IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

NETFLIX, INC.,

Petitioner,

v.

AVAGO TECHNOLOGIES INTERNATIONAL SALES PTE. LIMITED,

Patent Owner.

_____

Case No. IPR2020-01582

U.S. Patent 8,572,138

_____

REPLY DECLARATION OF PRASHANT SHENOY

I, Prashant Shenoy, declare as follows:

1. I am the same Prashant Shenoy that previously submitted a declaration in this proceeding which includes the scope of my work, qualifications, and background (Ex-1003). In this reply declaration, I have been asked to respond to certain opinions in Dr. Rosenblum's Declaration (Ex-2001), addressed below.

2. I have reviewed and considered each of the materials listed in the attached Appendix 1. I make this declaration based upon my own personal knowledge and, if called upon to testify, would testify competently to the matters contained herein.

I. Khandekar and Le teach the same VMs and VM images as the '138 patent.

3. As an initial matter, Dr. Rosenblum argues that the '138 patent uses a different type of VM than Khandekar. This is incorrect. Khandekar and Le—both VMware patents—use the same types of VMs and VM images as the '138 patent; they all use commercial VMware software to provide VMs and save images of VMs. Ex-1001, 32:59-63, 33:25-41; Ex-1004, 13:17-35; Ex-1005, 65:65-66:10, 13:22-30.

4. Khandekar at times uses shorthand to refer to its VM images based on the type of VM they contain: pre-built, custom-built, and instantly-deployable VM. Khandekar is clear that these are all "stored" images of VMs including virtual disk image files. Ex-1004, 15:40-50, 16:43-52, 17:30-50, FIGS 3, 4. While these stored

VM images are run on *host* computers using VMMs, Khandekar is clear that they are stored as VM images on the provisioning server. *See* Ex-1004, 13:17-35 (using existing VMware software to "load and run multiple VMs."), FIGS. 1-4; *see also* Ex-1003 ¶¶112-119; Ex-1005, 65:65-66:10 (discussing VMware ESX software includes a VMM that controls and manages VMs), 13:22-30. Khandekar's meaning is clear from its context. When Khandekar refers to the stored files of a VM, such as Figures 3 and 4 or with stored VMs on the provisioning server, Khandekar is referring to what is often described in the art as a VM image.

5. For example, Khandekar Figures 3 and 4 show stored VM images that contain a virtual disk image along with configuration files. *See* Ex-1004, FIGS. 3-4. Beyond storage, Khandekar also teaches execution of virtual machines, including using VMM software to run a VM with virtual hardware, such as virtual processor, virtual memory, and virtual devices. *See* Ex-1004, FIG. 1. This is consistent with how VMware software functioned, including how VMware software included functionality to create VM images (consistent with Khandekar's pre-built, custom-built, and instantly-deployable VM images) and to run them. A POSITA would have recognized that Khandekar's teachings of pre-built, custom-built, and instantly-deployable VM images relate to stored images, despite Khandekar's use of shorthand to sometimes refer to these based on the type of VM

stored in the image. This would have been clear from the context of Khandekar's teachings, as explained above. *See also* Ex-1003 ¶¶122-139.

6. I further note that Khandekar teaches a process for creating VM images that is analogous to the steps disclosed in the '138 patent specification. For example, Khandekar teaches creating a VM, installing and configuring a guest OS and applications inside the VM, and then save an image of the VM including the virtual disk image file. *See, e.g.*, Ex-1004, 15:40-50, 16:43-52, 17:30-50, FIGS 3, 4. Similarly, embodiments in the '138 patent, include creating a VM, installing and configuring the OS and applications, and then taking a snapshot of the VM including the virtual disk image file. Ex-1001, 33:55-67, 35:1-24. Both Khandekar and the '138 patent use VMware software to create these images. *See* Ex-1004, 13:17-35; Ex-1001, 32:59-64, 33:25-41.

II. Claim Construction

7. Dr. Rosenblum argues that a three-stage process from the embodiments is required by the challenged claims, including execution of VMM images at application nodes. *See* Ex-2001 ¶68. This is incorrect. A POSITA would have understood that these portions of the specification describe examples from embodiments. During my deposition, I was asked to describe the deployment process described in the specification. *See* Ex-2001 ¶69. I explained that as described in the exemplary embodiments in the specification, the deployment

process includes the three steps. I did not testify as to whether the claim 1 requires PO's three-stage process or any other steps described in the specification. It is my understanding that it is improper to import features from embodiments into the claims, particularly when, as here, the claim language is clear.

8. Limitation [1d] requires that "when executed" the VMM images provide a plurality of VMs. A POSITA would understand this to describe the functionality and capability of the VMM images *when* they are executed, as opposed to a requirement for VMM images to actually be executing. Similarly, Limitation [1e] requires image instances of software applications (VM image) to be "executable"—but actual execution is not required.

9. Limitations [1f] and [1g] recite "a control node that comprises an automation infrastructure to provide autonomic deployment"—referring to a capability rather than actual deployment. Specifically, these two limitations define how deployment is provided: autonomic deployment is provided "by causing the plurality of image instances … to be copied from the software image repository to the application nodes." There is no mention of executing any of the images. Therefore, based on the structure and language of claim 1—and similar language in claims 19 and 26—a POSITA would have understood that the challenged claims were carefully and deliberately drafted to avoid limitations that require actions to be taken by the application nodes, such as execution of images.

10. Dr. Rosenblum argued that the logic and grammar of claim 1 requires the VMM images to be executed to provide the VMs before deployment of the image instance of the software applications "**on the virtual machines**" could occur. Ex-2001 ¶¶82, 83 (emphasis in original). Dr. Rosenblum argues that the phrase "***the*** plurality of virtual machines" refers to specific virtual machines that exist at the application nodes and are provided by VMM software that is executing on the application nodes. This is incorrect. As evident from the plain language of claim 1, "the plurality of virtual machines" simply refers to the VMs that the VMM images are *capable* of creating. Claim 1 recites a "software image repository" that stores (i) VMM images and (ii) software application images. The VMM images have the capability, when executed, to "provide a plurality of virtual machines." The repository also stores "image instances of a plurality of software applications that are executable on ***the*** plurality of virtual machines," which a POSITA would have understood from the context to refer to the VMs that the VMM images are capable of creating.

11. Furthermore, Dr. Rosenblum's argument about "the" plurality of virtual machines contradicts claim 1 by creating a paradox out of its requirements. Claim 1 recites an image repository that stores "image instances of a plurality of software applications that are executable on ***the*** plurality of virtual machines." If "***the*** plurality of virtual machines" referred to specific VMs at the application

nodes, then the recited software application images cannot exist and cannot be stored in a repository until after VMM images have been deployed and executed on application nodes.[1] But claim 1 also requires the control node to "provide autonomic deployment" of the VMM images by causing them "to be copied ***from the software image repository*** to the application nodes." Dr. Rosenblum's arguments create self-contradictions within claim 1. A POSITA would, therefore, not have viewed the claims in that manner.

12. For the reasons explained above, a POSITA would not have understood the challenged claims to require PO's three-stage process.

III. Ground 1 Satisfies Patent Owner's Proposed Construction

13. Even if PO's proposed construction were adopted, Khandekar teaches or suggests PO's three-stage process, including through its teachings of deploying and executing VMM images, and deploying and executing VM images on remote host computers. This was explained in my opening declaration and further detailed below.

14. As discussed in my first declaration, a POSITA would have found it obvious, based on Khandekar's teachings, to deploy VMM images to hosts and use

[1] Contrary to Dr. Rosenblum's argument, the '138 patent specification states that images in the software image repository are created before deployment, and those software images are "bootable on the virtual machines" before deployment—meaning they are executable on the VMs before any images have been deployed. Ex-1001, 33:55-34:5.

VMMs to execute VM images on the hosts. *See e.g.*, Ex-1004, 8:36-44, 13:17-35, 4:66-5:8, Fig. 1; Ex-1003 ¶¶143-145. Therefore, Khandekar teaches or suggests executing VMM software, including VMware software, at the hosts for the purpose of executing VM images. At the very least, a POSITA would have found it obvious, from Khandekar's teachings, to execute VMM images at the hosts because Khandekar teaches executing VM images at the hosts, as explained below, and Figure 1 teaches executing VMMs to run VM images. *See* Ex-1004, Fig. 1.

15. Khandekar teaches deploying VM images on hosts by copying the VM image to the host. *See e.g.*, Ex-1003 ¶¶156-162. Khandekar further teaches booting the image. Ex-1004, 19:33-45 ("When the provisioning engine 810 deploys a VM on a host, such as host 1000, ***the VM is powered on and its guest OS is booted.***"), 13:36-40 ("Once a VM is deployed on a host, it may be started ('powered on') in any conventional manner, either by the provisioning server remotely, or by the user himself. Note that powering on a VM does not normally involve a manual action, but rather can be carried out through a software procedure."). Powering on the VM, also referred to in the art as booting the VM, causes the VM image to be booted on a virtual machine. For example, to deploy a custom-built VM to a host, Khandekar teaches copying VM image files to the host and then powering it on. Ex-1004, 21:25-64:

…

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

…

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

…

16. I note that a POSITA would have found it obvious to deploy custom-built and pre-built VM images in a similar fashion, including with respect to copying and booting the images, because both are staged in the VM staging repository 830 using the same process. Ex-1003 ¶¶126, 132; Ex-1004, 15:29-17:29, 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."). While custom-built VMs may additionally have other applications installed, a POSITA would have found it obvious to apply other aspects of deployment in a similar manner for both. *See* Ex-1003 ¶156; Ex-1004, 23:28-51.

17. Khandekar also teaches deploying an instantly-deployable VM image by copying it to the host and resuming the VM. Ex. 1004, 21:65-22:13:

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

…

> 4) Copy the VM's config file 842, REDO logs 843 for the disks 841 and suspend-to-disk image file 844 to the host.
>
> …
>
> 7) Call the API in the host ***to resume the VM***.
>
> …

18. As discussed in my first declaration, a POSITA would be motivated to automatically and remotely provision and start the VMs because it increases automation, and as Khandekar teaches, would be beneficial for testing environments and running a large number of different VMs. *See* Ex-1003 ¶146; Ex-1004, 22:53-67:

> This arrangement would be beneficial in the context of testing, in which a very large number VMs having different hardware/OS/application combinations need to be provisioned. The external computer could then be programmed to request all the needed combinations, ***run the VMs***, and log the results. Using the invention, such testing of many configuration combinations could therefore be made substantially wholly automatic.

19. In summary, a POSITA would have found it obvious from Khandekar's teachings to deploy VMM images to hosts, execute the VMM images, and then deploy and execute VMs on hosts.

IV. Ground 2 teaches deploying and executing VMM images, and deploying and executing VM images

20. My original declaration explained how the combination of Khandekar and Le (Ground 2) teaches deploying the *physical* image of a host computer (including the VMM) using Le's teachings, booting the host computer, and then deploying the *virtual* image of a VM to the host using Khandekar's teachings.

21. Ground 2 uses Le's teachings to initialize or reimage physical host computers. Ex-1003 ¶¶149-153. Le teaches techniques for writing a physical image onto the host's hard drive, which included the host OS. Ex-1005, 54:14-21, 10:55-62, 2:19-21, Fig. 5; Ex-1003 ¶149. The host OS is the VMM for Type 1 hypervisors such as VMware ESX. Ex-1003 ¶58; Ex-1005, 13:12-21, 66:2-10. Le teaches that during the imaging process, the destination disk is wiped clean, leaving only the physical image, and the computer reboots to load the new OS from the physical image. *See* Ex-1003 ¶151; Ex-1005, 27:1-9:

> Next, the imaging server leverages the server operating system's APIs and utilities to partition, then format, the destination disk 2230, ***destroying whatever partitions, file systems, data, etc., that was previously present on it***. Finally, the imaging server copies files and directories from the source file system to the destination file system, dismounts the two simulated disks 2210, 2230, and ***reboots*** the destination computer 1500 to allow it to ***load the new operating system*** deployed from the image 2015.

See also Ex-1005, 66:53-64 ("[A] physical computer typically reboots and becomes operational once it has been deployed from an image."). For the combination of Khandekar and Le, these teachings are used to deploy a hypervisor OS (the VMM). Ex-1003 ¶151. Therefore, the combination of Khandekar and Le teaches deploying a VMM image to a node and then executing the image (and VMM) at the node.

22. After deploying the physical VMM image, Ground 2 uses Khandekar's teachings to deploy virtual images of VMs, including virtual disk image files. Ex-1003 ¶¶168, 155-163. The combination uses separate images for VMMs and VMs because VMM images are images of *physical* disks while VM images include images of *virtual* disks. Ex-1003 ¶¶58, 151; Ex-1004, 4:66-5:8, 7:7-38, 13:17-35; Ex-1005, 26:52-27:9, 44:4-17, 66:33-47. In other words, the VMM image is an image of a physical hard drive, and the combination of Ground 2 used it to, for example, initialize a new host that is connected to the network or re-image a host that is not functioning properly. Ex-1003 ¶58, ¶¶149-153. As was typical in the art, Khandekar teaches that each physical host runs many virtual machines concurrently. *See* Ex-1004, Fig. 1. The VM image is that of a virtual machine and includes an image of a virtual disk. *See id.*

23. VM images would obviously have been deployed to hosts as a separate, subsequent step after deployment of the physical VMM image because,

as discussed above, the physical imaging process overwrites the entire hard drive, leaving only the hypervisor OS. *See also* Ex-1005, 27:1-9. Khandekar teaches copying and then executing VM images on hosts. *See supra* §III.

24. Le's UCMS embodiment further teaches PO's three-stage deployment process. My opening declaration explained how Le teaches that VMMs "are ***executed on the host computers***, where the image instances of the each virtual machine monitor and virtual machine manager provide a plurality of virtual machines, with each of the plurality of virtual machine operable to provide an environment that emulates a computer platform." Ex-1003 ¶119 (citing general teachings and UCMS embodiment). I further explained the motivation to apply Le's UCMS teachings to Khandekar's provisioning server. Ex-1003 ¶63.

25. Le teaches that VMMs are running and managing VMs at hosts. For example, Le teaches the server deploying VM images (e.g., template image 4020) to virtual machines. Ex-1005, 44:4-17 (describing deployment steps), 66:33-47. "Once a virtual machine is successfully deployed from a template image 4020, ***the virtual machine manager 6200 may optionally power it on***" or wait to power it on at a later time. Ex-1005, 66:53-64; Ex-1003 ¶96. Le includes numerous teachings where VMs are powered on. *See* Ex-1005 65:34-50, FIG.8 (illustrating a typical VM product with a VM being "powered-on"). "***When a virtual machine is powered on, a virtual machine monitor program 6300 controls it and manages***

the interactions …" Ex-1005, 65:65-66:10; *see also* Ex. 1005, 66:33-47 (deploying a template image as part of the UCMS embodiment).

26. Le teaches deploying a template image 4020 to an existing virtual machine with the option of the virtual machine manager powering the virtual machine on, or leaving it powered off and powering it on at a later time. Ex. 1003 ¶96; Ex. 1005, 66:53-64:

> Once a virtual machine is successfully deployed from a template image 4020, ***the virtual machine manager 6200 may optionally power it on*** (such as virtual machine 6010); this would mimic the results of a deployment to a physical computer, since ***a physical computer typically reboots and becomes operational once it has been deployed from an image***. The imaging server, or the user that requested the deploy-to-virtual machine operation, may also choose to have the destination virtual machine remain powered off (such as virtual machine 6020), with the option of powering it on at a later time using the virtual machine manager's fourth interface function.

27. A POSITA would be motivated to follow Le's teaching of powering on the VMs for several reasons. First, it is one of two options, and Le includes numerous teachings where VMs are powered on. *See* Ex-1005, 65:34-50, FIG.8 (illustrating a typical VM product with a VM being "powered-on"). "***When a virtual machine is powered on, a virtual machine monitor program 6300 controls it and manages the interactions*** between the software—commonly called 'guest

software'—running inside of the virtual machine and the host's physical resources, such as hardware devices." Ex-1005, 65:65-66:10. Additionally, as discussed above for Khandekar and in my opening declaration, a POSITA would be motivated to automatically and remotely provision and start the VMs because it increases automation, and would be beneficial for testing environments and running a large number of different VMs. Ex-1003 ¶146; Ex-1004 22:53-67.

28. In short, the combination of Khandekar and Le teaches PO's three-stage deployment process, including the execution of VMM images at host computers.

V. Khandekar teaches separate VMM and VM images

29. As discussed in my first declaration, a virtual machine ("VM") is a software abstraction (virtualization) of a physical machine such that software could run inside a VM without ever knowing that it was running inside a virtualized computer platform. Ex-1003 ¶11; Ex-1005, 11:42-54. VMs are easily relocatable and replicable to other machines and systems, which made them well-suited for the automated deployment and scaling of distributed computing systems. Ex-1003 ¶¶11-13; Ex-1004, 8:13-20.

30. The virtual machine manager/monitor ("VMM"), also referred to as a "hypervisor," is a piece of software that runs on a host machine and virtualizes the

resources of the machine for VMs. Ex-1003 ¶12; Ex. 1004, 7:7-27, 4:66-5:8, 13:17-35; Ex. 1005, 12:33-44.

31. Khandekar uses virtualization to allow the same VM image to run on a software abstraction—virtualized hardware—so that "[d]eployment of the VM is thereby made *substantially independent of the overall physical hardware configuration*." Ex-1004, 4:66-5:8, 13:17-35 ("[A]s long as the VMM on each host exports a known hardware interface to the VM deployed on it, then VM deployment will be substantially independent of the actual physical hardware configuration."), 7:7-27 ("Regardless of which level the VMM is implemented on, the interface exported to the respective VM is the same as the hardware interface of the machine, or at least of some predefined hardware platform, so that the guest OS 320 usually cannot determine the presence of the VMM."). Dr. Rosenblum agrees that a VM is an abstraction layer that provides independence from underlying physical hardware. Ex-1024, 22:18-23:2, 29:2-30:6, 44:12-45:2.

32. Therefore, using separate images for the VMM and VM is consistent with Khandekar's goals and teachings because it facilitates independence between the VM and underlying physical hardware, while a monolithic image (with both a VMM and VM) undermines Khandekar's goals by creating dependencies to physical hardware that otherwise would not exist. Having a combined image with both a VMM and VM would undermine the independence created by virtualization

because the VMM image is a physical image for a physical host computer. Khandekar does not teach or require a merged image, since merging them would tie the VM image to a physical computer, defeating the purpose of virtualization.

33. Dr. Rosenberg argues that Khandekar requires "including the VMM *within the same image* as the VM." Ex-2001 ¶139. This is incorrect, because the VMM image is a physical image while the VM image is a virtual image. It would not make sense for the VMM to be included within the same image as the VM, and it is contrary to Khandekar's teachings. For example, Khandekar teaches that a VMM may be already installed on the host or installed together with the VM. *See* Ex-1004, 13:17-35 ("An advantage of pre-installing a VMM, without a VM, on a plurality of hosts … is that it will then be possible to deploy a given VM on any arbitrary one (or more) of these hosts."). The fact that there is a choice to install these together or separately—including installation of a VMM "without a VM"—teaches or suggests that VMMs and VMs are separate images. Khandekar also teaches that one VMM may support multiple VMs (Ex-1004, 7:28-38), from which a POSITA would have found it obvious that the VMM image is separate from the multiple VM images.

34. Khandekar's use of separate images is consistent with the teachings of the prior art, where VMM images were stored separately from VMs. Ex-1003 ¶60; Ex-1014, ¶[0062]; Ex-1016, ¶¶[0043], [0049]. This is how VM images had been

used for years, including for VMware software, which had specific features for creating VM images.

35. I have a significant amount of experience in distributed computing and virtualization. Indeed, one of my papers is cited as the first non-patent publication on the face of the '138 patent. Ex-1001, 000002. In my experience, a POSITA reading Khandekar would have found it obvious, based on Khandekar's teachings, to use separate images for VMM and VMM, for the reasons explained above. This was how imaging was commonly used in the art because the central purpose for using VMs was to allow flexibility in running them on different machines.

36. Khandekar does not disclose a monolithic image. Dr. Rosenberg argues that "the included VMM *depends directly* on the VM." Ex-2001 ¶139. However, what Khandekar actually states is that the VMM must account for "the characteristics ***of the host*** on which the VM is to be installed[.]" Ex-1004, 13:25-28. As I explained above, the VMM image is a physical image and therefore must match the physical hardware of the ***host***, not the VM. That is why it would have been obvious to store a plurality of VMM images to account for the different host hardware. Ex-1003 ¶¶83-84.

VI. Conclusion

37. I declare that all statements made herein of my knowledge are true, that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed on September 28, 2021.

By:

Prashant Shenoy

APPENDIX 1: MATERIALS CONSIDERED IN THE PREPARATION OF THIS DECLARATION

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")
1024	Deposition Transcript of David Rosenblum (August 6, 2021)
2001	Declaration of David Rosenblum