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.

_____

IPR2020-01582

U.S. Patent No. 8,572,138

_____

PETITION FOR *INTER PARTES* REVIEW

OF U.S. PATENT NO. 8,572,138

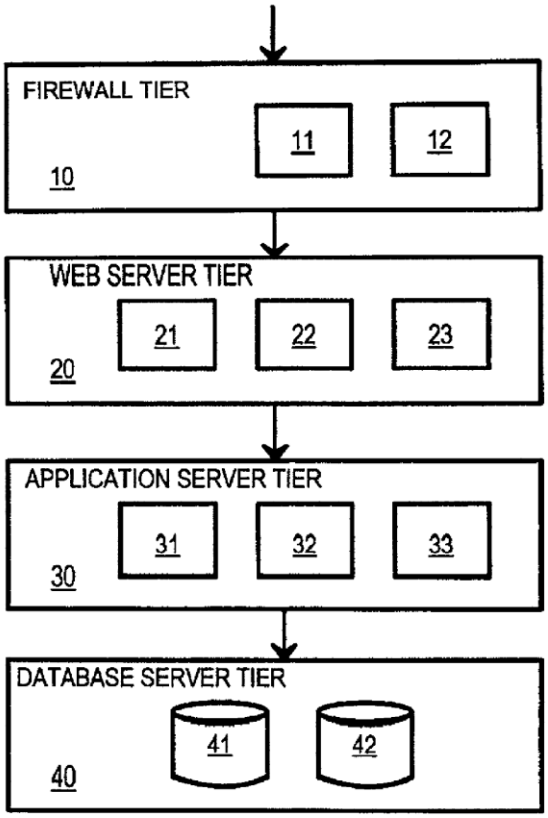
TABLE OF CONTENTS

EXHIBIT LIST

Exhibit No.	Description
1001	U.S. Patent No. 8,572,138 ("the '138 patent")
1002	Prosecution File History for the '138 patent
1003	Declaration of Prashant Shenoy
1004	U.S. Patent No. 7,577,722 to Khandekar *et al*. ("Khandekar")
1005	U.S. Patent No. 8,209,680 to Le *et al*. ("Le")
1006	U.S. Patent No. 7,962,633 to Sidebottom *et al*. ("Sidebottom")
1007	U.S. Patent Application Publication No. 2003/0036886 to Stone ("Stone")
1008	Khanna *et al*., *Application Performance Management in Virtualized Server Environments*, NOMS 2006 ("Khanna")
1009	Urgaonkar *et al*., *Dynamic Provisioning of Multi-tier Internet Applications*, ICAC'05 ("Urgaonkar")
1010	Excerpt of S. Russell, P. Norvig, *Artificial Intelligence - A Modern Approach* (2d. ed. Prentice-Hall 2003) ("Russell")
1011	Barham *et al.*, *Xen and the Art of Virtualization*, SOSP '03: ACM Symposium on Operating Systems Principles, 164-177 (Oct., 2003)
1012	U.S. Patent No. 6,496,847 to Bugnion *et al*. ("Bugnion")
1013	U.S. Patent Application Publication No. 2004/0019672 to Das *et al*. ("Das")
1014	U.S. Patent Application Publication No. 2007/0220246 to Powell *et al*. ("Powell")
1015	JP Patent Application Publication No. 2001-318797 to French
1016	English translation of JP Patent Application Publication No. 2001-318797 to French ("French")
1017	Declaration of Sylvia D. Hall-Ellis

I. BACKGROUND

For decades, websites such as e-commerce sites, online retail stores, and financial services sites, have been implemented with distributing computing systems where applications are assigned to networked computers, also referred to as servers, nodes, or hosts. Ex. 1003 ¶¶3-5. By the 1990s and early 2000s, it was common for nodes to be organized into tiers, including web server, application, and database tiers. *See* Ex. 1009 at 1, 3; Ex. 1007 ¶¶[0024]-[0039], Fig. 2, 1:



The multi-tier model facilitated monitoring, diagnosis, and automated control, which helped meet service-level objectives ("SLOs") by monitoring node status and using automated processes were often used to respond to performance

bottlenecks. *See* Ex. 1007 ¶¶[0037]-[0038]; Ex. 1009 at 3-4; Ex. 1003 ¶6. For example, automation was used to deploy new application instances to nodes, which allowed system capacity to scale with user demand. *See, e.g.*, Ex. 1009 at 4-5; Ex. 1007 ¶[0027]; Ex. 1003 ¶7.

By the 1990s and early 2000s, it was common for automated systems to use knowledge-based systems, such as rules engines, to select an appropriate node based on the monitored status of nodes. *See* Ex. 1013 ¶¶[0083]-[0084], [0163]-[0164], FIGS, 5, 10; Ex. 1006, 3:44-58; Ex. 1001, 26:40-61; Ex. 1010 at 286. Such systems and rule engines, including forward-chaining rule engines employing algorithms such as LEAP or RETE, were known in the art and commonly used for automated application deployment. *See id.* Ex. 1003 ¶¶8-9. To facilitate automated deployment, application images were often stored in a repository, typically with a database of available images and configurations. *See, e.g.*, Ex. 1004, 12:13-23; Ex. 1003 ¶10.

It became common in the early 2000s, for distributed computing systems to use virtual machines ("VM") for existing tiered architectures and to automate their deployment using rules engines. *Id.* ¶¶3-4. Virtualization allowed a physical server to host multiple VMs, each running a virtual server. *Id.* ¶11.

A VM is a software abstraction (virtualization) of a physical machine. A virtual machine monitor ("VMM") ran on a physical machine and virtualized its

resources for VMs. *See* Ex. 1004, 7:7-27. The VMM could suspend VMs by saving the VM state to its VM image, which were easily copied, transferred, and resumed, making them well-suited for the automated deployment and scaling of distributed computing systems. Ex. 1003 ¶¶11-13.

Virtualization simplified the management of distributed computer systems because the combination of an application running inside a VM could be saved together as a VM image. It was common to store VM images in a repository, so that the application could be easily deployed by copying the image from the repository to a node and then resumed. *See* Ex. 1004, 17:30-50; Ex. 1003 ¶14.

II. THE '138 PATENT

The provisional application for the '138 patent was filed in 2006.[1] According to the '138 patent, "[o]ne challenge with distributed computing systems is the organization, deployment and administration of such a system within an enterprise environment." Ex. 1001, 1:26-33. The '138 patent purports to solve this challenge with "a distributed computing system that conforms to a multi-level, hierarchical organizational model." *Id.*, 1:37-43. In embodiments, the tiers include

[1] Petitioner takes no position on whether the '138 patent is entitled to claim priority to the provisional but uses 2006 for simplicity because all relied-upon art predates 2006.

"a web server tier … a business application tier … and a database tier …" Ex. 1001, 13:27-56, FIG. 2.

The alleged invention also "includes a software image repository" that stores image instances of virtual machine managers and software applications, and a "control node that comprises an automation infrastructure to provide autonomic deployment of the image instances…" *Id.*, 2:15-30. The automation infrastructure may use a "forward chaining" rule engine. *Id.*, 22:20-62.

The '138 patent recites virtual machines and rules engines but does not purport to invent them. Its embodiments rely on commercially available software and image formats from VMware (Ex. 1001, 32:59-64, 33:25-41) and incorporate prior art papers on rules engines (*id.*, 26:26-61).

The alleged invention was already known in the art, including VMware prior art, which as explained above taught the claimed repository, automation, and virtual machine limitations. *See, e.g.*, *supra* §I; Ex. 1003 ¶¶18-19. The alleged invention is nothing more than a combination of prior art elements according to known methods to yield predictable results. Ex. 1003 ¶19.

III. IDENTIFICATION OF CHALLENGE

A. Statutory Grounds

Petitioner requests *inter partes* review and cancelation of the challenged claims on the following 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

B. Relied-Upon Art[2]

1. U.S. Patent No. 7,577,722 ("Khandekar") (Exhibit 1004)

Khandekar is a VMware patent that teaches a repository, called a provisioning server, for storing pre-configured, ready-to-deploy VM images, with operating systems and applications installed, and then automatically deploying those VM images to host computers. *See, e.g.*, Ex. 1004, 9:26-27 ("The main feature of the invention is a VM provisioning server…"), 4:30-35, 5:25-36, 10:15-26, Abstract. "[A] VMM is typically included for each VM in order to act as its software interface with the underlying host system." *Id.*, 13:17-35. Khandekar teaches a "heuristics/rules engine" for automated deployment based on the monitored status of the hosts. *See id.*, 4:58-65, 10:27-43; Ex. 1003 ¶26.

Khandekar was filed on April 5, 2002 and issued August 18, 2009. Khandekar is prior art under §102(e).

[2] Pre-AIA 35 U.S.C. §102 applies. MPEP §2159.02.

2. U.S. Patent No. 8,209,680 ("Le") (Exhibit 1005)

Le is a VMware patent that teaches techniques for managing distributed networks of physical computers and virtual machines. *See, e.g.*, Ex. 1005, 59:29-36. Le teaches using VMware software and techniques for capturing, configuring and deploying images. *See, e.g., id.*, Abstract; Ex. 1003 ¶28.

Le was filed on June 30, 2003, claims priority to a provisional application filed April 11, 2003, and issued June 12, 2012. Le is prior art under §102(e).

3. U.S. Patent No. 7,962,633 ("Sidebottom") (Exhibit 1006)

Sidebottom teaches a rules engine that applies a forward-chaining rules algorithm to monitored events on a network. *See* Ex. 1006, 9:36-45; Ex. 1003 ¶30.

Sidebottom was filed on October 13, 2005 and issued June 14, 2011. Sidebottom is prior art under §102(e).

4. U.S. Patent Application Publication No. 2003/0036886 ("Stone") (Exhibit 1007)

Stone teaches organizing nodes into tiers and monitoring events based on those tiers, so that automated agents may take action if needed. *See, e.g.*, Ex. 1007 ¶[0040], Abstract, FIG. 2:

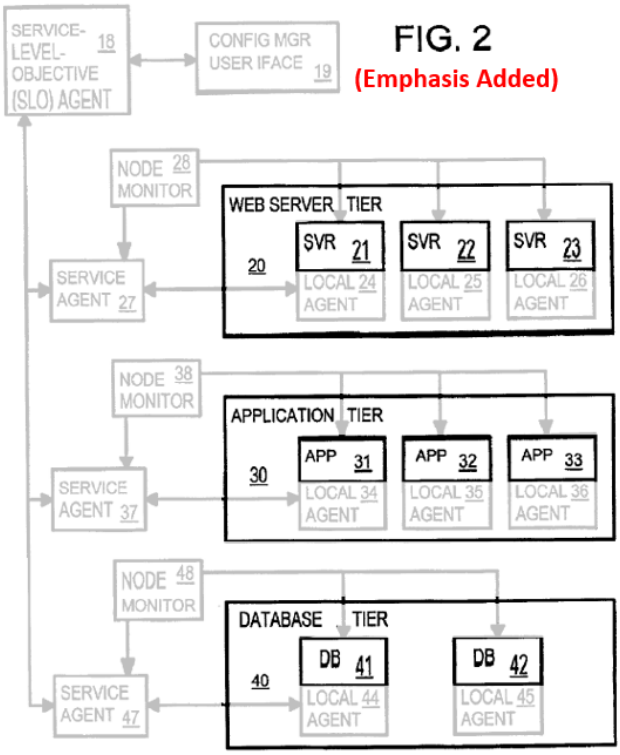


FIG. 2
(Emphasis Added)

Stone was published February 2003 and is prior art under §102(b).

C. Standing

Petitioners certify that Petitioners are not barred or estopped from requesting this inter partes review and that the '138 patent is eligible for IPR.

D. Discretionary Analysis for Review

There is no basis for discretionary denial of institution. None of the relied-upon art was considered during prosecution. Ex. 1003 ¶24. Nor is any reference substantially the same as those evaluated during prosecution. This is the only IPR ever filed against the '138 patent. No scheduling conference has been held, no claim construction briefing scheduled, nor any Markman hearing or trial date set.

Also, N.D. of California courts have been willing to stay cases upon the institution of IPRs against the asserted patents.

IV. LEVEL OF ORDINARY SKILL

A person of ordinary skill in the art ("POSITA") of the '138 patent would have had a bachelor's degree in electrical engineering, computer science, or a similar field with at least two years of experience in distributed computing systems or a person with a master's degree in electrical engineering, computer science, or a similar field with a specialization in distributed computing systems. Ex. 1003 ¶¶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;
- Use rules engines and other knowledge-based systems to automatically deploy images to the nodes of a distributed computing system;

- Implement multi-tier applications and web services on distributed computer systems; and
- Implement systems to monitor the status and state of distributed computer systems.

Ex. 1003 ¶35.

V. CLAIM CONSTRUCTION

A. "a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines" (Claim 1)

A POSITA would have understood this term is satisfied by, but not limited to, a plurality of virtual disk image files. Dependent claim 2 specifies that "the image instances of the plurality of software applications comprise virtual disk image files." A teaching sufficient to satisfy claim 2 must also be sufficient to satisfy claim 1. *Id.* ¶38.

In some embodiments, the virtual disk image files include an OS and application data. *See* Ex. 1001, 33:25-30, 33:55-67, 5:12-22, 5:40-56. To the extent it is argued that this is required, it is taught by the prior art as explained below. Ex. 1003 ¶¶39-40.

VI. CHALLENGED CLAIMS

A. Ground 1 (Khandekar) and Ground 2 (Khandekar in View of Le)

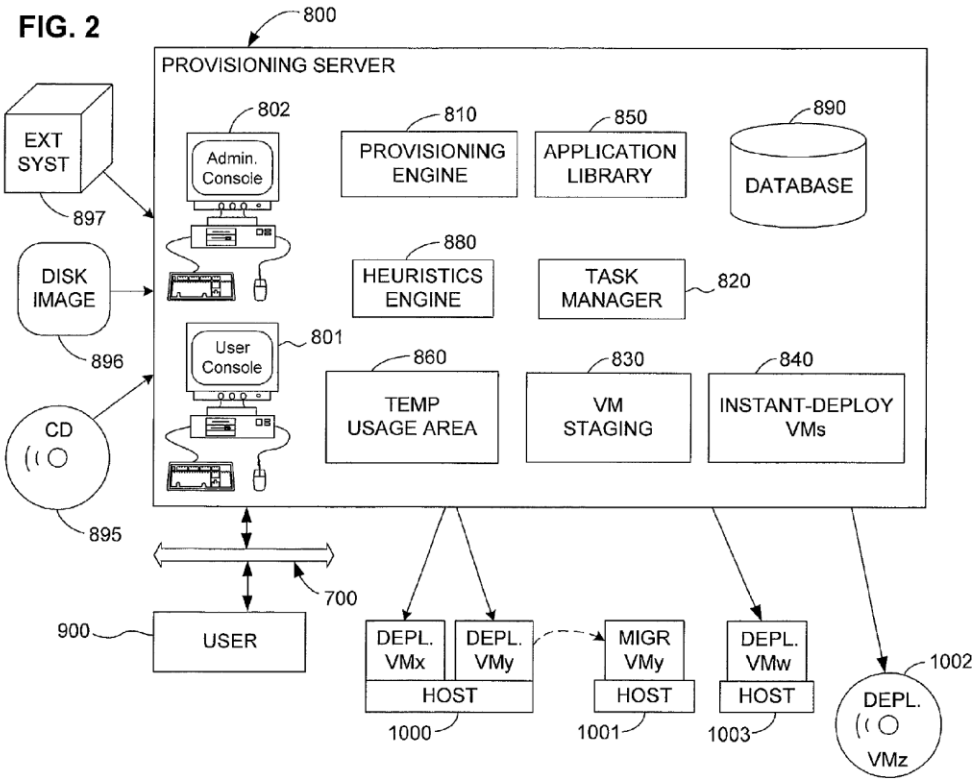
Khandekar renders the challenged claims obvious. Ex. 1003 ¶42. Le reinforces Khandekar's teachings with implementation details regarding VMware

software, the ratio of VMMs to VMs, and disk imaging techniques. Ex. 1003 ¶42. Le's teachings complement Khandekar's, and the combination further renders the challenged claims obvious. *Id.* ¶43.

1. Obviousness over Khandekar

While Khandekar describes aspects of its teachings in terms of "embodiments," a POSITA would have understood, or at least found it obvious, that Khandekar's teachings, as described for Grounds 1 and 2, are meant to be used together because they describe complementary aspects of the invention. Ex. 1003 ¶44. For example, Khandekar's Summary of the Invention describes "pre-built" and "custom-built" VMs as embodiments (*see* Ex. 1004, 4:36-47), but the detailed description teaches both options are implemented in the provisioning server, along with a third option for instantly-deployable VMs, and presented as choices to the user (*see id.*, 9:10-25); Ex. 1003 ¶45.

As another example, Khandekar teaches using its heuristics/rules engine together with VM provisioning because it is "another aspect" of the preferred embodiment. Ex. 1004, 4:58-65. This is illustrated in Figure 2, which teaches using many features of Khandekar's teachings together in the provisioning server, even though those features may be labelled as "embodiments" in other parts of the specification. *See id.*, Fig. 2:



Ex. 1003 ¶¶46-47. A POSITA would have had a reasonable expectation of success when applying Khandekar's teachings because Khandekar's specification and Figure 2 teach their combined use. *Id.* ¶50.

Khandekar teaches using VMMs, and a POSITA would have been motivated to use Khandekar's VMM teachings with the provisioning server because the provisioning server and the hosts include VMs, which require a VMM to run. This would have been obvious from basic virtualization knowledge and Khandekar's teachings. *See supra* §I; Ex. 1004, 7:7-27, Fig. 1; Ex. 1003 ¶48. Khandekar also teaches "provisioning more than one computer at a time" (Ex. 1004, 23:1-20),

which a POSITA would have been motivated to do because Web applications were common for distributed VM systems. Ex. 1003 ¶49.

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.

Khandekar explains that VMMs supported multiple VMs. *See* Ex. 1004, 7:28-38. While Khandekar teaches potential advantages to using one VM per VMM, a POSITA would have also found it obvious to use a single VMM to manage multiple VMs. Ex. 1003 ¶52. Khandekar was filed in 2002. By 2006, it was common to use multiple VMs per VMM, including with VMware ESX software. *See* Ex. 1011 at 1, 5, FIG. 1. This is confirmed by VMware patent Le, which teaches a single VMM hosting multiple VMs. *See* Ex. 1005, 65:43-66:10, FIG. 8; Ex. 1003 ¶53.

A POSITA in 2006 would have been motivated to apply Le's teachings to Khandekar, such that each VMM hosts multiple VMs, because it would be easier to scale a large number of VMs on a single physical server, reducing overhead, simplifying resource management, and facilitating consolidation. *See* Ex. 1005, 46:58-47:14, 18:6-19:23; Ex. 1008, FIG. 1; Ex. 1003 ¶¶53-54. Furthermore, Khandekar and Le were VMware patents with overlapping inventors—Khandekar was a named inventor for the Le patent, and vice versa. *See* Ex. 1005; Ex. 1004. A POSITA would have understood that Le reflects an updated view of the state of the art for hosting VMs on a VMM, including updated functionality in VMware's commercial ESX software. *See* Ex. 1005, 65:43-50, 65:65-66:10, FIG. 8; Ex. 1004, 7:28-38; Ex. 1003 ¶¶55-56.

Additionally, Le teaches implementation details of image creation and management for distributed computing and virtual machine systems. A POSITA would have naturally looked to other patents by the same company and inventors to implement and improve upon Khandekar's teachings. Ex. 1003 ¶57. Le's imaging teachings address two functions raised by Khandekar: they allow a VMM to be provided from the provisioning server to hosts, and they allow the provisioning server to know the characteristics of the host, so that an appropriate VMM image can be provided. *Id.* ¶59.

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. Le teaches techniques for 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). *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. *See* Ex. 1005, Figs. 3-5, 37:36-49; Ex. 1003 ¶58. A POSITA would have been motivated to apply prior art imaging solutions taught by Le, as well as the teachings of Le's embodiments.[3] Ex. 1003 ¶61.

[3] Le provides general teachings regarding VMs and imaging, and it also provides "exemplifying embodiments of the invention" including a "Universal Computer Management System (UCMS)." *See* Ex. 1005, 59:9-27. Since the UCMS embodies the principles of Le's invention, it elaborates on Le's other teachings and a POSITA would have been motivated to combine Le's UCMS

Furthermore, Khandekar provides express motivation to apply a solution for determining "the characteristics of the host on which the VM is to be installed" (Ex. 1004, 13:17-35), and Le teaches such a solution. Le's imaging client includes functionality that allows a central repository to determine the hardware characteristics of the remote computer. *See* Ex. 1005, 70:13-35 ("… The registration process causes the imaging client 1021 running from the secondary stack 4100 to ***analyze the computer's hardware configuration and transmit it over the network*** 3000 to the UCMS server 2101….").[4] A POSITA would have been motivated to combine Khandekar and Le because of the synergies described above. Ex. 1003 ¶59.

Le also teaches features of commercially available VMM software such as VMware ESX and GSX. *See* Ex. 1005, 65:65-66:10. A POSITA would have been motivated to use VMware software, including ESX server and known features such as Le's teaching of virtual machine monitors and managers, to implement Khandekar's teachings. *See* Ex. 1004, 13:17-35; Ex. 1005, 65:58-64, 66:11-19,

---

teachings with related teachings elsewhere in Le's specification. Ex. 1003 ¶63. A POSITA would have been motivated to apply Le's UCMS teachings related to virtual machines and imaging to Khandekar's provisioning server, which plays an analogous role as a central repository, with similar functionality. *See* Ex. 1005, 65:20-33; Ex. 1003 ¶63.

[4] All emphasis added unless stated otherwise.

13:22-30; Ex. 1003 ¶62. 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 and store images. *See* Ex. 1005, 60:64-61:8, 18:6-9, 44:58-48:55; Ex. 1003 ¶62.

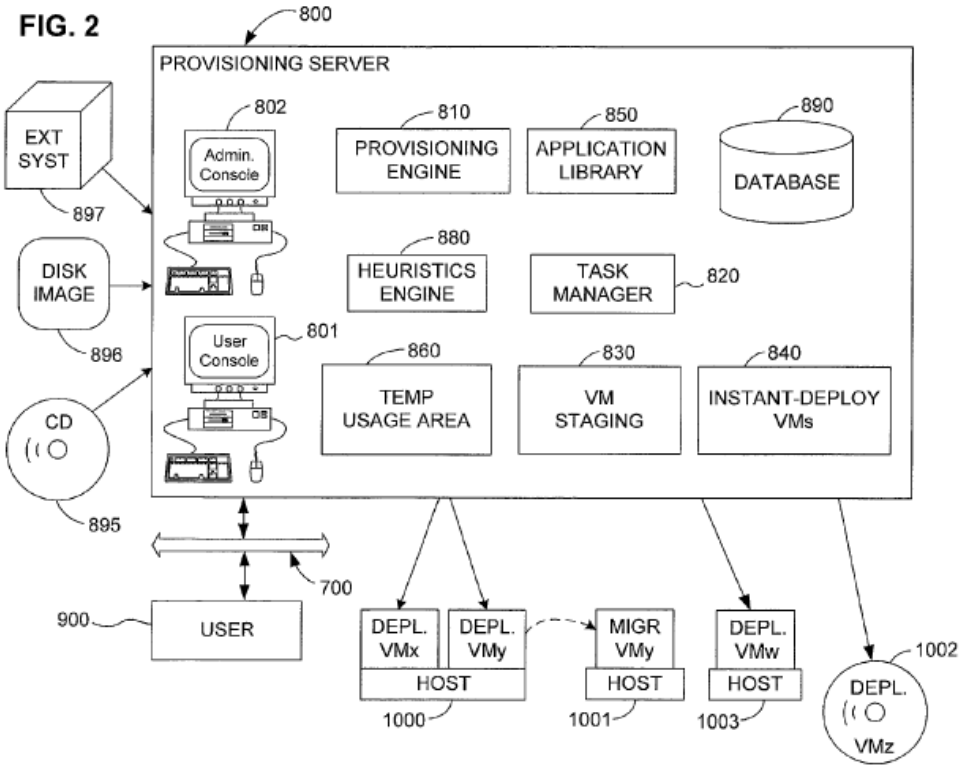
The prior art provides additional teaching, suggestion, and motivation to combine Khandekar with Le, using VMM images to provide a VMM to host computers. Several prior art references teach loading a VMM (hypervisor) using a VMM image. *See* Ex. 1014 ¶[0062] ("The hypervisor also may be loaded first or part of an OS-virtual machine manager image. It may be essentially part of a firmware image."); Ex. 1016 ¶¶[0049], [0043] ("The subsequent steps in the installation procedure of a fixed system remain essentially the same … The bootable core and the virtual machine manager image must be installed in a secure partition."); Ex. 1003 ¶60.

A POSITA would have had a reasonable expectation of success because Khandekar and Le contain compatible and complementary teachings. Both references are VMware patents directed to complementary aspects of virtual machine systems. Khandekar teaches using a VMM, and Le teaches VMware software for that purpose. Furthermore, Le's disk imaging teachings were well-known by 2006, and it would have been well within the level of ordinary skill to apply them. *See supra* §IV; Ex. 1003 ¶64.

3. Claim 1

1[a]. A distributed computing system comprising:

Khandekar teaches or suggests a distributed computing system with "a network of cooperating VMs … deployed on respective physical host platforms." Ex. 1004, 5:9-13, 23:1-20. "In the preferred embodiment of the invention … hosting is preferably provided using a bank or 'farm' of servers, each forming one of the hosts." *Id.*, 12:40-50, 4:58-5:8, 4:30-35, 23:26-51, 23:54-24:27, Abstract, FIG. 7, FIG. 2:



Ex. 1003 ¶¶67-68. Therefore, Khandekar renders [1a] obvious. Ex. 1003 ¶¶66-69.

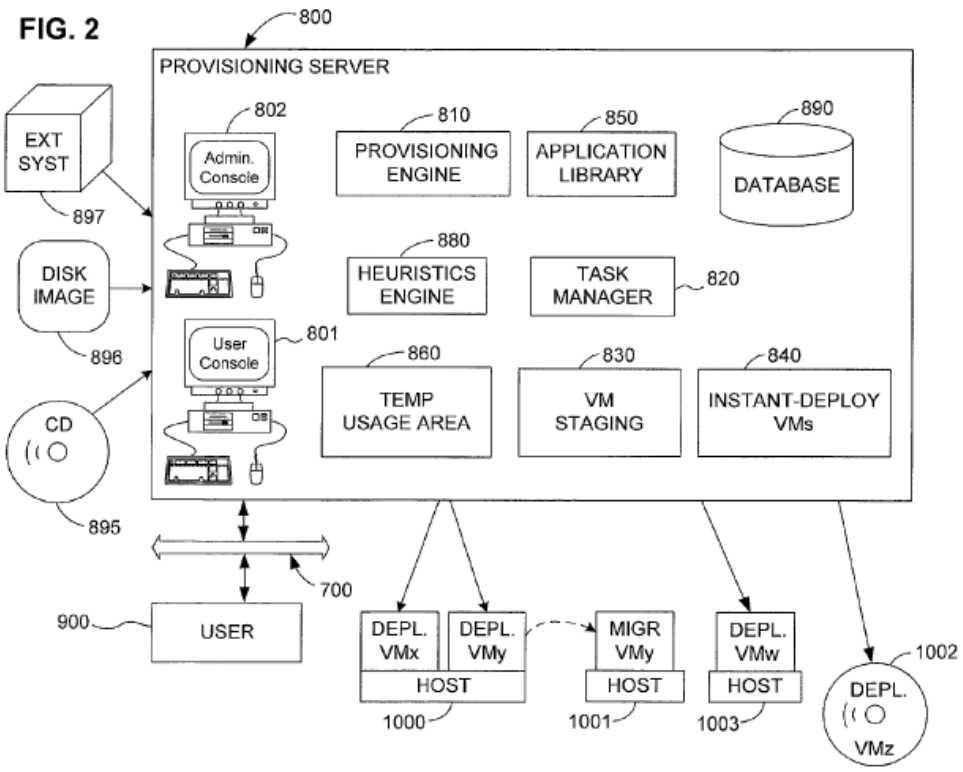
Ground 2: The Combination with Le Further Renders [1a] Obvious.

Le's teachings complement and elaborate on Khandekar's, further rendering this limitation obvious. *See* §VI.A.2; Ex. 1003 ¶¶70-73. Le teaches a distributed computing system including a UCMS that plays an analogous role to Khandekar's provisioning server.[5] *See* Ex. 1005, 59:29-36 ("The UCMS manages a set of physical computers and virtual machines residing on a network…"), 53:45-57, 60:19-28.

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

Khandekar teaches or suggests this limitation. Ex. 1003 ¶¶74-77. For example, Khandekar teaches a "provisioning server" that acts as a software image repository to store various images, e.g., as discussed below for [1c]-[1e]. *See, e.g.*, Ex. 1004, 4:36-41 ("… a plurality of pre-configured VMs having different configurations are pre-stored."), 11:36-49 ("[A]pplications may be stored in respective libraries …"), 23:28-51, 23:54-24:27, FIG. 2:

[5] *See supra* n. 3.



Ex. 1003 ¶74.

The provisioning server includes several libraries/repositories for software images, including 830 (VM staging repository), 840 (instantly-deployable VM repository), and 850 (applications library). *See* Ex. 1004, 11:21-25, 11:50-58, 16:43-63, 18:14-23, 18:56-19:2. A POSITA would have found it obvious to store VMM images in a single repository in the provisioning server because Khandekar teaches storing VMs there, and a corresponding VMM is included and installed together for each VM. *See infra* §VIII.A.3[1c]; Ex. 1004, 8:36-44, 13:17-35; Ex. 1003 ¶75. The provisioning server acts as a "central repository" with images of virtual machines and applications, including a database of related information. *See* Ex. 1004, 18:1-42; Ex. 1003 ¶76.

Khandekar teaches using standard computer components, which obviously would have included hard disks. *See* Ex. 1004, 9:26-32. A POSITA would have found it obvious to store image instances in non-transitory, computer-readable media such as hard drives. Ex. 1003 ¶77.

Ground 2: The Combination with Le Further Renders [1b] Obvious.

Le's teachings complement and elaborate on Khandekar's, further rendering this limitation obvious. *See* §VI.A.2; Ex. 1003 ¶¶78-80. For example, Le teaches a software image repository (e.g., server computer 2000 that includes server disk 2010 and template images 4020) comprising non-transitory, computer-readable media operable to store the image instances (e.g., image file 2015, template images 4020) discussed for limitation [1c] and [1e] below. *See* Ex. 1005, 25:34-41, 26:6-22, 26:63-27:9, 66:66-67:29, FIGS. 4-6:

Fig. 4

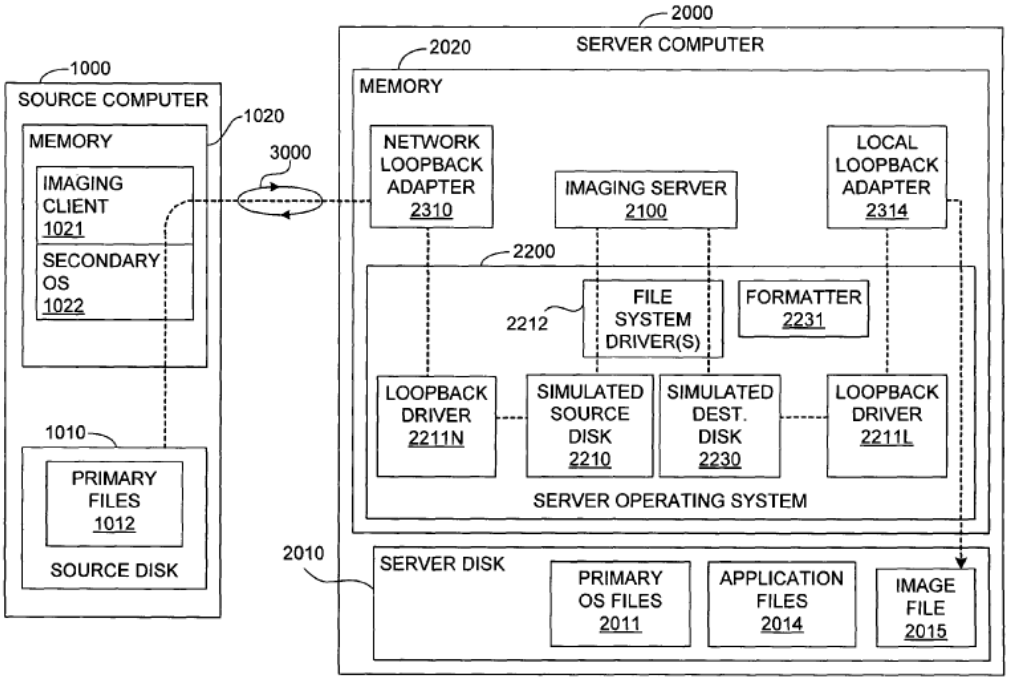

2000
SERVER COMPUTER
2020
MEMORY
1000
SOURCE COMPUTER
1020
3000
MEMORY
IMAGING CLIENT 1021
SECONDARY OS 1022
NETWORK LOOPBACK ADAPTER 2310
IMAGING SERVER 2100
LOCAL LOOPBACK ADAPTER 2314
2200
2212
FILE SYSTEM DRIVER(S)
FORMATTER 2231
LOOPBACK DRIVER 2211N
SIMULATED SOURCE DISK 2210
SIMULATED DEST. DISK 2230
LOOPBACK DRIVER 2211L
SERVER OPERATING SYSTEM
1010
PRIMARY FILES 1012
SOURCE DISK
2010
SERVER DISK
PRIMARY OS FILES 2011
APPLICATION FILES 2014
IMAGE FILE 2015

Fig. 5

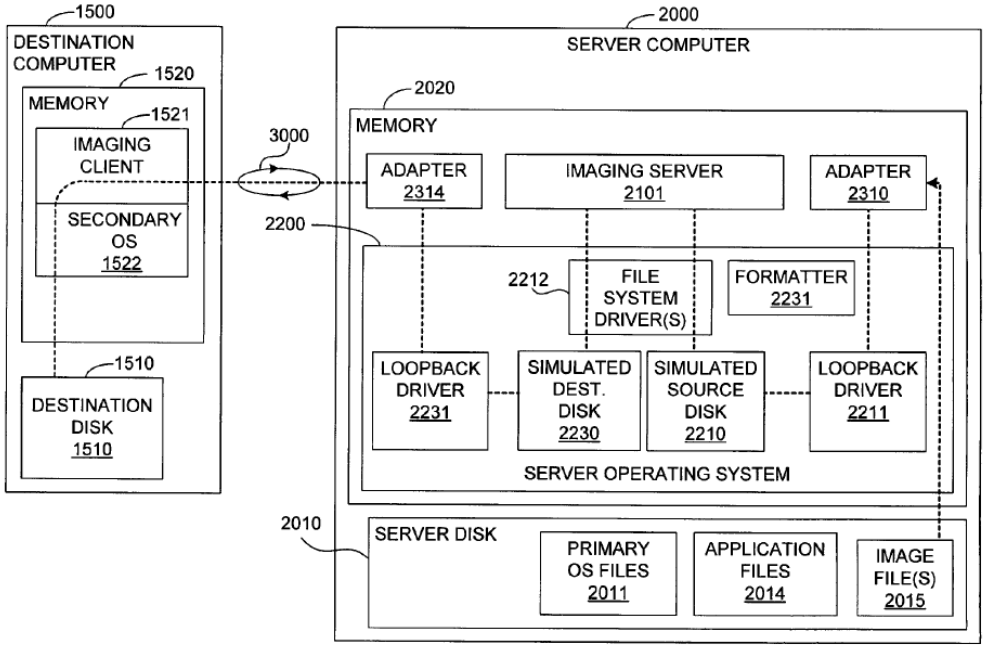
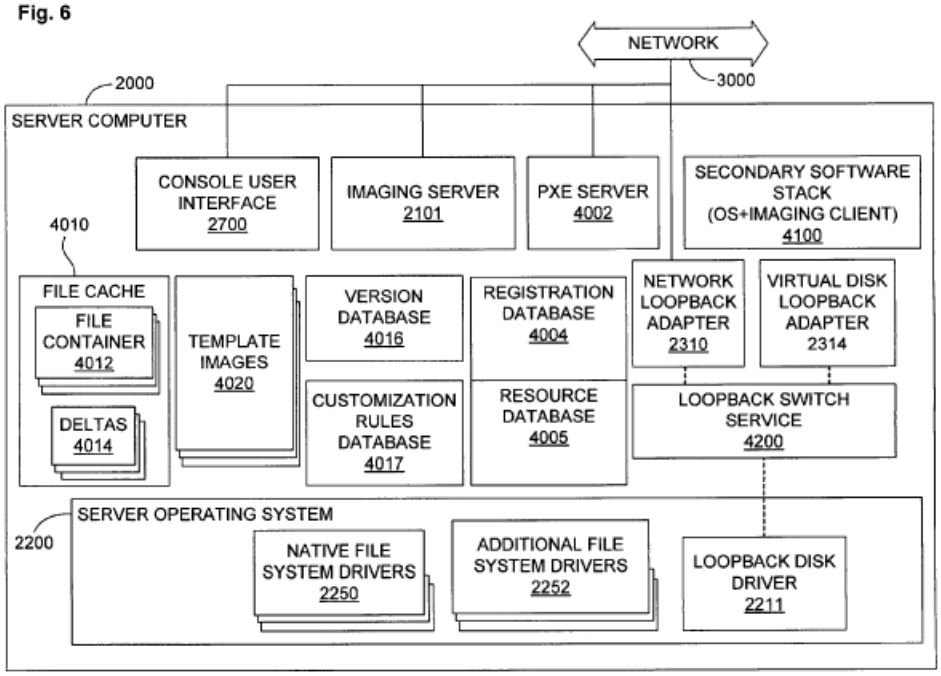

1500
DESTINATION COMPUTER
1520
MEMORY
1521
IMAGING CLIENT
SECONDARY OS 1522
3000
2000
SERVER COMPUTER
2020
MEMORY
ADAPTER 2314
IMAGING SERVER 2101
ADAPTER 2310
2200
2212
FILE SYSTEM DRIVER(S)
FORMATTER 2231
LOOPBACK DRIVER 2231
SIMULATED DEST. DISK 2230
SIMULATED SOURCE DISK 2210
LOOPBACK DRIVER 2211
SERVER OPERATING SYSTEM
1510
DESTINATION DISK 1510
2010
SERVER DISK
PRIMARY OS FILES 2011
APPLICATION FILES 2014
IMAGE FILE(S) 2015

Fig. 6



Le teaches the use of hard disks. *See* Ex. 1005, Figs. 4-6, 26:38-51, 63:20-34, 60:64-61:8; Ex. 1003 ¶79.

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

Khandekar teaches or suggests this limitation. Ex. 1003 ¶¶81-94. First, Khandekar teaches a virtual machine manager (e.g., "VMM") that is executable on a plurality of application nodes (e.g., hosts). *See* Ex. 1004, 4:66-5:8 ("… a virtual machine-to-hardware interface, such as a virtual machine monitor (VMM), is installed on each of a plurality of hardware hosts."). "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." *Id.*, 7:7-27, 13:17-35; Ex. 1003 ¶81.

The provisioning server stores a plurality of image instances of VMs. *See infra* §VIII.A.3[1e]. Khandekar teaches including a corresponding VMM with each VM. *See* Ex. 1004, 13:17-35:

> 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 … 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.***

When the provisioning server provides a VM to a host, it may also include a VMM together with the VM. *See id.*, 8:36-44 ("… 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."); Ex. 1003 ¶82.

Since VM images are stored in the provisioning server, a POSITA would have found it obvious to likewise store a plurality of VMM images in the provisioning server so they may be deployed together to hosts, as taught by Khandekar. *See* Ex. 1004, 8:36-44, 13:17-35. Doing so would have allowed the hosts to be remotely initialized with VMM software and configured to support

virtual machines, facilitating Khandekar's goal of automated, flexible deployment of VMs to hosts. *See id.*, 4:30-35; Ex. 1003 ¶83. This removes the need to manually install VMMs on hosts, which is advantageous with a large number of hosts. *See id.*

Furthermore, it would have been obvious to store a plurality of VMM images to account for the different characteristics of the hosts, which can have different software and hardware configurations. *See* Ex. 1004, 8:20-36, 6:13-22. Khandekar teaches that a VMM may be provided to a host "as long as ***the characteristics of the host*** on which the VM is to be installed are known." *Id.*, 13:17-35. Thus, a POSITA would have found it obvious to store a plurality of VMM images to accommodate different software and hardware configurations; this was common in the art. Moreover, it would have been obvious to include at least two VMM images: an image for Type 1 hypervisors, and another for Type 2 hypervisors, because Khandekar teaches both options for the VMM. *See id.*, 7:7-27. Since Khandekar teaches that the provisioning server acts as a repository for storing VM images (*see id.*, 4:36-41, 11:21-25, 16:43-63), a POSITA would have found it obvious for the provisioning server to also store VMM images so that it can provide the VMMs together with VMs to hosts, as taught by Khandekar. *See id.*, 8:36-44, 13:17-35; Ex. 1003 ¶48. Khandekar teaches that the provisioning server includes a database and several libraries; therefore, Khandekar teaches or

suggests that the provisioning server includes storage infrastructure, and a POSITA would have found it obvious and been motivated to store VMM images in the provisioning server for deployment. Ex. 1003 ¶84.

Application Nodes. Khandekar teaches application nodes in the form of "hosts." *See* Ex. 1004, 5:25-36, 5:52-58, 12:40-50, 10:27-43, FIG. 2. A POSITA would have found it obvious that Khandekar's hosts satisfy the claimed application nodes because the hosts are computing devices that form the computing nodes of the distributed computing system. Ex. 1003 ¶87.

This is confirmed by the '138 patent, which explains that "application nodes" are "computing nodes." *See* Ex. 1001, 3:59-64, 4:26-24, FIG. 1:

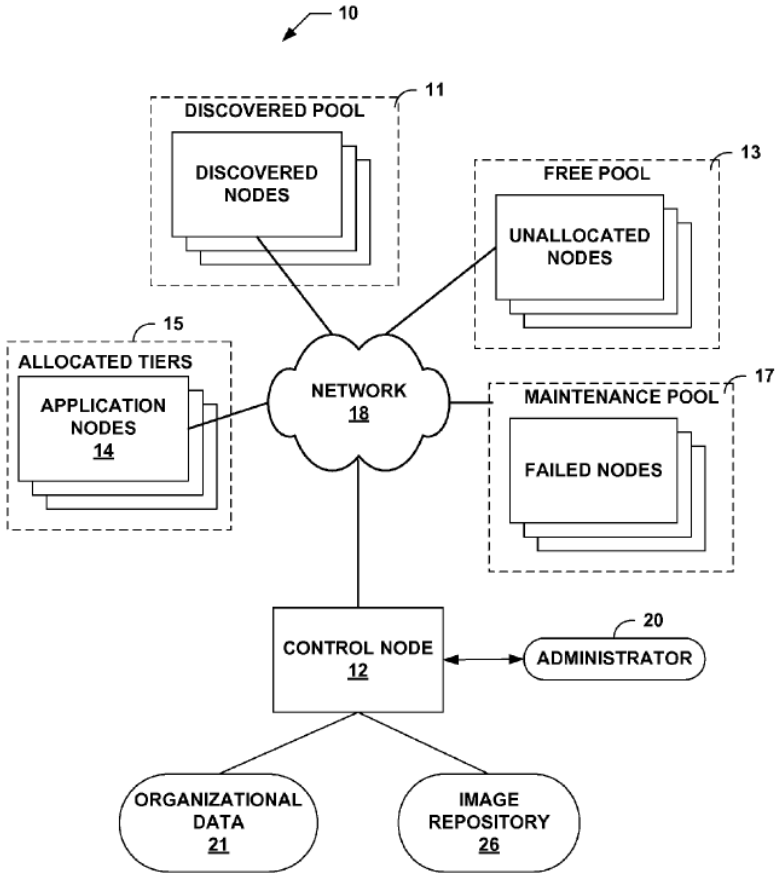


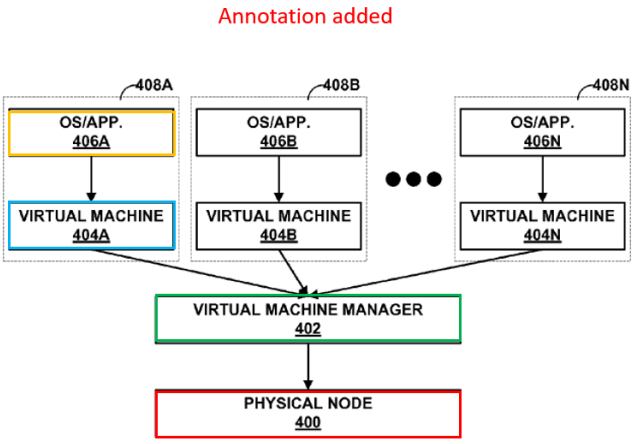
FIG. 1

"Within distributed computing system 10, a computing node refers to ***the physical computing device***." Ex. 1001, 3:65-4:5, 1:16-25; Ex. 1003 ¶86.

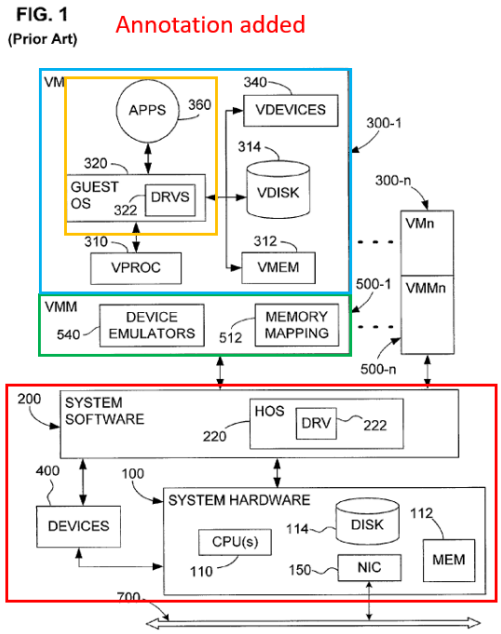
Virtual Machine Manager. Khandekar's "virtual machine monitor" satisfies the claimed "virtual machine manager" because the references use their respective terms to refer to the same functionality. Ex. 1003 ¶81. In the prior art, the term "virtual machine manager" was often used to refer to a virtual machine monitor. *See, e.g.*, Ex. 1014 ¶[0029] ("further note that as used herein, virtual

machine monitor and virtual machine manager, as well as VMM, can be considered equivalent"); Ex. 1003 ¶88.

Khandekar and the '138 patent use these terms to refer to the same functionality, e.g., the software that manages a VM. *Compare* Ex. 1001, FIG. 24 with Ex. 1004, FIG. 1.



Ex. 1001, FIG. 24



Ex. 1004, FIG. 1

Ex. 1003 ¶89.

The '138 patent explains that a virtual machine manager is a software program that is capable of managing one or more virtual machines, for example "an instance of VMWare ESX software." *See* Ex. 1001, 32:59-33:24; Ex. 1003 ¶90.

Khandekar, a VMware patent, teaches that a virtual machine monitor manages the execution of virtual machines. *See, e.g.*, Ex. 1004, 6:62-7:27; Ex.

1003 ¶91. A POSITA would have found it obvious to use known VMM software, including VMware ESX software. Ex. 1003 ¶¶92-93.

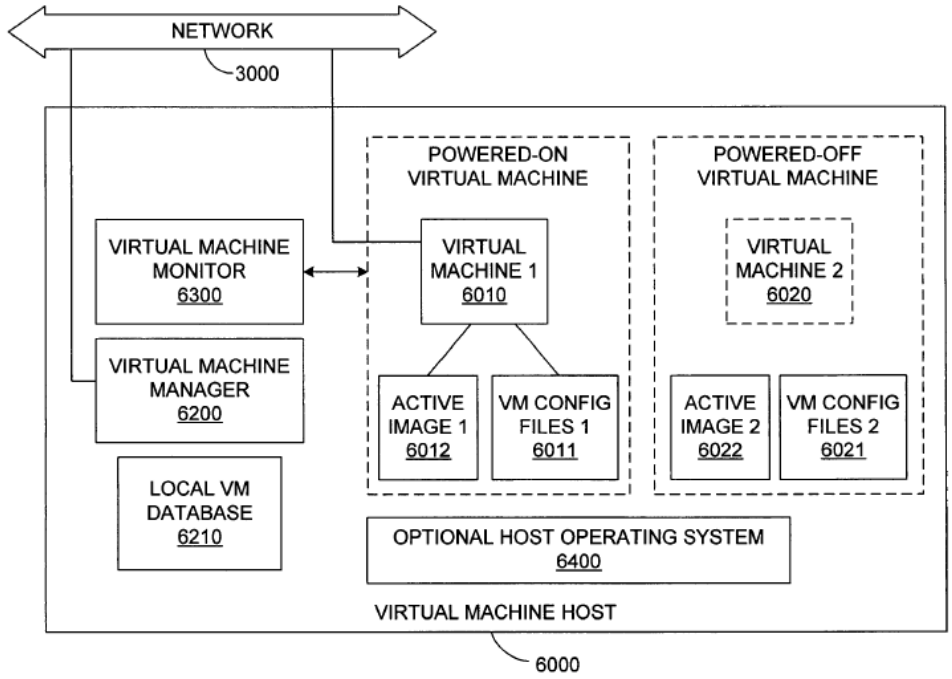
Ground 2: The Combination with Le Provides an Alternative Basis for the "Virtual Machine Manager"

The phrase "virtual machine manager" did not have a universal definition in the art. Ex. 1003 ¶95. The '138 patent uses the term to refer to a virtual machine monitor, which is taught by Khandekar. *See* Ex. 1001, 32:59-67 ("… a virtual machine manager is a software program that is capable of ***managing one or more virtual machines***. …"). Le confirms this, teaching that the virtual machine monitor controls and manages VMs by managing their interactions with the host's physical resources. *See* Ex. 1005, 65:65-66:2 ("When a virtual machine is powered on, a virtual machine monitor program 6300 controls it and ***manages the interactions*** between the software … running inside of the virtual machine and the host's physical resources, such as hardware devices."); Ex. 1003 ¶97.

To the extent it is argued otherwise, Le expressly teaches a virtual machine "manager" (*see* §VI.A.2; Ex. 1003 ¶¶95-98), using the phrase to refer to other aspects of the system, including creating, destroying, and maintaining VM files on the host. *See* Ex. 1005, 66:11-32 ("The virtual machine manager 6200 is typically also responsible for creating, destroying and maintaining virtual machine files residing on the host 6000. …"), 67:20-32, 66:48-64, 67:9-19; Ex. 1003 ¶96.

Moreover, Le is a VMware patent that teaches the use of VMware software including "VMware ESX Server" (*see* Ex. 1005, 65:65-66:10, 13:22-30)—the same software the '138 patent relies on for a virtual machine manager. *See* Ex. 1001, 32:62-64. Therefore, Le teaches the claimed "virtual machine manager," and the combination of Khandekar and Le teaches a virtual machine manager that is executable on a plurality of application nodes (e.g., virtual machine host 6000). *See* §VI.A.2; Ex. 1005, FIG. 8:

Fig. 8



Ex. 1003 ¶98.

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

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; Ex. 1003 ¶¶99-111. As explained above, Khandekar teaches the provisioning server providing VMMs to hosts. Le teaches 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. Ex. 1003 ¶99.

Le teaches "us[ing] disk imaging to quickly provision a bare-metal computer on the network ***with a complete software stack consisting of an operating system and a set of core applications***." Ex. 1005, 54:14-21, 15:63-65, 16:29-30, 8:10-23, 22:32-44, 67:31-40. "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." *Id.*, 10:55-62; Ex. 1003 ¶¶100-101.

VMMs acted as an OS (Type 1, such as ESX) or ran as a software application on top of a separate OS (Type 2, such as GSX). *See* Ex. 1004, 7:7-14; Ex. 1005, 13:5-21, 66:2-10. Therefore, a POSITA would have found it obvious 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* Ex. 1005, 15:63-65, 16:29-30. It would have been obvious to use images of VMware ESX and GSX software to initialize hosts. Ex. 1003 ¶102.

The use of VMM images in this manner was known in the art. *See, e.g.*, Ex. 1014 ¶[0062] ("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 ¶¶[0049], [0043]. Thus, the prior art includes teachings, suggestions, and motivations to combine Khandekar and Le as described above, using a plurality of VMM images to practice Khandekar's teachings. Ex. 1003 ¶103.

Le further teaches storing a plurality of such images (e.g., image file 2015, template images 4020) in a server computer (e.g., server computer 2000 that includes server disk 2010 and template images 4020), which acts as a central repository analogous to the provisioning server in Khandekar. *See* Ex. 1005, 26:53-62 ("… a stored disk image 2015 (more than one may of course be stored)"), 26:6-22, 26:53-57, 54:14-21 ("A single template image, or ***a small set of core template images***, is used to create clones."), 66:66-67:8, 67:25-20, FIGS. 4-5, FIG. 6 ("TEMPLATE IMAGES 4020"), FIG. 9, 67:31-40 ("UCMS users and, in particular, a UCMS administrator, decide how many images 4020 to maintain on the UCMS server 2000…"); Ex. 1003 ¶104. Le teaches a "Universal Computer

Management System (UCMS)," (Ex. 1005, 59:21-27), which is analogous to Khandekar's provisioning server and manages and deploys images to virtual machines and physical computers. *See* Ex. 1005, 65:20-33; *supra* §VI.A.2; Ex. 1003 ¶105.

A POSITA would have been motivated to follow Le's teachings because storing a plurality of images would account for different computer hardware and software platforms for the host computers, as explained above. Ex. 1003 ¶105. Khandekar teaches consideration of host characteristics when providing a VMM. *See* Ex. 1004, 13:17-35, 8:20-36, 6:13-22. Le reinforces the obviousness of this approach, explaining that it was common to "to create one image file per family of similar computers." *See* Ex. 1005, 9:66-10:9, 16:29-41. Le teaches this was commonly used despite being potentially inelegant, which was often the case for real-world engineering solutions. *See id.*; Ex. 1003 ¶106.

Additionally, Le teaches using a Type 1 VMM (VMware ESX) and Type 2 VMM (VMware GSX). *See* Ex. 1005, 66:2-10. Thus, it would have been obvious for the provisioning server to store at least two VMM images, for GSX and ESX software. Ex. 1003 ¶107.

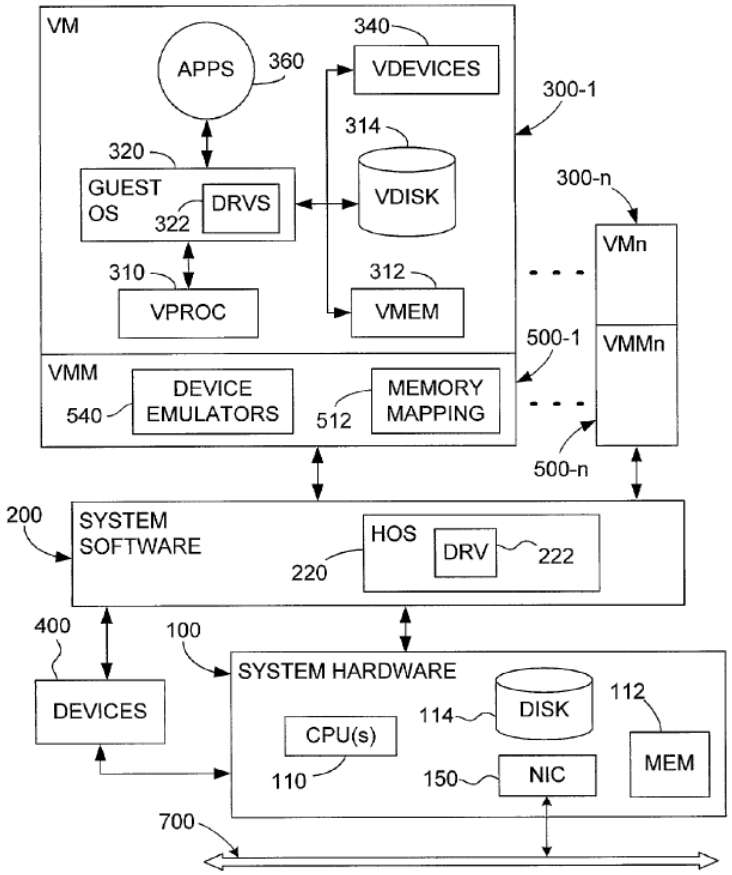
While Le discusses various challenges and solutions for the details of disk imaging, those details are beyond the scope of the challenged claims and would not detract from the obviousness or motivation of combining Khandekar and Le. *See*

§VIII.A.2. A POSITA would also have found it obvious to use prior art solutions to fit the needs of particular projects. Ex. 1003 ¶¶108-110.

[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**

Khandekar renders this limitation obvious. Ex. 1003 ¶¶112-116. First, Khandekar teaches or suggests that, when executed on the applications nodes (e.g., hosts), the image instances of the virtual machine manager (e.g., VMM, for example labelled 500-1 "VMM" through 500-n "VMMn" in Fig. 1) are executed on the host computer and provide a plurality of virtual machines (e.g., labelled 300-1 "VM" through 300-n "VMn"). *See* Ex. 1004, Fig. 1:

FIG. 1
(Prior Art)



Ex. 1003 ¶112.

"FIG. 1 shows the main components of a computer system that **includes one or more virtual machines (VMs)**. The illustrated system includes an underlying system hardware platform 100, system software 200, and **a plurality of virtual machines (VMs) 300-1, …, 300-n** that run on the system software 200; the hardware platform 100 and the system software 200 thus form a 'host' for the VMs." Ex. 1004, 5:51-58, 13:17-35, 7:7-27; Ex. 1003 ¶113.

To the extent PO contends [1d] requires that *each* image instance of a VMM provides a plurality of virtual machines, Khandekar teaches this. Khandekar teaches that commercial VMware products allow computers "to load and run multiple VMs." Ex. 1004, 13:17-35. Khandekar teaches "[i]t would also be possible to use a single VMM to act as the interface to all VMs…" *Id.*, 7:28-38. Khandekar notes potential difficulty switching between VM contexts if different guest operating systems are used. *See id.* Nonetheless, by 2006 using multiple VMs for each VMM was common because of its known advantages and support in commercial software such as VMware software. Ex. 1003 ¶¶52-54. A POSITA would have been motivated to do so. *Id.* ¶114.

Khandekar further teaches or suggests each of the plurality of virtual machines (VMs) is operable to provide an environment that emulates a computer platform. *See* Ex. 1004, 7:39-47 ("… the VM is structured and performs as a complete 'real' computer system, …"), 6:23-43:

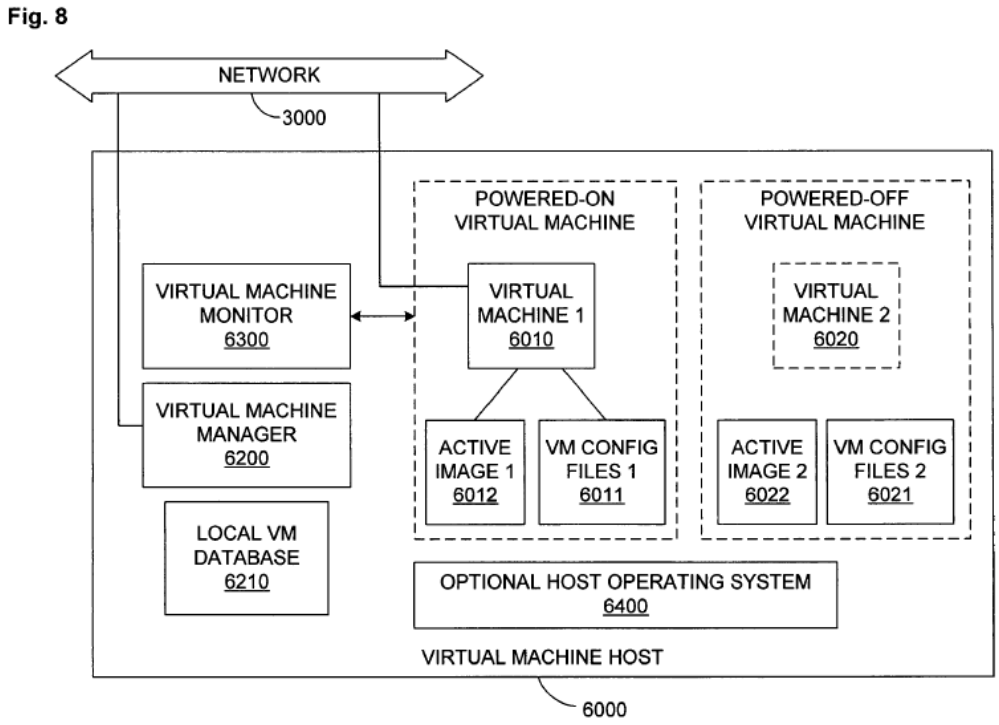
> As is well known in computer science, a virtual machine (VM), which is sometimes also referred to as a "virtual computer," is a software abstraction—a "virtualization"—of an actual physical computer system. … each VM will typically include one or more virtual CPUs 310 (VPROC), … virtual system memory 312 (VMEM), a virtual disk 314 (VDISK), optional virtual peripheral devices 340 (VDEVICES) and drivers 322 (DRVS) for handling the virtual devices 340, ***all of***

which are implemented in software to emulate the corresponding components of an actual computer. …

Id., 6:44-61; Ex. 1003 ¶115.

Ground 2: The Combination with Le Further Renders [1d] Obvious.

To the extent PO contends that *each* image instance of a VMM must provide a plurality of virtual machines, the combination of Khandekar and Le further renders this obvious. *See* §VI.A.2; Ex. 1003 ¶¶117-121. For example, Le teaches a single virtual machine monitor and a single virtual machine manager that support a plurality of virtual machines on a host. *See* Ex. 1005, FIG. 8:



"[T]wo virtual machines 6010 and 6020 are shown" but "[a]ny number of virtual machines may be loaded onto the host …" Ex. 1005, 65:43-50; Ex. 1003 ¶118.

This was a feature of commercially available software from VMware and others. *See* Ex. 1005, 65:65-66:10; Ex. 1011 at 1, 5, FIG. 1; Ex. 1016 ¶[0076]. Khandekar and Le teach a conventional usage of VMM/VM software, and a POSITA would have been motivated to combine those teachings. Ex. 1003 ¶119.

Le further teaches that each VM provides an environment that emulates a computer platform. *See* Ex. 1005, 65:65-66:10, 66:11-32; *supra* §VIII.A.3[1c]; Ex. 1003 ¶119.

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

Khandekar teaches or suggests this limitation. *See e.g.*, Ex. 1004, 18:56-19:2, 10:15-26, 14:4-18; Ex. 1003 ¶¶122-139. Limitation [1e] may be satisfied by a plurality of virtual disk image files (*see* §V.A), which Khandekar teaches three ways: with pre-built, custom-built, and instantly-deployable VMs. Khandekar presents the user with three choices for storing and deploying image instances. *See* Ex. 1004, 9:10-25:

> … The invention provides the user with three choices:
> **1)** Clone an existing, ***pre-built model VM*** and configure a few parameters, such as host name, IP address, etc.

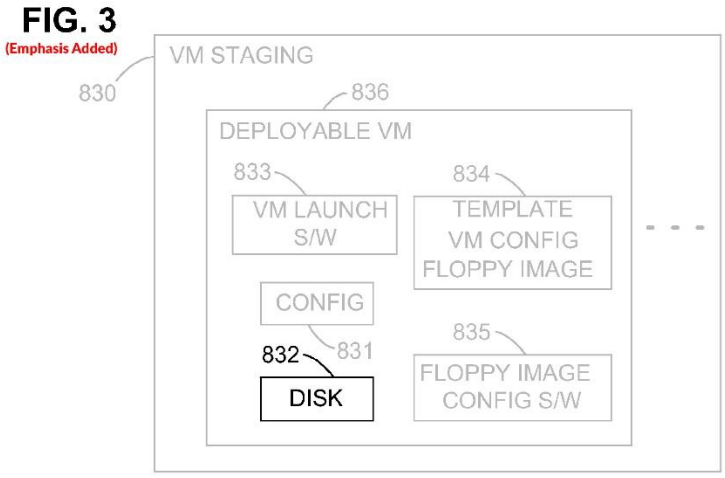
> **2)** Install an operating system and/or one or more applications to a VM created from scratch and then configure them. Such a VM is referred to below as a ***"custom-built" VM***.
>
> **3)** ***Instantly provision*** a VM by resuming it from a suspended state. Such a VM is referred to below as a an "instant-deploy" VM.

See §VI.A.1, §VI.A.3[1g]; Ex. 1003 ¶125.

Each teaching independently renders limitation [1e] obvious by teaching a plurality of virtual disk image files. For pre-built and instantly-deployable VMs, Khandekar teaches that each virtual disk image file includes an operating system and application data. Ex. 1003 ¶123. To the extent PO contends the software applications are somehow distinct from the virtual disk image file, Khandekar teaches custom-built VMs where application images (installation files) are stored in an application library and copied to hosts. *Id.* ¶124.

(1) Pre-Built, Staged VMs. Khandekar teaches that its provisioning server includes a plurality of images of pre-built VMs that are stored in a repository (e.g., library 830). *See* Ex. 1004, 10:27-29 ("a library 830 of staged VMs"), 16:43-63, 18:16-23 ("Staging VMs: This data structure is required for provisioning pre-built or custom-built VMs and contains … [l]ocation of the VM in the VM staging repository 830."), 22:17-22, 23:54-24:27, FIGS. 2-3; Ex. 1003 ¶126.

Pre-built VMs include a virtual disk image file, for example as indicated by the 832 "disk" in Figure 3. *See* Ex. 1004, FIG. 3:



See also Ex. 1004, 15:62-16:12, 16:21-29.; Ex. 1003 ¶127. Pre-built VMs also include operating systems and application data installed. *See, e.g.*, Ex. 1004, 15:40-16:12 ("… pre-built VMs will be stored with their ***operating systems and applications installed*** … In the first step, a model VM is created and ***the necessary OS and applications are installed*** in it."), 5:25-36, 14:19-25, 18:16-23, 10:15-26, 8:50-55, FIG. 3; Ex. 1003 ¶¶126, 129.

A "VM Staging" process is used to create new VM images, including for pre-built, custom-built, and instantly-deployable VMs, which are then shut down and stored in a library and database for future deployment. *See* Ex. 1004, 15:29-40, 16:43-52; Ex. 1003 ¶128. The staged VMs each include virtual disk image files along with configuration files and supporting configuration and launch components. *See* Ex. 1004, 16:47-63 ("For each VM, the data sets kept in the VM

staging server 830 are illustrated in FIG. 3 and are as follows: … ***2) VM disk files 832***. …"); Ex. 1003 ¶¶127, 129.

Khandekar teaches that "any number of deployable VMs may be staged." Ex. 1004, 15:46-50. A POSITA would have been motivated to follow this teaching by staging a plurality of VM images in the "VM staging repository 830" because doing so would have increased the speed and efficiency in which VMs can be deployed—particularly for frequently deployed VMs. Ex. 1003 ¶130. This is a preferred embodiment of Khandekar. *See* Ex. 1004, 4:36-47 ("In a preferred embodiment of the invention, ***a plurality of pre-configured VMs having different configurations are pre-stored***…"), FIG. 7. Furthermore, it would have been obvious to keep staged VMs for a period of time after they have been deployed, so they are readily accessible if requested again. Ex. 1003 ¶130.

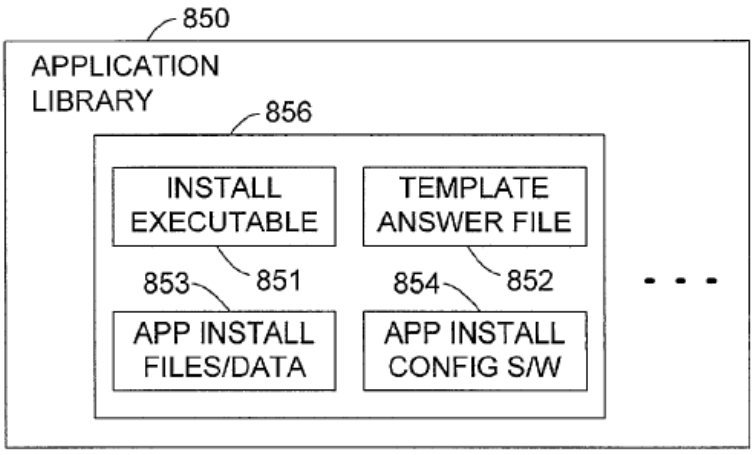
(2) Custom-Built VMs. Custom-built VMs are also staged using the process described above but may have additional applications installed. *See* Ex. 1004, 15:29-17:29. Custom-built VMs include VM disk files that are stored in the provisioning server and copied to hosts when deployed. *See id.*, 16:43-17:13. For custom-built VMs, the virtual disk image only includes the operating system. *See id.*, 15:41-46.

Application images (application installation files) are stored in the provisioning server's application library and copied to hosts. *See id.*, 19:20-20:30

(explaining the "application install software is copied to the temporary usage area" which "will reside on the host server where the VM is being deployed."), 23:28-51, 21:25-64, 4:42-51; Ex. 1003 ¶132.

Thus, Khandekar teaches images of software applications stored in an applications library 850 that are executable on VMs. *See e.g.*, Ex. 1004, 18:56-19:2 ("The applications library 850 is a repository of the software and data needed to install an application 856 in a VM. Since the applications are installed by the provisioning engine 810 during deployment phase (see below), they are installed unattended, that is, without any human intervention."), 10:27-42, 11:50-58 ("… library 850 … contain[s] code for any collection of the myriad of existing applications, from web servers, database servers, spreadsheets and word-processing programs, to games and browsers."), 12:13-23, FIG. 5:

FIG. 5



Ex. 1003 ¶131. Khandekar teaches "stor[ing] in the library 850 different copies of applications, with different settings for the different possible operating systems,

different copies of operating systems configured for different processors, etc." Ex. 1004, 12:13-23; Ex. 1003 ¶138.

Khandekar teaches that "any number of deployable VMs may be staged." Ex. 1004, 15:46-50. Thus, a POSITA would have been motivated to stage a plurality of VM images in the "VM staging repository 830" to increase the speed and efficiency in which VMs can be deployed—particularly for a set of frequently requested configurations. For example, it would have been obvious to keep staged VMs for a period, so they are readily accessible if they are requested again, avoiding the need to re-build custom VMs that are frequently requested. Ex. 1003 ¶133.

(3) Instantly-Deployable VMs. Unlike the other approaches, instantly-deployable VMs are suspended and resumed in a fully-booted state. *See* Ex. 1004, 17:30-50; Ex. 1003 ¶135. "[T]hese ready-to-deploy VMs … are created as follows. First, a new VM is created according to the steps outlined above in the section on VM Staging." Ex. 1004, 17:30-50. "In the first step, a model VM is created ***and the necessary OS and applications are installed in it***." *Id.*, 15:49-52; Ex. 1003 ¶134.

As discussed above, the staging process creates a VM with "VM disk files 832" and various configuration and software files/modules. *See* Ex. 1004, 16:43-

63. Therefore, instantly-deployable VMs include a virtual disk image file with an operating system and application data. Ex. 1003 ¶134.

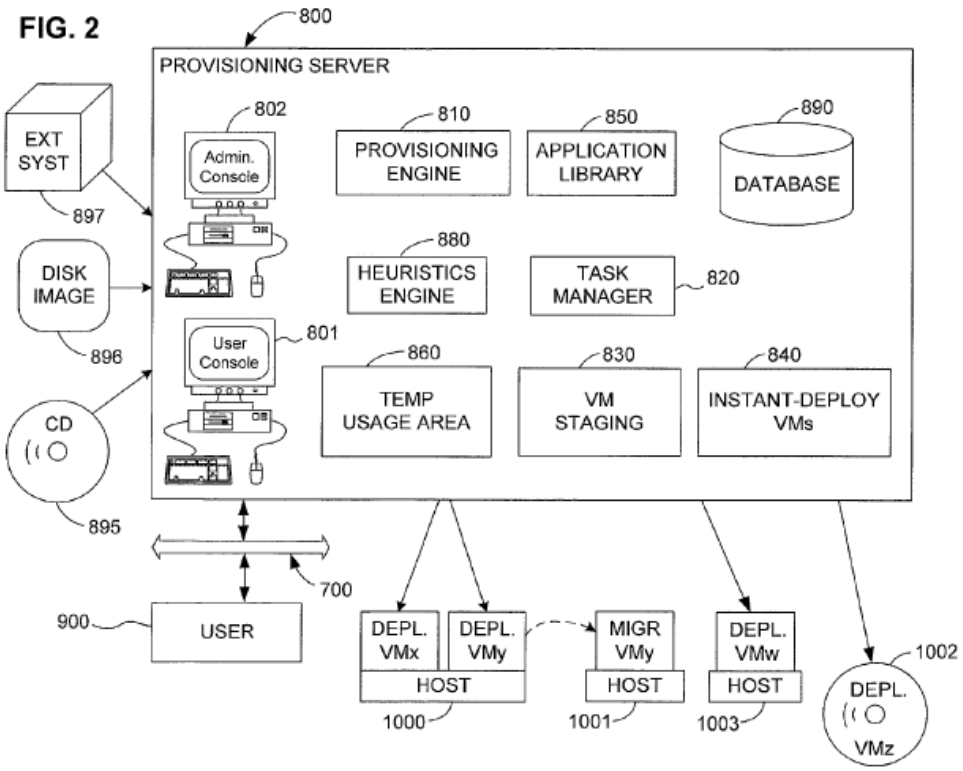
Khandekar teaches that images of instantly-deployable VMs are stored in the "instantly-deployable VMs repository 840." Ex. 1004, 18:1-15, 14:4-18, 18:6-15, FIGS. 2, 4. Therefore, Khandekar teaches or suggests that the provisioning server stores a plurality of instantly-deployable VMs. Ex. 1003 ¶136.

Other Limitations in [1e]. Khandekar teaches the image instances of a plurality of software applications that are executable on the plurality of virtual machines. As explained above, each pre-built, custom-built, and instantly-deployable VM includes a virtual disk image file with an OS and applications that are executable on the virtual machines. *See* Ex. 1004, 6:43-56; Ex. 1003 ¶137.

[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**

Khandekar teaches or suggests a control node (e.g., "provisioning server") that comprises an automation infrastructure (e.g., "provisioning engine," "heuristics/rules engine," and supporting functionality in the provisioning server)

to provide autonomic[6] deployment of the plurality of image instances of the virtual machine manager (e.g., VMM) on the application nodes (e.g., "hosts") by causing the plurality of image instances of the virtual machine manager to be copied from the software image repository to the application nodes. *See, e.g.*, Ex. 1004, 5:25-36 ("The invention enables users to configure, deploy and get access to substantially 'customized' computer systems … with no need for human intervention."), 9:9-25, 23:54-24:27, FIG 2:



[6] The '138 patent uses "autonomic" and "automatic" to refer to automated processes that do not require human intervention. *See*, *e.g.*, Ex. 1001, 8:13-23, 35:58-67, 7:5-16, 20:4-6; Ex. 1003, n.3.

Ex. 1003 ¶¶140-147.

Khandekar's provisioning server is a node in the network that controls the automated deployment of VMM and VM images from the provisioning server's software image repository (*see* §VI.A.3[1b]-[1e]) to hosts. *See, e.g.*, Ex. 1004, 5:25-36, 9:9-25, 23:28-51, FIGS. 2, 7. For example, Khandekar teaches selecting "one of the pre-configured VMs, which is then ***automatically prepared for deployment***." *Id.*, 4:36-41; Ex. 1003 ¶141. Image deployment is automatic. After a request is made, a heuristics engine "applies heuristics/rules to select the most appropriate VM from a library 830 of staged VMs." Ex. 1004, 10:27-42. "At least one host computer 1000, 1001 is made available to the provisioning server, and ***the heuristics engine 880 also selects which host machine 1000, 1001 is most appropriate to provision and deploy the VM on***." *Id.* "[T]here is a plurality of physical hosts on which VMs can be deployed. … the status of the hosts is monitored and the host on which a configured VM is to be deployed is selected heuristically." *Id.*, 4:58-65; *see also* §VIII.A.3[1g]; Ex. 1005, 68:21-24 (teaching monitoring status and performance information); Ex. 1003 ¶¶141-142.

A POSITA would have found it obvious to autonomically deploy image instances of VMMs from the provisioning server to the hosts because Khandekar teaches (1) automatic deployment of VM images, and (2) providing a VMM together with the VM. *See* Ex. 1004, 4:36-41, 4:58-65, 10:27-42, 8:36-44 ("…

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***."), 13:17-35; *see supra* §VI.A.1, §VI.A.2, §VI.A.3[1c] (explaining motivation to apply this teaching); Ex. 1003 ¶¶143-144. Therefore, it would have been obvious based on Khandekar's teachings that the provisioning server is a control node within the network that includes an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes. Ex. 1003 ¶144.

Khandekar teaches that the autonomic deployment of images is accomplished by copying the image from the provisioning server's repository to the host computer without human intervention. *See e.g.*, Ex. 1004, 18:56-19:2, 5:25-36, 4:36-41. This includes copying VM images, as explained below (*see* §VI.A.3[1g]) and a VMM that "is included and installed together with the VM." (*see* Ex. 1004, 8:36-44, 13:25-28). Since the VMM is included with the VM, it obviously is copied along with the VM to the host computer. Ex. 1003 ¶145. This conventional usage of VMM images was known and used in the art. *See, e.g.*, Ex. 1014 ¶[0062]; Ex. 1016 ¶¶[0049], [0043]; Ex. 1003 ¶145.

Claim 1 recites "autonomic deployment" but does not recite autonomic *requests* for VMs. Nonetheless, Khandekar teaches both, explaining that a user may be "any person or entity" but "could also be a software application or tool"

(Ex. 1004, 9:49-52) that "interact[s] with the provisioning server using a well-defined programmatic interface (API), which may be designed in a known manner." *Id.*, 9:56-59, 22:53-67. Khandekar teaches autonomic requests from software applications, and a POSITA would have been motivated to apply these teachings because it increases automation in a well-known manner. *See id.* And Khandekar provides express motivation by explaining the benefits of automation for testing environments. *See id.*, 22:53-67; Ex. 1003 ¶146.

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 desk 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.

Le teaches that disk imaging was conventionally used to clone, i.e., copy, an image of a computer, including OS and applications, onto other machines. *See, e.g.*, Ex. 1005, 2:19-21, 54:14-21, 10:55-62:

> ***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.

Ex. 1003 ¶150.

Le teaches techniques for copying images from a central repository to remote destination computers. *See* Ex. 1005, FIGS. 3-5:

Fig. 5 **(Emphasis Added)**

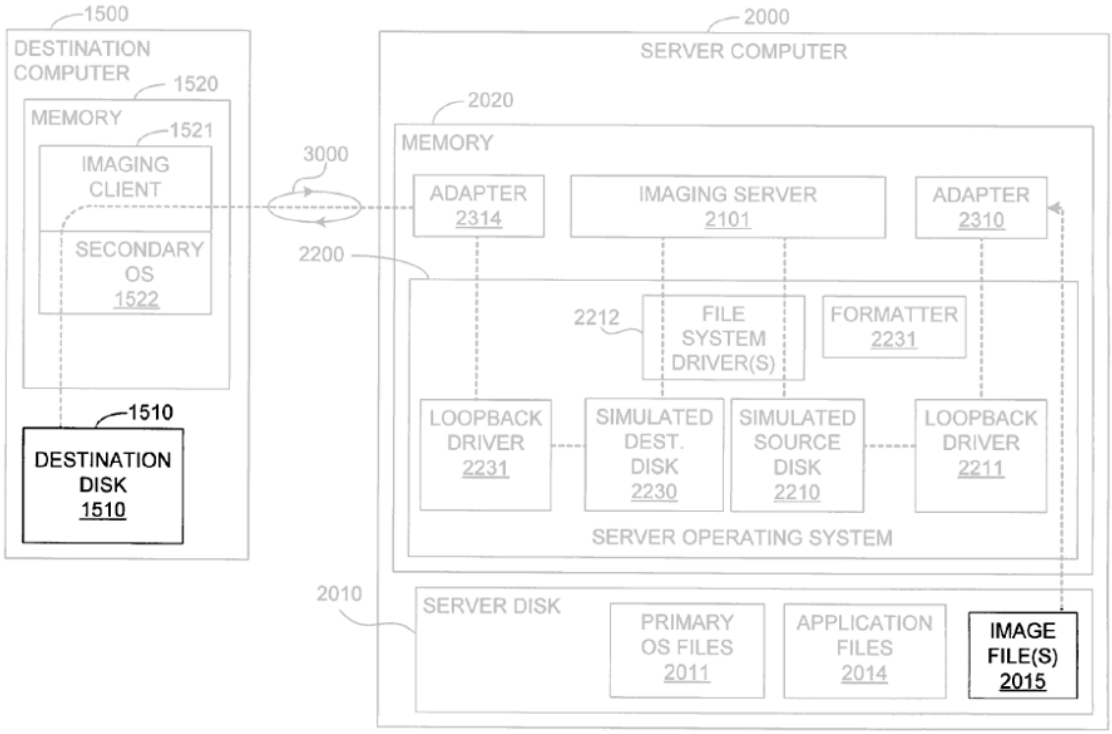


Fig. 3
(Prior Art)

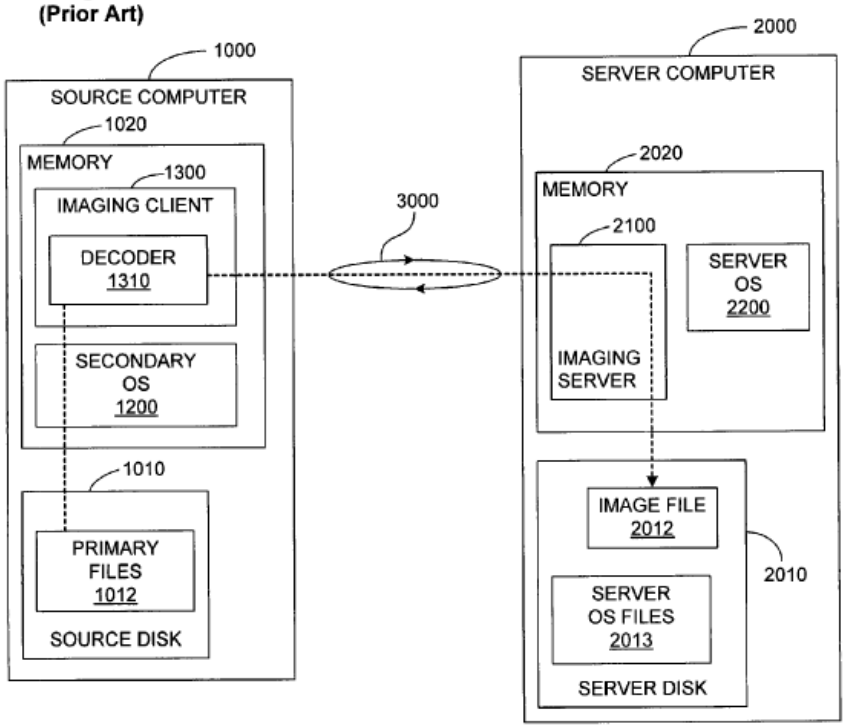
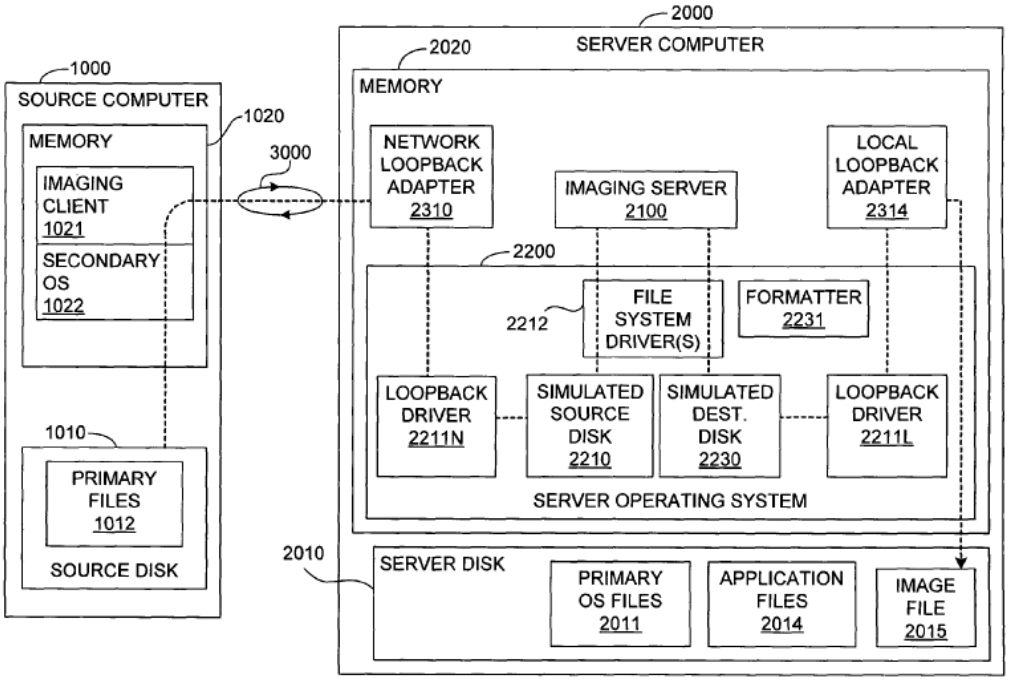


Fig. 4



Le describes the process of creating an image and explains that deployment is the inverse process whereby an image is copied to the destination computer. *See* Ex. 1005, 26:52-27:9 ("…the imaging server copies files and directories from the source file system to the destination file system…"), 72:14-22 ("… It then copies all files and folders from the source to the destination."), 20:25-37, FIG. 9; *infra* §VI.A.3[1g]; Ex. 1003 ¶151.

Le teaches a central repository, called the UCMS, deploying images to virtual machines and physical computers. *See* 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 (*see id.*, 70:12-35), which a POSITA would have been motivated to use to address Khandekar's teaching that

host characteristics should be determined before sending a VMM (*see* Ex. 1004, 13:17-35). Thus, it would have been obvious for the provisioning server to deploy VMM images to physical computers and VM images for virtual machines, using the hardware configuration of the host computer to guide the automation process and send an appropriate image to the host computer. *See* Ex. 1005, 65:20-33, 59:29-36, 70:12-35; Ex. 1003 ¶¶152-153.

[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.**

Khandekar teaches or suggests a control node (e.g., "provisioning server") that comprises an automation infrastructure (e.g., "provisioning engine," "heuristics/rules engine," and supporting functionality in the provisioning server) 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 (e.g., hosts). *See* Ex. 1004, 23:54-24:27, FIGS. 2, 7:

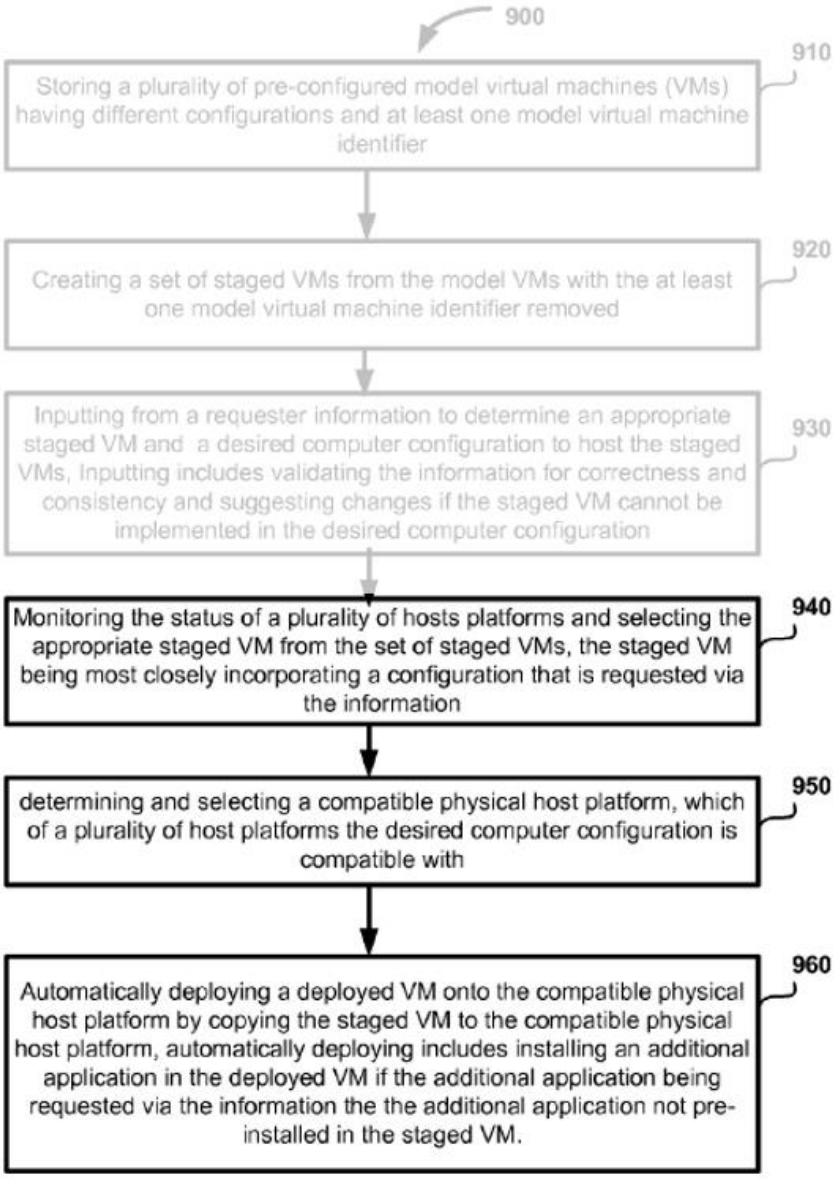


FIG. 7
(Emphasis Added)

See supra §§VI.A.3VIII.A.3[1e]-[1f]; Ex. 1003 ¶¶155-163.

Khandekar's provisioning server is a node in the network that controls the automated deployment of VM images from the provisioning server's repository to host computers. *See supra* §VI.A.3[1f]; Ex. 1004, 4:36-41, 4:58-65, 10:27-42. The

autonomic deployment process was described above. *See* §VI.A.3[1f]. For example, "a VM is ***automatically deployed*** onto the compatible physical host platform by ***copying the staged VM to the compatible physical host platform***." *See id.*, 23:28-51 (discussing Fig. 7), 5:25-36 (without human intervention); Ex. 1003 ¶156.

Khandekar provides additional details for deploying image instances of pre-built, custom, and instantly-deployable VMs; each is sufficient by itself to render this limitation obvious. *See* Ex. 1004, 9:10-25; *supra* §VIII.A.3[1e]. Khandekar's general teachings for deploying VMs apply to all three approaches. Ex. 1003 ¶157.

(1) Pre-Built, Staged VMs. Khandekar teaches copying pre-built VMs, with OS and applications installed, from the VM staging repository (library 830) to hosts that are automatically selected by a heuristics engine. *See* Ex. 1004, 9:15-25, 16:42-63, 15:40-16:12, 10:15-26; *supra* §VI.A.3[1e]. For staged VMs, Khandekar teaches that "'[d]eployment' is the step of taking a staged VM, ***copying*** it to a host machine and letting it configure with a new identity." Ex. 1004, 8:64-67, 9:1-3. Khandekar teaches that VMs are deployed "from a library 830 of staged VMs" to a "host computer" that the heuristics engine 880 determines is most appropriate. *Id.*, 10:27-42; Ex. 1003 ¶158.

Knowledge-based systems for autonomically deploying applications were well-known in the art. Khandekar explains that "[t]he system designer will be able

to determine which heuristics/rules to implement in the heuristics engine 880 using known design criteria." Ex. 1004, 10:35-42. This includes "[k]nown expert systems, 'intelligent assistants,' and other knowledge-based systems and engines…" *Id.*; Ex. 1003 ¶159.

(2) Custom-Built VMs. Custom-built VMs are also staged and are therefore copied to host computers as explained above. Ex. 1003 ¶160. Custom-built VMs may have additional applications, where installation images of those applications are copied from the applications library to the host and installed at the host. *See* Ex. 1004, 18:56-19:2 ("The applications library 850 is a repository of the software and data needed to install an application 856 in a VM."); Ex. 1003 ¶160.

The heuristics engine automatically determines a host for deployment; then the VM files, including its VM disk files with operating system and application installation software, are copied from the provisioning server to the host. *See* Ex. 1004, 21:25-64:

> …
> 2) Call the heuristics engine 880, to determine the VM's hardware configuration and the host to deploy on.
> …
> 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.
>
> …

Ex. 1003 ¶160. The application installation images are copied to the host and then installed at the end of the process. *See* Ex. 1004, 4:42-51, 23:54-24:27, 23:28-51 ("…The automatically deploying includes installing an additional application in the deployed VM if the additional application … [is] not pre-installed in the staged VM."); Ex. 1003 ¶161.

(3) Instantly Deployable VMs. Khandekar teaches copying fully-configured, instantly-deployable VMs, with pre-installed OS and software applications, from the instantly-deployable VMs repository (library 840) to host computers that are automatically selected by a heuristics engine. *See* Ex. 1004, 14:4-18 ("A brief description of the operating system, service pack and ***set of applications installed*** in each instantly deployable VM is preferably also included … the heuristics engine 880 applies the heuristics and rules to determine the best host, and provisions the VM on that host."), 11:20-35:

> 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. … Since the host servers have to have a hardware configuration identical to that where the VM was suspended, the heuristics engine will select the appropriate host server to deploy the VM on.

See also Ex. 1004, 21:65-22:13 (copying, e.g., the "suspend-to-disk image file 844 to the host."); Ex. 1003 ¶162.

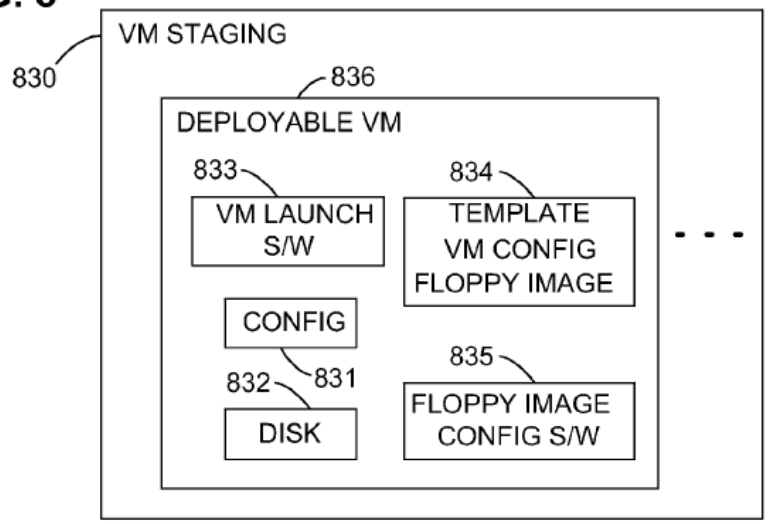
4. Claim 2

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

For Grounds 1 and 2, Khandekar teaches or suggests the additional limitation of claim 2, wherein the image instances of the plurality of software applications comprise virtual disk image files (e.g., "VM disk file 832," "virtual disk file 841"); Ex. 1003 ¶¶164-169.

For staged VMs, including pre-built and custom built VMs, Khandekar teaches the staged VMs include "VM disk files." *See* Ex. 1004, 16:47-63 ("For each VM, the data sets kept in the VM staging server 830 are illustrated in FIG. 3 and are as follows: … 2) VM disk files 832. …"), FIG. 3 (showing "DISK" 832):

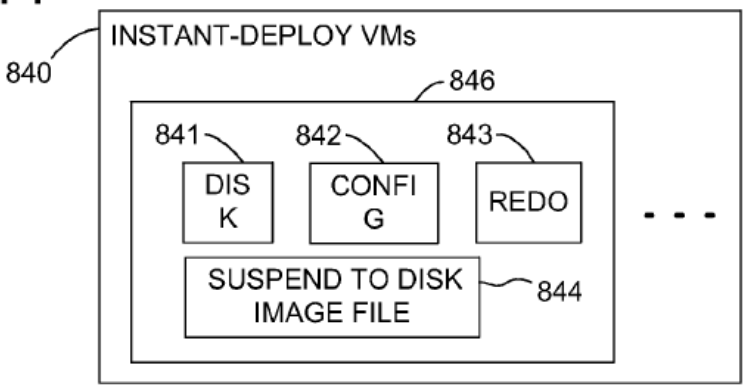
FIG. 3



Ex. 1003 ¶166.

Instantly-deployable VMs include a "virtual disk file 841." Ex. 1004, 16:43-63, 17:30-50, FIG. 4:

FIG. 4



Ex. 1003 ¶167.

A POSITA would have found it obvious that Khandekar teaches the claimed virtual disk image files because the '138 patent relies on VMware software for its embodiments, and Khandekar is a VMware patent. The '138 patent states that "if virtual machine manager 402 is an instance of ***VMWare ESX***, virtual machine disk

image files 408 may be specially captured ***.vmdk files***." Ex. 1001, 33:25-41. The VMware VMDK format is short for "Virtual Machine Disk." Ex. 1003 ¶168. Therefore, it would have been obvious that Khandekar's teachings include the virtual machine disk image files used in VMware products, which are used by the embodiments of the '138 patent. *Id*.

5. Claim 3

3.	**The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises an operating system.**

For Grounds 1 and 2, Khandekar teaches or suggests the additional limitation of claim 3. *See* Ex. 1004, 10:15-26, 14:4-18, 11:36-49 (teaching "operating systems such as Linux Red Hat, Windows 2000, Windows Me, etc."); Ex. 1003 ¶¶170-175. Khandekar teaches pre-built, custom-built, and instantly-deployable VMs. As explained for claim 1, each approach includes a virtual disk image with an operating system installed. *See* §VIII.A.3[1e]; Ex. 1004, 9:10-25; Ex. 1003 ¶¶171-175.

6. Claim 4

4.	**The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises a server application.**

For both Grounds 1 and 2, Khandekar teaches or suggests the additional limitation of claim 4, including server applications such as a web server and

database server. *See* Ex. 1004, 11:50-58 ("… library 850 … contain[s] code for any collection of the myriad of existing applications, from ***web servers, database servers*** … to games and browsers."), 12:24-39 ("For example, a user could indicate that he wants the system according to the invention to build a (virtual) web server running a Window OS"). Therefore, Khandekar's teachings satisfy this limitation. Ex. 1003 ¶¶176-179.

This is confirmed by the '138 patent, which includes an embodiment where the server application is a web server (Microsoft Internet Information Server). *See* Ex. 1001, 33:11-24; Ex. 1003 ¶178.

7. Claim 5

5.	**The distributed computing system of claim 1, wherein a first one of the virtual machines provides an environment that emulates an x86 platform.**

For both Grounds 1 and 2, Khandekar teaches or suggests the additional limitation of claim 5. Ex. 1003 ¶¶180-183. For example, Khandekar teaches the Linux Red Hat, Windows 2000, and Windows Me operating systems. *See supra* §VIII.A.5; Ex. 1004, 11:36-49. These operating systems typically ran on x86 platforms—indeed, x86 was the primary target platform for Windows 2000 and Windows Me. Ex. 1003 ¶182. The x86 platform was the predominant computing platform at the time, and VMware's ESX and GSX software emulated x86

platforms. *Id.*; *see also* Ex. 1004 13:17-35 ("Several products are available from VMware"). Thus, Khandekar renders claim 5 obvious. Ex. 1003 ¶¶182-183.

Ground 2: The Combination with Le Further Renders Claim 5 Obvious.

Le provides additional teachings on implementation details that render claim 5 obvious. *See* §VI.A.2; Ex. 1003 ¶¶184-186. Le teaches support for x86 virtual machines as described in Bugnion, which it incorporates by reference. *See* Ex. 1005, 12:63-13:11; Ex. 1012, 18:28-38 (teaching "Intel x86 architecture in a virtual machine"); Ex. 1003 ¶¶185-186.

8. Claim 9

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

For both Grounds 1 and 2, Khandekar teaches or suggests the additional limitation of claim 9, wherein the control node (e.g., "provisioning server") further comprises one or more rule engines (e.g., heuristics/rules engine) that provide autonomic deployment of the software applications to the virtual machines in accordance with a set of one or more rules. *See, e.g.,* Ex. 1004, 10:27-42:

> … the heuristics engine 880 also selects which host machine 1000, 1001 is most appropriate to provision and deploy the VM on. … ***The system designer will be able to determine which heuristics/rules to implement in the heuristics engine 880 using known design criteria***.

> Known expert systems, "intelligent assistants," and other ***knowledge-based systems and engines*** that can be embedded in other products and loaded with domain knowledge may be included in the heuristics engine 880 to implement the various decision functions.

See also id., 20:31-62, 11:11-19, 12:24-39, 14:4-18, 14:35-49, FIG 2; Ex. 1003 ¶¶187-193.

Khandekar teaches the use of rules and "knowledge-based systems and engines," which were known in the art to include rules engines. *See* Ex. 1004, 10:27-42; Ex. 1003 ¶188. For example, Khandekar teaches monitoring the status of hosts, with the heuristics engine providing autonomic deployment by applying rules based on the monitored status. *See* Ex. 1004, 4:58-65 ("… the status of the hosts is monitored and the host on which a configured VM is to be deployed is selected heuristically…"), 23:54-24:27, 12:57-13:8:

> Information about the current state of the different hosts is preferably input dynamically into a database or other data structure in the heuristics engine 880. … ***The heuristics engine may then also efficiently direct deployment*** and migration in order to balance the load on the different available hosts, for example by ***deploying a selected, staged VM to the host whose processor currently has the most available cycles***.

As an example, Khandekar teaches monitoring whether a host is overloaded beyond a threshold. *See id*., 13:44-56 ("Hosts can be monitored for load and ***if the load crosses a threshold*** …"); Ex. 1003 ¶190.

Khandekar teaches the use of known rules engines and applies rules based on whether host servers are overloaded. Therefore, Khandekar's teachings render this claim limitation obvious. Ex. 1003 ¶¶189-192.

This is confirmed by the '138 patent, which does not purport to invent a rules engine and relies on rules engines that were known in the art. *See* Ex. 1001, 26:45-61. Like Khandekar, the '138 patent looks to whether a server status is "overloaded" and then takes action accordingly with its rules engine. *See id*., 25:60-26:25; Ex. 1003 ¶¶189-191.

9. Claim 19

19[a]. A method comprising:

Khandekar renders [19a] obvious. *See* Ex. 1004, 4:30-35 ("a method and system implementation…"), 23:1-20, 4:30-35, 23:26-51, FIG. 7; Ex. 1003 ¶¶194-197. Khandekar teaches various steps and algorithms performed by its provisioning server, as explained for claim 1. *See* §VI.A.3.

Ground 2 (Le)

Le further teaches a method. *See* §VI.A.2; Ex. 1003 ¶¶198. For example, Le teaches an "Image Capture and Deployment Method." *See, e.g.*, Ex. 1005, 59:21-37, 64:59, 65:21-33. Le teaches various steps and algorithms, as explained for claim 1. *See* §VI.A.3; Ex. 1003 ¶199.

[19b]	**…**

Limitation [19b] is similar to [1c], with overlapping language shown in green below:

Limitation [19b]	**Limitation [1c]**
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	**a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes**

The analysis from [1c] applies here and renders this shared language obvious. *See* §VI.A.3[1c]; Ex. 1003 ¶200.

Limitation [19b] also recites "storing" the plurality of image instances "in a distributed computing system." Khandekar alone (Ground 1), and in combination with Le (Ground 2), teach a distributed computing system (*see* §VI.A.3[1a]) comprising a software image repository for storing the plurality of image instances recited in [1c] (*see* §VI.A.3[1b], [1c], [1e]). That analysis applies here, rendering this limitation obvious. Ex. 1003 ¶¶201-202.

[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;**

Limitation [19c] is encompassed by [1d]. That analysis applies here, rendering this limitation obvious. *See* §VIII.A.3[1d]. Ex. 1003 ¶203.

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

Limitation [19d] is identical to limitation [1e] except [19d] further requires "storing." The analysis from [1e] applies here, rendering this [19d] obvious. *See* §VIII.A.3[1e]; Ex. 1003 ¶204.

Regarding "storing," that limitation is taught by Khandekar alone (Ground 1), and in combination with Le (Ground 2), which teach a software image repository for storing the plurality of image instances recited in limitation [1e]. *See* §VIII.A.3[1b], [1e]. That analysis applies here, rendering this limitation obvious. Ex. 1003 ¶¶205-206.

[19e]	**…**

Limitation [19e] is similar to [1f], with overlapping language shown in green below:

Limitation [19e]	**Limitation [1f]**
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**	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**

Ex. 1003 ¶207. The analysis from [1f] applies here and renders this shared language obvious. *See* §VI.A.3[1f]. Claim 19's omission of "plurality of" and "from the software image repository" does not materially alter this analysis because, while this part of [19e] might be broader than the corresponding part of [1f], it is still satisfied by teachings that render [1f] obvious. Ex. 1003 ¶208.

Regarding the first portion of [19e], Khandekar teaches or suggests a control node that comprises an automation infrastructure (e.g., "heuristics/rules engine") to *provide autonomic deployment* of the plurality of image instances of the virtual machine manager on the application nodes, as explained for [1f]. *See* §VIII.A.3[1f]. That analysis applies here. Therefore, Khandekar renders obvious the step of executing a rules engine (e.g., the "heuristics/rules engine") to perform operations to autonomically deploy the "image instances…" recited in the remainder of the claim. *See id.*; Ex. 1003 ¶¶209-210.

[19f] …

Limitation [19f] is similar to [1g], with overlapping language shown in green below:

Limitation [19f]	**Limitation [1g]**
executing a 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**	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**

Ex. 1003 ¶211. The analysis from [1g] applies here and renders this shared language obvious. *See* §VI.A.3[1g]. Claim 19 omits some limitations from the latter portion, but this does not materially alter the analysis because, while this part of [19f] might be broader, it is satisfied by teachings that render [1g] obvious. Ex. 1003 ¶212.

Regarding the remainder of [19f], Khandekar teaches or suggests a control node that comprises an automation infrastructure (e.g., "heuristics/rules engine") to *provide autonomic deployment* of the plurality of image instances of the software applications on the virtual machines. *See* §VIII.A.3[1g], VI.A.9[19e]. That analysis applies here, rendering obvious the step of executing the rules engine (e.g., the "heuristics/rules engine") to perform operations to autonomically deploy the "image instances…" recited in the remainder of the claim. *See id.*; Ex. 1003 ¶¶213-214.

10. Claim 20

20.	**The method of claim 19, wherein storing the image instances of the plurality of software applications comprises storing virtual disk image files.**

Claim 20 is nearly identical to claim 2. That analysis applies here, rendering claim 20 obvious. *See supra* §VI.A.4. Claim 20 additionally recites "storing." This is taught by Khandekar alone (Ground 1), and in combination with Le (Ground 2), which teach or suggest *storing* the plurality of image instances of a plurality of software applications. *See* §VIII.A.9[19d], §VIII.A.3[1b]; Ex. 1003 ¶216.

11. Claim 21

21.	**…**

The limitations of Claim 21 are identical to those recited for Claim 3. That analysis applies here, rendering claim 21 obvious. *See* §VI.A.5; Ex. 1003 ¶¶219-220.

12. Claim 26

26[a].	**A non-transitory, computer-readable medium comprising instructions stored thereon, that when executed by a programmable processor:**

Khandekar renders [26a] obvious. Ex. 1003 ¶¶222-224. Khandekar teaches that its provisioning server includes standard computer components, which obviously would have included a processor to execute software (instructions) that implement the functions of the provisioning server. *See* Ex. 1004, 9:26-45. It

would also have been obvious to use a non-transitory, computer-readable medium, such as a hard disk, to store the provisioning server's software, as was common for the vast majority of computers. Ex. 1003 ¶224.

Ground 2: The Combination with Le Further Renders [26a] Obvious.

Le teaches implementation details that further render [26a] obvious. *See* §VI.A.2; Ex. 1003 ¶¶225-227. For example, Le teaches using software installed on a computer to implement the features of its UCMS, which plays an analogous role to Khandekar's provisioning server (*see supra* §VI.A.2), with further teachings that software programs comprise computer-executable instructions and stored on disks. *See* Ex. 1005, 59:29-36, 60:13-18, 23:44-52, 89:8-31, 26:38-52, FIG. 4; Ex. 1003 ¶226. It would have been obvious to apply these teachings to the provisioning server. Ex. 1003 ¶227.

[26b]	**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,**

Aside from verb tense, limitation [26b] is identical to [19b]. That analysis applies here, rendering this limitation obvious. *See* §VI.A.9[19b]; Ex. 1003 ¶228.

[26c]	**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;**

Limitation [26c] is identical to [19c]. That analysis applies here, rendering this limitation obvious. *See* §VI.A.9[19c]; Ex. 1003 ¶229.

[26d]	**store a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines;**

Aside from verb tense, limitation [26d] is identical to [19d]. That analysis applies here, rendering this limitation obvious. *See* §VI.A.9[19d]; Ex. 1003 ¶230.

[26e]	**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**

Limitation [26e] is identical to [19e] except [26e] omits "executing a rules engine to." This omission does not materially alter the analysis because, while [26e] might be broader than [19e], it is still satisfied by teachings that render [19e] obvious. That analysis applies here, rendering this limitation obvious. *See* §VI.A.9[19e]; Ex. 1003 ¶231.

[26f]	**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.**

Limitation [26f] is identical to [19f] except [26f] omits "executing a rules engine to." This omission does not materially alter the analysis because, while

[26f] might be broader than [19f], it is still satisfied by the teachings that render [19f] obvious. That analysis applies here, rendering this limitation obvious. *See* §VI.A.9[19f]; Ex. 1003 ¶232.

B. Ground 3: Khandekar in View of Le and Sidebottom

1. Motivation to Combine

The combination of Khandekar and Le was explained for Ground 2. *See* §VI.A.2. A POSITA would have been further motivated to apply Sidebottom's teachings regarding a rules engine with a forward-chaining algorithm. *See, e.g.*, Ex. 1006, Abstract, 1:36-44; Ex. 1003 ¶234.

The combination of Khandekar and Le includes automatic deployment using a heuristics/rules engine. *See* §VI.A.3[1f]-[1g], §VI.A.8. Khandekar provides express motivation to apply known "knowledge-based systems and engines" to its heuristics/rules engine. *See* Ex. 1004, 10:27-42; Ex. 1003 ¶235. Sidebottom provides such a teaching in the form of a "rules engine" that applies a "forward-chaining rule-based algorithm, such as LEAP or RETE." *See* Ex. 1006, 3:44-58, 9:36-45. Khandekar's express motivation to combine its teachings with known knowledge-based systems and engines would have included the forward-chaining rule algorithms taught by Sidebottom. Ex. 1003 ¶236.

Furthermore, the prior art provides motivation to combine these teachings. For example, Khandekar teaches that "[h]osts can be monitored for load…" *See*

Ex. 1004, 13:44-56; *supra* §VI.A.8. Das teaches monitoring "facts" such as "an average load on a CPU" and applying a "Chaining of Rules." *See* Ex. 1013 ¶¶[0083]-[0084], [0163]-[0164]. Therefore, Das provides a teaching, suggestion, and motivation to apply chaining rules, such as a forward-chaining rules engine, in response to monitored load, as taught by Khandekar. Ex. 1003 ¶237.

As another example, Russell teaches that forward-chaining algorithms, such as the widely-used RETE, apply Modus Ponens inference rules, which are useful to make inferences in response to newly-arrived information. *See* Ex. 1010 at 280, 286 n. 4, 310. Khandekar teaches taking actions based on newly-arrived information (e.g., monitored load or status). *See*, *e.g.*, Ex. 1004, 13:44-56. Therefore, Russell provides a teaching, suggestion, and motivation to apply forward-chaining rules such as RETE to Khandekar. Ex. 1003 ¶¶238-239.

Sidebottom provides additional motivation to combine, explaining that its teachings are broadly applicable for defining policies in a computer network. *See* Ex. 1006, 1:36-38. Sidebottom teaches monitoring events in the network, evaluating if the condition of a condition/action rule is satisfied using a rules engine, and if so, implementing the action. *See id*., 2:65-3:17, 3:32-43, 5:63-6:17, 4:26-44. As was typical for rules engines, Sidebottom teaches that its rules were "specified in terms of high-level business information." *See id.*, 1:41-44.

Therefore, Sidebottom's high-level rules were adaptable to a wide variety of applications and business needs. Ex. 1003 ¶240.

Forward-chaining rules engines, such as LEAP or RETE, could be readily utilized in the combination of Khandekar and Le because they are general algorithms, not tied to any particular implementation. Sidebottom and Khandekar both teach the use of rules for defining condition/action rules based on monitored events in a network. *See* Ex. 1004, 10:27-4, 13:44-56; *See* Ex. 1006, 3:32-43, 5:63-6:17, 4:26-44. Therefore, the combination would have used Sidebottom's rules engine teachings in a natural and conventional manner. Ex. 1003 ¶241.

A POSITA would have had a reasonable expectation of success in combining the teachings of Khandekar, Le, and Sidebottom given the similarity in their teachings, which are directed to complimentary aspects of distributed computing systems. The combination of Khandekar and Le was explained for Ground 2. Khandekar teaches the use of known knowledge-based systems and engines and Sidebottom teaches such a rules engine, with forward-chaining rules algorithms, such as LEAP or RETE, that were known in the art. Their application to Khandekar's heuristics/rules engine would have been for their known and conventional purpose, as explained above. Given a POSITA's knowledge of distributed computer systems, related monitoring techniques, and rules engines, the combination would have been well within the level of ordinary skill in the art. *See*

§IV. Ex. 1003 ¶242. The combination applies known techniques (e.g., forward-chaining rules algorithms, such as LEAP or RETE) to improve similar devices (e.g., distributed computing systems in a network) in the same way as they were used in the prior art. Ex. 1003 ¶243.

2. Claim 9

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

The combination renders claim 9 obvious. Ex. 1003 ¶¶244-251. Khandekar teaches the control node of claim 9. *See* §VI.A.8; Ex. 1003 ¶245. A POSITA would have been motivated to apply Sidebottom's teachings to Khandekar's heuristics/rules engine. *See* §VI.B.1; Ex. 1003 ¶246.

Sidebottom teaches a rules engine using "forward-chaining rule-based algorithm, such as LEAP or RETE." *See* Ex. 1006, 3:44-58, 9:36-45. Sidebottom teaches high-level condition/action rules. *See id.*, 1:41-44 ("The business logic ***rules may be defined as condition/action rules specified in terms of high-level business information*** that 'fire' in response to one or more low-level network events."), 9:36-45 ("Rules engine 20 begins a condition/action rule processing cycle … For example, rules engine 20 may use a forward-chaining algorithm to evaluate the 'condition' attributes."); Ex. 1003 ¶247. For example, Sidebottom teaches monitoring events in the network, evaluating if the condition of a

condition/action rule is satisfied using a rules engine, and if so, implementing the action. *See* Ex. 1006, 2:65-3:17, 3:32-43, 5:63-6:17, 4:26-44, Abstract, FIG. 2:

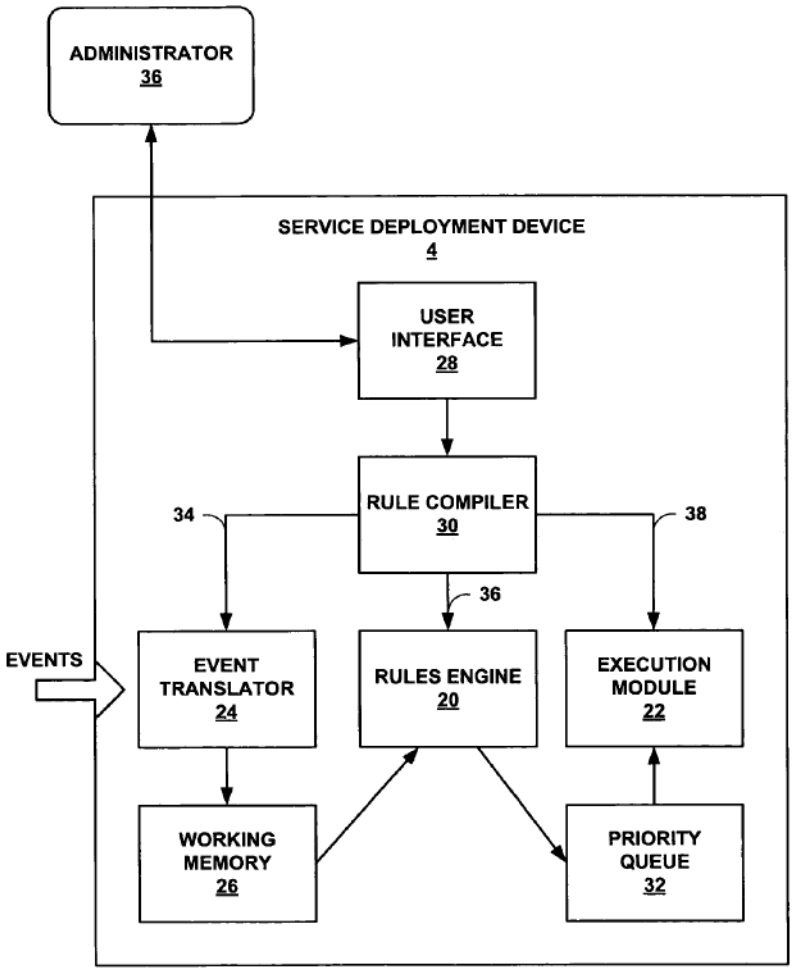


FIG. 2

Ex. 1003 ¶248.

Similarly, Khandekar similarly teaches monitoring host status, such as whether the load of a host has exceeded a threshold, and using that information for its heuristics/rules engine. *See*, *e.g.*, Ex. 1004, 4:58-65, 13:44-56. Therefore, a POSITA would have found it obvious to apply Sidebottom's teachings to Khandekar by monitoring events in hosts—such as whether a host is overloaded—

evaluating if the condition of a condition/action rule is satisfied using a forward-chaining rules engine, and if so, implementing the action to guide the deployment of software applications. *See* Ex. 1006, 2:65-3:17, 3:32-43, 5:63-6:17, 4:26-44; *supra* §VI.A.3[1g] (explaining autonomic deployment), §VI.A.8 (same); Ex. 1003 ¶249.

This is consistent with the specification of the '138 patent, which similarly looks to whether a server status is "overloaded" and then takes action accordingly with its rules engine. *See* Ex. 1001, 25:60-26:25; Ex. 1003 ¶250.

3. Claim 10

10.	**The distributed computing system of claim 9, wherein the one or more rule engines are forward-chaining rule engines.**

The combination renders claim 10 obvious. Ex. 1003 ¶¶252-255; *see supra* §VI.B.1, §VI.B.2 (explaining the application of Sidebottom's forward-chaining rule engines to Khandekar and Le). Sidebottom teaches the use of forward-chaining rules algorithms for rules engines. *See* Ex. 1006, 3:44-58 ("… In one embodiment, SDD 4 uses a forward-chaining rule-based algorithm, such as LEAP or RETE."), 9:36-45 ("… rules engine 20 may use a forward-chaining algorithm to evaluate the 'condition' attributes."); Ex. 1003 ¶253.

The '138 patent does not purport to invent a rules engine and instead relies on known rules engines including the RETE and LEAP algorithms. *See* Ex. 1001, 26:45-61; Ex. 1003 ¶254.

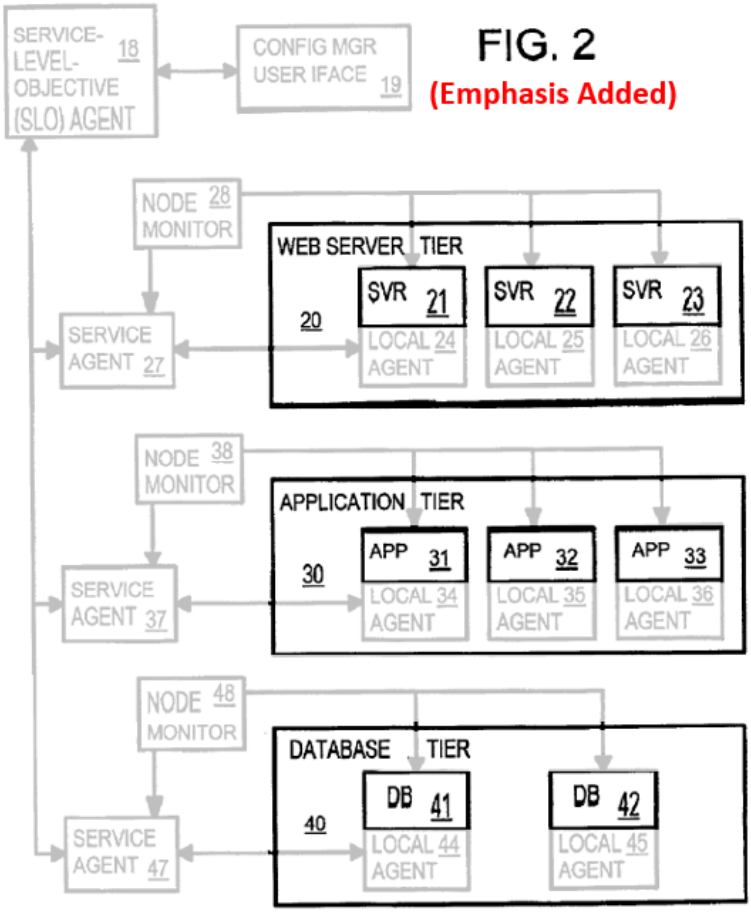
C. Ground 4: Khandekar in View of Le and Stone

1. Motivation to Combine

The combination of Khandekar and Le was explained for Ground 2. *See* §VI.A.2. It includes automatic deployment using a heuristics/rules engine that automatically deploys images or takes other actions based on the monitored status of the hosts. *See* §VI.A.3[1f]-[1g], §VI.A.8; Ex. 1003 ¶258. A POSITA would have been further motivated to apply Stone's teachings regarding a multi-tier web service and monitoring service-level-objectives (SLO) based on those tiers. *See, e.g.*, Ex. 1007 ¶¶[0028]-[0031], [0034], [0040]. Ex. 1003 ¶257.

Khandekar and Le teach techniques for virtual machine systems, which were used in a variety of web and enterprise applications. Khandekar explains its teachings may be applied to a "web application" that includes a ***web server***, an ***application server***, and a ***database server*** on a network of host computers. *See* Ex. 1004 23:1-21. This three-tiered architecture was common for web sites. Ex. 1003 ¶259.

Like Khandekar, Stone teaches a common three-tiered architecture for web sites, with servers organized into "***web server***," "***application***," and "***database***" tiers. *See* Ex. 1007, FIG. 2:



Ex. 1003 ¶260.

A POSITA would have been motivated to combine Stone's tiered architecture with Khandekar and Le because Khandekar provides an express motivation to apply its teachings to known web applications, including applications with Stone's three-tiered architecture. *Id*. ¶261.

Moreover, Stone's tiered service model offered several benefits by organizing components into functional tiers corresponding to the data flow for each service. *See* Ex. 1007 ¶[0037]. Collected statistics are organized by service and

tier, allowing automated diagnosis based on service-level objectives ("SLOs"). *See id.* ¶¶[0037]-[0038]. The tiered model also allows automatic reallocation within a tier. *See id.* ¶[0055]. Thus, a POSITA would have been motivated to apply Stone's tier-based teachings to improve the distributed computing system taught by the combination of Khandekar and Le. Ex. 1003 ¶261.

A POSITA would have been further motivated to apply Stone's teaching of monitoring nodes based on tiers. *See* Ex. 1007 ¶[0001]. Stone teaches an SLO monitoring and control system that collect statistics and monitors events by tier (*see id.* ¶¶[0037]-[0038], [0040], FIG. 2), allowing an automated management process (with an SLO agent) to diagnose and correct problems if an SLO is not met, such as by replicating nodes or increasing resources. *See id.* ¶[0057], Abstract; Ex. 1003 ¶¶262-263. These teachings complement and enhance the automation of Khandekar's teachings for monitoring the status of hosts and taking actions in response. *See* Ex. 1004, 13:44-56; Ex. 1003 ¶¶263-264.

A POSITA would have had a reasonable expectation of success in combining Khandekar, Le, and Stone given the similarities in their teachings, which are directed to complementary aspects of distributed computing systems. Khandekar provides express motivation to apply Stone's tiered architecture, and both teach similar monitoring of the hosts/nodes in distributed computing for web applications. VMware systems were commonly used for Web applications, and the

use of VMware patents Khandekar and Le for a common Web site architecture would have been well within their intended scope. Distributed computing systems, virtual machines and related monitoring techniques were well known at the time. Given a POSITA's knowledge of these topics, the combination would have been well within the level of ordinary skill in the art. *See* §IV; Ex. 1003 ¶265.

The combination applies an obvious combination of well-known prior art elements according to known methods to yield predictable results. Ex. 1003 ¶266.

2. Claim 17

17.	**The distributed computing system of claim 1, wherein the application nodes are organized into tiers, and wherein status data is collected based on the tiers.**

The combination renders claim 17 obvious. Ex. 1003 ¶¶267-278. Khandekar teaches collecting status data of the hosts. *See* Ex. 1004, 4:58-65 ("… the status of the hosts is monitored and the host on which a configured VM is to be deployed is selected heuristically."), 12:57-13:8, 13:44-56, 23:28-51; Ex. 1003 ¶269. Le also teaches status data is collected. *See* Ex. 1005, 68:21-24 ("… the agent 7300 can also report status and system performance information to the imaging server 2101, allowing the UCMS to detect the presence of the agent and to monitor the deployed computer's health."); Ex. 1003 ¶270. A POSITA would have been motivated to combine Khandekar and Le with Stone's teachings regarding the

monitoring of status data in a tiered distributed computing system, including web server, application, and database tiers. *See supra* §VI.C.1; Ex. 1003 ¶271.

Stone teaches a distributed computing system (*see* Ex. 1007 ¶[0001], ¶[0026]), with application nodes organized into tiers. For example, Stone teaches "a multi-tiered service model of an e-business web service" with, e.g., web server, application, and database tiers. *See id.* ¶¶[0028]-[0031], [0034], FIG. 1:

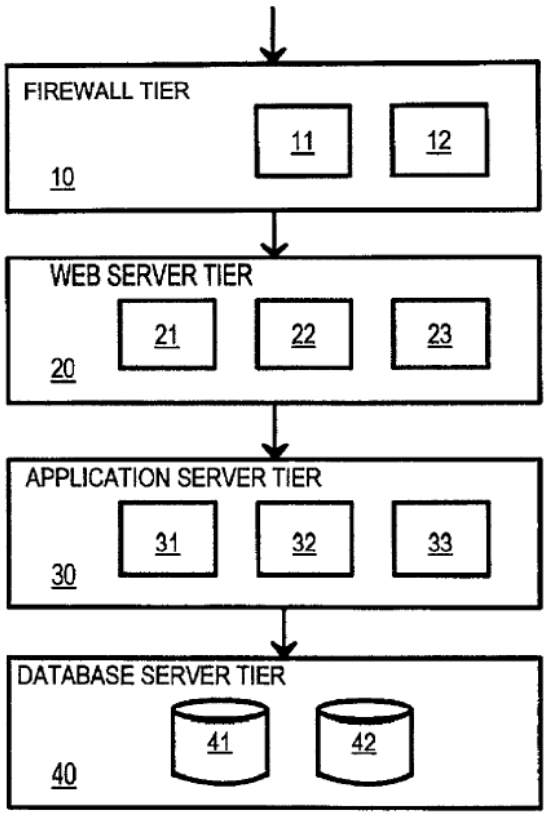
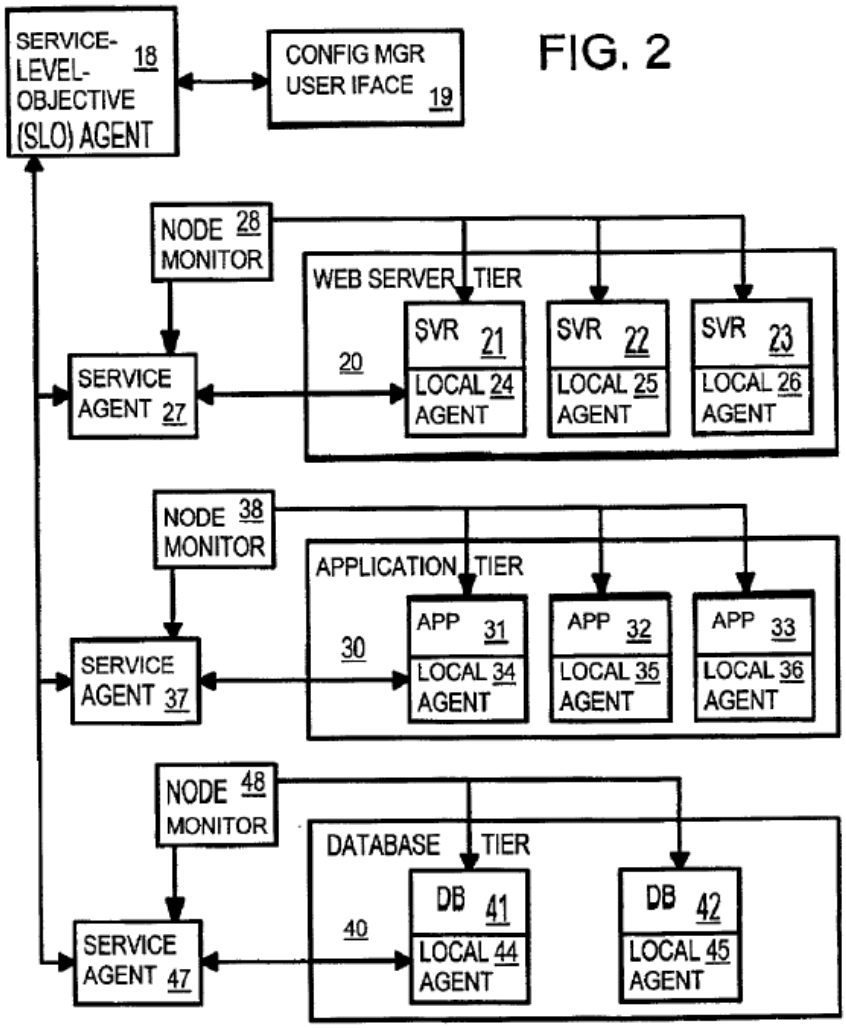


FIG. 1

Ex. 1003 ¶272.

Stone teaches status data is collected based on those tiers. For example, Stone teaches "a service-level-objective (SLO) monitoring and control system for a multi-tier web site." *See* Ex. 1007 ¶[0040], [0037] ("The tiered model organizes these components into functional levels or tiers that correspond to the data flow for each Service. ***Statistics collected and events monitored can thus be organized by Service and tier*** …"), FIG.2:



Ex. 1003 ¶273.

Stone teaches that "service agents" collect status data based on tiers; for example, service agent 27 collects status data for the web server tier 20. *See* Ex. 1007 ¶¶[0042]-[0046]; Ex. 1003 ¶274. Stone also teaches "node monitors" that collect status data based on the tiers, for example, node monitor 28 which collects status data for the web server tier 20. *See* Ex. 1007 ¶[0044], ¶[0046]; Ex. 1003 ¶275.

Based on Stone's teachings, a POSITA would have found it obvious to organize application nodes, such as Khandekar's hosts, into tiers, including web server, application, and database tiers. Indeed, Khandekar already teaches or suggests organizing hosts into those tiers. *See* Ex. 1004 4:58-65, 23:1-21.

Additionally, Stone teaches collecting status data based on those tiers. All three references teach monitoring node/host status, and Stone teaches an obvious way to organize the collection of status data when application nodes are organized into tiers, as explained above. Stone teaches an improved way to monitor a Web service in a tiered environment. *See* Ex. 1007 ¶[0038]; *see* §VIII.C.1. Therefore, the combination renders claim 17 obvious. Ex. 1003 ¶276.

This is confirmed by the '138 patent, which discloses application nodes organized into the same three tiers as Stone, with status data collected based on the tiers. *See* Ex. 1001, 13:40-56 ("…For example, storefront domain 34A includes ***a***

web server tier (labeled 'web tier') 36A, a ***business application tier*** (labeled 'app tier') 36B, and a ***database tier*** (labeled 'DB tier') 36C …"), Fig. 2:

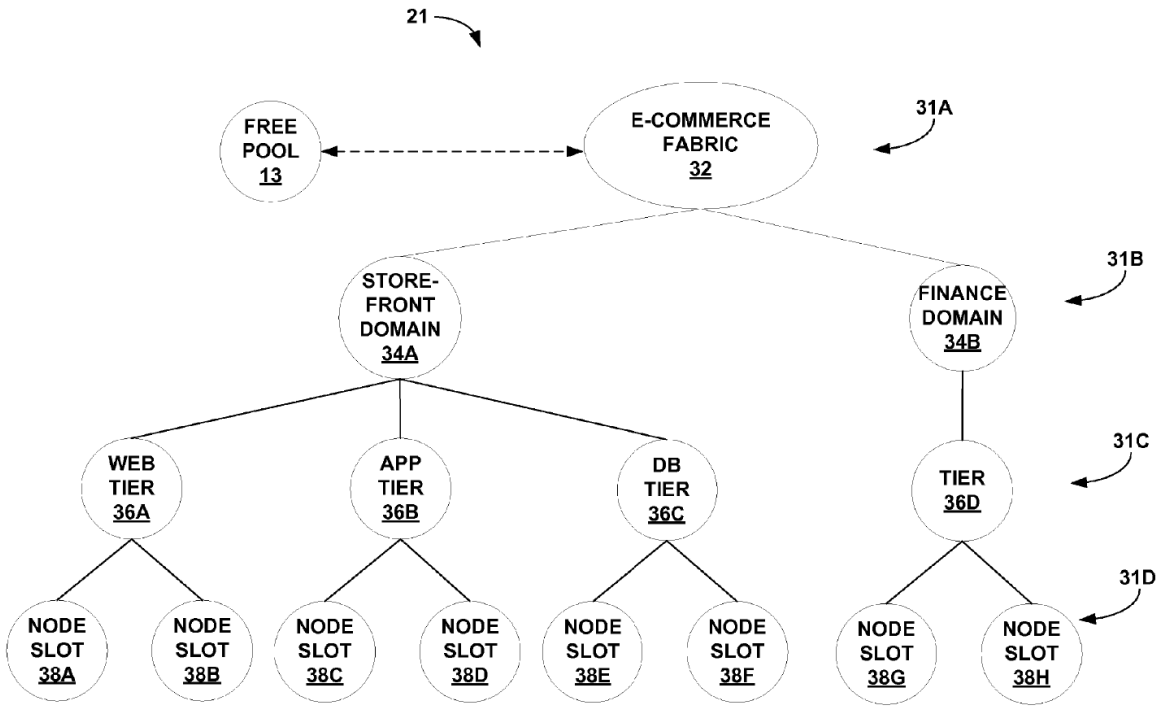


FIG. 2

Ex. 1003 ¶¶277-278.

VII. CONCLUSION

For the foregoing reasons, Petitioner requests *inter partes* review and that the challenged claims be canceled as unpatentable.

VIII. MANDATORY NOTICES AND FEES

Real Party In Interest: The real parties-in-interests here are Netflix, Inc. and Netflix Streaming Services, Inc. No other parties directed, controlled, or funded this Inter Partes Review proceeding (IPR).

Related Matters: The '138 patent is the subject of the following civil actions and administrative proceedings:

- *Broadcom Corp. et al. v. Netflix, Inc.*
  Case No. 3:20-cv-04677-JD (N.D. Cal) transferred from 8:20-cv-00529 (C.D. Cal)

Lead and Backup Counsel: Pursuant to 37 C.F.R. §42.8(b)(3) and 42.10(a), Petitioner designates Harper Batts (Reg. No. 56,160) as lead counsel and Jeffrey Liang (Reg. No. 69,043) and Chris Ponder (Reg. No. 77,167) as back-up counsel, each of Sheppard, Mullin, Richter & Hampton LLP.

Service Information: Petitioner consents to service by electronic mail addressed to:

HBatts@sheppardmullin.com,
CPonder@sheppardmullin.com,
JLiang@sheppardmullin.com, and
Legal-Netflix-Broadcom-IPRs@sheppardmullin.com

Petitioner's counsel may also be served by mail or hand delivery at Sheppard, Mullin, Richter & Hampton LLP, 379 Lytton Ave., Palo Alto, CA

94301. Petitioner's counsel may be reached by telephone at (650) 815-2673 and by facsimile at (650) 815-4668.

Fees: The Office is authorized to charge fees for this Petition to Deposit Account 50-4561, Ref. 64MK-316262-138.

Respectfully submitted,

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/Harper Batts/
Harper Batts (Reg. No. 56,160)

CLAIM APPENDIX

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.

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

3. The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises an operating system.

4. The distributed computing system of claim 1, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises a server application.

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

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

10. The distributed computing system of claim 9, wherein the one or more rule engines are forward-chaining rule engines.

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

19[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.

20. The method of claim 19, wherein storing the image instances of the plurality of software applications comprises storing virtual disk image files.

21. The method of claim 19, wherein a first image instance of the plurality of image instances of the plurality of software applications comprises an operating system.

26[a]. A non-transitory, computer-readable medium comprising instructions stored thereon, that when executed by a programmable processor:

[26b] 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,

[26c] 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;

[26d] store a plurality of image instances of a plurality of software applications that are executable on the plurality of virtual machines;

[26e] 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

[26f] 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.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on September 10, 2020, true and correct copies of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 8,572,138 and Exhibits 1001-1017 were served in its entirety on the Patent Owner at the following addresses of record as listed on PAIR via Priority Mail Express® or Express Mail:

CA, Inc.
One CA Plaza
Islandia, NY 11749

Official Correspondence Address

Vanessa Elfend
15101 Alton Parkway
Irvine, CA 92618

Patent Owner Correspondent Address

The undersigned certifies that true and correct copies of the Petition for *Inter Partes* Review of U.S. Patent No. 8,572,138 and Exhibits 1001-1017 were also sent via electronic mail to the attorneys of record for Plaintiff in the concurrent litigation matter(s):

Adrienne Dominguez (Pro Hac Vice) - adrienne.dominguez@tklaw.com
Bruce S. Sostek (Pro Hac Vice) - bruce.sostek@tklaw.com
Cary Chien, Bar No. 274078 - cchien@hopkinscarley.com
Christopher A. Hohn, Bar No. 271759 - chohn@hopkinscarley.com
John V. Picone, III - jpicone@hopkinscarley.com
Michael D. Karson (Pro Hac Vice) - michael.karson@tklaw.com
Richard L. Wynne, Jr. (Pro Hac Vice) - richard.wynne@tklaw.com
Robert K. Jain, Bar No. 309728 - rjain@hopkinscarley.com
Vishal Patel (Pro Hac Vice) - vishal.patel@tklaw.com

True and correct copies of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 8,572,138 and Exhibits 1001-1017 will also be served by September 11, 2020 in its entirety on the Patent Owner at the following address of record as listed on PAIR via Priority Mail Express® or Express Mail:

Avago Technologies International Sales Pte. Limited
1 Yishun Avenue 7
Singapore, Singapore 768923

Patent Owner

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

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

Date: September 10, 2020
379 Lytton Ave.
Palo Alto, CA 94301
Tel: (650) 815-2673

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS, AND TYPE STYLE REQUIREMENTS

1. This Petition complies with the type-volume limitation of 14,000 words, comprising 13938 words, as counted using the Microsoft Word software that was used to prepare this paper, excluding the parts exempted by 37 C.F.R. § 42.24(a).

2. This Petition complies with the general format requirements of 37 C.F.R. § 42.6(a) and has been prepared using Microsoft® Word 2016 in 14-point Times New Roman.

Respectfully submitted,

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

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

Date: September 10, 2020

379 Lytton Ave.
Palo Alto, CA 94301
Tel: (650) 815-2673