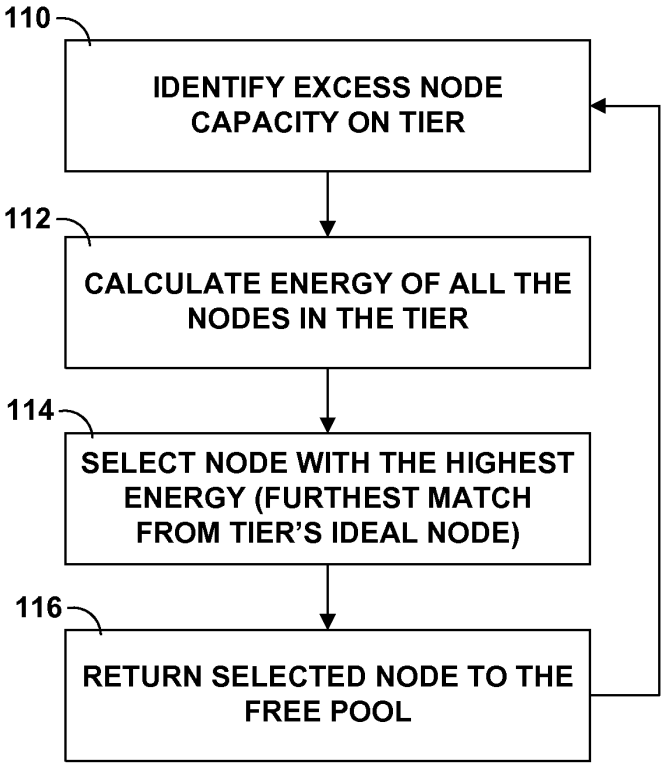

110
IDENTIFY EXCESS NODE CAPACITY ON TIER
112
CALCULATE ENERGY OF ALL THE NODES IN THE TIER
114
SELECT NODE WITH THE HIGHEST ENERGY (FURTHEST MATCH FROM TIER'S IDEAL NODE)
116
RETURN SELECTED NODE TO THE FREE POOL

FIG. 6

120
122
Cassat Collage Controller 3.0
Cassat
Collage 3.0
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Audit Log (0)
Control Node Log (0)
Reports
Active History
Hardware Inventory
Image Utilization
Images
Users
Domain: Collage Domain
Tier List
Properties
Tier Order
System
Domain: Collage Domain
Domain Actions
Status: Unknown
Description: Cassat Collage Root Domain
Tiers
Status - All -
Filter
NEW TIER
Status
Tier
Nodes
Description
Image
Last Modified
Test Tier 1
4
my test tier 1
appliance(1.0.0)
Test Tier 2
2
my test tier 2
appliance(1.0.0)
Test Tier 3
3
my test tier 3
Tier Actions
Done

FIG. 7

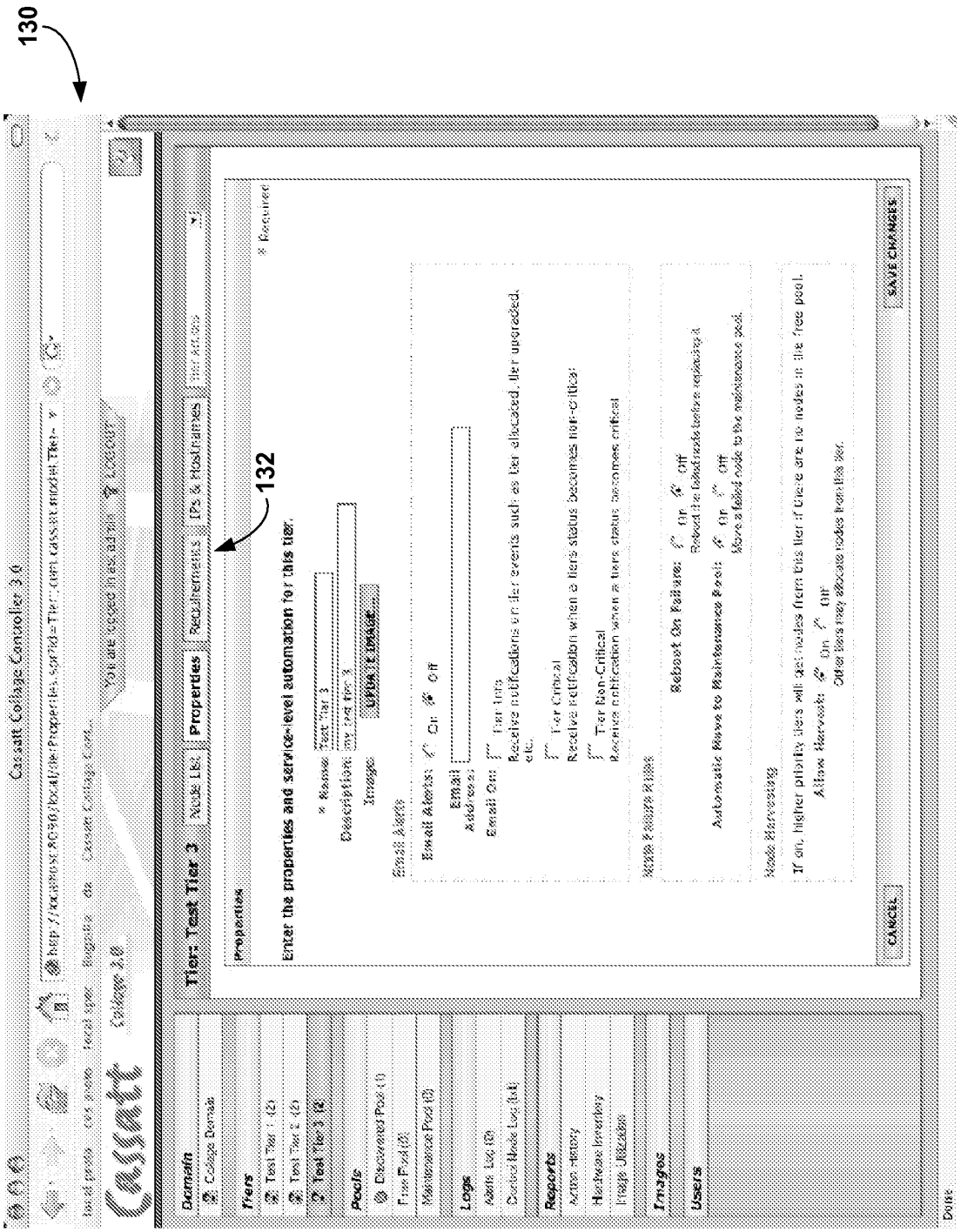

130
132
Cassatt Collage Console 3.0
Cassatt
Collage 3.0
Domain
College Domain
Tiers
Test Tier 1 (0)
Test Tier 2 (0)
Test Tier 3 (1)
Pools
Discovered Pool (0)
Free Pool (0)
Maintenance Pool (0)
Logs
Admin Log (0)
Control Node Log (0)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Tier: Test Tier 3
Node List
Properties
Requirements
IPs & Hostnames
Test Actions
Properties
* Required
Enter the properties and service-level automation for this tier.
* Name: Test Tier 3
Description: my test tier 3
Image: Create Image...
Email Alerts
Email Alerts: On Off
Email Addresses:
Email On: Test Info
Receive notifications on test events such as tier allocated, tier upgraded, etc.
Tier Critical
Receive notification when a tiers status becomes non-critical
Tier Non-Critical
Receive notification when a tiers status becomes critical
Node Failure Action
Reboot On Failure: On Off
Reboot the failed node before repairing it
Automatic Move to Maintenance Pool: On Off
Move a failed node to the maintenance pool.
Node Harvesting
If on, higher priority tiers will take nodes from this tier if there are no nodes in the free pool.
Allow Harvesting: On Off
Other tiers may allocate nodes from this tier.
CANCEL
SAVE CHANGES

FIG. 8

140
Canatt Cottage Controller 3.0
Node: Node - 1
Enter the properties for this node.
Domain
Trace
Pods
Logs
Reports
Images
Users
Interface Information
Component
IP Address
MAC Address
Custom Requirements

FIG. 9

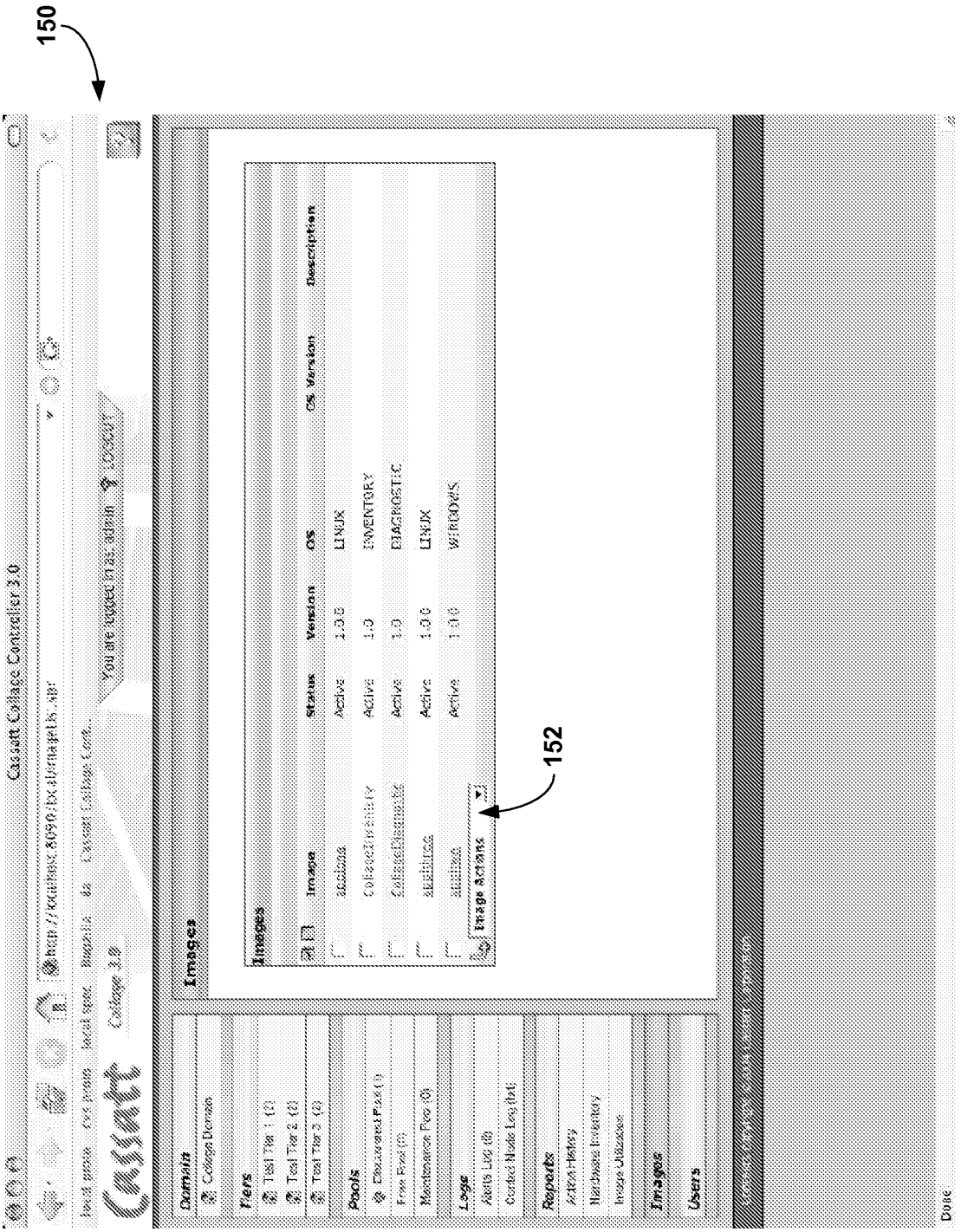

150
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
Domain
Collage Domain
Tiers
Test Tier 1 (1)
Test Tier 2 (1)
Test Tier 3 (1)
Pools
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (0)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Images
Images
Image
Status
Version
OS
OS Version
Description
Active
LINUX
Active
INVENTORY
Active
DIAGNOSTIC
Active
LINUX
Active
WINDOWS
Image Actions
152

FIG. 10

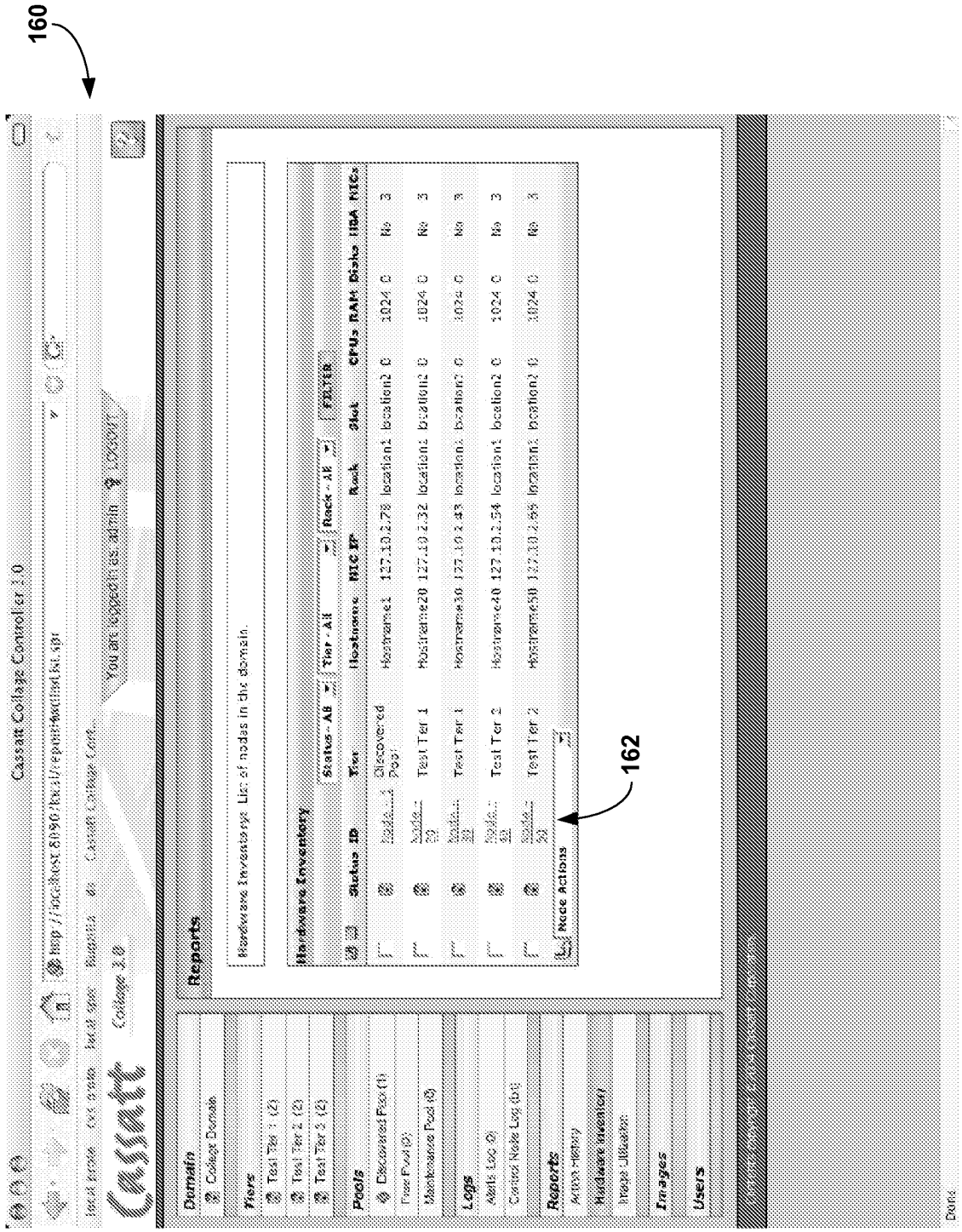

160
Cassat Cottage Controller 3.0
Cassat
Cottage 3.0
Reports
Hardware Inventory: List of nodes in the domain.
Hardware Inventory
Status - All
Tier - All
Rack - All
FILTER
Node ID
Tier
Hostname
NIC IP
Rack
Slot
CPUs
RAM
Disks
HBA
NICs
Discovered Pool
Test Tier 1
Test Tier 2
Maintenance Pool
Node Actions
162

FIG. 11

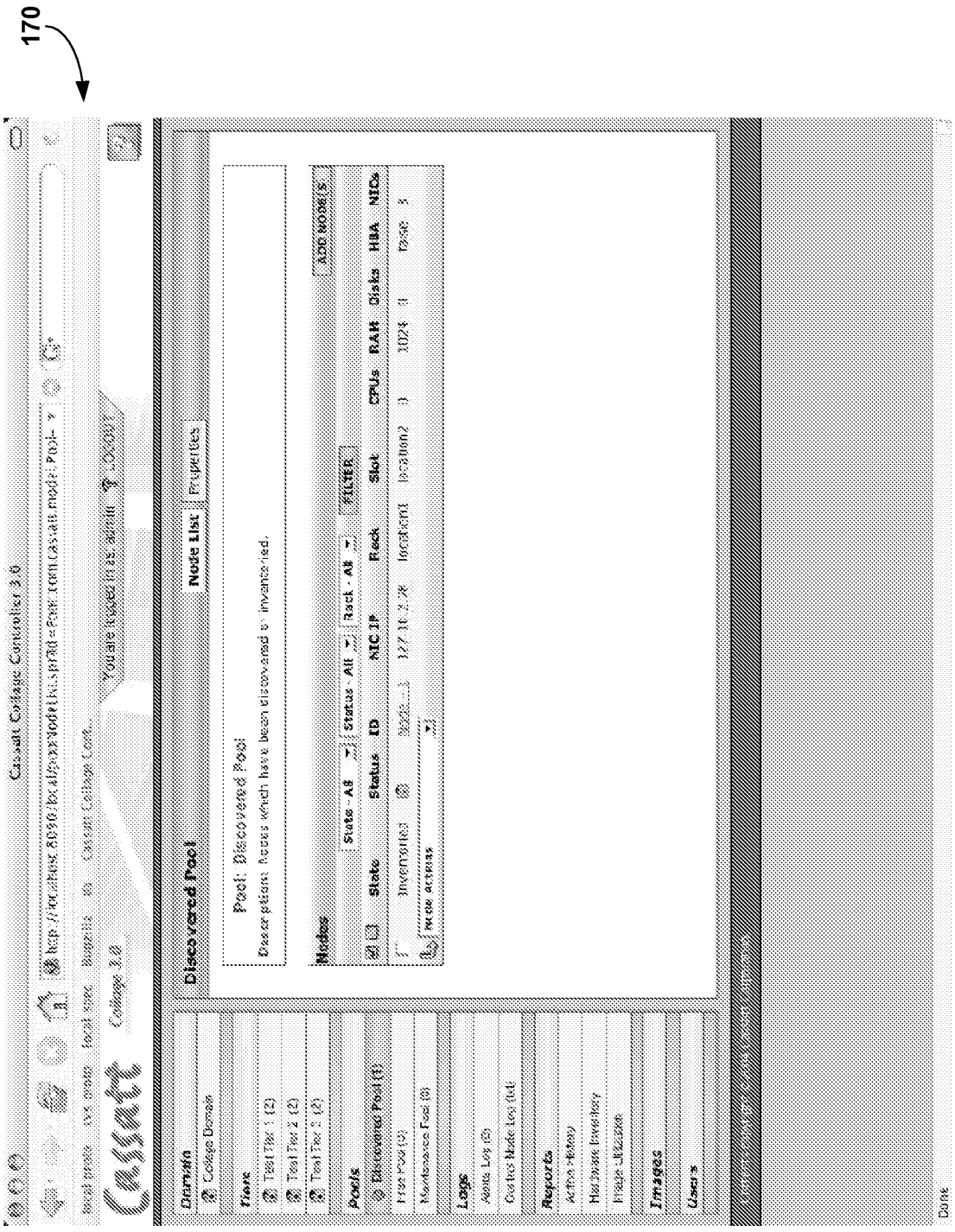

170
Cobalt
Discovered Pool
Node List
Properties
Pool: Discovered Pool
Nodes
ADD NODES
Domain
Tiers
Pools
Logs
Reports
Images
Users

FIG. 12

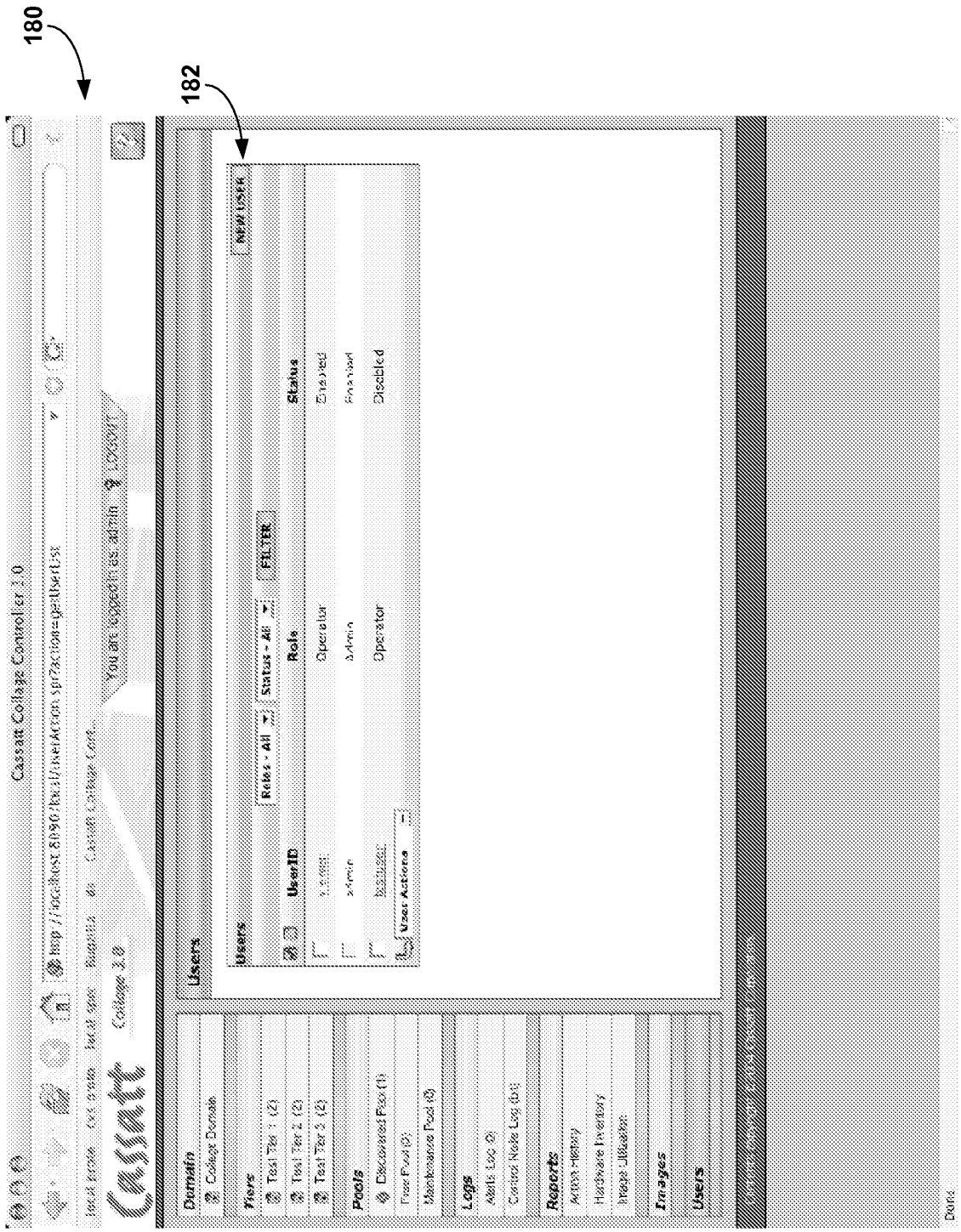

180
182
Cassat Collage Controller 3.0
Cassat
Collage 3.0
Users
Domain
College Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Discovered Pool (1)
Free Pool (0)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (0)
Reports
Active History
Hardware Inventory
Image Utilization
Images
Users
NEW USER
FILTER
Roles - All
Status - All
UserID
Role
Status
Operator
Disabled
Admin
Enabled
Operator
Disabled
User Actions

FIG. 13

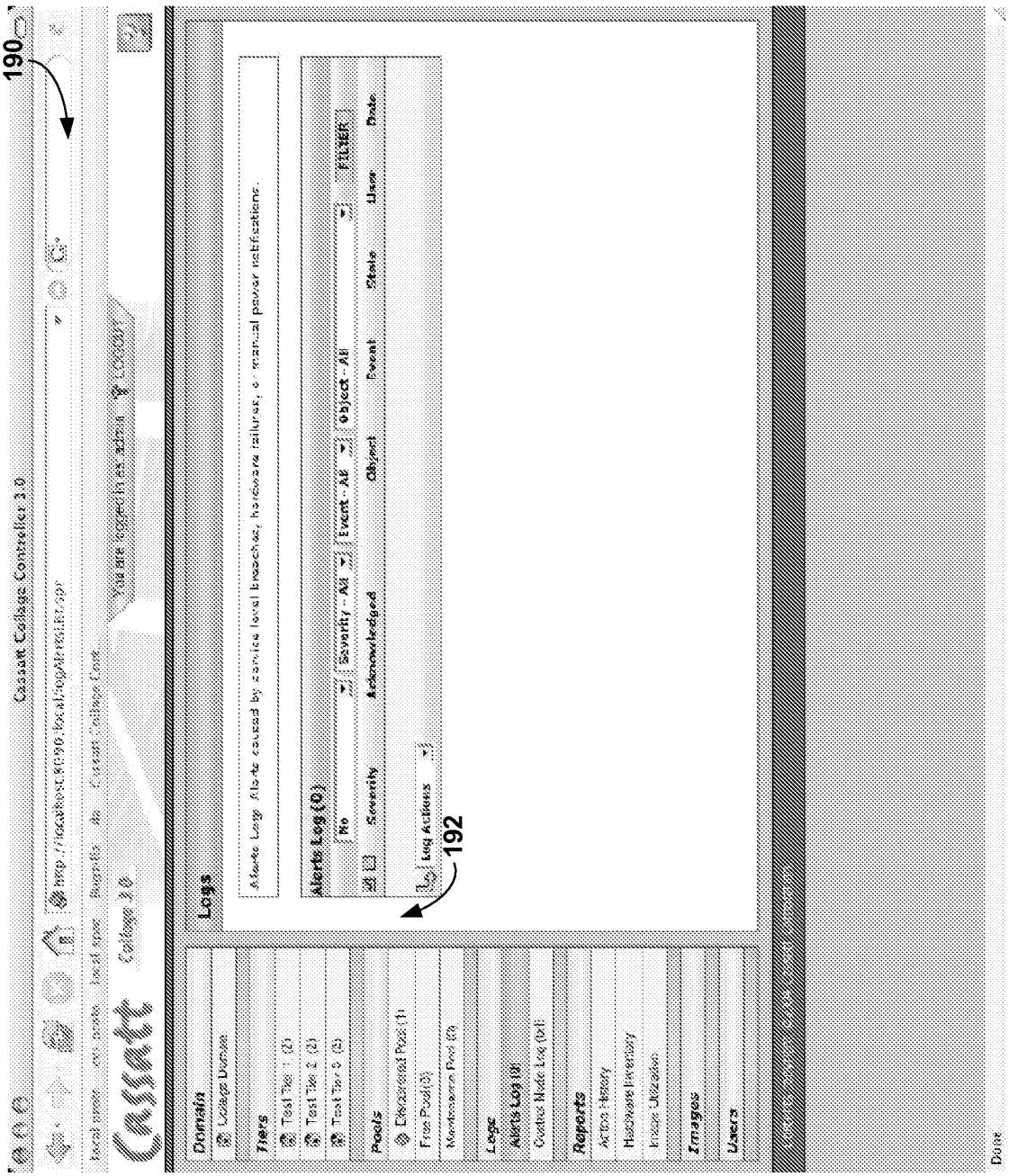

190
Cassatt Collage Controller 3.0
Cassatt
Collage 3.0
Logs
Domain
Collage Domain
Tiers
Test Tier 1 (2)
Test Tier 2 (2)
Test Tier 3 (2)
Pools
Development Pool (1)
Free Pool (3)
Maintenance Pool (0)
Logs
Alerts Log (0)
Control Node Log (0)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Alerts Log (0)
Log Actions
192
Severity
Acknowledged
Object
Event
State
User
Date
FILTER
Done

FIG. 14

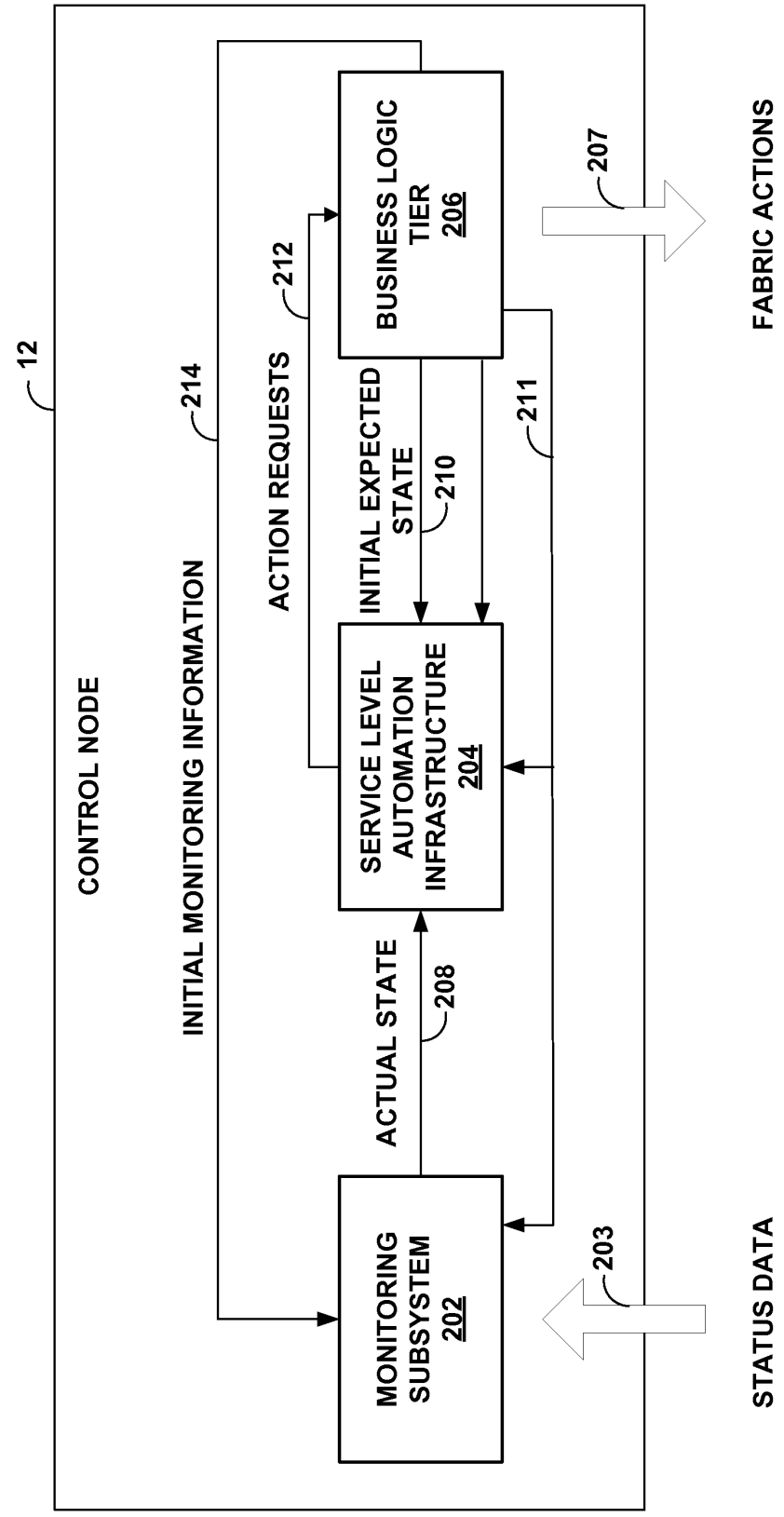

12
CONTROL NODE
214
INITIAL MONITORING INFORMATION
212
ACTION REQUESTS
BUSINESS LOGIC TIER 206
207
FABRIC ACTIONS
INITIAL EXPECTED STATE
210
211
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
ACTUAL STATE
208
MONITORING SUBSYSTEM 202
203
STATUS DATA

FIG. 15

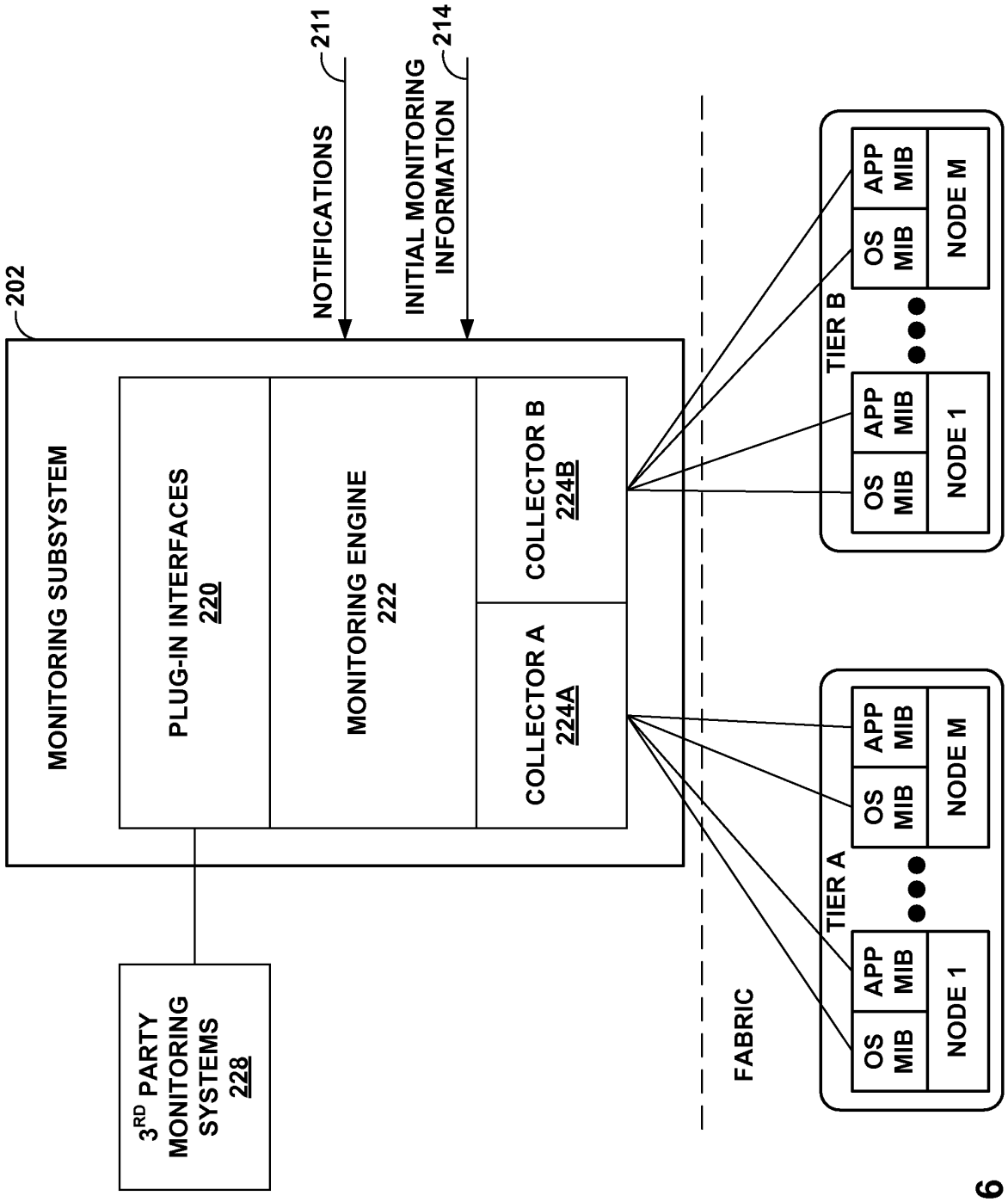

202
MONITORING SUBSYSTEM
PLUG-IN INTERFACES 220
MONITORING ENGINE 222
COLLECTOR B 224B
COLLECTOR A 224A
NOTIFICATIONS
211
INITIAL MONITORING INFORMATION
214
3RD PARTY MONITORING SYSTEMS 228
FABRIC
TIER A
TIER B
OS MIB
APP MIB
NODE 1
NODE M

FIG. 16

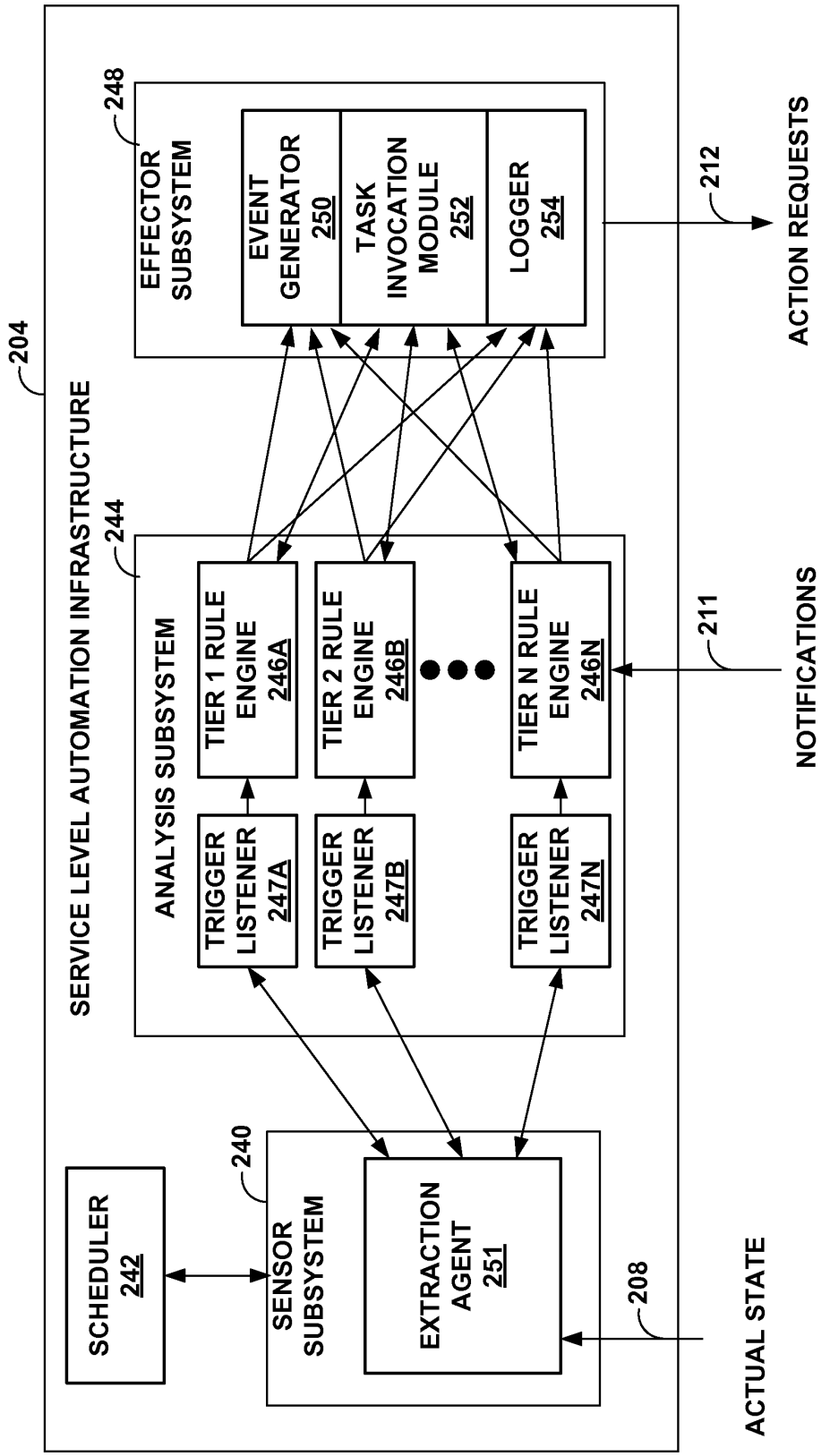

204
SERVICE LEVEL AUTOMATION INFRASTRUCTURE
248
EFFECTOR SUBSYSTEM
EVENT GENERATOR 250
TASK INVOCATION MODULE 252
LOGGER 254
212
ACTION REQUESTS
244
ANALYSIS SUBSYSTEM
TIER 1 RULE ENGINE 246A
TIER 2 RULE ENGINE 246B
TIER N RULE ENGINE 246N
211
NOTIFICATIONS
TRIGGER LISTENER 247A
TRIGGER LISTENER 247B
TRIGGER LISTENER 247N
240
SENSOR SUBSYSTEM
EXTRACTION AGENT 251
SCHEDULER 242
208
ACTUAL STATE

FIG. 17

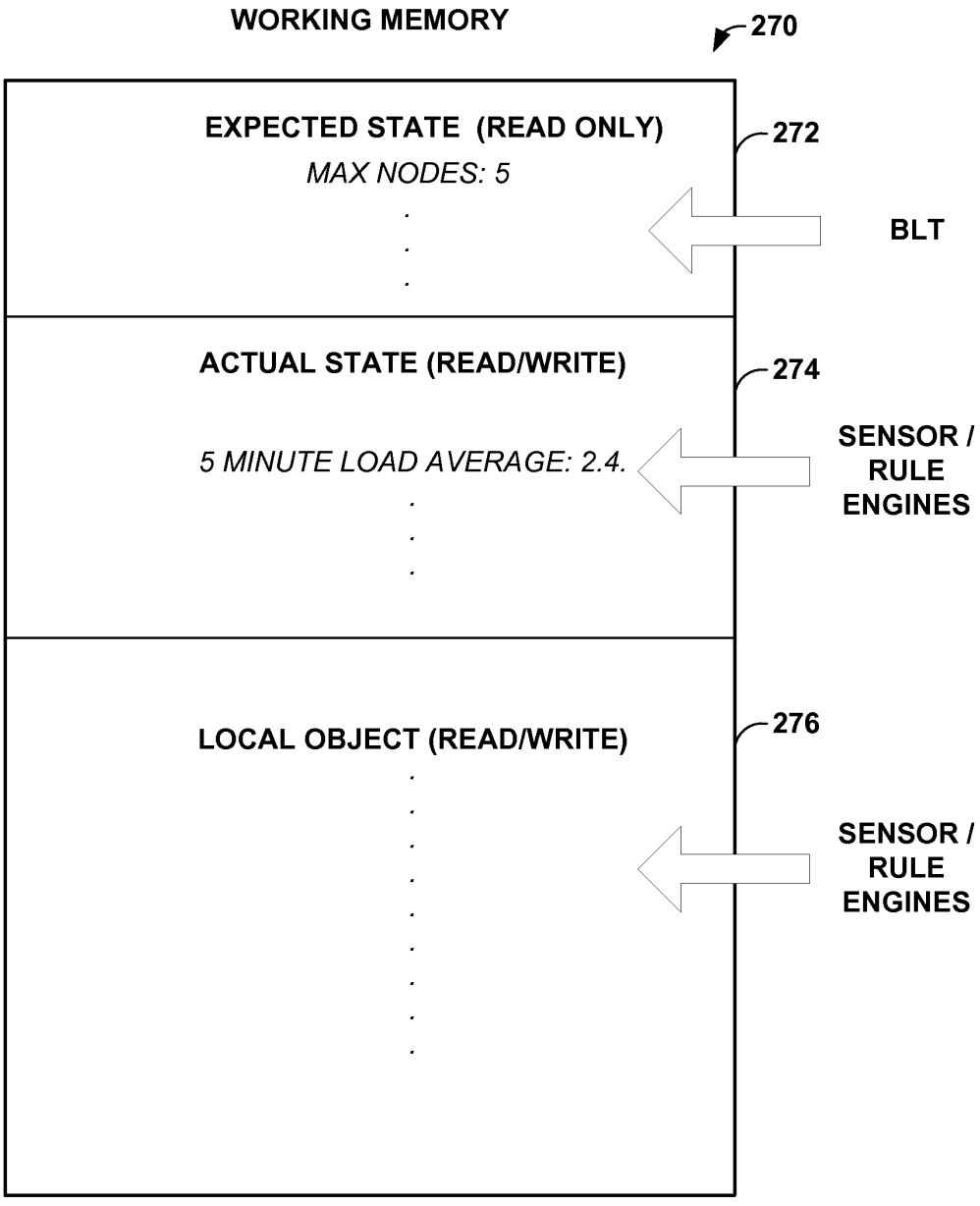

WORKING MEMORY
270
EXPECTED STATE (READ ONLY)
MAX NODES: 5
272
BLT
ACTUAL STATE (READ/WRITE)
274
5 MINUTE LOAD AVERAGE: 2.4.
SENSOR / RULE ENGINES
LOCAL OBJECT (READ/WRITE)
276
SENSOR / RULE ENGINES

FIG. 18

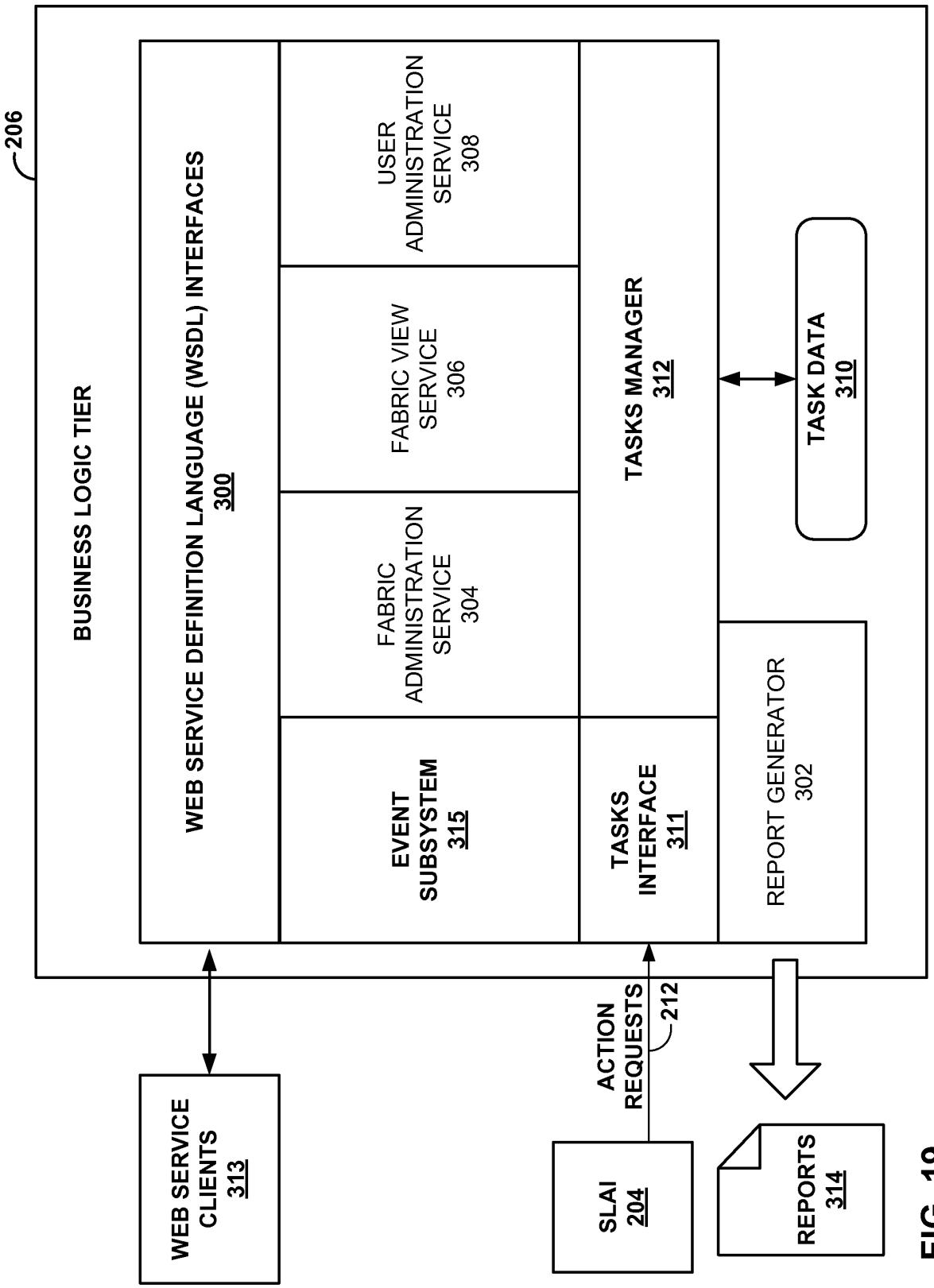

206
BUSINESS LOGIC TIER
WEB SERVICE DEFINITION LANGUAGE (WSDL) INTERFACES 300
USER ADMINISTRATION SERVICE 308
FABRIC VIEW SERVICE 306
FABRIC ADMINISTRATION SERVICE 304
EVENT SUBSYSTEM 315
TASKS MANAGER 312
TASKS INTERFACE 311
REPORT GENERATOR 302
TASK DATA 310
WEB SERVICE CLIENTS 313
ACTION REQUESTS 212
SLAI 204
REPORTS 314

FIG. 19

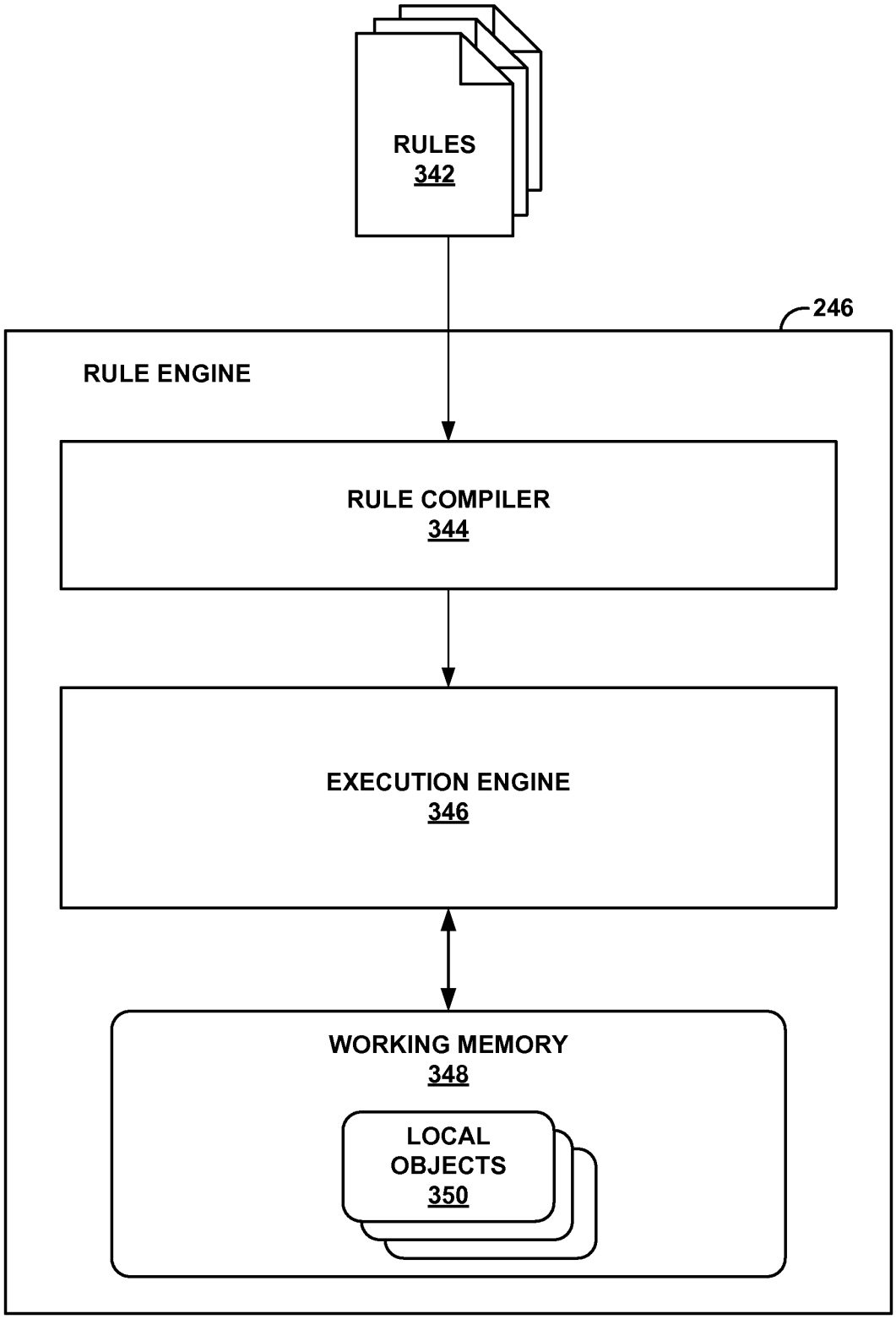

RULES
342
246
RULE ENGINE
RULE COMPILER
344
EXECUTION ENGINE
346
WORKING MEMORY
348
LOCAL
OBJECTS
350

FIG. 20

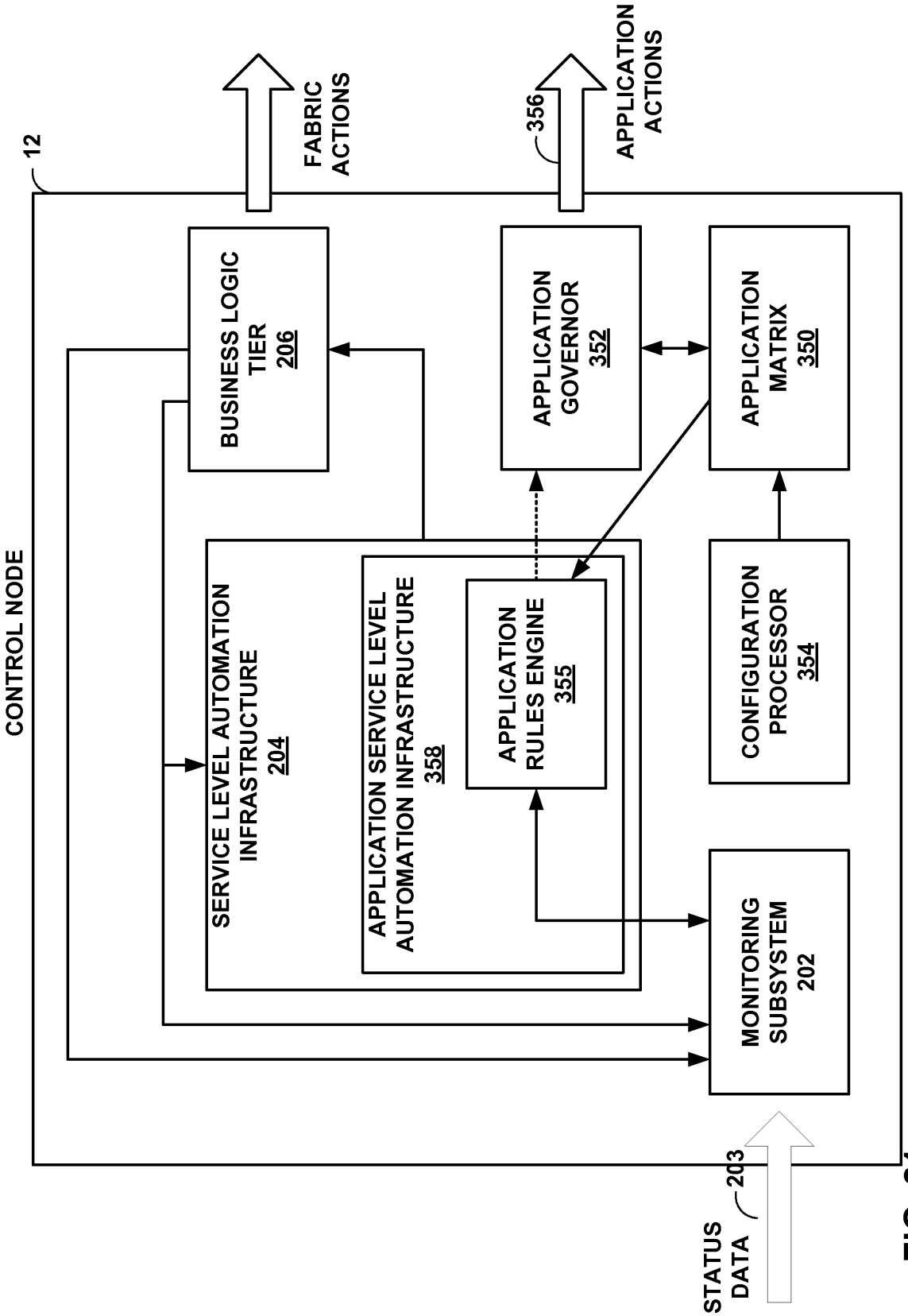

12
CONTROL NODE
FABRIC ACTIONS
356
APPLICATION ACTIONS
BUSINESS LOGIC TIER 206
APPLICATION GOVERNOR 352
APPLICATION MATRIX 350
SERVICE LEVEL AUTOMATION INFRASTRUCTURE 204
APPLICATION SERVICE LEVEL AUTOMATION INFRASTRUCTURE 358
APPLICATION RULES ENGINE 355
CONFIGURATION PROCESSOR 354
MONITORING SUBSYSTEM 202
STATUS DATA
203

FIG. 21

360 APPLICATION	362 APPLICATION NODES	364 SERVICES	366 NODE MONITORED VALUES	368 SERVICE MONITORED ATTRIBUTES	370 SERVICE LEVELS	372 DEPLOYMENT CONSTRAINTS
Name= DataDomain **adminIP=** "172.31.64.201" **adminPort=** "1100" **adminTier=** "Web Admin" **clusterName=** "PathCluster" **expectedStartupDelay**=120 **loadDelay**=120 **maxNodes**=2 **minNodes**=1	**IP=** 172.31.64.201 **Name=** managedServer_0 **IP=** 172.31.64.202 **Name=** managedServer_1	**name**="AI Design-time" **path**="/lib/ai-designtime.ear" **name**="System EJBs" **path**="/lib/ejbs.ear" **name**="Worklist Worker User Interface" **path**="/lib/worklist/worklist.ear" **name**="DBMS_ADK" **path**="/lib/DBMS_ADK.ear" **name**="App" **path**="/user_projects/domains/data.ear"	**name**= Load5Average **name**= PercentFree-Memory	**name**= PendRequests **name**= ExecuteThread-IdleCount **name**= ExecuteThread-TotalCount	**attribute**= "BusyThread-Percentage" **expression**= "(ExecuteThreadTotalCount-ExecuteThreadIdleCount)*100/ExecuteThreadTotal Count" **frequency**="15" **maxThreshold**="20.0" **minThreshold**="5.0" **period**="30" **attribute**= "PendingRequests" **expression**= "PendingRequests" **frequency**="15" **maxThreshold**="20" **minThreshold**="5" **period**="30"	**attribute**= "LoadAverage" **expression**= "Load5Average" **frequency**="15" **maxThreshold**="4" **minThreshold**="-1" **period**="15" **attribute**= "FreeMemory" **expression**= "PercentMemoryFree *100" **frequency**="15" **maxThreshold**="-1" **minThreshold**="1" **period**="60"
name= Portal-Domain						

376 374

FIG. 22

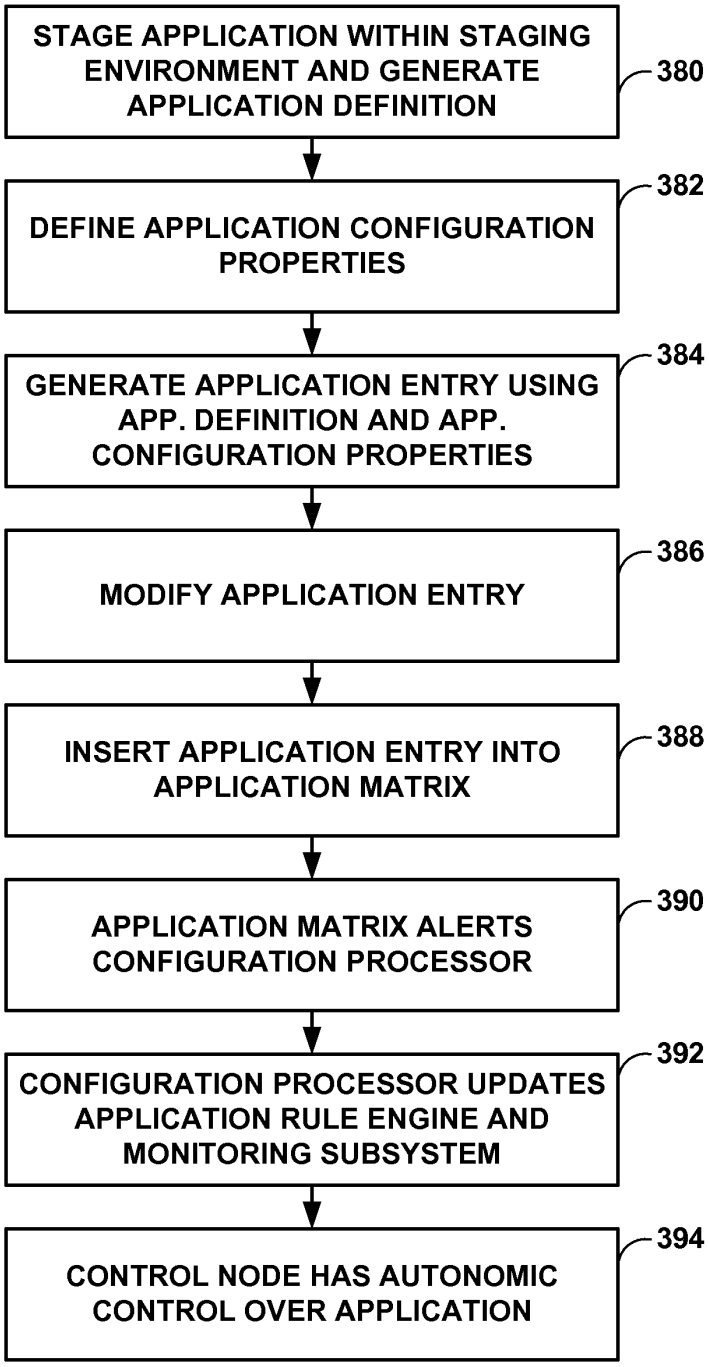

STAGE APPLICATION WITHIN STAGING ENVIRONMENT AND GENERATE APPLICATION DEFINITION
380
DEFINE APPLICATION CONFIGURATION PROPERTIES
382
GENERATE APPLICATION ENTRY USING APP. DEFINITION AND APP. CONFIGURATION PROPERTIES
384
MODIFY APPLICATION ENTRY
386
INSERT APPLICATION ENTRY INTO APPLICATION MATRIX
388
APPLICATION MATRIX ALERTS CONFIGURATION PROCESSOR
390
CONFIGURATION PROCESSOR UPDATES APPLICATION RULE ENGINE AND MONITORING SUBSYSTEM
392
CONTROL NODE HAS AUTONOMIC CONTROL OVER APPLICATION
394

FIG. 23

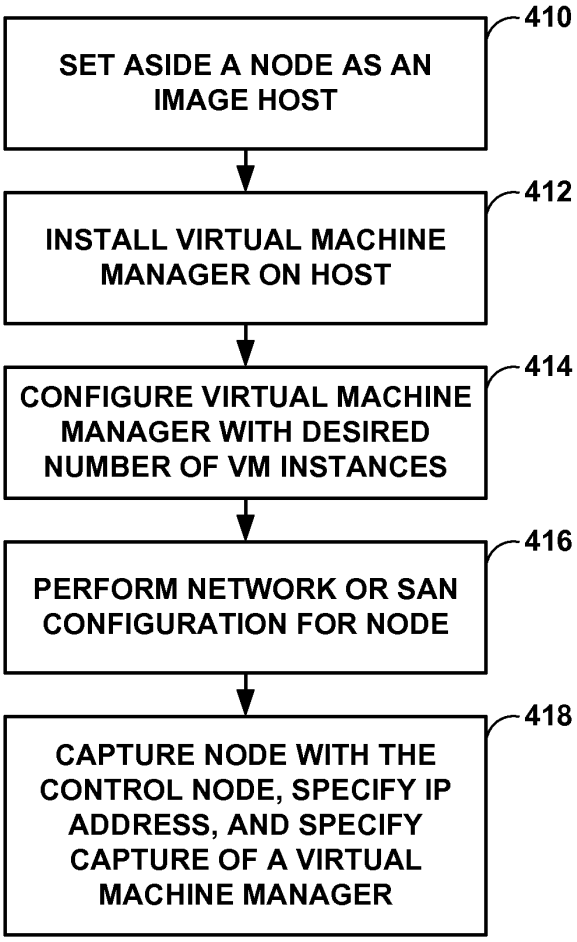

410
SET ASIDE A NODE AS AN IMAGE HOST
412
INSTALL VIRTUAL MACHINE MANAGER ON HOST
414
CONFIGURE VIRTUAL MACHINE MANAGER WITH DESIRED NUMBER OF VM INSTANCES
416
PERFORM NETWORK OR SAN CONFIGURATION FOR NODE
418
CAPTURE NODE WITH THE CONTROL NODE, SPECIFY IP ADDRESS, AND SPECIFY CAPTURE OF A VIRTUAL MACHINE MANAGER

FIG. 25

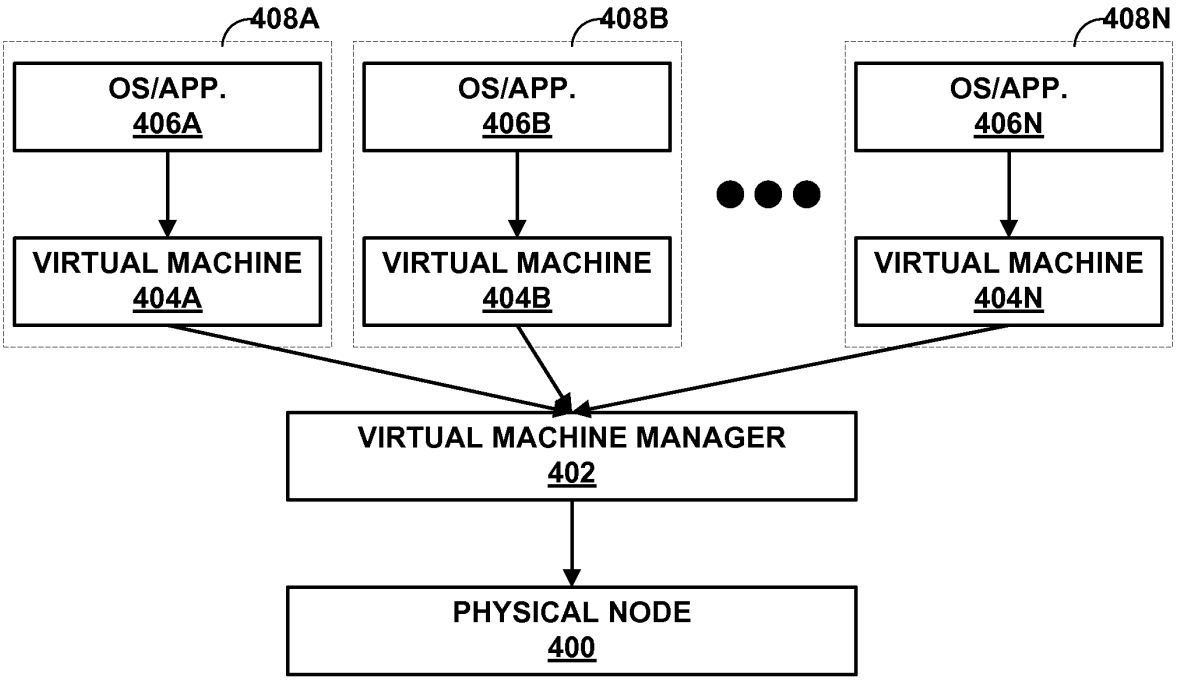

408A
408B
408N
OS/APP. 406A
OS/APP. 406B
OS/APP. 406N
VIRTUAL MACHINE 404A
VIRTUAL MACHINE 404B
VIRTUAL MACHINE 404N
VIRTUAL MACHINE MANAGER 402
PHYSICAL NODE 400

FIG. 24

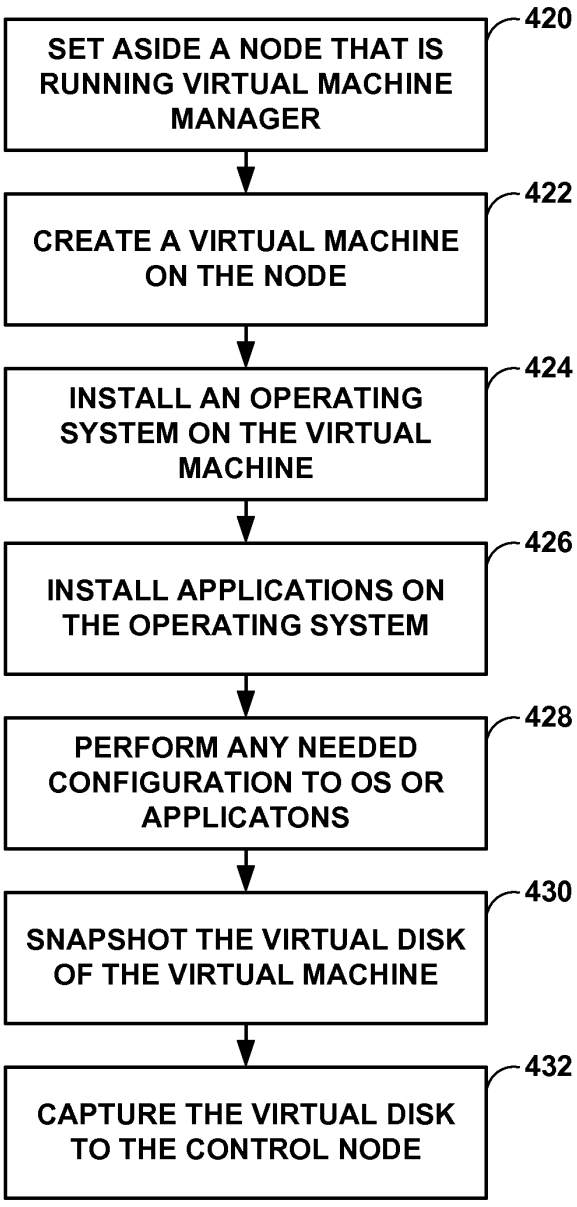

SET ASIDE A NODE THAT IS RUNNING VIRTUAL MACHINE MANAGER
420
CREATE A VIRTUAL MACHINE ON THE NODE
422
INSTALL AN OPERATING SYSTEM ON THE VIRTUAL MACHINE
424
INSTALL APPLICATIONS ON THE OPERATING SYSTEM
426
PERFORM ANY NEEDED CONFIGURATION TO OS OR APPLICATONS
428
SNAPSHOT THE VIRTUAL DISK OF THE VIRTUAL MACHINE
430
CAPTURE THE VIRTUAL DISK TO THE CONTROL NODE
432

FIG. 26

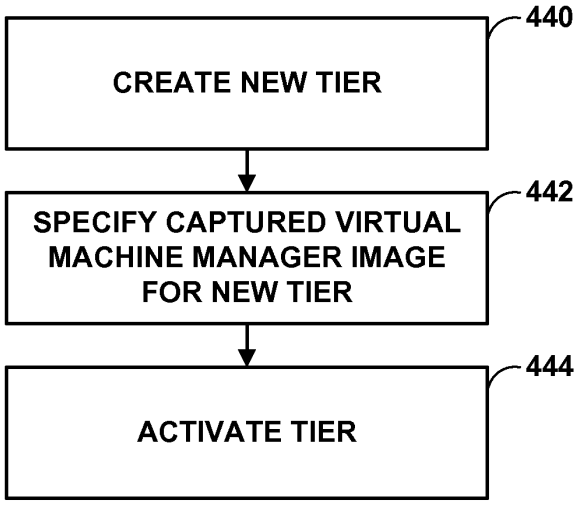


FIG. 27A

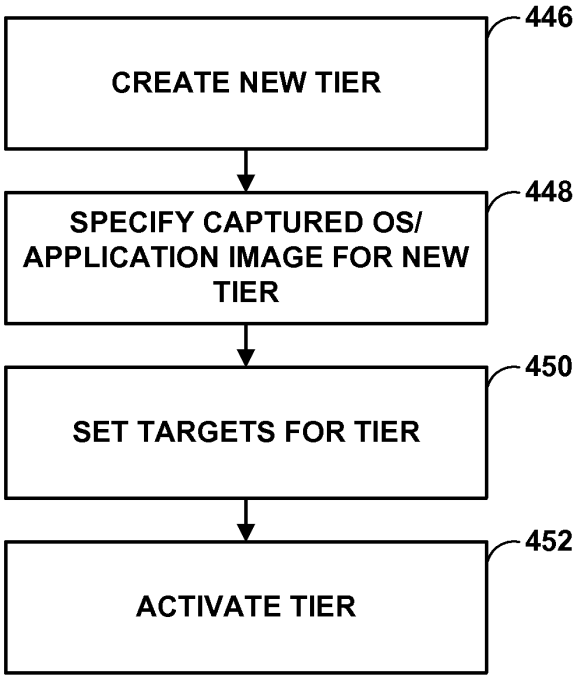


FIG. 27B

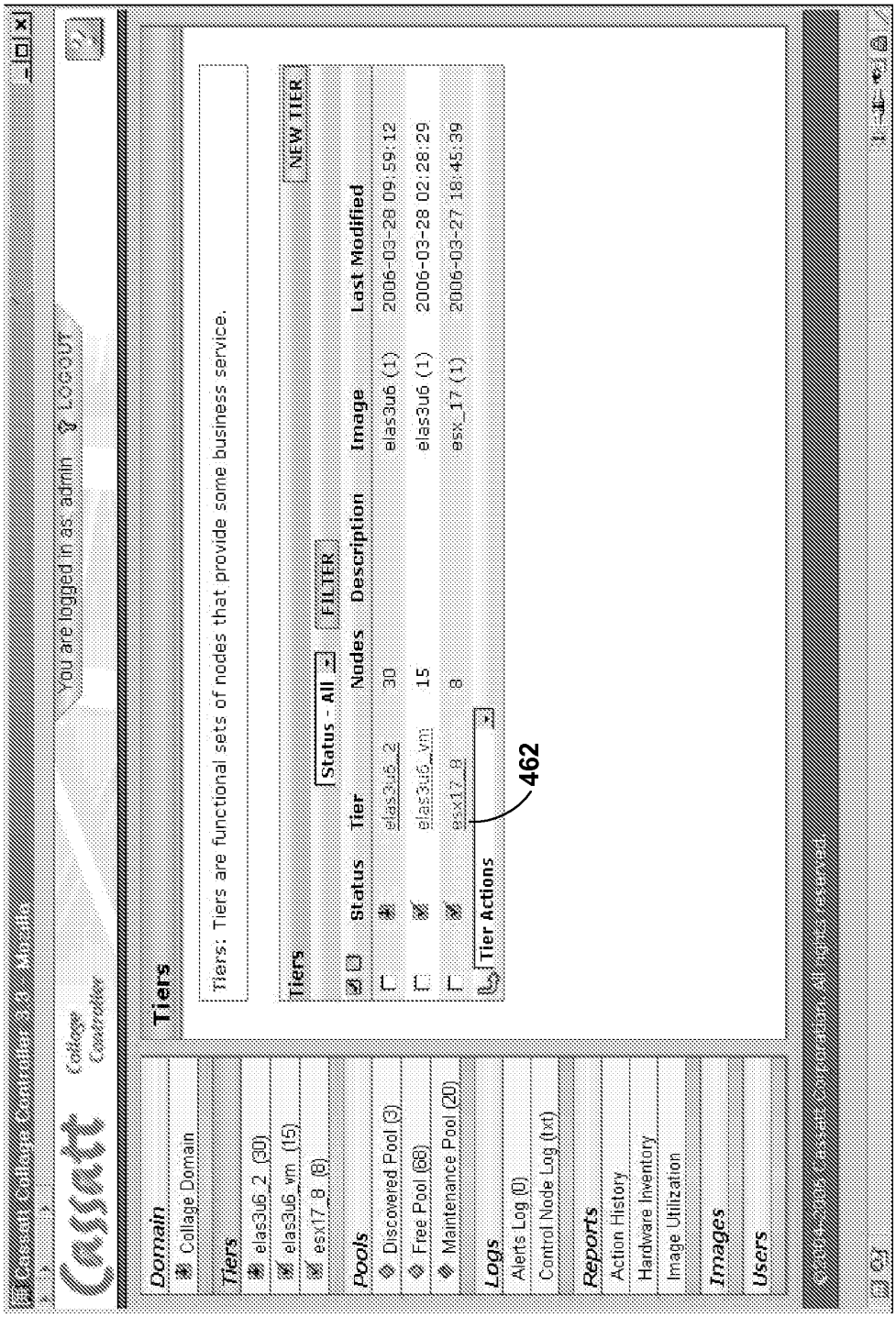

460
Tiers
Domain
Collage Domain
Tiers
elas3u6_2 (30)
elas3u6_vm (15)
esx17_8 (8)
Pools
Discovered Pool (3)
Free Pool (68)
Maintenance Pool (20)
Logs
Alerts Log (0)
Control Node Log (txt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
You are logged in as admin
Logout
Tiers: Tiers are functional sets of nodes that provide some business service.
Tiers
Status - All
FILTER
NEW TIER
Status
Tier
Nodes
Description
Image
Last Modified
elas3u6_2
30
elas3u6 (1)
2006-03-28 09:59:12
elas3u6_vm
15
elas3u6 (1)
2006-03-28 02:28:29
esx17_8
8
esx_17 (1)
2006-03-27 18:45:39
Tier Actions
462

FIG. 28

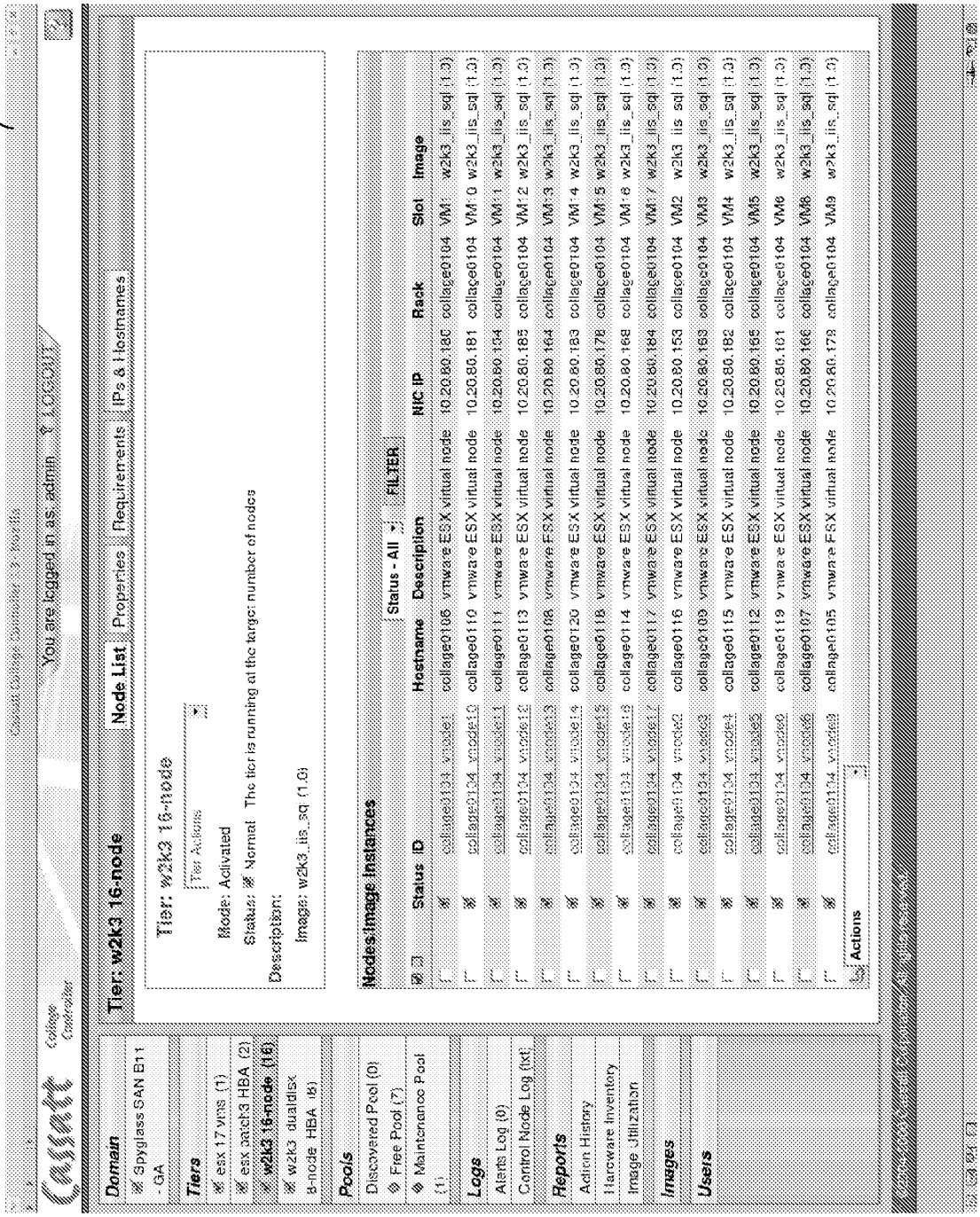

470
Cassatt
Cassatt Active Control 3.1 Portlets
Cassatt College Controller
You are logged in as: admin
Domain
Spyglass SAN B11 - GA
Tiers
esx 17 vms (1)
esx patch3 HBA (2)
w2k3 16-node (16)
w2k3 dualdisk 8-node HBA (8)
Pools
Discovered Pool (0)
Free Pool (7)
Maintenance Pool (1)
Logs
Alerts Log (0)
Control Node Log (fxt)
Reports
Action History
Hardware Inventory
Image Utilization
Images
Users
Tier: w2k3 16-node
Node List
Properties
Requirements
IPs & Hostnames
Tier: w2k3 16-node
Tier Actions
Mode: Activated
Status: Normal The tier is running at the target number of nodes
Description:
Image: w2k3_iis_sql (1.0)
Nodes:Image Instances
Status - All
FILTER
Status ID Hostname Description NIC IP Rack Slot Image
Actions

FIG. 29

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Filer:	Kent Joseph Sieffert/Karen Sorensen
Attorney Docket Number:	1072-021US01

Filed as Small Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Utility filing Fee (Electronic filing)	4011	1	75	75
Utility Search Fee	2111	1	250	250
Utility Examination Fee	2311	1	100	100
Pages:				
Claims:				
Claims in excess of 20	2202	8	25	200
Miscellaneous-Filing:				
Petition:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
		Total in USD ($)		**625**

Electronic Acknowledgement Receipt

EFS ID:	1642118
Application Number:	11694483
International Application Number:	
Confirmation Number:	1040
Title of Invention:	DISTRIBUTED COMPUTING SYSTEM HAVING AUTONOMIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES
First Named Inventor/Applicant Name:	Jagane Sundar
Customer Number:	28863
Filer:	Kent Joseph Sieffert/Karen Sorensen
Filer Authorized By:	Kent Joseph Sieffert
Attorney Docket Number:	1072-021US01
Receipt Date:	30-MAR-2007
Filing Date:	
Time Stamp:	17:17:14
Application Type:	Utility

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)
1		Application.pdf	304190	yes	59

	Multipart Description/PDF files in .zip description				
	Document Description		Start	End	
	Specification		1	53	
	Claims		54	58	
	Abstract		59	59	
Warnings:					
Information:					
2	Application Data Sheet	ADS.pdf	19061	no	3
Warnings:					
Information:					
This is not an USPTO supplied ADS fillable form					
3	Transmittal of New Application	Transmittal.pdf	33674	no	2
Warnings:					
Information:					
4	Drawings	Drawings.pdf	2250063	no	29
Warnings:					
Information:					
5	Fee Worksheet (PTO-06)	fee-info.pdf	8544	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	2615532		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

11/694,483

APPLICATION AS FILED – PART I

	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)		OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
FOR							
BASIC FEE (37 CFR 1.16(a), (b), or (c))				75			
SEARCH FEE (37 CFR 1.16(k), (i), or (m))				250			
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))				100			
TOTAL CLAIMS (37 CFR 1.16(i))	28 minus 20 =	8	X$25=	200	OR	X$50	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	3 minus 3 =		X$100=			X$200=	
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR						
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			N/A			N/A	
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	625		TOTAL	

APPLICATION AS AMENDED – PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))							OR		
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))							OR		
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comment on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.