

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NETFLIX, INC.,

Petitioner,

v.

AVAGO TECHNOLOGIES INTERNATIONAL SALES PTE. LIMITED,

Patent Owner.

Case No. IPR2020-01582

U.S. Patent 8,572,138

PETITIONER'S ORAL HEARING DEMONSTRATIVES

**NETFLIX, INC.,
Petitioner,**

v.

**AVAGO TECHNOLOGIES INTERNATIONAL
SALES PTE. LIMITED,
Patent Owner.**

Case No. IPR2020-01582
U.S. Patent 8,572,138

Petitioner's Demonstratives

Oral Argument

January 5, 2022

Claim 1

1[a]. A distributed computing system comprising:

[1b] a software image repository comprising non-transitory, computer-readable media operable to store:

[1c] (i) a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes,

[1d] 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

[1e] (ii) a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines; and

[1f] 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

[1g] 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.

Claim 19

19[a]. A method comprising:

[19b] 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,

[19c] 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;

[19d] storing a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines;







[19e] 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

[19f] 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.

Instituted Grounds

Grounds	Claims	Statutory Basis	Prior Art
1	1-5, 9, 19-21, and 26	§ 103	Khandekar
2	1-5, 9, 19-21, and 26	§ 103	Khandekar in view of Le
3	9, 10	§ 103	Khandekar in view of Le and Sidebottom
4	17	§ 103	Khandekar in view of Le and Stone

Remaining Disputes

	Petitioner	Patent Owner
1. Claim Construction (whether PO's 3-step process should be imported into claims)	 <p>The claims include an express definition for deployment.</p> <p>Even if adopted, PO's construction is satisfied by Khandekar and Le.</p>	 <p>PO's construction contradicts the claims, specification, and PO's district court positions.</p>
2. Virtual Machines (whether Khandekar and Le teach virtual machines)	 <p>The '138 patent relies on VMware software.</p> <p>Khandekar and Le are VMware references that teach VMware software.</p>	 <p>PO's arguments are contrary to the '138 patent and prior art.</p>
3. VMM Images (whether Khandekar and Le teach separate VMM images)	 <p>The VMM is included in the image of a physical computer.</p> <p>The prior art includes express motivation for the combination.</p>	 <p>PO failed to address differences between physical and virtual images.</p> <p>PO fails to address the <i>combination</i> of Ground 2.</p>

The Petition is supported by testimony from Dr. Shenoy

Dr. Shenoy is co-author of prior art cited by the applicant on the face of the '138 patent



US008572138B2

(12) **United States Patent**
Sundar et al. (10) **Patent No.:** **US 8,572,138 B2**
(45) **Date of Patent:** **Oct. 29, 2013**

(54) **DISTRIBUTED COMPUTING SYSTEM HAVING AUTOMATIC DEPLOYMENT OF VIRTUAL MACHINE DISK IMAGES**
(75) **Inventors:** Jagann Sundar, Saranga, CA (US);
Sanjay Radia, Fremont, CA (US);
David A. Henseler, Maplewood, MN (US)

(73) **Assignee:** CA, Inc., San Jose, CA (US)
(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1339 days.

(21) **Appl. No.:** 11/094,403
(22) **Filed:** Mar. 30, 2007

(65) **Prior Publication Data**
US 2007/0233608 A1 Oct. 4, 2007

Related U.S. Application Data
(60) Provisional application No. 60/787,280, filed on Mar. 30, 2006.

(51) **Int. Cl.**
G06F 12/00 (2006.01)
G06F 17/10 (2006.01)
(52) **U.S. Cl.**
USPC
(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**
U.S. PATENT DOCUMENTS

6,256,637 B1 * 7/2001 Venkatesh
6,511,059 B1 * 1/2003 Gupta et al.
6,775,829 B1 8/2004 Kozminski
6,865,737 B1 3/2005 Lucas et al.
7,091,239 B1 * 8/2006 van der

US 8,572,138 B2
Page 2

References Cited	FOREIGN PATENT DOCUMENTS
U.S. PATENT DOCUMENTS	WO 2006/081727 A1 8/2006
2003/0135009 A1 * 7/2003 Davis et al. 707/100	WO 2006/081801 A1 8/2006
2003/0135008 A1 * 7/2003 Haggas et al. 709/312	WO 2006/081804 A1 8/2006
2003/0140282 A1 7/2003 Kahr et al.	WO 2006/081805 A1 8/2006
2003/017175 A1 9/2003 Hirschfeld et al.	WO 2006/081901 A1 8/2006
2003/0182035 A1 * 10/2003 Daenert et al. 717/138	
2004/0008804 A1 5/2004 Ho	
2004/0162741 A1 8/2004 Flaxer et al.	
2004/0181794 A1 9/2004 Coleman et al.	
2004/0187104 A1 9/2004 Sathya et al.	
2004/0230948 A1 * 11/2004 Tabbar et al. 717/114	
2004/0260714 A1 12/2004 Rao et al.	
2005/0065200 A1 1/2005 Matsui et al.	
2005/0091040 A1 * 2/2005 Fulkerson et al. 718/1	
2005/0138370 A1 * 6/2005 Gould et al. 713/164	
2005/0193261 A1 9/2005 Liu et al.	
2006/017806 A1 8/2006 Jackson et al.	
2006/0178377 A1 8/2006 Jackson	
2006/0178907 A1 8/2006 Engquist et al.	
2006/0179084 A1 8/2006 Finck et al.	
2006/0179901 A1 8/2006 Henseler et al.	
2006/0179994 A1 8/2006 Finck et al.	
2006/0174238 A1 8/2006 Henseler et al.	
2006/0259202 A1 * 11/2006 Solomon et al. 707/27	
2007/0168019 A1 * 7/2007 Henseler et al. 717/101	
2007/0169049 A1 * 7/2007 Gungell et al. 717/151	

* cited by examiner

IPR2020-0

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.

Netfix, Inc. - Ex. 1001, Page 000002
IPR2020-01582 (Netfix, Inc. v. Avago Technologies International Sales PTE. Limited)

Patent Owner's Three-Stage Process Should Not Be Imported Into the Claims

1. Claim Construction
2. Virtual Machines
3. Image Instances of Virtual Machine Manager

PO in District Court: “no construction necessary” for any claim terms of the ’138 patent

PO’s District Court Positions:

Claim Term	PO’s Position
“virtual machine”	No construction necessary. Plain and ordinary meaning.
“virtual machine manager”	No construction necessary. Plain and ordinary meaning.
“image instance”	No construction necessary. Plain and ordinary meaning.
“rules engine”	No construction necessary. Plain and ordinary meaning.

PO asks the Board to import a three-step process from the embodiments into the claims

POR:

The logic and grammar of this paragraph mandate this three-step deployment process.

- **Step 1** – To begin the deployment process, the specification uses the phrase "Once control node 12 deploys the image instance of virtual machine manager 402 to a node" (*i.e.*, this step must happen first). Ex. 1001 at 34:5-6.
- **Step 2** – The specification next states that the virtual machine manager supports the virtual machine(s) on the nodes (*i.e.*, the virtual machine manager provides the virtual machine on the node). *Id.* at 34:7-11. This of course cannot happen as a matter of logic until after the control node deploys the image instance of the virtual machine manager onto the node.
- **Step 3** – The specification states "Subsequently, control node 12 may deploy an image instance of virtual disk image files 408 to one of the new unallocated nodes" (*i.e.*, the image instances of the software applications are deployed onto the virtual machine after the deployment of the image instance of the virtual machine manager). *Id.* at 34:11-13 (emphasis added); *see also id.* at 32:67-33:2.

Ex. 2001, ¶¶ 67-68.

PO never identified any claim language that requires construction

34

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.

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.

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 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.

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.

35

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). 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.

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.

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 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 associated SAN, and finally boots the virtual machine from the virtual disk image file.

Claim 1

PO never identified any claim language that requires construction



“[W]e agree that safety at once upon removal from the patient is an essential element of the invention as described by MBO. However, we cannot endorse a construction analysis that does not identify ‘a textual reference in the actual language of the claim with which to associate a proffered claim construction.’ *Johnson Worldwide Assocs., Inc. v. Zebco Corp.*, 175 F.3d 985, 990 (Fed. Cir. 1999); see also *Renishaw PLC v. Marposs S.p.A.*, 158 F.3d 1243, 1248 (Fed. Cir. 1998) (‘[I]t is manifest that a claim must explicitly recite a term in need of definition before a definition may enter the claim from the written description.’); *E.I. du Pont de Nemours & Co. v. Phillips Petroleum Co.*, 849 F.2d 1430, 1433 (Fed. Cir. 1988) (finding it improper to impose ‘a limitation read into a claim from the specification wholly apart from any need to interpret what the patentee meant by particular words or phrases in the claim’).”

MBO Laboratories, Inc. v. Becton, Dickinson & Co., 474 F.3d 1323, 1330-31 (Fed. Cir. 2007)

PO argues that the VMM must execute and actually manage one or more virtual machines – but PO took a contrary position in District Court

PO's PTAB Arguments

POR:

Step 2 – The specification next states that the virtual machine manager supports the virtual machine(s) on the nodes (*i.e.*, the virtual machine manager provides the virtual machine on the node). *Id.* at 34:7-11. This

PO's expert:

- Q. So Claim 1 requires that the virtual machine manager must **actually manage one more virtual machines**; is that right?
- A. Yes.

PO's District Court Briefing

“when executed.” ’138 Patent at claim 1. The patentee made this clear by expressly defining “virtual machine manager” as “a software program that is **capable of managing one or more virtual machines**,” not a program that *actually* does so. *Id.* at 32:60–62 (emphasis added).

PO admits: its construction would convert claim 1 into a “forbidden mix” of system and method claims that claim 1 seeks to avoid

Patent Owner’s District Court Briefing on Construction of Claim 1 (Ex-1027):

As the Federal Circuit held in *Chrimar Holding Co., LLC v. ALE USA Inc.*, 732 F. App’x 876 (Fed. Cir. 2018), as amended (June 1, 2018), “it is hardly uncommon for an apparatus or system claim, as a claim to a product rather than a process (or a forbidden mix), to be directed to capability, instead of actual operation.” *Id.* at 885. And “to infringe a claim that recites capability and not actual operation, an accused device need only be capable of operating in the described mode.” *Finjan, Inc. v. Secure Computing Corp.*, 626 F.3d 1197, 1204 (Fed. Cir. 2010) (citation and quotation marks omitted). Netflix’s attempt to import an actual-management limitation into

’138 Patent, Claim 1:

1. A distributed computing **system** comprising:
...

PO's expert admits claim 1 does not require the VMM to be executing

PO's expert:

- Q. Does claim 1 require the virtual machine manager to actually be executing on an application node?
- A. **It doesn't require anything to be executed. It describes what happens *when* the virtual machine manager is executed.**

Claim 1 does not require actual execution

'138 Patent, Claim 1:

1. 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
...



“Because the language of claim 1 refers to ‘programmable selection means’ and states ‘whereby **when** said alternate addressing mode is selected’ (emphases added), the accused device, to be infringing, **need only be capable of operating in the page mode**. Contrary to GI/M's argument, actual page mode operation in the accused device is not required.”

Intel Corp. v. U.S. Int'l Trade Comm'n, 946 F.2d 821, 832 (Fed. Cir. 1991)

“The plurality of virtual machines” refers to the virtual machines that the VMM images are *capable* of creating

VMM images have the capability, *when executed*, to provide a plurality of virtual machines.

“[E]xecutable on the plurality of virtual machines” simply refers to the virtual machines that the VMM images are *capable* of creating.

1. 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.

Claim 1 includes express definitions for deployment

'138 Patent, Claim 1:

1. 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 application 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.

The express definitional language in claim 1 is complete and unambiguous

“When claim language has as plain a meaning on an issue as the language does here, leaving no genuine uncertainties on interpretive questions relevant to the case, it is particularly difficult to conclude that the specification reasonably supports a different meaning.”

Straight Path IP Grp., Inc. v. Sipnet EU S.R.O., 806 F.3d 1356, 1361 (Fed. Cir. 2015).

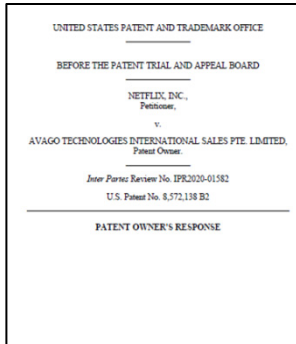


“The claim, however, provides its own definition of ‘remission.’ That is, claim 1 specifically recites: ‘remission is defined as a DAI score of 0 or 1.’ That express definitional language in the claim itself is complete and unambiguous, and it cannot be overridden by a description of exemplary embodiments in the Specification.”

GeneriCo, LLC v. Dr. Falk Pharma GmbH, IPR2016-00297, Paper 55, 10-11 (PTAB May 19, 2017).

PO imports an entire three-stage process from the specification but cites no words of manifest exclusion or restriction

Patent Owner:



a computer-readable medium. Each of these independent Challenged Claims requires the three-step deployment process (or include claim elements that must perform the three-step deployment process) described in the specification of the '138 Patent.

Federal Circuit:



“Even when the specification describes only a single embodiment, the claims of the patent will not be read restrictively unless the patentee has demonstrated a clear intention to limit the claim scope using ‘words or expressions of manifest exclusion or restriction.’”

Liebel-Flarsheim Co. v. Medrad, Inc., 358 F.3d 898, 906-08 (Fed. Cir. 2004) (internal citations omitted).

PO relies on descriptions of “exemplary” procedures

Ex-1001 ('138 Patent):

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

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

PO's expert:

Q. So therefore, the challenged claims do not require all the steps described for 27B, correct?

THE WITNESS: That's correct.

Q. The challenged claims do not require all the steps that are described with respect to Figure 27A, correct?

THE WITNESS: That's correct.

Most of Figure 27B relates to tiers, which are not required by the claims

Ex-1001 ('138 Patent):

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 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.

PO's expert:

Q. You'd agree that the challenged claims do not require the use of tiers, correct?

THE WITNESS: That's correct, they don't require the use of tiers. The tiers are discussed in some of the dependent claims. The independent claims certainly don't require the use of tiers.

Q. The challenged claims do not require creating a new tier, correct?

THE WITNESS: That's correct.

...

Q. The challenged claims do not require specifying captured OS/application image for a new tier, correct?

THE WITNESS: That's not required. It's not explicitly mentioned and certainly not in the independent claims.

Q. And the challenged claims do not require setting targets for a tier, correct?

THE WITNESS: Again, that's not explicitly mentioned in the independent claims.

Q. The challenged claims do not require activating a tier, correct?

THE WITNESS: Same answer as before.

Q. Does Claim 1 require booting of virtual machines?

THE WITNESS: No. I don't see any language in Claim 1 where I would make that interpretation.

PO's arguments require deployment and VMM execution *before* software application images can be stored

PO's logic: "the plurality of virtual machines" refers to specific VMs.

- Software application images cannot be stored in the repository until *after* VMMs are deployed and executed at application nodes.

PO's expert contradicted PO's logic:

Q. Now, just looking at the first paragraph of Claim 1, would you agree that the software image repository recited here in Claim 1 can contain the images described in Claim 1 even if there's no control node that provides additional functionality?

A. ... I suppose a simple answer is yes. I mean, **a software image repository can exist in the absence of a control node that does anything with that software image repository.**

1. 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.

PO's arguments are inconsistent with the '138 patent specification

'138 patent: software application images are created and “bootable on the virtual machines” before deployment

'138 patent:

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 deployment.

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

PO's expert:

Q. So then in this description, spanning columns 33 and 34, the two-phase capture process generates the image instances of the virtual machine managers and image instances of software applications, correct?

A. Yes.

Q. The images are then stored in a software image repository, correct?

A. Yes.

Q. And then the control node deploys images after the image instances of virtual machine managers and image instances of software applications are generated and stored on the control node, correct?

A. As described at the top of column 34, yes, that's the way that happens.

Each independent claim defines autonomic deployment as being provided by copying the image to an application node

Claim 19:

19. 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.

Claim 26:

26. 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.

Claim 19 recites steps performed at the control node

19. 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.

Claim 19 does not recite any actions taken by the application nodes.

PO's construction lacks written description


19. 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.



PO would require the rules engine to execute VMM images at application nodes

The specification does not mention the rules engine executing any images at application nodes

Ex-1001 ('138 Patent):

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.



No mention of executing images



No mention of rules engines in the portions of the specification that describe VMs or VMMs

Even if Adopted, PO's Construction Is Satisfied by Grounds 1 and 2

1. Claim Construction
2. Virtual Machines
3. Image Instances of Virtual Machine Manager

It was obvious to use Le's disk imaging teachings to provide VMMs to host computers

Claim 1:

1[a]. A distributed computing system comprising:

[1b] a software image repository comprising non-transitory, computer-readable media operable to store:

[1c] (i) a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes,

[1d] 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

[1e] (ii) a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines; and

[1f] 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

[1g] 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.

Petition:

Ground 2: Khandekar in View of Le Further Renders [1f] Obvious

Khandekar teaches autonomic deployment of VM and VMM images, as explained above for Ground 1. Le provides additional disk imaging teachings that complement and elaborate on Khandekar's teachings with techniques for copying images from a repository to remote computers, further rendering this limitation obvious. Ex. 1003 ¶¶148-154.

As explained previously, a POSITA would have found it obvious to use Le's disk imaging teachings to provide VMMs from the provisioning server to hosts.

See §VI.A.2, §VI.A.3[1c]. Disk images include the operating system and software applications, which would include Type 1 and 2 VMMs, respectively. See, e.g., Ex. 1005, 8:10-23, 10:55-62, 22:32-44; Ex. 1003 ¶149.

The last step of Le's imaging process reboots the computer to load the newly-deployed operating system

Dr. Shenoy's
Original Declaration:

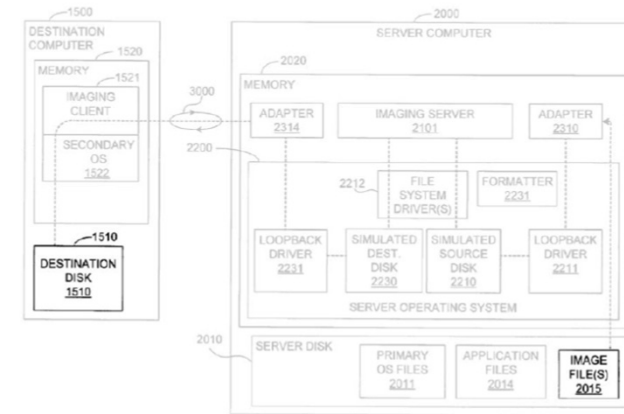
[1f] a control node that comprises an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes by causing the plurality of image instances of the virtual machine manager to be copied from the software image repository to the application nodes and

* * *

Ex-1005 (Le):

on it. Finally, the imaging server copies files and directories from the source file system to the destination file system, dismounts the two simulated disks 2210, 2230, and reboots the destination computer 1500 to allow it to load the new operating system deployed from the image 2015.

Fig. 5 (Emphasis Added)



See Ex. 1005, 26:52-27:9 (“... The invention’s image deployment process is illustrated in FIG. 5 and is symmetrical to the novel image capture process described above. ... The adapters 2310, 2314 and the file system drivers 2212 will perform the same functions as for image capture, but in the reverse direction. ... Finally, the imaging server copies files and directories from the source file system to the destination file system, dismounts the two simulated disks 2210, 2230, and reboots the destination computer 1500 to allow it to load the new operating system deployed from the image 2015.”) (emphasis added); see also *id.*,

PO's expert admits that execution of VMware ESX software on the application node creates a virtual machine

PO's expert:

Q. The execution of VMware ESX software on the application node creates a virtual machine which is an emulated computer platform, correct?

A. Yes.

Q. Does the 138 patent explain how exactly a virtual machine is created by a virtual machine manager?

A. Only to the extent that it says that the virtual machines are created when the virtual machine manager is executed.

PO's expert: once an image instance of the VMM (VMware ESX) is executed on the node, it creates the VM

PO's expert:

69. I note that Dr. Shenoy characterizes the specification of the '138 Patent as describing the three-step deployment process. For example, Dr. Shenoy recognizes that the image instance of the virtual machine manager deploys before the image instance of the software applications. *See* Ex. 2002 at 63:10-64:6; 68:12-69:2; and 73:2-74:3. Dr. Shenoy also recognizes that, once the image instance of the virtual machine manager is deployed on the node, it creates the virtual machine. *See id.* at 68:12-69:2, 73:2-74:3. Dr. Shenoy also acknowledges that the image instance of the software application (*e.g.*, virtual disk image) is deployed after the deployment of the virtual machine manager and after the creation of the virtual machine (by execution of the deployed virtual machine manager), because the image instance of the software application is deployed onto the virtual machine (*i.e.*, the unallocated node that is executing the virtual machine manager). *See id.* at 63:10-64:6; 73:2-74:3.

PO's expert:

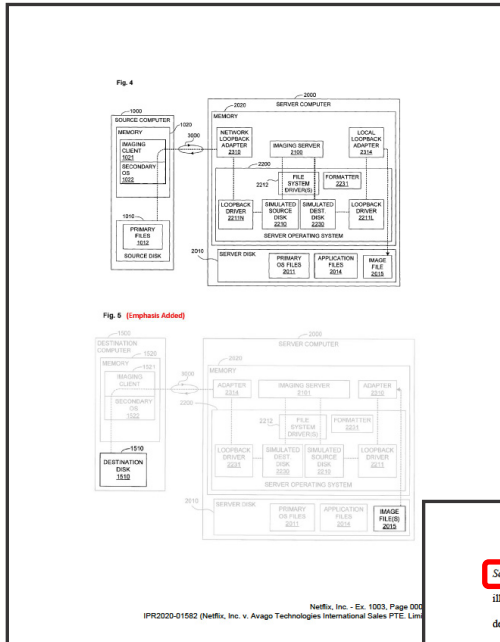
- Q. For the virtual machines in the embodiment of the 138 patent, what is the specific software that is used to emulate a computer platform?
- A. It's the software capability, the software that's provided by the virtual machine manager so that when you execute the virtual machine manager, the functionality of the virtual machine manager is to create these virtual machines.

Virtual machine images are deployed as a subsequent step because physical VMM imaging overwrites the entire hard drive

Dr. Shenoy's Original Declaration:

Ex-1005 (Le):

operating system 2200. Next, the imaging server leverages the server operating system's APIs and utilities to partition, then format, the destination disk 2230, destroying whatever partitions, file systems, data, etc., that was previously present on it. Finally, the imaging server copies files and directories



See Ex. 1005, 26:52-27:9. The invention's image deployment process is illustrated in FIG. 5 and is symmetrical to the novel image capture process described above. ... The adapters 2310, 2314 and the file system drivers 2212 will perform the same functions as for image capture, but in the reverse direction. ... Finally, the imaging server copies files and directories from the source file system to the destination file system, dismounts the two simulated disks 2210, 2230, and reboots the destination computer 1500 to allow it to load the new operating system deployed from the image 2015." (emphasis added); see also *id.*, 72:14-22 ("... It then copies all files and folders from the source to the destination."), 20:25-37 ("A virtual disk can also be used as a regular disk image for physical computers, i.e., it can be archived, cataloged, then later deployed to one or multiple physical computers for backup/restore or cloning purposes."), FIG. 9; *infra* §VIA.3[1g].

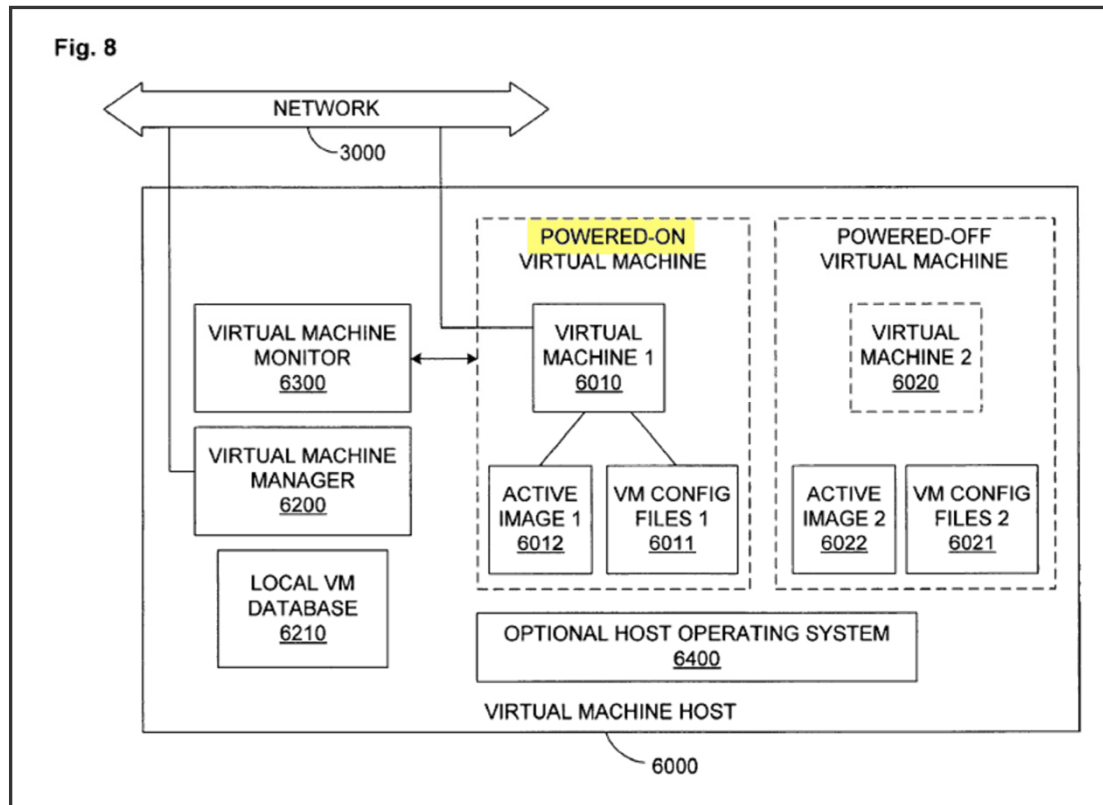
152. Le teaches an "exemplifying embodiment," the "Universal Computer Management System (UCMS)," (Ex. 1005, 59:21-27), which is similar to Khandekar's provisioning server. See *supra* §VIA.2. The UCMS manages and deploys images to virtual machines and physical computers. See, e.g., Ex. 1005, 65:20-33, 59:29-36. To guide deployment to physical computers, the UCMS includes functionality that analyzes the hardware configuration of new computers when they are registered. See Ex. 1005, 70:12-35.

Netfix, Inc. - Ex. 1003, Page 000101
IPR2020-01582 (Netfix, Inc. v. Avago Technologies International Sales PTE. Limited)

Le teaches that VMMs are running and managing virtual machines at hosts

Le includes numerous teachings where VMs are powered on

Ex-1005 (Le):



When a virtual machine is powered on, a virtual machine monitor program 6300 controls it and manages the interactions between the software—commonly called “guest” software—running inside of the virtual machine and the host’s physical resources, such as hardware devices. Some virtual machine products, such as VMware GSX Server and Connectix Virtual Server, access the host’s physical resources through the services of an industry-standard host operating system 6400 such as Linux or Windows; other products, like VMware ESX Server, include a virtual machine monitor 6300 and system-level kernel capable of managing physical resources directly without the need for a host operating system.

PO's expert admitted that for a VM to be resumed on a node, a VMM must be executing on that node

PO's expert:

Q. And after a virtual machine disk image file is copied to an application node, the virtual machine manager software allows that virtual machine disk image file to be resumed on a virtual machine at that node, correct?

A. Yes.

Q. And in order for a virtual machine to be resumed on a node, that means virtual machine manager software must be executing on that node, correct?

A. Correct.

Q. And as a general matter, if a virtual machine is running on a computer, then a virtual machine manager must also be running to provide and manage that virtual machine, correct?

A. Yes.

Q. And if a virtual machine exists on a node, then a virtual machine manager must be running on that node, correct?

A. Yes.

Ground 1: Khandekar teaches deploying and executing VMM images at host computers

Dr. Shenoy's Original Declaration (Ex-1003):

VMs.” Ex. 1004, 5:51-58 (emphasis added). The image instances of the VMM are executed on the host computer, allowing it to run a plurality of VMs. *See* Ex. 1004, 13:17-35 (emphasis added):

As is mentioned above, a VMM is typically included for each VM in order to act as its software interface with the underlying host system. ... Several products are available from VMware, Inc. of Palo Alto, Calif., that configure conventional computers, both stand-alone desktop computers and multi-user servers, **to load and run multiple VMs.** ...

Ex. 1004, 7:7-27 (“A VMM is usually a thin piece of software that runs directly on top of the host and virtualizes all, or at least some of the resources of the machine, or at least of some machine.”).

Khandekar's VMMs manage the execution of the virtual machines

Ex-1004 (Khandekar):

Dr. Shenoy:

92. Furthermore, Khandekar is a VMware patent that teaches the use of VMMs that run directly on the underlying system hardware or alternatively on a Host OS, but in either case, the VMMs manage the execution of the VMs. *See id.*, 6:62-7:27. VMMs had long been known in the art; Khandekar teaches that “[t]he general features of VMMs are known in the art and are therefore not discussed in detail here.” *Id.* Given Khandekar’s teachings and the fact that it is a VMware patent, a POSITA would have found it obvious to use known VMM software, including VMware ESX software. *See id.*

Some interface is usually required between a VM and the underlying host platform, in particular, between the VM and the host OS 220, which is the “real” OS in the sense of being either the native OS of the underlying physical computer, or the OS or other system-level software that handles actual I/O operations, takes faults and interrupts, etc. As is mentioned above, the host platform is responsible for actually executing VM-issued instructions and transferring data to and from the actual, physical memory 112 and storage devices 114. This interface is often referred to as a “virtual machine monitor” (VMM).

A VMM is usually a thin piece of software that runs directly on top of the host and virtualizes all, or at least some of, the resources of the machine, or at least of some machine. In some systems, the VMM may even be included in the system software itself, or operate alongside it at system level. In some conventional systems, VMMs run directly on the underlying system hardware 100, and will thus act as the “real” operating system for its associated VM. In other systems, such as the one shown in FIG. 1, the HOS 220 is interposed as a software layer between VMMs and the hardware. Still other arrangements are possible, all of which may be used in the invention. Regardless of which level the VMM is implemented on, the interface exported to the respective VM is the same as the hardware interface of the machine, or at least of some predefined hardware platform, so that the guest OS 320 usually cannot determine the presence of the VMM. The VMM also usually tracks and either forwards (to the HOS 220) or itself schedules and handles all requests by its VM for machine resources as well as various faults and interrupts. The general features of VMMs are known in the art and are therefore not discussed in detail here.

Ex-1004 (Khandekar):

In the following description of the invention, merely for the sake of simplicity, only one VM/VMM pair is discussed. The discussion applies equally, however, to all such VM/VMM pairs that may be included in any given implementation of the invention. Moreover, it will be assumed below that when a VM is installed on a host, then a corresponding VMM is either already installed on the host, or is included and installed together with the VM.

According to another embodiment of the invention, a virtual machine-to-hardware interface, such as a virtual machine monitor (VMM), is installed on each of a plurality of hardware hosts. Deployment of the VM is thereby made substantially independent of the overall physical hardware configuration. At least one of the hosts is then selected as the host for actual VM deployment. Because each host plus interface forms a separate physical host platform, the actual host may be selected substantially arbitrarily, according to any given criterion.

For images of pre-built and custom-built VMs, Khandekar teaches copying the virtual disk image file to the host and booting the VM with the image

Ex-1004 (Khandekar):

Once a VM is deployed on a host, it may be started (“powered on”) in any conventional manner, either by the provisioning server remotely, or by the user himself. Note that powering on a VM does not normally involve a manual action, but rather can be carried out through a software procedure.

Provisioning Engine 810

The provisioning engine is the component that ties together the various components described above. The provisioning engine preferably takes each user request and runs the corresponding provisioning operation as a separate process. An algorithm used in a prototype of the invention for provisioning a fully configurable (custom-built) VM is as follows:

- 1) Get a new task ID from the task manager 820.
- 2) Call the heuristics engine 880, to determine the VM's hardware configuration and the host to deploy on.
- 3) If the heuristics engine returns an error, call the task manager to abort this task and inform the user about the error, or invoke and implement some other rule base to suggest action.
- 4) Create a new temporary usage area 860 for this task on the host server.
- 5) Store the user input in the user input (choices) file 863 in the temporary usage area.
- 6) Create the VM config floppy image 861 from the template floppy image 834 for the VM, as described above in the “Temporary Usage Area” section.
- 7) For each application selected by the user, create the custom answer file 864 and the application installation software 862 using the process described above in the “Application Library” section.
- 8) Copy the VM's config file and the disk files to the host.
- 9) Call the API in the host to register this new VM.
- 10) Call the API in the host to connect the VM config floppy image to the VM.
- 11) Call the API in the host to power on the VM. This causes the configuration script in the VM to configure the operating system parameters using the data in the config floppy image; it also causes installation of the applications.
- 12) If there is any error during step 11, call the task manager to abort the task.
- 13) When the configuration step successfully completes, call the API in the host to power the VM down.
- 14) Clean up the temporary usage area and inform the user of the successful completion of the task.

7) For each application selected by the user, create the custom answer file 864 and the application installation software 862 using the process described above in the “Application Library” section.

8) Copy the VM's config file and the disk files to the host.

9) Call the API in the host to register this new VM.

10) Call the API in the host to connect the VM config floppy image to the VM.

11) Call the API in the host to power on the VM. This causes the configuration script in the VM to configure the operating system parameters using the data in the config floppy image; it also causes installation of the applications

For Instantly-Deployable VMs, Khandekar teaches deploying images by coping the images and resuming them on the host

Ex-1004 (Khandekar):

According to an alternative “instant-deployment” embodiment of the invention, the user selects one of a plurality of fully configured VMs, which are stored in a library 840 of instant deployment VMs and are kept in a suspended state until deployed. The provisioning server then selects an appropriate host 1000, 1001 on which to provision the VM, copies the necessary data on that host, and resumes the execution of the VM on the new host. The VM is thus instantly available to

For provisioning an instantly-deployable VM, the algorithm followed in the prototype of the invention was as follows:

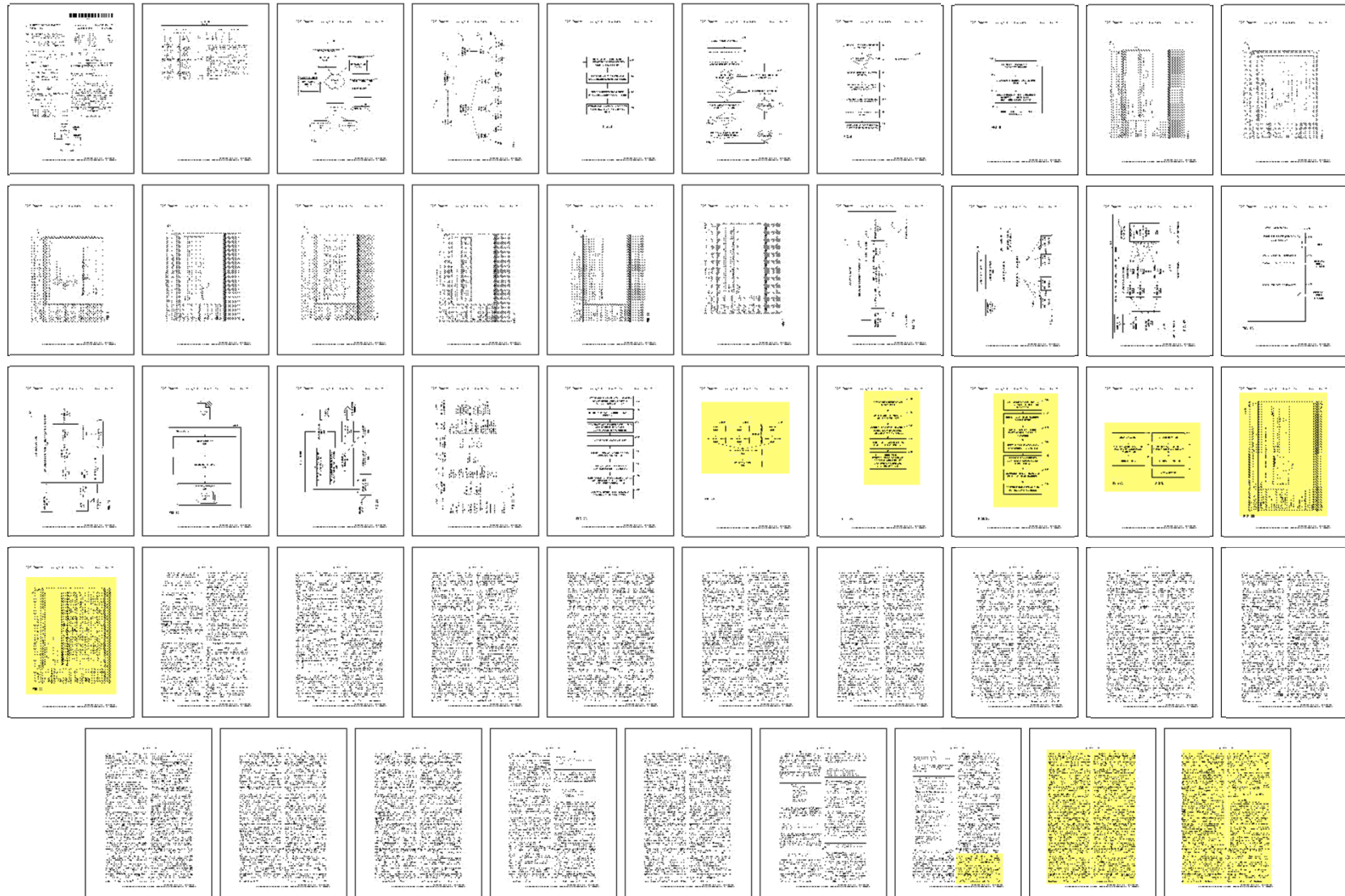
- 1) Get a task ID from the task manager 820.
- 2) Call the heuristics engine to determine the host to deploy on.
- 3) Create a temporary usage area 860 and store the user input in it.
- 4) Copy the VM's config file 842, REDO logs 843 for the disks 841 and suspend-to-disk image file 844 to the host.
- 5) Modify the config file for the VM to access the disk remotely on the server.
- 6) Call the API on the host to register the VM.
- 7) Call the API in the host to resume the VM.
- 8) Clean up the temporary usage area and inform the user of successful completion of the task.

Khandekar and Le Teach “Virtual Machines”

1. Claim Construction
2. Virtual Machines
3. Image Instances of Virtual Machine Manager

The '138 patent mentions virtual machines as an afterthought

The specification's first 31 columns describe non-virtualized applications:



Only the last 6 figures and 5 columns mention virtual machines

PO's expert admits that virtual machines were known in the art

PO's expert:

Q. You'd agree that virtual machines were known and used before the 138 patent, correct?

A. Yes.

Q. Why were virtual machines used in the art before the 138 patent?

THE WITNESS: The primary reason for using virtual machines is to provide a layer of – or to provide a measure of independence from underlying hardware.

PO's argument

POR:

One of the primary shortcomings of the prior art cited by Petitioner is that the term virtual machine (or "VM"), as used in Petitioner's prior art, has a very different meaning than the term "virtual machine," as used in the '138 Patent – an issue that Petitioner ignores, but which permeates throughout its invalidity analysis. When the

PO's expert admits: the '138 patent does not disclose a new kind of VMM

PO's expert:

Q. Does the 138 patent disclose how a POSITA would go about creating new virtual machine manager software?

A. It discloses how image instances of virtual machine managers could be created. It doesn't -- **doesn't disclose how to, for example, design and implement a new kind of virtual machine manager.**

Q. Is there a part of the 138 patent that describes the specific software components that are used to emulate a computer platform?

THE WITNESS: I'm not aware. **I don't recall any part of the 138 patent that describes, as I would put it, the internal workings, let's say, of a virtual machine.**

The '138 patent relied on existing VMware software

PO's expert:

- Q. VMware software included the functionality to execute image instances of software applications on virtual machines, correct?
- A. It included the functionality to provide -- or includes the functionality to provide the virtual machines on which software applications could be executed.
- Q. And VMware software also has the functionality to load and execute those software applications on its virtual machines, correct?

THE WITNESS: Yes.

The '138 patent only discloses using commercial VMware software to provide virtual machines

Ex-1001 ('138 Patent):

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, Calif.

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.

PO's expert:

Q. The *only* example of a virtual machine manager disclosed in the embodiments of the 138 patent is *VMware ESX software*, correct?

A. Yes. That appears to be the case, yes.

Q. ... in those embodiments of the 138 patent, the virtual machines that are emulating bare metal hardware, those are provided by, in some embodiments, VMware ESX software, correct?

A. Yes.

Khandekar and Le are VMware patents

(12) United States Patent Khandekar et al.	(10) Patent No.: US 7,577,722 B1 (45) Date of Patent: Aug. 18, 2009
(54) PROVISIONING OF COMPUTER SYSTEMS USING VIRTUAL MACHINES	7,448,079 B2 * 11/2008 Tremain 726/14 2002/0069369 A1 * 6/2002 Tremain 713/201 2002/0120660 A1 * 8/2002 Hay et al. 709/100 2002/0144257 A1 * 10/2002 Matsushima 717/178 2002/0156877 A1 * 10/2002 Lu et al. 709/221 2004/0268348 A1 * 12/2004 Waki et al. 718/1
(75) Inventors: Dilip Khandekar , San Jose, CA (US); Dragutin Petkovic , Saratoga, CA (US); Pratap Subrahmanyam , Sunnyvale, CA (US); Bich Cau Le , San Jose, CA (US)	FOREIGN PATENT DOCUMENTS
(73) Assignee: VMware, Inc. , Palo Alto, CA (US)	WO WO 02/03220 A2 1/2002
(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 582 days.	OTHER PUBLICATIONS
	"ImageCast—Total PC Deployment and Image Management, Enterprise Edition," StorageSoft, Inc., 2000, (http://www.storagesoft.com/support/doc/imagecast/DS_imagecast_v45.pdf).

(12) United States Patent Le et al.	(10) Patent No.: US 8,209,680 B1 (45) Date of Patent: Jun. 26, 2012
(54) SYSTEM AND METHOD FOR DISK IMAGING ON DIVERSE COMPUTERS	6,108,147 A * 8/2000 Jeon 360/15 6,108,697 A * 8/2000 Raymond et al. 709/218 6,205,450 B1 * 3/2001 Kanome 707/203 6,253,300 B1 * 6/2001 Lawrence et al. 711/173 6,477,624 B1 * 11/2002 Kedem et al. 711/147 6,519,762 B1 * 2/2003 Colligan et al. 717/170 6,598,131 B2 * 7/2003 Kedem et al. 711/147 6,658,435 B1 * 12/2003 McCall 707/204 6,721,846 B2 * 4/2004 Mund et al. 711/118 6,804,774 B1 * 10/2004 Larvoire et al. 713/2 7,000,231 B1 * 2/2006 Gold 717/174 2003/0191911 A1 * 10/2003 Kleinschnitz et al. 711/154
(75) Inventors: Bich C. Le , San Jose, CA (US); Dilip Khandekar , San Jose, CA (US); Sirishkumar Raghuram , Maharashtra (IN)	OTHER PUBLICATIONS
(73) Assignee: VMware, Inc. , Palo Alto, CA (US)	Lee et al., "Petal: Distributed Virtual Disks," 1996, ACM, p. 84-92.*
(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1435 days.	

Khandekar and Le teach VMs provided by VMware software

Ex-1004:

As is mentioned above, a VMM is typically included for each VM in order to act as its software interface with the underlying host system. In the preferred embodiment of the invention, a VMM is already pre-installed on each host 1000, 1001 so as to ensure compatibility and proper configuration. Several products are available from VMware, Inc. of Palo Alto, Calif., that configure conventional computers, both stand-alone desktop computers and multi-user servers, to load and run multiple VMs. It would also be possible, however, for a corresponding VMM to be included along with each staged VM, as long as the characteristics of the host on which the VM is to be installed are known. An advantage of pre-installing a VMM, without a VM, on a plurality of hosts, such as hosts 1000, 1001, 1003, is that it will then be possible to deploy a given VM on any arbitrary one (or more) of these hosts. Moreover, as long as the VMM on each host exports a known hardware interface to the VM deployed on it, then VM deployment will be substantially independent of the actual physical hardware configuration.

Ex-1005:

When a virtual machine is powered on, a virtual machine monitor program 6300 controls it and manages the interactions between the software—commonly called “guest” software—running inside of the virtual machine and the host’s physical resources, such as hardware devices. Some virtual machine products, such as VMware GSX Server and Connectix Virtual Server, access the host’s physical resources through the services of an industry-standard host operating system 6400 such as Linux or Windows; other products, like VMware ESX Server, include a virtual machine monitor 6300 and system-level kernel capable of managing physical resources directly without the need for a host operating system.

Compared with the hosted deployment, a kernel may offer greater performance because it can be co-developed with the VMM and be optimized for the characteristics of a workload consisting of VMMs. The ESX Server product of VMware, Inc., has such a configuration. A kernel-based virtualization system of the type illustrated in FIG. 2 is described in U.S. patent application Ser. No. 09/877,378 (“Computer Configuration for Resource Management in Systems Including a Virtual Machine”), which is also incorporated here by reference.

It was obvious to use VMware software including ESX

Dr. Shenoy (Ex-1003):

host, so that an appropriate VMM image can be provided. *See id.* These synergies would have motivated a POSITA to combine Khandekar and Le by using Le's imaging teachings to deliver virtualization infrastructure to hosts, including VMware software such as ESX and GSX software, and Le's teaching of virtual machine monitors/managers to the extent Patent Owner attempts to distinguish between these terms—both were components of VMware software that would have obviously been included in the disk imaging techniques described above.

of VMware software such as the ESX Server. *See, e.g., id.*, 65:58-64, 66:11-19. A POSITA would have been motivated to use VMware ESX software to implement Khandekar's teachings because Khandekar is a VMware patent, and its VMM teachings would obviously have applied to ESX and other VMware software. *See*

computers and multi-user servers, to load and run multiple VMs."'). Moreover, a POSITA would have been motivated to apply known features of VMware software and systems, including ESX server and Le's teachings of virtual machine monitors and managers, to Khandekar's VMM teachings. *See* Ex. 1005, 13:22-30. Additionally, Khandekar teaches that VM images are created (*see, e.g.,* Ex. 1004, 9:10-25, FIG. 7) and a POSITA would have naturally looked to Le for additional details on how to create images and how to use them to deploy software, including VMware software such as ESX, GSX, and related components, to remote hosts.

virtual machine manager provide a plurality of virtual machines. By 2006, VMM systems that were known in the art supported multiple VMs per instance, including commercially available software such as VMware GSX and ESX Server (*see* Ex. 1005, 65:65-66:10) and the Xen hypervisor, an x86 virtual machine monitor, (*see* Ex. 1011 at 1, 5, FIG. 1). *See also* Ex. 1016 ¶[0076] ("VM manager 20 creates

PO's Expert:

Q. Do you agree that a POSITA, in 2006, would have had the knowledge and skills to use virtualized computing systems including the use of commercially available software from VMware such as ESX and GSX software?

A. Yes.

Q. So for the virtual machine manager, a POSITA reading the 138 patent would use software that was known in the art such as VMware software, correct?

A. Yes. And I believe the Sundar, the 138 patent basically encourages that view, encourages that approach.

Khandekar uses VMM software to provide virtual machines containing the same components identified by PO's expert

PO's Expert:

Q. A newly-created virtual machine would contain a **virtual processor**, correct?

A. Yes.

Q. It would also contain a **virtual memory**, correct?

A. Yes.

Q. And a newly-created virtual machine would include **virtual devices**, correct?

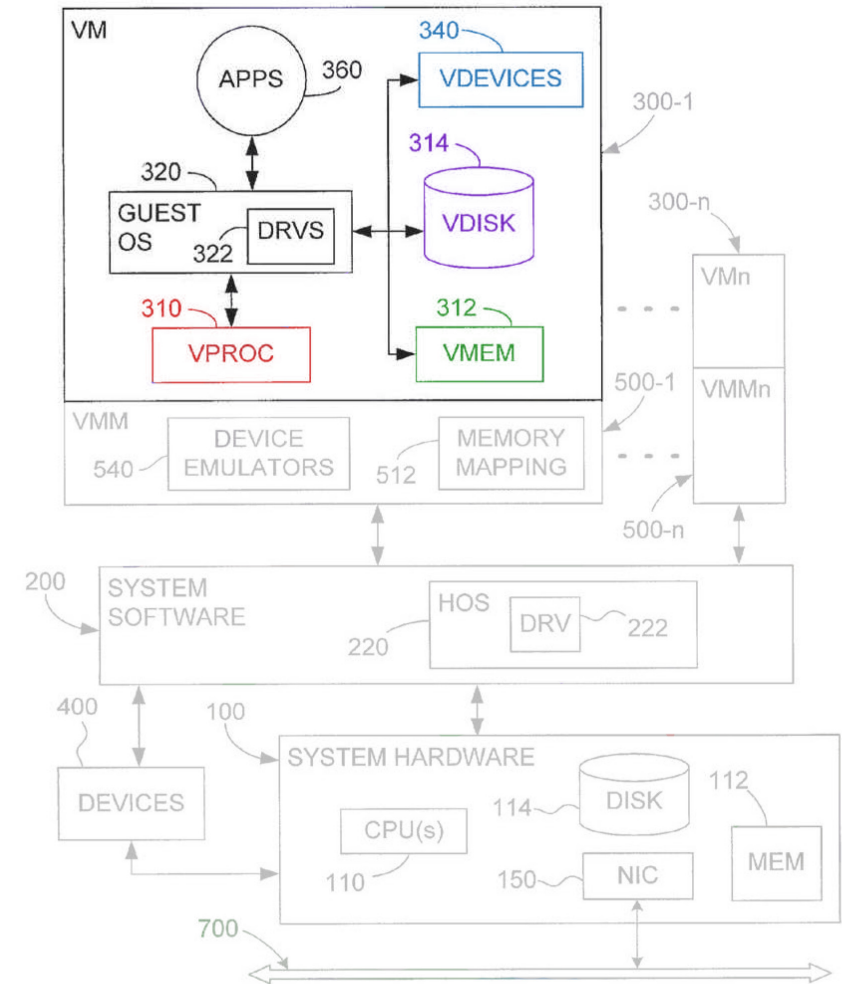
A. Yes.

Q. And a newly-created virtual machine would include a blank **virtual disk**; is that right?

A. Possibly.

Ex-1004 (Khandekar):

FIG. 1
(Prior Art)



The '138 patent relies on VMware software to create virtual machine images

PO's expert:

Q. **VMware ESX software captures virtual machine disk image files as .vmdk files, correct?**

THE WITNESS: **Yes.**

Q. So then to take a snapshot of the virtual disk of the virtual machine in step 430, a POSITA would have used VMware ESX software to capture a .vmdk file, correct?

THE WITNESS: They could have, yes.

Q. That's one of the ways that's disclosed by the 138 patent, right?

A. Yes, it's taught as an example in passing in the discussion of this image capture process described in Figure 26.

...

Q. Right. And does the 138 patent disclose any other specific ways to capture a snapshot other than using VMware software?

A. Does it mention -- I don't remember exactly, but I believe it mentions, is it the GSX product as well, perhaps not in this particular place in the patent specification, but possibly elsewhere where it discusses virtual machine manager products in general.

Q. Okay. Can you -- do you know where -- can you point to any specific places where the 138 patent discusses?

A. I could have a look here. Okay. Well, I see two instances of the specific mention of VMware ESX. I don't see mentions of other VMware products. It could be that I'm mixing this up with one of the other patents where other products were mentioned.

Khandekar teaches virtual machine disk image files

Ex-1004 (Khandekar):

FIG. 3

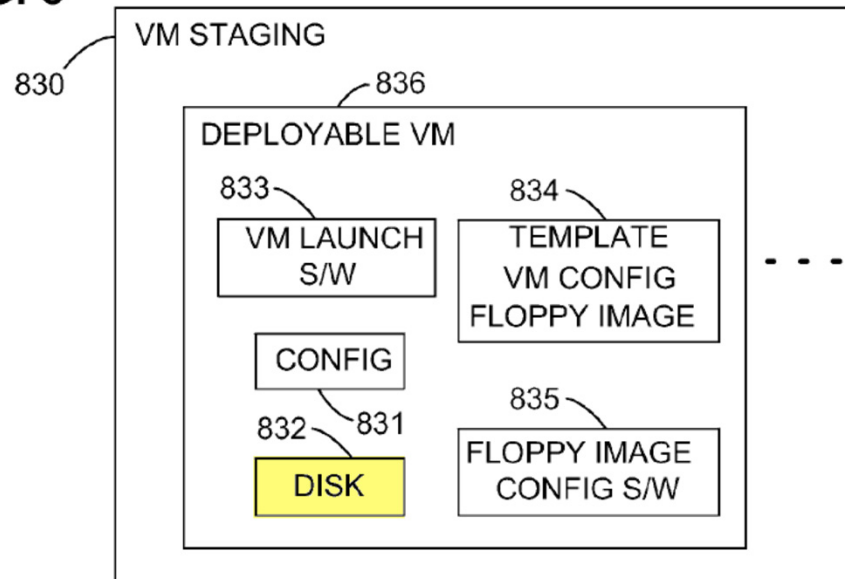
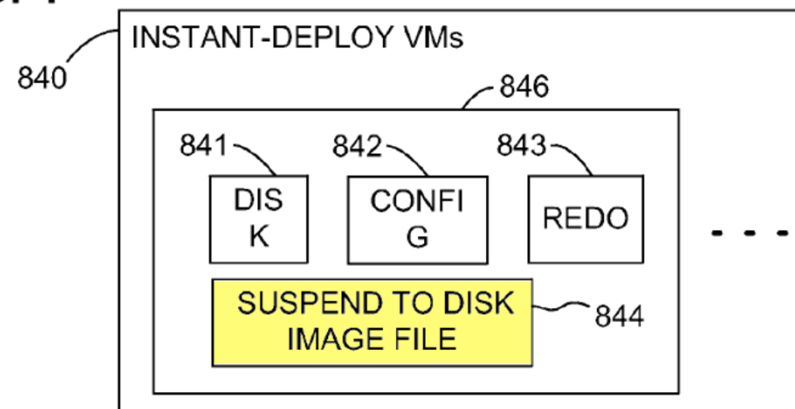


FIG. 4



Khandekar and Le (Ground 2) Teach Image Instances of a Virtual Machine Manager

1. Claim Construction
2. Virtual Machines
3. Image Instances of Virtual Machine Manager

The POR does not address the *combination* of Khandekar and Le

The POR attacks Khandekar alone (pp. 45-64):

[illegible]

The POR spends *only a single page* on Le (pp. 64-65):

[illegible]

POR:

2. Ground 2: Khandekar and Le

Petitioner further alleges that the combination of Khandekar and Le renders independent Claims 1, 19, and 26, and dependent Claims 2-5, 9, 20, and 21, obvious under 35 U.S.C. § 103. *See* Pet. at 9. As noted above, Khandekar fails to disclose or suggest each limitation of any independent claim of the '138 Patent. Le fails to remedy these deficiencies of Khandekar, and the Challenged Claims are equally valid over the combination of Khandekar and Le. *See* Ex. 2001, ¶¶ 148-152.

With regard to the installation of a VMM image instance, Le teaches only that, "[t]o power on and run the cloned virtual machine, all that the host computer needs is to have the appropriate VMM software installed." Ex. 1005 at 14:28-30; *see also* Ex. 2001, ¶ 149. Thus, Le, like Khandekar, presumes that a VMM has already been installed when discussing deployment, and provides no enabling disclosure of how the pre-existing installation of a VMM might have occurred. *See* Ex. 2001, ¶ 150. Le also appears to teach that the "cloned virtual machine," which would correspond to the software application image instance, is copied to the host before the VMM image instance is executed. *See id.* Certainly, the Petitioner fails to identify, and a

Obviousness cannot be overcome by attacking references individually

“Non-obviousness cannot be established by attacking references individually where the rejection is based upon the teachings of a combination of references. Thus, Petersen must be read, not in isolation, but for what it fairly teaches in combination with the prior art as a whole.”

In re Merck & Co., 800 F.2d 1091, 1097 (Fed. Cir. 1986) (citations omitted).



“The remainder of Bradium's arguments concerning this claim limitation lack merit because they attack the disclosures of the two references individually. A finding of obviousness, however, cannot be overcome ‘by attacking references individually where the rejection is based upon the teachings of a combination of references.’”

Bradium Techs. LLC v. Iancu, 923 F.3d 1032, 1050 (Fed. Cir. 2019).

Le reinforces Khandekar's teachings with implementation details for disk imaging techniques

Dr. Shenoy's Opening Declaration (Ex-1003):

A. Grounds 1 and 2: Claims 1-5, 9, 19-21, and 26 are Rendered Obvious by Khandekar Alone (Ground 1), and Khandekar in View of Le (Ground 2)

* * *

42. It is my opinion that a POSITA would have found Claims 1-5, 9, 19-21, and 26 obvious based on the teachings of Khandekar by itself. Le, another reference by the same inventors and assignee VMware, reinforces Khandekar's teachings with implementation details regarding, for example, VMware software, the ratio of VMMs to VMs, and disk imaging techniques.

43. For example, Khandekar teaches that each VMM may support a single VM, or each VMM may support multiple VMs. *See, e.g.*, 1004, 7:28-38. Khandekar therefore teaches a VMM providing a plurality of VMs. To the extent it is argued otherwise, Le teaches that VMM providing a plurality of VMs. Le further teaches implementation details regarding VMware software and techniques for creating and deploying images to physical and virtual computers. Le's teachings complement Khandekar's teachings, and the combination further renders the challenged claims obvious.

Le reinforces Khandekar's teachings with implementation details for disk imaging techniques

Petition:

2. Motivation to Combine Khandekar and Le

Khandekar teaches a provisioning server with a repository of VM images that automatically deploys those images to hosts using a heuristics/rules engine. *See, e.g.,* Ex. 1004, 4:30-35, 4:58-65, 10:27-43, Abstract. Khandekar teaches deploying a VMM along with the VM image. *See, e.g., id.*, 8:36-44, 13:17-35 (“a VMM is typically included for each VM in order to act as its software interface with the underlying host system”). Le reinforces Khandekar's teachings with implementation details regarding, for example, the ratio of VMMs to VMs, VMware software, and disk imaging techniques. Ex. 1003 ¶52.

The Motivation to Combine Khandekar and Le

Dr. Shenoy's Opening Declaration (Ex-1003):

58. For example, Khandekar teaches that the provisioning server may provide a VMM along with a VM to a host computer. *See, e.g.*, Ex. 1004, 13:17-35. Le teaches imaging techniques that a POSITA would have been motivated to use for this purpose. A VMM is a program running on a physical computer, either as the operating system (Type 1 hypervisor OS) or an application running on top of a separate operating system (Type 2 hypervisor). *See, e.g.*, Ex. 1004, 7:7-27; Ex. 1005, 66:2-10. Le teaches techniques for physically imaging the hard disk of a remote computer, to deploy a cloned image with an operating system (which would include a Type 1 VMM) and applications (which would include a Type 2 VMM) installed. *See, e.g.*, Ex. 1005, 2:19-21, 15:63-65, 16:29-30, 54:14-21, 1:64-3:13; *infra* §VI.A.3[1c], [1f]. Le therefore teaches a mechanism for deploying VMMs to remote computers, which is useful in a variety of common scenarios in distributed computing including (a) installing a VMM onto a computer that does not currently support virtual machines, (b) initializing a new host that is connected to the network, and (c) reimaging a host that is not functioning properly. Le teaches the use of imaging clients to handle the imaging process at the remote computer (*see*

Ex. 1005, Figs. 3-5), and the use of the PXE standard to automatically initialize new computers by installing and then executing the imaging client. *See, e.g.*, Ex. 1005, 37:36-49 (“For example, the PXE (pre-boot execution environment) standard specifies a mechanism for **automatically initializing a new, bare-metal computer when it first appears on the network**. ... the server can download an arbitrary program over the network into the computer’s memory. The computer executes the program once it is downloaded.”) (emphasis added). Therefore, Le teaches techniques for implementing Khandekar’s teachings of providing a VMM to host computers, and a POSITA would have found it obvious and been motivated to apply those teachings to Khandekar.

Ground 2 provides an additional basis for the plurality of image instances of a virtual machine manager

Dr. Shenoy's Opening Declaration (Ex-1003):

[1c] (i) a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes,

* * *

Ground 2 (Khandekar in View of Le) Provides an Additional Basis for the Plurality of Image Instances of a Virtual Machine Manager

99. Le provides additional disk image teachings that complement and elaborate on Khandekar's teachings, further rendering it obvious for a plurality of VMM images to be stored in the provisioning server. *See supra* §VI.A.2 (explaining how and why Le's disk image teachings would have been combined with Khandekar). As explained above, Khandekar teaches the provisioning server providing VMMs to hosts. Le further teaches various disk-imaging techniques that facilitate the remote deployment of software onto host machines, techniques which a POSITA would have found obvious to use for deploying VMM images.

It was obvious to use Le's disk imaging teachings to provide VMMs to host computers

Petition:

Ground 2: Khandekar in View of Le Further Renders [1f] Obvious

Khandekar teaches autonomic deployment of VM and VMM images, as explained above for Ground 1. Le provides additional disk imaging teachings that complement and elaborate on Khandekar's teachings with techniques for copying images from a repository to remote computers, further rendering this limitation obvious. Ex. 1003 ¶¶148-154.

As explained previously, a POSITA would have found it obvious to use Le's disk imaging teachings to provide VMMs from the provisioning server to hosts. See §VI.A.2, §VI.A.3[1c]. Disk images include the operating system and software applications, which would include Type 1 and 2 VMMs, respectively. See, e.g., Ex. 1005, 8:10-23, 10:55-62, 22:32-44; Ex. 1003 ¶149.

Ground 2 uses Le's teachings to image physical host computers

Le teaches techniques for writing a physical image onto the host's hard drive

Ex-1005 (Le):

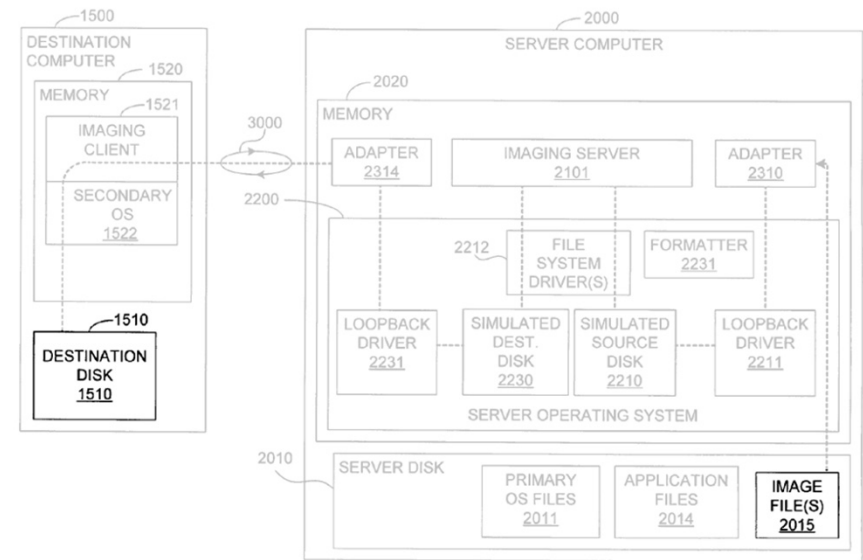
Disk imaging is often used as a cloning technology for rapid computer deployment. Image-based computer cloning

represents a snapshot of a second computer's disk. Image capture is the process of creating an image file from a computer's disk. A common disk imaging setup is to use two

computer. During capture, disk data is transferred from the client to the server over a network or a cable.

The reverse of the capture process is called deployment.

Fig. 5 (Emphasis Added)



It would have been obvious to use Le's disk imaging techniques to install a VMM onto a host computer

Ex-1005 (Le):

An image is often used to make multiple clones of a base computer. The approach is to capture an image from the base computer's disk, and then deploy the same image to multiple destination computers. Before the initial image is captured, the base computer is configured with an operating system and common set of software applications that are required on all clones. Any computer deployed from this image would inherit the same set of software.

Dr. Shenoy:

102. Type 1 VMMs, also referred to as the hypervisor OS, acted as the operating system. *See* Ex. 1004, 7:7-14; Ex. 1005, 13:12-21. Type 2 VMMs ran as software application on top of a separate OS, such as Windows or Linux. *See* Ex. 1004, 7:14-27; Ex. 1005, 13:5-11. Le teaches the use of VMware ESX software, which is a Type 1 VMM, and VMware GSX software, which is a Type 2 VMM. *See, e.g.*, 1005, 66:2-10. Thus, a POSITA would have found it obvious and been motivated to use disk imaging techniques to install a VMM onto a host computer because disk imaging techniques allowed rapid deployment of disk images—including the operating system and software applications such as Type 1 and Type 2 VMMs, respectively—onto remote computers over a network. *See, e.g.*, Ex. 1005, 15:63-65, 16:29-30. As an example, a POSITA would have found it obvious to use images of VMware ESX and GSX software, as explained above, to initialize a host if it does not already have a VMM so that the host can support VMs. As another example, disk imaging techniques can also be used to re-initialize a host that is no longer functioning properly.

There is no dispute that the VMM is the OS for Type 1 hypervisors (VMware ESX) or that the VMM is an application for Type 2 hypervisors (VMware GSX)

The Level of Ordinary Skill

Petition:

¶¶33-34. A person with less formal education but more relevant practical experience may also meet this standard. *Id.* A POSITA in 2006 with the above level of skill would have had the knowledge and skills necessary to:

- Create a distributed computing system with a network of computing devices/servers;
- Use virtualized computing systems, including the use of commercially available software from VMware such as ESX and GSX software;
- Create images of physical and virtual machines, including images that include VMMs, operating systems, and applications;

PO's expert:

Q. Do you agree that a POSITA, in 2006, would have had the knowledge and skills to create images of physical and virtual machines including images that include virtual machine managers, operating systems and applications?

A. Yes.

The prior art includes express teaching, suggestion, and motivation to use a VMM image

Ex. 1014:

[0062] Turning to the concept of a boot sequence, as can be readily understood, the first partition does not need a virtual machine manager 202. Instead the virtual machine manager 202 may be loaded at the time of concurrency need along with the loading the general purpose operating system, for example. However, to assist in the transition, since late binding of a virtual machine manager below an existing partition may be complex, the specialty partition may anticipate a concurrent general purpose operating system and associate with a virtual machine manager, which may be on disk or non-volatile memory, such that the boot occurs with this thin virtual machine manager working. For late binding, the virtual machine manager may describe or construct a partition that describes the pre-existing OS and then subjugate or hoist it onto the virtual machine manager. The hypervisor also may be loaded first or part of an OS-virtual machine manager image. It may be essentially part of a firmware image.

Ex. 1016:

[0043] The subsequent steps in the installation procedure of a fixed system remain essentially the same regardless of how the system is actually managed. Utilities (running on that machine or on another system) are required for formatting and partitioning the bootable hard drive. The boot ROM on the target hardware must be reprogrammed with an appropriate boot environment (note that it is also possible to save the original ROM as an image for use by guest systems). The bootable core and the virtual machine manager image must be installed in a secure partition.

PO mischaracterized Dr. Shenoy's testimony

Sur-Reply:

⁷ When questioned extensively where this argument appeared in his original declaration, Dr. Shenoy could identify only his "reply dec." EX-2007, 133:15-140:5 (emphasis added).

Dr. Shenoy pointed to ¶58 of his *opening* declaration:

- Q. Is there any discussion in Paragraph 149 through 153 of your declaration that talks about physical images?
- A. In Paragraph 149, I said: "As explained previously, a POSITA would have found it obvious to use Le's disk imaging techniques to install a VMM, including the operating system and software applications as Type 1 and Type 2 VMM respectively on the remote computers, and it talks about the previous section, 6831C.[sic]
- Q Are you going to that section?
- A I'm looking at that section right now.
- Q Okay.
- A. So if you look at **Paragraph 58** of the declaration, it talks about Le using imaging techniques that a POSITA would have used. It talks about a VMM is a program running on a physical computer and talks about Type 1 hypervisor, OS and talks about Type 2. It says Le teaches techniques for physically imaging the hard disk of the remote computer to deploy a cloned image with an operating system which would include a Type 1 VMM.
- Q. It says a cloned image with an operating system and applications, correct?
- A. Well, when it says "applications," that's referring to the Type 2 VM, because a Type 2 VMM is an application from an operating system's perspective.

PO ignores the fact that the VMM image is *physical* while the virtual machine image is *virtual*

PO's expert:

139. Not only does Khandekar lack an explicit teaching that images of a VMM would be stored separately from images of VMs (*see supra* ¶¶ 43-44, 53-54, 138), I believe that Khandekar teaches that including a VMM with a VM requires including the VMM *within the same image* as the VM. As stated in Khandekar, "[i]t would also be possible, however, for a corresponding VMM to be included along with each staged VM, **as long as the characteristics of the host on which the VM is to be installed are known.**" Ex. 1004 at 13:25-28 (emphasis added). The highlighted phrase teaches an understanding whereby the included VMM *depends directly* on the VM with which it is to be associated, and that dependence thereby constrains the choice of VMM to include with the VM. If the VMM were imaged and stored separately from the VM, then there would be no dependence between the VMM and the VM and therefore no need to know *a priori* the characteristics of the host on which the VM is to be installed.

PO's expert offers no explanation of how the VMM (an OS or application on the physical host) could be included in the virtual disk image

Khandekar Alone (Ground 1) Teaches Image Instances of a Virtual Machine Manager

1. Claim Construction
2. Virtual Machines
3. Image Instances of Virtual Machine Manager

Khandekar teaches separate images for VMMs and virtual machines

Ex-1004
(Khandekar):

As is mentioned above, a VMM is typically included for each VM in order to act as its software interface with the underlying host system. In the preferred embodiment of the invention, a VMM is already pre-installed on each host 1000, 1001 so as to ensure compatibility and proper configuration. Several products are available from VMware, Inc. of Palo Alto, Calif., that configure conventional computers, both stand-alone desktop computers and multi-user servers, to load and run multiple VMs. It would also be possible, however, for a corresponding VMM to be included along with each staged VM, as long as the characteristics of the host on which the VM is to be installed are known. An advantage of pre-installing a VMM, without a VM, on a plurality of hosts, such as hosts 1000, 1001, 1003, is that it will then be possible to deploy a given VM on any arbitrary one (or more) of these hosts. Moreover, as long as the VMM on each host exports a known hardware interface to the VM deployed on it, then VM deployment will be substantially independent of the actual physical hardware configuration.

That there is a choice to install these either together or separately indicates that they are separate images

Khandekar uses virtualization to allow the same virtual machine image to be deployed independent of underlying physical hardware

Ex-1004 (Khandekar):

According to another embodiment of the invention, a virtual machine-to-hardware interface, such as a virtual machine monitor (VMM), is installed on each of a plurality of hardware hosts. Deployment of the VM is thereby made substantially independent of the overall physical hardware configuration. At least one of the hosts is then selected as the host for actual VM deployment. Because each host plus interface forms a separate physical host platform, the actual host may be selected substantially arbitrarily, according to any given criterion.

Using separate images is consistent with Khandekar's goals of virtualization

Dr. Shenoy:

32. Therefore, using separate images for the VMM and VM is consistent with Khandekar's goals and teachings because it facilitates independence between the VM and underlying physical hardware, while a monolithic image (with both a VMM and VM) undermines Khandekar's goals by creating dependencies to physical hardware that otherwise would not exist. Having a combined image with both a VMM and VM would undermine the independence created by virtualization because the VMM image is a physical image for a physical host computer. Khandekar does not teach or require a merged image, since merging them would tie the VM image to a physical computer, defeating the purpose of virtualization.

PO's expert admits: primary reason for using virtual machines was to provide independence from underlying physical hardware

PO's expert:

Q. You'd agree that virtual machines were known and used before the 138 patent, correct?

A. Yes.

Q. Why were virtual machines used in the art before the 138 patent?

THE WITNESS: The primary reason for using virtual machines is to provide a layer of -- or to provide a measure of independence from underlying hardware.

Q. Why was it beneficial for virtual machines to provide a measure of independence from underlying physical hardware?

A. Basically otherwise it complicates the development of software systems. It provides portability, for example. If I have a software application that I want to be able to run on, let's say, three different hardware platforms, if I can -- if I can provide virtual machines that can run on top of those three different platforms, the development of the virtual machines would be much simpler than the development of three different versions of the presumably larger software application.

Q. I see. So the use of virtual machines allows the same software application to run on different underlying physical hardware platforms; is that right?

A. Yes.

PO Mischaracterization #1

PO's expert:

139. Not only does Khandekar lack an explicit teaching that images of a VMM would be stored separately from images of VMs (*see supra* ¶¶ 43-44, 53-54, 138), I believe that Khandekar teaches that including a VMM with a VM requires including the VMM *within the same image* as the VM. As stated in Khandekar, "[i]t would also be possible, however, for a corresponding VMM to be included along with each staged VM, **as long as the characteristics of the host on which the VM is to be installed are known.**" Ex. 1004 at 13:25-28 (emphasis added). The highlighted phrase teaches an understanding whereby the included VMM *depends directly on the VM* with which it is to be associated, and that dependence thereby constrains the choice of VMM to include with the VM. If the VMM were imaged and stored separately from the VM, then there would be no dependence between the VMM and the VM and therefore no need to know *a priori* the characteristics of the host on which the VM is to be installed.

Ex-1004 (Khandekar):

load and run multiple VMs. It would also be possible, however, for a corresponding VMM to be included along with each staged VM, as long as the characteristics of the host on which the VM is to be installed are known. An advantage of

The VMM image must match the physical hardware of the host, not the VM.

PO Mischaracterization #2

PO's expert:

140. I believe that this teaching of Khandekar is further reinforced by the fact that Khandekar teaches that VMMs and VMs are always to be considered in pairs:

In the following description of the invention, merely for the sake of simplicity, **only one VM/VMM pair** is discussed. The discussion applies equally, however, to **all such VM/VMM pairs that may be included** in any given implementation of the invention. Moreover, it will be assumed below that when a VM is installed on a host, **then a corresponding VMM is either already installed on the host, or is included and installed together with the VM.**

Id. at 8:36-43 (emphasis added). Khandekar thus teaches that any VMM included

Ex-1004 (Khandekar):

In the following description of the invention, merely for the sake of simplicity, only one VM/VMM pair is discussed. The discussion applies equally, however, to all such VM/VMM pairs that may be included in any given implementation of the invention. Moreover, it will be assumed below that when a VM is installed on a host, then a corresponding VMM is either already installed on the host, or is included and installed together with the VM.

In FIG. 1, VMMs 500-1, . . . , 500-*n*, are shown, acting as interfaces for their respective attached VMs 300-1, . . . , 300-1*n*. It would also be possible to use a single VMM to act as the interface to all VMs, although it will in many cases be more difficult to switch between the different contexts of the various VMs (for example, if different VMs use different guest operating systems) than it is simply to include a separate VMM for each VM. The important point is simply that some well-defined, known interface should be provided between each VM and the underlying system hardware 100 and software 200.

Khandekar teaches that one VMM may support multiple VMs, which the POR did not address or dispute

PO Mischaracterization #3

POR:

teaching from Khandekar. *See* Ex. 2001, ¶ 141. Indeed, given Khandekar's teaching that VMMs and VMs are paired (and in fact "form a single cooperating system,") a POSITA would find no reason to store a VMM separately from a VM, except by using hindsight in light of the disclosure of the '138 Patent. *Id.*, ¶¶ 140-41 (quoting Ex. 1004 at 7:40-42).

Ex-1004 (Khandekar):

Because operation of the VM requires some form of interface with the host platform, from the perspective of the system designer the VM and VMM can be considered to form a single cooperating system, although, as is mentioned above the VMM is typically transparent to the VM. The important point to keep in mind when it comes to this invention is that the VM is structured and performs as a complete "real" computer system, even though it is a software construct and assumes an interface such as the VMM.

This refers to the execution of a VM on a VMM; it has nothing to do with images

Reserve Slides

A VMM manages the execution of virtual machines

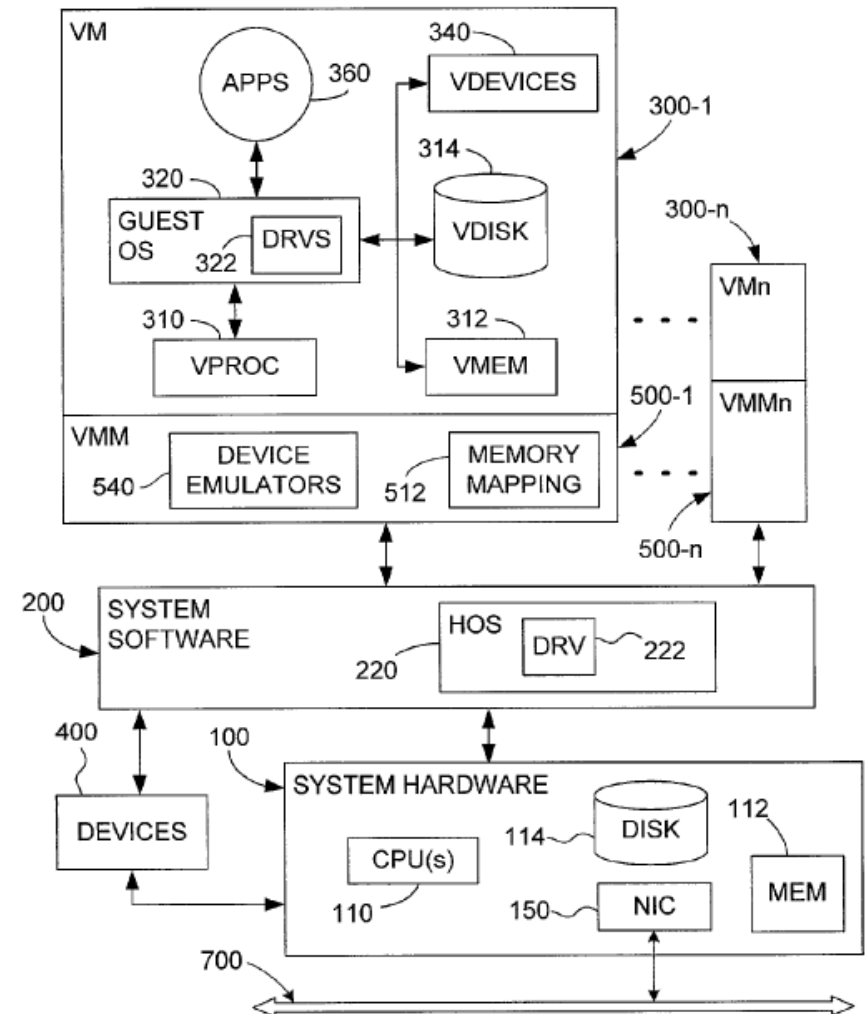
Ex-1004 (Khandekar):

Virtual Machine Monitors

Some interface is usually required between a VM and the underlying host platform, in particular, between the VM and the host OS 220, which is the “real” OS in the sense of being either the native OS of the underlying physical computer, or the OS or other system-level software that handles actual I/O operations, takes faults and interrupts, etc. As is mentioned above, the host platform is responsible for actually executing VM-issued instructions and transferring data to and from the actual, physical memory 112 and storage devices 114. This interface is often referred to as a “virtual machine monitor” (VMM).

A VMM is usually a thin piece of software that runs directly on top of the host and virtualizes all, or at least some of, the resources of the machine, or at least of some machine.

FIG. 1
(Prior Art)



PO's expert admitted that VM images cannot execute unless a VMM image is already running and providing the VMs

PO's expert:

- Q. A virtual machine manager is required for a virtual machine to run; is that right?
- A. Yes.
- Q. And for the 138 patent, the virtual machine managers are executed on application nodes; is that right?
- A. Yes.
- Q. The virtual machine manager runs and manages the virtual machine; is that correct?
- A. Yes.
- Q. A virtual machine cannot exist without a virtual machine manager that is running and managing it; is that right?
- A. That's right. I would say at least in the -- you know, in the context of the Sundar patent. If you look in the Khandekar patent, Khandekar calls the captured images virtual machines. So Khandekar might argue that virtual machines can exist without the virtual machine manager.
- Q. And is it your view that Khandekar uses the term "virtual machine" to refer to what the 138 patent describes as the virtual disk image?
- A. Yes.
- Q. So for a virtual machine, as that term is used in the 138 patent, then a virtual machine cannot exist without a virtual machine manager that is running and managing it, correct?
- A. Yes, that's correct. I think the 138 patent clearly spells out every time there's a discussion of a virtual machine, that virtual machine is created or provided as a result of executing a virtual machine manager.

The image instances of software applications can be snapshots of already- or previously-running VMs

PO's expert:

Q. And then after this snapshot of a previously-running virtual machine is deployed to a node with a virtual machine manager, then the snapshot can be resumed on that different node, correct?

THE WITNESS:- So the snapshot of the virtual disk can be deployed on some virtual machine on the new node. And if memory serves, the 138 patent talks about persistent and nonpersistent virtual disks where the persistent ones would be a resumption essentially.

A reference must be considered for everything that it teaches

PO's expert:

Q Okay. So you'd agree, then, that when analyzing Khandekar for obviousness, Khandekar must be considered for everything it teaches, not simply the described invention or preferred embodiments?

THE WITNESS: I agree with that. And maybe to give an example to convey a sense of what my thinking is, it might discuss things in there that are background IP to the disclosed invention. And one could take that description of background IP as part of the teachings of the patent.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on December 27, 2021, complete copy of the foregoing document was served via email to all parties to this proceeding at the addresses indicated:

Daniel S. Young (Registration No. 48,277)
dyoung@adseroip.com
dyoung@sbiplaw.com

Chad E. King (Registration No. 44,187)
chad@adseroip.com
cking@sbiplaw.com

ADSERO IP LLC
d/b/a Swanson & Bratschun LLC
8210 Southpark Terrace
Littleton, CO 80120

Date: December 27, 2021

/Harper Batts/
Harper Batts (Reg. No. 56,160)
Attorney for Petitioner