

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ORACLE CORPORATION,
Petitioner

v.

VIRTAMOVE, CORP.,
Patent Owner

Case No.: IPR2025-01002
U.S. Patent No. 7,519,814
Issue Date: April 14, 2009

Title: SYSTEM FOR CONTAINERIZATION OF APPLICATION SETS

**PETITION FOR *INTER PARTES* REVIEW
OF CLAIMS 3, 5, 7, 11-12, 15-16, AND 31-34
OF U.S. PATENT NO. 7,519,814**

TABLE OF CONTENTS

	Page
INTRODUCTION	1
I. BACKGROUND	2
A. Containers Versus Virtual Machines	2
B. Containers Versus Shared Application Environment	3
C. Containers in the Prior Art	3
1. Linux VServer (Gélinas).....	4
2. Solaris Zones (Tucker).....	5
3. Zap Pods (Osman).....	6
II. THE '814 PATENT	7
A. Overview	7
B. Prosecution History	8
III. STATEMENT OF RELIEF REQUESTED	9
A. Grounds	9
B. The References Are Prior Art.....	10
1. The Patent's Filing Date	10
2. Osman	12
3. Tucker	13
4. Bandhole	16
5. Gélinas.....	16

TABLE OF CONTENTS
(continued)

	Page
IV. LEVEL OF ORDINARY SKILL	17
V. CLAIM CONSTRUCTION	18
A. “Container” and “System Files”	18
B. “Disparate Computing Environments”	18
VI. GROUNDS OF UNPATENTABILITY.....	20
A. Ground 1: Claims 3, 5, 7, 11-12, 15, and 31-34 are Unpatentable as Obvious in View of Osman.....	20
1. Claim 3.....	20
a. Limitation 1[pre][i]: “In a system having a plurality of servers”	20
b. Limitation 1[pre][ii]: “with operating systems that differ”	21
c. Limitation 1[pre][iii]: “operating in disparate computing environments”.....	21
d. Limitation 1[pre][iv]: “wherein each server includes a processor and an operating system including a kernel”.....	22
e. Limitation 1[pre][v]: “a set of associated local system files compatible with the processor”	22
f. Limitation 1[pre][vi]: “a method of providing at least some of the servers in the system with secure, executable applications related to a service”	23
g. Limitation 1[pre][vii]: “wherein the applications are executed in a secure environment”	23

TABLE OF CONTENTS
(continued)

	Page
h. Limitation 1[pre][viii]: “wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service”	24
i. Limitation 1[a][i] “the method comprising: storing in memory accessible to at least some of the servers a plurality of secure containers of application software”	24
j. Limitation 1[a][ii]: “each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications”	26
k. Limitation 1[a][iii]: “for use with a local kernel residing permanently on one of the servers”	27
l. Limitation 1[a][iv]: “wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems”	27
m. Limitation 1[a][v]: “the containers of application software excluding a kernel”	28
n. Limitation 1[a][vi]: “wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server”	28
o. Limitation 1[a][vii]: “wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server”	29

TABLE OF CONTENTS
(continued)

	Page
p. Limitation 1[a][viii]: “wherein the application software cannot be shared between the plurality of secure containers of application software”	29
q. Limitation 1[a][ix]: “wherein each of the containers has a unique root file system that is different from an operating system’s root file system”	30
r. Limitation 2: “wherein each container has an execution file associated therewith for starting the one or more applications”	30
s. Limitation 3: “wherein the execution file includes instructions related to an order in which executable applications within will be executed.”	30
2. Claim 5	31
3. Claim 7	31
4. Claim 11	32
5. Claim 12	32
6. Claim 15	33
a. Limitation 14[a][i]: “creating containers prior to said step of storing containers in memory, wherein containers are created by:”	33
b. Limitation 14[a][ii]: “a) running an instance of a service on a server”	33
c. Limitation 14[a][iii]: “b) determining which files are being used”	34

TABLE OF CONTENTS
(continued)

	Page
d. Limitation 14[a][iv]: “c) copying applications and associated system files to memory without overwriting the associated system files so as to provide a second instance of the applications and associated system files”	34
e. Limitation 15[a]: “assigning an identity to the containers including at least one of a unique IP address, a unique Mac address and an estimated resource allocation”	34
f. Limitation 15[b]: “installing the container on a server”	34
g. Limitation 15[c]: “testing the applications and files within the container”	35
7. Claim 31	35
a. Limitation 31[pre]: “A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:”	35
b. Limitation 31[a][i]: “a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers”	35
c. Limitation 31[a][ii]: “each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers”	36
d. Limitation 31[a][iii]: “each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address”	36

TABLE OF CONTENTS
(continued)

	Page
e. Limitation 31[a][iv]: “wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes”	36
f. Limitation 31[a][v]: “and associated system files for use in executing the one or more processes”	37
g. Limitation 31[a][vi]: “wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system”	37
h. Limitation 31[a][vii]: “each container having its own execution file associated therewith for starting one or more applications”	37
i. Limitation 31[a][viii]: “in operation, each container utilizing a kernel resident on the server”	37
j. Limitation 31[a][ix]: “wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel”	38
k. Limitation 31[b]: “a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications”	38
8. Claim 32	38
9. Claim 33	39

TABLE OF CONTENTS
(continued)

	Page
a. Limitation 33[a][i]: “the run time module includes an intercepting module associated with the plurality of containers for intercepting system calls from any of the plurality of containers”	39
b. Limitation 33[a][ii]: “and for providing values alternate to values the kernel would have assigned in response to the system calls”	39
c. Limitation 33[a][iii]: “so that the containers can run independently of one another without contention, in a secure manner”	39
d. Limitation 33[a][iv]: “the values corresponding to at least one of the IP address, the host name and the Mac_Address.”	40
10. Claim 34.....	40
a. Limitation 34[a]: “monitoring resource usage of applications executing;”	40
b. Limitation 34[b]: “intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;”.....	41
c. Limitation 34[c]: “comparing the monitored resource usage of the at least one respective application with the resource limits; and,”	41
d. Limitation 34[d]: “forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.”	41
B. Ground 2: Claim 16 is Unpatentable as Obvious in View of Tucker and Bandhole.....	42

TABLE OF CONTENTS
(continued)

	Page
1. Claim 16.....	44
a. Limitation 1[pre][i]: “In a system having a plurality of servers”	44
b. Limitation 1[pre][ii]: “with operating systems that differ”	45
c. Limitation 1[pre][iii]: “operating in disparate computing environments”	45
d. Limitation 1[pre][iv]: “wherein each server includes a processor and an operating system including a kernel”	46
e. Limitation 1[pre][v]: “a set of associated local system files compatible with the processor”	46
f. Limitation 1[pre][vi]: “a method of providing at least some of the servers in the system with secure, executable applications related to a service”	47
g. Limitation 1[pre][vii]: “wherein the applications are executed in a secure environment”	48
h. Limitation 1[pre][viii]: “wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service”	48
i. Limitation 1[a][i] “the method comprising: storing in memory accessible to at least some of the servers a plurality of secure containers of application software”	48

TABLE OF CONTENTS
(continued)

	Page
j. Limitation 1[a][ii]: “each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications”	49
k. Limitation 1[a][iii]: “for use with a local kernel residing permanently on one of the servers”	50
l. Limitation 1[a][iv]: “wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems”	50
m. Limitation 1[a][v]: “the containers of application software excluding a kernel”	50
n. Limitation 1[a][vi]: “wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server”	51
o. Limitation 1[a][vii]: “wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server”	52
p. Limitation 1[a][viii]: “wherein the application software cannot be shared between the plurality of secure containers of application software”	53
q. Limitation 1[a][ix]: “wherein each of the containers has a unique root file system that is different from an operating system’s root file system”	53

TABLE OF CONTENTS
(continued)

	Page
r. Limitation 16: “creating containers prior to said step of storing containers in memory, wherein a step of creating containers includes: using a skeleton set of system files as a container starting point and installing applications into that set of files.”	53
C. Ground 3: Claim 16 is Unpatentable as Obvious in View of Gélinas.....	54
1. Claim 16.....	55
a. Limitation 1[pre][i]: “In a system having a plurality of servers”	55
b. Limitation 1[pre][ii]: “with operating systems that differ”	55
c. Limitation 1[pre][iii]: “operating in disparate computing environments”	55
d. Limitation 1[pre][iv]: “wherein each server includes a processor and an operating system including a kernel”	57
e. Limitation 1[pre][v]: “a set of associated local system files compatible with the processor”	57
f. Limitation 1[pre][vi]: “a method of providing at least some of the servers in the system with secure, executable applications related to a service”	58
g. Limitation 1[pre][vii]: “wherein the applications are executed in a secure environment”	58

TABLE OF CONTENTS
(continued)

	Page
h. Limitation 1[pre][viii]: “wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service”	58
i. Limitation 1[a][i] “the method comprising: storing in memory accessible to at least some of the servers a plurality of secure containers of application software”	58
j. Limitation 1[a][ii]: “each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications”	59
k. Limitation 1[a][iii]: “for use with a local kernel residing permanently on one of the servers”	59
l. Limitation 1[a][iv]: “wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems”	60
m. Limitation 1[a][v]: “the containers of application software excluding a kernel”	60
n. Limitation 1[a][vi]: “wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server”	61
o. Limitation 1[a][vii]: “wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server”	61

TABLE OF CONTENTS
(continued)

	Page
p. Limitation 1[a][viii]: “wherein the application software cannot be shared between the plurality of secure containers of application software”	62
q. Limitation 1[a][ix]: “wherein each of the containers has a unique root file system that is different from an operating system’s root file system”	62
r. Limitation 16: “creating containers prior to said step of storing containers in memory, wherein a step of creating containers includes: using a skeleton set of system files as a container starting point and installing applications into that set of files.....	63
VII. SECONDARY CONSIDERATIONS OF NONOBVIOUSNESS	63
VIII. DISCRETIONARY DENIAL IS UNWARRANTED	64
IX. MANDATORY NOTICES	65
X. CONCLUSION.....	70

TABLE OF AUTHORITIES

	Page(s)
Cases	
<i>Amazon.com, Inc. v. CustomPlay, LLC</i> , IPR2018-01496, Paper 34 (P.T.A.B. Mar. 4, 2020)	13
<i>Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.</i> , 800 F.3d 1375 (Fed. Cir. 2015)	10, 13
<i>In re GPAC Inc.</i> , 57 F.3d 1573 (Fed. Cir. 1995)	17
<i>Halliburton Energy Servs., Inc. v. Dynamic 3D Geosolutions LLC</i> , IPR 2014-01186, 2015 WL 5565065 (PTAB Dec. 15, 2015)	17
<i>Hulu, LLC v. Sound View Innovations, LLC</i> , IPR2018-01039, Paper 29 (P.T.A.B. Dec. 20, 2019)	17
<i>Intel Corp. v. Qualcomm Inc.</i> , 21 F.4th 801 (Fed. Cir. 2021)	20
<i>Keysight Techs., Inc. v. Centripetal Networks, LLC</i> , IPR2022-01525, Paper 27 (P.T.A.B. Apr. 15, 2024)	17
<i>Kolcraft Enters., Inc. v. Graco Children’s Prods., Inc.</i> , 927 F.3d 1320 (Fed. Cir. 2019)	12
<i>Leap-frog Enters. v. Fisher-Price, Inc.</i> , 485 F.3d 1157 (Fed. Cir. 2007)	64
<i>Medtronic, Inc. v. Teleflex Innovations S.A.R.L.</i> , 68 F.4th 1298 (Fed. Cir. 2023)	12
<i>Microsoft Corp. v. VirtaMove, Corp.</i> , IPR2025-00852, Paper 3 (PTAB Apr. 18, 2025)	64
<i>New Railhead Mfg., LLC v. Vermeer Mfg. Co.</i> , 298 F.3d 1290 (Fed. Cir. 2002)	10

<i>Newell Cos. v. Kenney Mfg. Co.</i> , 864 F.2d 757 (Fed. Cir. 1988)	63
<i>Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co. Ltd.</i> , 868 F.3d 1013 (Fed. Cir. 2017)	18
<i>VirtaMove, Corp. v. Amazon.com, Inc.</i> , No. 7:24-cv-00030	18, 19
<i>Visa, Inc. v. Cortex MCP, Inc.</i> , IPR2024-00487, Paper 8 (PTAB Aug. 2, 2024).....	65
<i>Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.</i> , 200 F.3d 795 (Fed. Cir. 1999)	18
<i>Voter Verified, Inc. v. Premier Election Sols., Inc.</i> , 698 F.3d 1374 (Fed. Cir. 2012)	17

Statutes

35 U.S.C. §102	12
35 U.S.C. §102(a)	12, 16
35 U.S.C. §102(b)	12, 16
35 U.S.C. §102(e)	13, 16
35 U.S.C. §103	9
35 U.S.C. § 325(d)	65

Other Authorities

37 C.F.R. §42.10(b)	72
37 C.F.R. §42.15(a).....	72
37 C.F.R. §42.103	72
37 C.F.R. §42.104(a).....	72
37 C.F.R. § 42.122	65

77 Fed. Reg. 4875966

TABLE OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 7,519,814 (“the ’814 patent”)
1002	Declaration of Dr. Darrell Long, Ph.D.
1003	Steven Osman et al., <i>The Design and Implementation of Zap: A System for Migrating Computing Environments</i> , 5 Proc. of the Symposium on Operating Systems Design and Implementation (2002) (“Osman”)
1004	U.S. Patent No. 7,437,556 (“Tucker”)
1005	U.S. Provisional Patent Application No. 60/469,558 (“Tucker Provisional”)
1006	U.S. Patent Publication No. 2002/0171678A1 (“Bandhole”)
1007	<i>Virtual Private Servers and Security Contexts</i> (“Gélinas”)
1008	File history of the ’814 patent
1009	Solaris 9 press release from Sun Microsystems
1010	B. Walters, “VmWare Virtual Platform.” Linux Journal, 1999.
1011	Stephen Soltesz et al, <i>Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors</i> (2007)
1012	D. Price and A. Tucker. <i>Solaris zones: Operating system support for consolidating commercial workloads</i> . In Proceedings of the 18th Usenix LISA Conference, 2004.
1013	U.S. Provisional Patent Application No. 60/502,619
1014	U.S. Provisional Patent Application No. 60/512,103
1015	Declaration of Rachel Watters Regarding Osman
1016	Declaration of Jacques Gélinas Regarding Linux VServer

Exhibit No.	Description
1017	Message to Linux Kernel Mailing List Regarding Linux VServer
1018	Slashdot post Regarding Linux VServer
1019	Amazon’s Opening Claim Construction Brief in <i>VirtaMove, Corp. v. Amazon.com, Inc. et al.</i> , No. 7:24-cv-30-ADA-DTG (W.D. Tex.) (the “Amazon Litigation”)
1020	Patent Owner’s Sur-Reply Claim Construction Brief from the Amazon Litigation
1021	Excerpts from deposition of named inventor Donn Rochette from the Amazon Litigation
1022	J. Ball, “Managing Initscripts with Red Hat’s chkconfig.” <i>Linux Journal</i> , 2001.
1023	Kravetz, et al. “Enhancing Linux scheduler scalability.” <i>Proceedings of the Ottawa Linux Symposium, Ottawa, CA. 2001.</i>
1024	Scheduling order from the Amazon Litigation
1025	Order cancelling <i>Markman</i> hearing in the Amazon Litigation
1026	Order granting transfer in the Amazon Litigation
1027	<i>Curriculum vitae</i> of Dr. Darrell Long
1028	Order Overruling Objections and Transferring Case to Northern District of California, in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (May 7, 2025)
1029	Plaintiff VirtaMove Corp.’s Preliminary Disclosure of Asserted Claims and Infringement Contentions, in <i>VirtaMove, Corp. v. Oracle Corp.</i> , 7:24-cv-00339 (W.D. Tex.) (Mar. 28, 2025)

Petitioner Oracle Corporation (“Petitioner” or “Oracle”) requests *inter partes* review of claims 3, 5, 7, 11-12, 15-16, and 31-34 (the “challenged claims”) of U.S. Patent No. 7,519,814 (“the ’814 patent”), which VirtaMove, Corp. (“Patent Owner” or “PO”) purportedly owns.

INTRODUCTION

The challenged claims recite methods for running software applications in “containers.” A container is a set of files needed to execute an application on a computer. The files in a container are grouped together and isolated from other files and applications on the same computer. Containers prevent different applications on the same computer from interfering with each other.

A host of prior art container technologies—Solaris zones, Zap pods, Linux VServers, and more—provide the same functionality described in the challenged claims. All of these container technologies (and several others) were available and well known before the ’814 patent’s earliest claimed priority date in September 2003. Yet the patent fails to acknowledge these earlier container technologies.

The Examiner was aware of at least one of these technologies—VServer—and expressly recognized its relevance to the patent claims. But the only reference the Examiner cited concerning VServer published in 2007 and thus was not prior art. This 2007 publication omits aspects of VServer that are material to the patent claims.

The Examiner never considered the 2002 VServer reference raised in this Petition, which is prior art. Nor did the Examiner review the prior art references that this Petition relies on concerning Zap pods and Solaris zones. Each of these references discloses the elements that were missing from the Examiner's prior art and each of them renders the challenged claims unpatentable.

The '814 patent claims exclusive rights to container technology that belongs in the public domain. Because the patent contributed nothing to the art, PO is not entitled to exclude the public from practicing the challenged claims. Thus, the Board should cancel the claims.

I. BACKGROUND

A. Containers Versus Virtual Machines

Instead of addressing the many container systems in the prior art, the '814 patent frames its contribution as an advance over “Virtual Machine technology, pioneered by VmWare” and released commercially in 1999. (Ex. 1001, 1:51-56; Ex. 1010.) Like containers, multiple virtual machines can be hosted on a single physical computer and each one can be customized to meet the unique needs of the applications it contains. (Ex. 1001, 1:27-56.) But the patent identifies a “key difference” between the Virtual Machine (“VM”) approach and the patent's container-based approach. (Ex. 1001, 1:56-61.) While VMs require a copy of the operating system “for each application,” the container approach required only one

copy of the operating system (OS) “regardless of the number of application containers deployed.” (*Id.*) Because containers do not require multiple copies of the OS, they avoid the “performance overhead” associated with VMs. (*Id.*, 1:62-63.)

The claims of the ’814 patent capture this distinction over VMs by specifying that containers do not contain their own “kernel” (which is the core of an OS). (*E.g.*, *id.*, 17:41-50 (claim 1); *see also id.*, 2:39-42 (containers share a kernel from the underlying OS).) However, the patent’s distinction does not apply to the prior-art systems presented in this Petition—all of which used containers rather than VMs.

B. Containers Versus Shared Application Environment

The ’814 patent acknowledged that, outside of VMs, multiple applications could share an operating system in a prior-art environment called “SoftGrid.” (*Id.*, 2:4-12.) The patent distinguished SoftGrid in that it did “not isolate applications into distinct environments.” (*Id.*) In contrast, containers isolate applications to prevent them from interfering with each other. (*Id.*, 4:39-42.)

C. Containers in the Prior Art

The ’814 patent fails to distinguish the many prior-art container systems that provided the same capabilities the patent describes—including the isolation and kernel sharing that the patent relies on to distinguish other prior art.

1. Linux VServer (Gélinas)

In October 2001, Jacques Gélinas disclosed a system for Linux computers that would allow “several independant [sic] virtual servers running on the same box (*sharing the same kernel as well*).” (Ex. 1017 (emphasis added).) This system became known as Linux VServer. (Ex. 1002 ¶38.) By 2002, Gélinas’s website described that VServer “split a Linux server into virtual ones with as much *isolation* as possible between each one, looking like real servers, yet sharing some common tasks.” (Ex. 1007, 1 (emphasis added).)

To provide isolation, VServer built on a well-known computer command called “chroot,” which had been part of conventional operating systems for decades. (Ex. 1002 ¶39.) The chroot command confined a particular application process to a limited view of the computer’s file system, locking it out of other areas of the computer. (*Id.*) VServer built on chroot to also prevent processes from communicating with each other and isolate them from network resources. (Ex. 1007, 2-3; Ex. 1002 ¶39.) Thus, VServer created secure containers that isolate and confine processes while allowing them to share the kernel of the underlying server. (Ex. 1002 ¶39.)

Although the Examiner cited a 2007 article that mentioned VServer (Ex. 1008, 15), the Examiner never considered the 2002 publication on which this Petition relies. (Ex. 1001, 1-2.) The 2007 article omits key features of VServer that

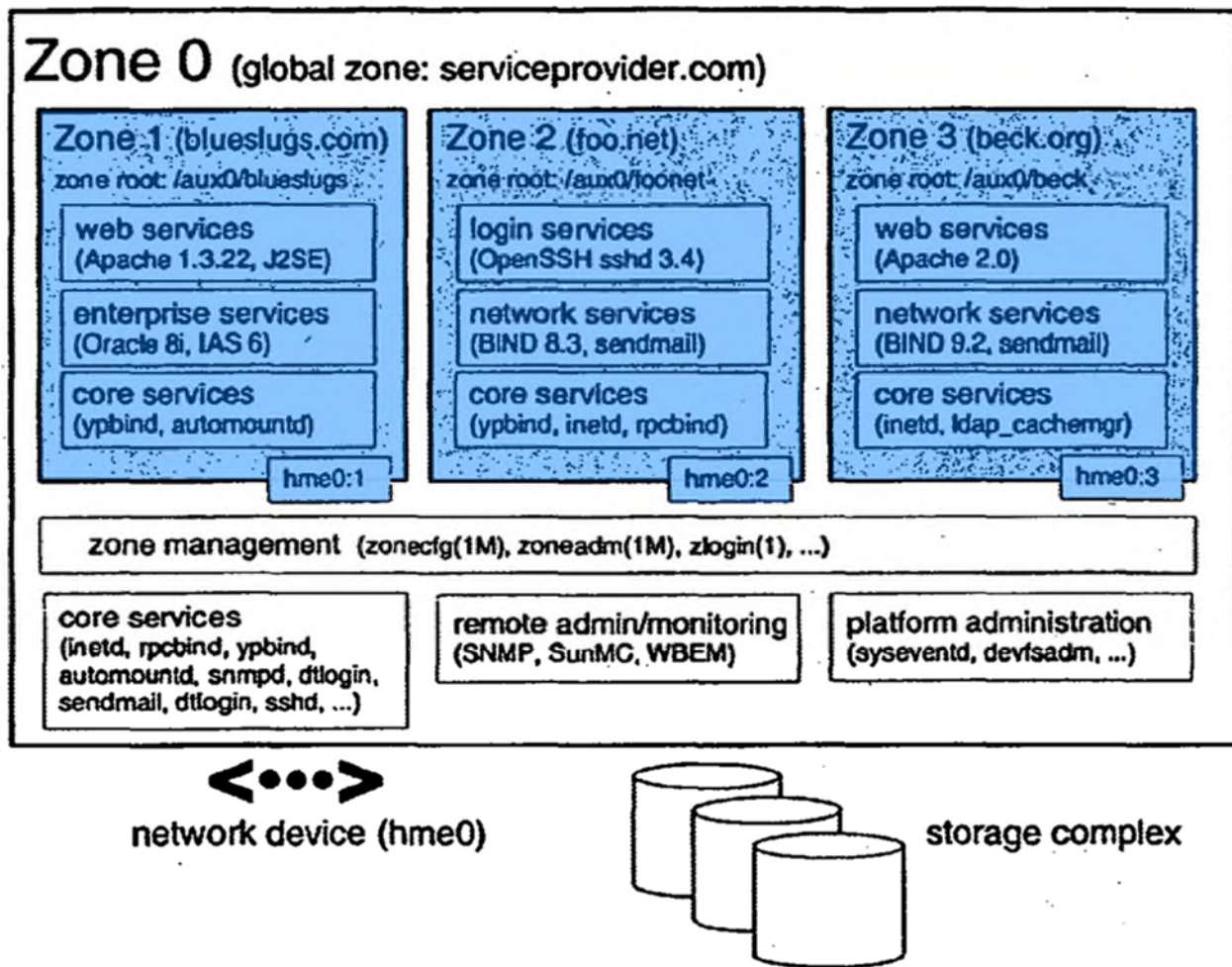
satisfy elements of the asserted patent claims and that are disclosed in the 2002 publication presented here. (*Infra* §VI.C.)

2. Solaris Zones (Tucker)

While Gélinas was developing VServer, Sun Microsystems was working on containers for its popular Solaris operating system. Sun published a press release in May 2002 that announced “containers” as part of Solaris 9. (Ex. 1009, 2-3.) Sun’s containers allowed “customers to run multiple applications on a single server, with fault, security and resource containment built-in.” (*Id.*, 3)

As Sun continued to develop its container technology, it filed a provisional patent application describing a system called Solaris Zones in May 2003. (Ex. 1005.) Sun’s provisional explained that a “zone is an application container[.]” (*Id.*, 92.) That provisional eventually matured into an issued patent. (Ex. 1004.)

Sun’s technology allowed an operating system to be partitioned into a “global zone” and one or more “non-global zones”—containers that would isolate groups of processes from each other and from the underlying operating system. (Ex. 1005, 1-5.) Sun’s provisional also described non-global zones as isolated ““sandbox[es]” within which one or more applications can run without affecting or interacting with the rest of the system.” (*Id.*, 1.) Figure 1.1 from Sun’s provisional shows non-global zones 1, 2, and 3 (**containers**) running services to host three different web sites:



(Ex. 1005, 3 (annotated).) Thus, Solaris Zones are also secure containers that isolate and confine processes that share an underlying operating system. (Ex. 1002 ¶42.)

3. Zap Pods (Osman)

Containers were also the subject of academic research. In December 2002, researchers from Columbia University presented a paper on “Zap,” a system designed to allow groups of application processes to be easily moved from server to server. (E.g., Ex. 1003, 361, 367.) Zap isolated groups of processes into “pods,” which are “a group of processes with a private namespace that presents the process

group with the same virtualized view of the system.... This decouples processes in a pod from dependencies on the host operating system and from other processes in the system.” (*Id.*) Zap’s pods were isolated from the underlying server and from other pods so that they could be paused, saved, and resumed on another server without interruption. (*See id.*; Ex. 1002 ¶43.)

II. THE ’814 PATENT

A. Overview

The ’814 patent describes a system of “associating applications with secure containers.” (Ex. 1001, 7:4-15.) As Figure 2 shows, each **container** (20a, 20b) is comprised of applications and system files, and the container is thus “an aggregate of all files required to successfully execute a set of software applications on a computing platform.” (*Id.*, 7:22-29.)

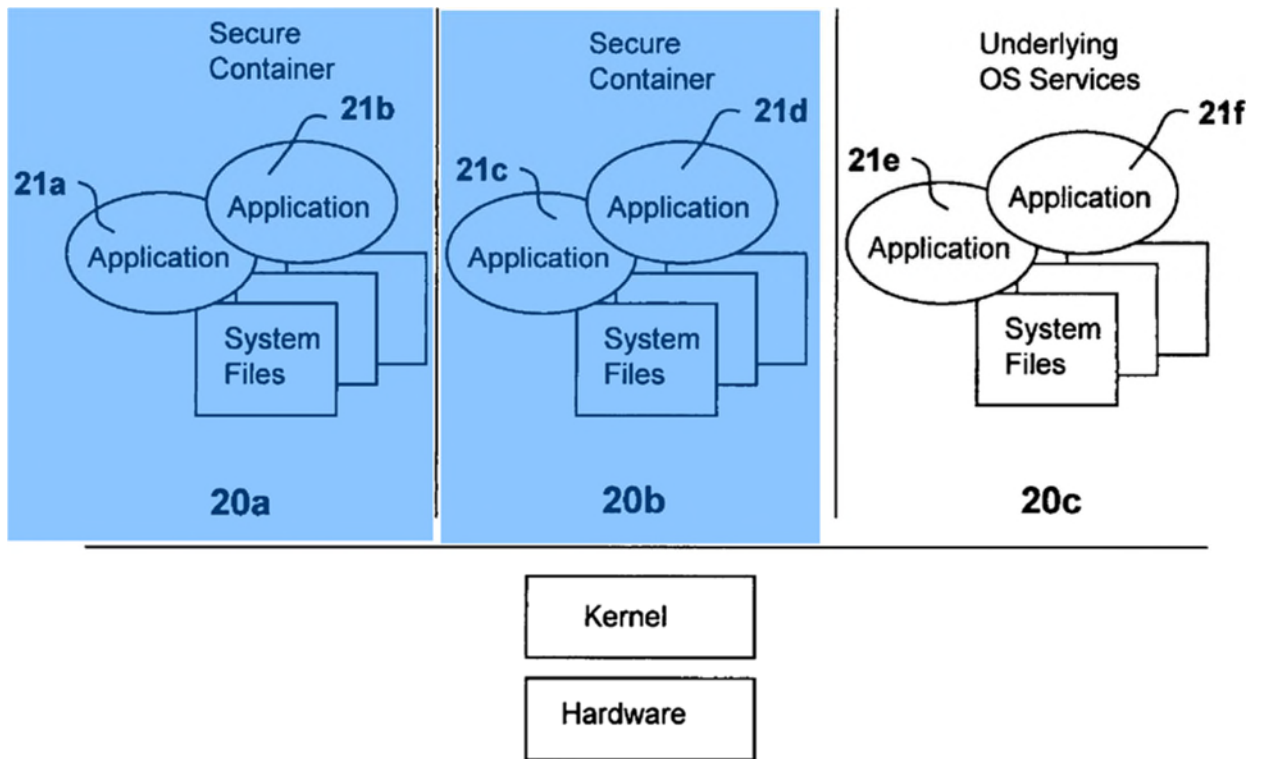


Figure 2

(*Id.*, Fig. 2.)

As in the prior art, “each application executing associated with a container ... is able to accesses [sic] files that are dedicated to the container” but is “not able to access the files contained in another container” or “the file provided with the underlying operating system” that are outside of any container. (*Id.*, 8:66-9:7.)

B. Prosecution History

The Examiner cited, among other references, a 2007 article by Soltesz et al. that was not prior art. (Ex. 1008, 15.) Soltesz discussed various aspects of the prior-art VServer system, but omitted aspects material to the claims at issue here. For example, the Examiner incorrectly determined that the prior art failed to disclose

(1) “applications software not being sharable between the plurality of secure (and isolated) containers”; (2) “unique root file systems different from an operating system’s root file system”; and (3) “different versions of the same operating system running on the same system/server environment.” (Ex. 1008, 15-16.) These features were all disclosed in Gélinas (Ex. 1007), which describes VServer in far more detail than Soltesz and was not before the Examiner (Ex. 1001, 1-2).

Finally, the Soltesz article also cites (among a list of 25 references) the Osman paper relied on in this Petition and a 2004 paper about Solaris zones. (Ex. 1011 (Soltesz), 286-87.) That Solaris paper is not prior art because it published in November 2004. (Ex. 1012.) As for the Osman paper, the Examiner never cited it and the patent applicant never submitted it. (Ex. 1001, 1-2.) Thus, the Patent Office never compared the claims of the ’814 patent to any of the prior art presented here.

III. STATEMENT OF RELIEF REQUESTED

A. Grounds

The challenged claims should be canceled under 35 U.S.C. §103 as follows:

Ground	Claims Challenged	References
1	3, 5, 7, 11-12, 15, 31-34	Osman
2	16	Tucker & Bandhole
3	16	Gélinas

B. The References Are Prior Art.

1. The Patent's Filing Date

The '814 patent claims priority to two provisional applications. (Ex. 1001.) But the challenged claims are not entitled to the filing date of either provisional application. For a patent to effectively claim priority to a provisional application, the specification of the provisional must contain a written description of the invention that enables a POSITA to practice the invention claimed in the patent. *Dynamic Drinkware, LLC v. Nat'l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015); *New Railhead Mfg., LLC v. Vermeer Mfg. Co.*, 298 F.3d 1290, 1294 (Fed. Cir. 2002). Neither provisional to which the '814 patent claims priority satisfies this standard.

First, the '619 provisional deals with subject matter entirely different from that addressed by challenged claim 1. (*Compare* Ex. 1013 *with* Ex. 1001, claim 1.) The '619 provisional is entitled “Drag & Drop Application Management” and deals with “software that manages the operation of a data center” through drag-and-drop installation. (Ex. 1013, 1-7.) The word “container” appears nowhere in this provisional. (*Id.*)

Second, while the '103 provisional deals with a container system, it fails to disclose all of claim 1. For example, it does not disclose claim 1's limitation that “said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files *that remain resident*

on the server.” (Ex. 1001, 17:53-57.) The antecedent basis for “the server” is “one of the servers” in the claimed system on which a “local kernel” resides. (*Id.*, 17:43-46.) This “local kernel” is used to execute the applications and system files in each container. (*Id.*) Thus, claim 1 requires system files in a container to be copies of system files that “remain resident” on the server where the containers run.

In contrast, the ’103 provisional discloses that “data and configuration information are copied from the computing platform from which they originate” and packaged into a container before being installed on a “remote computing platform.” (Ex. 1014, 18-19). The provisional *does not* disclose that the “platform from which they originate” and the “remote” platform are the same server. (*Id.*) For at least this reason, the ’103 provisional fails to support claim 1 and its dependents.

The prosecution history of the ’814 patent confirms that the ’103 provisional fails to support claim 1. The applicant added the “copies or modified copies” limitation by amendment. (Ex. 1008, 28-29.) In doing so, the applicant identified alleged support for the limitation. (*Id.*, 37-41) Specifically, the applicant identified Figure 9 and paragraph 91 of the non-provisional application. (*Id.*, 38.) But this figure and paragraph do not appear in the ’103 provisional. (Ex. 1014.) Thus, the prosecution history confirms that the patent is not entitled to the filing date of the ’103 provisional.

In litigation, VirtaMove has asserted that claims of the '814 patent are entitled to effective filing dates in September 2002, earlier than either of the provisional applications. To establish the earlier date, VirtaMove would have to show that the '814 patent's inventors conceived of the claimed invention by that date and either reduced it to practice by that date or exercised reasonable diligence thereafter. *Medtronic, Inc. v. Teleflex Innovations S.A.R.L.*, 68 F.4th 1298, 1302-03 (Fed. Cir. 2023). Such an assertion must be shown by independent evidence corroborating an inventor's testimony. *Kolcraft Enters., Inc. v. Graco Children's Prods., Inc.*, 927 F.3d 1320, 1324 (Fed. Cir. 2019). VirtaMove has identified no such evidence.

The references herein are prior art under pre-AIA 35 U.S.C. §102 because they predate the '814 patent's filing date as explained below.

2. Osman

Osman is prior art under at least 35 U.S.C. §102(b) because it is a printed publication that published no later than December 11, 2002—when it was presented at a major industry conference. (Ex. 1003, cover; Ex. 1015, 1 (citation listing conference date).) The publication was also received by the Kurt F. Wendt Library at the University of Wisconsin-Madison on or before April 14, 2003 and available to library patrons by May 2003—more than a year before the '814 patent's filing date. (Ex. 1015, 1-2.) Even if the '814 patent were entitled to its earliest provisional priority date, Osman would still be prior art under at least 35 U.S.C. §102(a).

3. Tucker

Tucker is prior art under at least 35 U.S.C. §102(e) because it is an issued patent filed on January 21, 2004. (Ex. 1004.) Further, Tucker is entitled to the priority date of its own provisional application (the “Tucker Provisional”). The Tucker Provisional was filed May 9, 2003 (Ex. 1005). Tucker was filed within one year of the Tucker Provisional’s filing, names at least one inventor in common, and specifically references the Tucker Provisional. (Ex. 1004, 1, 1:6-10; Ex. 1005.)

As shown below, the Tucker Provisional fully supports claim 1 of Tucker. Thus, Tucker is entitled to the Tucker Provisional’s May 9, 2003 priority date. *Dynamic Drinkware*, 800 F.3d at 1381-82; *Amazon.com, Inc. v. CustomPlay, LLC*, IPR2018-01496, Paper 34, at 44-48 (P.T.A.B. Mar. 4, 2020).

Tucker, Claim 1	Exemplary Support in Tucker Provisional (Ex. 1005)
A method comprising:	1 (“Zones provide a means of virtualizing operating system services[.]”)
establishing a global zone, wherein the global zone is a global operating system environment that can support execution of one or more processes;	2 (“Zone 0, enclosing zones 1-3, is the <i>global</i> zone. Processes running in this zone have (by and large) the same set of privileges available on a Solaris system today[.]”), 3 (“The global zone always exists, and acts as the ‘default’ zone in which all processes run if no zones have been explicitly created by the administrator.”), 4-5, 9, 34-35, Fig. 1.1.

Tucker, Claim 1	Exemplary Support in Tucker Provisional (Ex. 1005)
<p>establishing a non-global zone within the global zone, wherein the non-global zone is a partition of the global operating system environment, wherein the non-global zone operates as a separate and distinct operating system environment, and wherein the non-global zone can support execution of one or more processes;</p>	<p>1 (“A zone is a ‘sandbox’ within which one or more applications can run without affecting or interacting with the rest of the system.”), 2 (“Zones 1, 2 and 3 are each running a disjoint workload in a sample consolidated environment. ... Each zone can provide an almost arbitrarily rich and customized set of services.”), 2 (“each zone is provided a portion of the file system hierarchy”), 3-5, 9 (“The first step toward bringing up a zone is to create a configuration which specifies various parameters for that zone. Once a configuration is complete, it can be installed onto the system, then it can be booted, logged into, and administered.”), 9-20 (discussing creation and configuration of non-global zones), 25, 33-37, 71-74, Fig. 1.1 (showing non-global zones within the global zone), Fig. 3.1.</p>
<p>isolating a first process executing within the non-global zone to the non-global zone so that the first process does not have visibility or access to processes and objects that are not associated with the non-global zone;</p>	<p>1, 2 (“Basic process isolation is also demonstrated. Each zone is given access to a logical network interface; applications running in distinct zones cannot observe the network traffic of the other[.] ... Finally, each zone is provided a portion of the file system hierarchy. Because each zone is confined to its subtree of the file system hierarchy, the workloads cannot access each other’s on-disk data.”), 3, 4 (“No zone (other than the global zone) should be able to access objects belonging to</p>

Tucker, Claim 1	Exemplary Support in Tucker Provisional (Ex. 1005)
	another zone (including the global zone), either to control or modify those objects in some way, or to simply monitor or read them.”), 4 (“processes in a non-global zone should not be able to interfere with the execution of processes in other zones in the system”), 5, 25-31, 33-42, 68, 71-73, Fig. 1.1.
permitting a second process executing within the global zone to have visibility and access to processes and objects associated with the global zone; and	2 (processes in the global zone “have (by and large) the same set of privileges available on a Solaris system today”), 3-5, 25 (“Processes running in the global zone will still have the full set of privileges, allowing them to affect activity in any zone of the system.”), 33, 34 (“By default, access to system objects from the global zone will be restricted to those objects associated with the global zone.”), 35-36, Fig. 1.1.
permitting the second process executing within the global zone to have access to processes and objects associated with the non-global zone, if the second process has a privilege to cross zone boundaries.	3, 4 (“Appropriately privileged processes in the global zone can access objects associated with other zones.”), 5, 25 (“Processes running in the global zone will still have the full set of privileges, allowing them to affect activity in any zone of the system.”), 33-36, Fig. 1.1.

This Petition cites to the Tucker Provisional in the obviousness ground below (*infra*, §VI.B) because the entirety of the Tucker Provisional was incorporated by

reference into Tucker. (Ex. 1004, 1:6-10). This incorporated disclosure qualifies as prior art for the reasons set forth above.

4. Bandhole

Bandhole is prior art under at least 35 U.S.C. §102(b) because it published on November 21, 2002. (Ex. 1006.) Even if the '814 patent were entitled to its earliest provisional priority date, Bandhole would be prior art under 35 U.S.C. §102(a). It is also prior art under §102(e) because it is a published patent application filed January 30, 2002. (*Id.*)

5. Gélinas

Gélinas (Ex. 1007) is prior art under at least 35 U.S.C. §102(b) because it is a printed publication that published no later than Aug. 14, 2002. (Ex. 1016 ¶¶4-9.) Gélinas is a set of pages from a public website. (*Id.* ¶9.) The website's address was circulated as part of the VServer project's announcement on the Linux Kernel Mailing List. (*Id.* ¶5; Ex. 1017.)

Subscription to the Linux Kernel Mailing List was common for skilled artisans at the time, so the Gélinas website was disseminated to the interested public via the Mailing List, and a POSITA therefore would have been independently aware of and able to locate Gélinas. (Ex. 1016 ¶¶5-7.)

The Mailing List post was also publicized on Slashdot, a popular link aggregation service with numerous daily readers, many of whom were members of

the interested public. (Ex. 1016 ¶¶6-7; Ex. 1018.) Thus, Gélinas is a prior-art printed publication. *Hulu, LLC v. Sound View Innovations, LLC*, IPR2018-01039, Paper 29 at 8-12 (P.T.A.B. Dec. 20, 2019) (precedential); *Voter Verified, Inc. v. Premier Election Sols., Inc.*, 698 F.3d 1374 (Fed. Cir. 2012); *Keysight Techs., Inc. v. Centripetal Networks, LLC*, IPR2022-01525, Paper 27 (P.T.A.B. Apr. 15, 2024).

Gélinas consists of four pages from the VServer website—a main page titled “Virtual private servers and security contexts” and three pages in a “Howto/FAQ” section. (Ex. 1016 ¶¶ 8-9.) The main page includes a hyperlink to the Howto/FAQ section. (Ex. 1007 at 28 (“A FAQ can be found at [link]”).) Because all four pages were on the same website (hosted at www.solucorp.qc.ca) discussing the same software (VServer), they constitute a single publication. *Halliburton Energy Servs., Inc. v. Dynamic 3D Geosolutions LLC*, IPR 2014-01186, 2015 WL 5565065, *10 (PTAB Dec. 15, 2015). Regardless, even if the Gélinas pages were separate references, it would have been obvious to combine their teachings because a POSITA “would have been interested in reading the [Gélinas] pages, all collected at a single location, to obtain an understanding of a single coherent system, the [VServer] system.” *Id.*

IV. LEVEL OF ORDINARY SKILL

Based on the relevant factors, *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995), a person of ordinary skill in the art (“POSITA”) would have had a

minimum of a bachelor's degree in computer science, computer engineering, software engineering, or a similar field, and approximately two years of industry or academic experience in a field related to operating system design. (Ex. 1002 ¶31.) Work experience could substitute for formal education and additional formal education could substitute for work experience. (*Id.*)

V. CLAIM CONSTRUCTION

Petitioner submits that no claim terms require construction to resolve the obviousness challenges here; however, this is not a waiver of potential claim construction arguments. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co. Ltd.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017); *Vivid Techs., Inc. v. Am. Sci. & Eng'g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

A. “Container” and “System Files”

Petitioner notes that the terms “container” and “system files,” which are used in the claims, are defined in the patent's specification. (Ex. 1001, 2:16-3:19.) No explicit construction for these terms is necessary because the prior art relied on herein discloses the claims under the definitions in the specification and the plain meaning of the claims. (Ex. 1002 ¶48.)

B. “Disparate Computing Environments”

Petitioner notes that in a related district court case, *VirtaMove, Corp. v. Amazon.com, Inc.*, No. 7:24-cv-00030 (“Amazon Litigation”), Defendant

(“Amazon”) asserted that the phrase “disparate computing environments,” recited in claim 1 (from which several challenged claims depend), is indefinite. The patent defines “disparate computing environments” as “Environments where computers are *stand-alone* or where there are plural computers and where they are *unrelated*.” (Ex. 1001, 2:17-19 (emphases added).) Defendant argued in the Amazon Litigation that “unrelated” is a subjective term of degree. (Ex. 1019.) However, PO argued that the district court need not resolve the meaning of “unrelated” because, in the context of claim 1, the computers cannot be “unrelated.” (Ex. 1020¶49.) In PO’s view, “disparate computing environments” would thus be limited to “environments run by ... standalone computers.” (*Id.* at 4)

PO further asserted that a computer can be “standalone” even if it is connected to other computers in a larger system, as long as the computer is “independently operable.” (*Id.*) The prior art here discloses “disparate computing environments” under PO’s interpretation. (Ex. 1002 ¶50; *infra* §VI.A.1.c, §VI.B.1.c, §VI.C.1.c.)

Another potential interpretation of “disparate computing environments” was suggested by the patent’s first named inventor, Donn Rochette. In his deposition in the Amazon litigation, Rochette testified that the patent’s definition of “disparate computing environments” would cover computers that each “have a different network address,” which “would make them standalone and separate.” (Ex. 1021, 71:17-19, 73:18-74:18.) The prior art discloses “disparate computing environments”

under Rochette’s interpretation also. (Ex. 1002 ¶51; *infra* §VI.A.1.c, §VI.B.1.c, §VI.C.1.c.)

Because the prior art discloses the claim limitations under any available interpretation, no limitations require express construction here. *Intel Corp. v. Qualcomm Inc.*, 21 F.4th 801, 812-13 (Fed. Cir. 2021).

VI. GROUNDS OF UNPATENTABILITY

A. Ground 1: Claims 3, 5, 7, 11-12, 15, and 31-34 are Unpatentable as Obvious in View of Osman.

1. Claim 3

Claim 3 depends from claim 2, which depends from claim 1. Thus, the limitations of claims 1 and 2 are addressed below, along with the added limitation of claim 3.

a. Limitation 1[pre][i]: “In a system having a plurality of servers”

Osman discloses that its system is designed for an environment where “compute machines [plural] typically run completely independently of one another” and applications are migrated from one machine to another. (Ex. 1003, 363.) Osman’s “compute machines” include “servers, where application processing takes place.” (*Id.*) Osman also discloses multiple “network file servers” for storing “applications and user data.” (*Id.*) Thus, Osman discloses this limitation. (Ex. 1002 ¶54.)

b. Limitation 1[pre][ii]: “with operating systems that differ”

Osman discloses that its system was “successfully used ... to migrate pods across Linux kernels with minor version differences.” (Ex. 1003, 372.) The Linux kernel is the core of the Linux operating system, so a POSITA would understand from this disclosure that Osman’s system was used with Linux operating systems having minor kernel-version differences. (Ex. 1002 ¶55.) Thus, Osman discloses this limitation. (Ex. 1002 ¶55.)

Additionally, Osman discloses that its system lacks “dependencies on the host operating system.” (Ex. 1003, 361 (system “should not necessitate use of new operating systems or substantial modifications to existing ones”), 362 (discussing system’s “compatib[ility] with existing operating systems”—plural), 363, 373.) Based on this express teaching, a POSITA would have found it obvious to use Osman’s system with other operating systems in addition to Linux. (Ex. 1002 ¶56.)

c. Limitation 1[pre][iii]: “operating in disparate computing environments”

As discussed above, Osman discloses that its system is used in environments where “compute machines typically run completely independently of one another, each running its own independent operating system.” (Ex. 1003, 363.) This disclosure satisfies the claim limitation under PO’s interpretation, which requires “independently operable” computers. (*Supra*, §V.B.)

Osman also discloses that pods running on different host machines have different network addresses. (Ex. 1003, 369 (“external IP address ... changes as the given pod moves from host to host”).) This disclosure satisfies the claim limitation under the inventor’s interpretation, which is satisfied by separate computers that each “have a different network address.” (*Supra*, §V.B.)

Thus, Osman discloses this limitation. (Ex. 1002 ¶59.)

d. Limitation 1[pre][iv]: “wherein each server includes a processor and an operating system including a kernel”

As discussed above, Osman discloses servers that each run their own independent operating system. (Ex. 1003, 363.) A POSITA would have understood that each such operating system included a kernel—which is the core of an operating system. (Ex. 1002 ¶60; *see also* Ex. 1003, 362 (Osman’s system is “implemented ... as a loadable kernel module”).) Furthermore, Osman discloses implementing its system on servers with “933 MHz Intel Pentium-III” processors. (Ex. 1003, 372.) Thus, Osman discloses this limitation. (Ex. 1002 ¶60.)

e. Limitation 1[pre][v]: “a set of associated local system files compatible with the processor”

The ’814 patent defines “System files” as “files provided within an operating system and which are available to applications as shared libraries and configuration files.” (Ex. 1001, 2:52-54.) As an example, the ’814 patent lists “configuration files” from the “Apache” web server. (*Id.*, 3:6-17.) Osman discloses that pods may

contain such configuration files. (Ex. 1003, 367 (describing example where “each pod maintains its own configuration” used to run “apache”).) These configuration files are in “private pod directories” that can be stored “locally on the host machine.” (*Id.*) Thus, Osman discloses this limitation. (Ex. 1002 ¶61.)

f. Limitation 1[pre][vi]: “a method of providing at least some of the servers in the system with secure, executable applications related to a service”

Osman discloses executing eleven different applications related to a “VNC” service. (Ex. 1003, 374.) VNC is a service used to connect to a remote computer and run graphical applications such as a “web browser” or “PDF viewer” in a pod. (*Id.*) As another example, Osman discloses executing “Apache” in a pod. (*Id.*) The ’814 patent admits that Apache is an application related to a service. (Ex. 1001, 4:26-31.)

Further, applications in Osman’s pods are secure because only authorized users “are allowed to manipulate the pod[.]” (Ex. 1003, 371.) Thus, Osman discloses this limitation. (Ex. 1002 ¶63.)

g. Limitation 1[pre][vii]: “wherein the applications are executed in a secure environment”

Osman discloses that its “processes are created inside of a pod and spend their entire lifetimes in the context of that pod; they are not allowed to leave” the pod. (Ex. 1003, 364.) “Other processes outside a pod ... are therefore not able to interact with processes inside a pod” as they otherwise would. (*Id.*) Because application

processes are isolated in Osman's pods, the applications are executed in a secure environment. (Ex. 1002 ¶64; *see also infra*, §VI.A.1.p, §VI.A.1.q.) Thus, Osman discloses this limitation. (*Id.* ¶64.)

h. Limitation 1[pre][viii]: “wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service”

Osman discloses that its applications perform tasks related to a service. (Ex. 1003, 374 (listing tasks performed by Apache and VNC applications).) For example, Osman discloses that its system may run a web server, which a POSITA would have understood necessarily includes at least one object executable by the server's operating system for performing a task related to web hosting. (*Id.*, 367, 373 (web server delivered “54 web pages”); Ex. 1002 ¶65.) Thus, Osman discloses this limitation. (Ex. 1002 ¶65.)

i. Limitation 1[a][i] “the method comprising: storing in memory accessible to at least some of the servers a plurality of secure containers of application software”

The '814 patent defines a “container” as an “aggregate of files required to successfully execute a set of software applications on a computing platform[.]” (Ex. 1001, 2:23-25.) Osman's pods are containers because they comprise a set of files needed to execute the applications running in the pod. (Ex. 1003, 367-68.)

Osman discloses, for example, that when a pod is to be migrated from one place to another, the system “saves and restores the information it needs to

reconstruct the pod virtual file system, including a list of all files opened by the processes within a pod and the access rights with which the files were opened.” (*Id.*, 368.) For any files that are not already saved in network file storage, the system “saves the contents of the file and recreates the file when the pod is restarted.” (*Id.*, 365, 368; Ex. 1002 ¶67.)

Osman also discloses storing multiple pods together in memory accessible to one or more servers. (Ex. 1003, 367-68 (on “network file server” and local host’s file system), 372, 374.) Thus, Osman discloses storing a “plurality” of containers. (Ex. 1002 ¶68.)

In addition to the basic definition of “container” addressed above, the ’814 patent further describes the attributes of containers as follows:

Each container for use on a server is mutually exclusive of the other containers, such that read/write files within a container cannot be shared with other containers. . . . A container comprises one or more application programs including one or more processes, and associated system files for use in executing the one or more processes; but containers do not comprise a kernel; each container has its own execution file associated therewith for starting one or more applications. In operation, each container utilizes a kernel resident on the server that is part of the operating system (OS) the container is running under to execute its applications.

(Ex. 1001, 2:29-42.) To the extent “container” is construed to require any of these attributes, Osman discloses them. For example, Osman discloses that files within a container cannot be shared (unless the container is specifically configured to allow sharing). (Ex. 1003, 367; *infra*, §VI.A.1.p, § VI.A.1.q.) Osman also discloses the

other attributes of containers, as explained below in connection with other claim limitations. (*Infra*, §VI.A.1.j (applications and system files in containers); §VI.A.1.m (no kernel in containers); §VI.A.1.r (execution file in containers); §VI.A.1.k (containers use server’s kernel).) Thus, Osman discloses this limitation under any construction of “container.” (Ex. 1002 ¶¶66-68.)

j. Limitation 1[a][ii]: “each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications”

Osman discloses that each pod (container) may comprise one or more applications. For example, Osman describes testing that involved two pods—one for VNC and one for Apache. (Ex. 1003, 374.) The VNC pod contained eleven applications and the Apache pod contained one application. (*Id.*) Apache relies on system files as explained above. (*Supra*, §VI.A.1.e.) Additionally, a POSITA would understand that applications in the VNC pod relied on shared libraries and configuration files, both of which are “system files” under the ’814 patent’s definition (Ex. 1001, 2:52-54), because shared libraries and configuration files were routinely used in complex applications like the ones installed in the VNC pod. (Ex. 1002 ¶¶69.) Accordingly, Osman discloses this limitation. (*Id.*)

k. Limitation 1[a][iii]: “for use with a local kernel residing permanently on one of the servers”

Osman states that its system “can be dynamically loaded on a running Linux system to provide ... functionality without modifying, recompiling, or reinstalling the Linux kernel.” (Ex. 1003, 372.) Osman accomplishes this by “leveraging the loadable kernel module interface available in many commodity operating systems” and implementing its system “as a Linux kernel module.” (*Id.*) Thus, Osman discloses this limitation. (Ex. 1002 ¶70.)

l. Limitation 1[a][iv]: “wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems”

Osman discloses that its system has been used “to migrate pods across Linux kernels with minor version differences.” (Ex. 1003, 372.) A POSITA would have understood this to mean that associated system files within the pods were compatible with local kernels of at least several different Linux operating systems. (Ex. 1002 ¶71.) More specifically, the system files in Osman’s pods were compatible with the Linux kernels in the operating systems running on both the source computer and the destination computer. (*Id.*) Otherwise, the migration of the pods would not have been successful because the applications in the pod could not have been running both before and after the migration. (*Id.*) Thus, Osman discloses, or at least suggests, this limitation. (*Id.*)

m. Limitation 1[a][v]: “the containers of application software excluding a kernel”

As discussed above, Osman discloses that its system makes use of a local kernel. (*Supra* §VI.A.1.k.) Further, Osman discloses that its system may “migrate pods across Linux kernels.” (Ex. 1003, 372.) Such migration is accomplished by saving information that the pod was using to run on a host with a first kernel and loading that information when the pod is restarted on another host with a different kernel. (*Id.*) Thus, a POSITA would understand that the pod (container) does not include the kernel. (Ex. 1002 ¶72.) Accordingly, Osman discloses this limitation. (*Id.*)

n. Limitation 1[a][vi]: “wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server”

Osman discloses an example in which one copy of the Apache web server runs inside a pod and another copy runs outside the pod on the same computer. (Ex. 1003, 373.) As explained above, Apache uses system files to control its configuration. (*Supra*, §VI.A.1.e.) When one copy of Apache is inside a pod, that copy uses the system files inside the pod instead of the system files outside the pod—because system files outside of the pod are inaccessible from inside the pod. (Ex. 1002 ¶73; Ex. 1003, 367.) Thus, Osman discloses this limitation. (Ex. 1002 ¶73.)

- o. Limitation 1[a][vii]: “wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server”**

When copies of Apache run both inside and outside of a pod (*supra*, §VI.A.1.n), the Apache configuration files inside the pod are copies or modified copies of the Apache configuration files outside the pod. (Ex. 1002 ¶74; Ex. 1003, 367, 374.) Thus, Osman discloses this limitation. (Ex. 1002 ¶74.)

- p. Limitation 1[a][viii]: “wherein the application software cannot be shared between the plurality of secure containers of application software”**

Osman discloses creating a private directory for each pod and ensuring that “this directory is not accessible by processes on the host machine that are not in the given pod.” (Ex. 1003, 367.) These private directories prevent applications running in different pods from interfering with each other. (*Id.*)

Osman also discloses that pods *may* be configured to share application software in a common location. (*Id.*) But absent such configuration, a POSITA would understand that application software in one pod would not be shared with other pods because Osman’s system would “prevent processes within a pod from breaking out of their virtual file system environment.” (*Id.*; Ex. 1002 ¶76.) Osman also discloses that processes in a pod are prevented from sharing memory with processes outside the pod. (Ex. 1003, 364.) Thus, Osman discloses this limitation. (Ex. 1002 ¶76.)

- q. Limitation 1[a][ix]: “wherein each of the containers has a unique root file system that is different from an operating system’s root file system”**

Osman discloses that when a pod is created, the system “then uses the `chroot` call to set the staging area [for the pod’s virtual file system] as the root directory for the pod,” thereby giving the pod its own root file system separate from the operating system’s root file system. (Ex. 1003, 367; Ex. 1002 ¶77.) Thus, Osman discloses this limitation.

- r. Limitation 2: “wherein each container has an execution file associated therewith for starting the one or more applications”**

Osman discloses claim 2’s additional limitation because its pods include an execution file used by the “create_pod” command to specify “what applications if any should be launched once the pod is created[.]” (Ex. 1003, 371.)

- s. Limitation 3: “wherein the execution file includes instructions related to an order in which executable applications within will be executed.”**

Osman discloses claim 3’s additional limitation because the create_pod command can also be used “with /etc/init.d for automatically starting up services like a web server” in a pod. (Ex. 1003, 371.) In Linux, init.d was a directory used to store scripts that ran when the system started up. (Ex. 1002 ¶80.) Such scripts executed in a specific order based on their names. For example, an init.d script named “S90mysql” (to start a database application) would run before “S95httpd” (to

start the Apache web server application) because “90” comes before “95.” (Ex. 1022 at 3 (emphasis added).)

For the reasons above, Osman renders claim 3 obvious. (Ex. 1002 ¶80.)

2. Claim 5

Claim 5 depends from claim 2. The limitations of claim 2 are addressed above. (*Supra*, §§VI.A.1.a - r.) Claim 5 adds “the step of modifying at least some of the associated system files in plural containers to provide an association with a container specific identity assigned to a particular container.” Osman discloses claim 5’s additional limitation because, in one example, “each pod maintains its own configuration” for the Apache web server. (Ex. 1003, 367.) The ’814 patent states that Apache configuration files are an example of system files. (Ex. 1001, 3:6-17.) The Apache configuration for each pod is stored in that pod’s private directory and is therefore associated with the specific identity of that pod. (Ex. 1003, 367; Ex. 1002 ¶82.) Thus, Osman renders claim 5 obvious. (Ex. 1002 ¶82.)

3. Claim 7

Claim 7 also depends from claim 2. Claim 7 adds “the step of modifying at least some of the system files to define container specific mount points associated with the container.” Osman discloses claim 7’s additional limitation because a pod’s virtual file system is created “by mounting various NFS mount points from a file server[.]” (Ex. 1003, 372.) It would have been obvious to a POSITA that these file-

system mount points are defined by system files because Osman discloses that the “file system configuration” for each pod is specified by options in a configuration file. (*Id.*, 371.) Configuration files are system files, according to the ’814 patent. (Ex. 1001, 2:52-54.) Thus, Osman renders claim 7 obvious. (Ex. 1002 ¶84.)

4. Claim 11

Claim 11 also depends from claim 2. Claim 11 further recites that “containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location.” Osman discloses this because “various network-accessible directories that [a] pod is configured to access will be mounted from a network file server.” (Ex. 1003, 367.) Additionally, pods can be “checkpointed to and restarted from an image on the local disk,” which is a location separate from the network file server. (*Id.*, 374.) This checkpointed state includes various parameters describing a pod, such as the set of running processes, the contents of memory, and any “directories and files not mounted on a network file system.” (*Id.*, 365 (Table 1).) Thus, Osman renders claim 11 obvious. (Ex. 1002 ¶92.)

5. Claim 12

Claim 12 depends from claim 11 and further recites “the step of merging the files stored in network storage with the parameters to affect the step of storing in claim 1.” Osman discloses the additional limitation of claim 12 by describing how

a pod's virtual file system is staged when a pod is moved to a new host. (Ex. 1003, 367-68.) The virtual file system merges "network-accessible directories that the pod is configured to access ... from a network file server" with files stored on the local host. (*Id.* (describing local files in "/proc" and "unlinked" local files).) The files on the local host are specified by parameters in the checkpointed state information, as described above for claim 11. (*Id.*, 365 (Table 1).) Thus, Osman renders claim 12 obvious. (Ex. 1002 ¶93.)

6. Claim 15

Claim 15 depends from claim 14, which depends from claim 1. The limitations of claim 1 are addressed above. (*Supra*, §§VI.A.1.a - q.) Osman discloses each further limitation of claims 14 and 15 as explained below.

a. Limitation 14[a][i]: "creating containers prior to said step of storing containers in memory, wherein containers are created by:"

Osman discloses pods running on a computer before they are suspended and stored, e.g., in network file storage. (Ex. 1003, 365-371; *supra*, §VI.A.3.) Such pods are created on a first server before being migrated to a second server. (*Id.*; Ex. 1002 ¶¶81, 97-100.)

b. Limitation 14[a][ii]: "a) running an instance of a service on a server"

Osman discloses executing applications in a pod to provide a service. (*Supra* §VI.A.1.f.) For example, Osman discloses executing an Apache web server. (*Id.*)

c. Limitation 14[a][iii]: “b) determining which files are being used”

Osman discloses that when a pod is suspended, the system saves “a list of all files opened by the processes within a pod.” (Ex. 1003, 368.)

d. Limitation 14[a][iv]: “c) copying applications and associated system files to memory without overwriting the associated system files so as to provide a second instance of the applications and associated system files”

Osman discloses an example in which a first copy of Apache runs outside a pod and a second copy of Apache runs inside a pod. (*Supra*, §VI.A.1.n, §VI.A.1.o.) In this example, a POSITA would understand that the Apache application and its associated system files were copied into the pod without overwriting the copy of Apache outside the pod. (Ex. 1002 ¶100.) Osman also discloses migrating the Apache pod. (Ex. 1003, 374-75.)

e. Limitation 15[a]: “assigning an identity to the containers including at least one of a unique IP address, a unique Mac address and an estimated resource allocation”

Osman discloses that each pod is assigned a “[p]er-pod IP address.” (Ex. 1003, 365, 369; Ex. 1002 ¶103.)

f. Limitation 15[b]: “installing the container on a server”

Osman discloses installing containers on servers. (*E.g.*, Ex. 1003 at 373 (installing pods on computers that act as “web server”); Ex. 1002 ¶104.)

g. Limitation 15[c]: “testing the applications and files within the container”

Osman discloses testing the applications and files in the installed containers. (Ex. 1003 at 373-75; Ex. 1002 ¶105.) For example, “Application benchmarks” from this testing are shown in Osman’s Table 2. (*Id.*, 373.) One of these benchmarks involved downloading 54 web pages from an Apache web server. (*Id.*)

Thus, Osman renders claim 15 obvious. (Ex. 1002 ¶106.)

7. Claim 31

a. Limitation 31[pre]: “A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:”

Osman discloses computing systems performing tasks such as hosting web pages using “worker processes” from a web server application, or hosting applications (and thus processes) in a “VNC thin-client computing user session.” (Ex. 1003, 374; Ex. 1002 ¶107.)

b. Limitation 31[a][i]: “a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers”

Osman discloses this limitation as explained above for Limitation 1[a][i]. (*Supra*, §VI.A.1.i; Ex. 1002 ¶108.)

- c. **Limitation 31[a][ii]: “each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers”**

Osman discloses this limitation for reasons explained above in connection with Limitations 1[a][viii]-[ix]. (*Supra*, §VI.A.1.p, §VI.A.1.q; Ex. 1002 ¶109.) Osman also discloses functionality that “prevent[s] processes within a pod from breaking out of their virtual file system environment,” where the pod’s files are stored. (Ex. 1003, 367.) This functionality prevents files in a pod from being shared with other pods (unless the pods are specifically configured to allow such sharing). (Ex. 1002 ¶109.)

- d. **Limitation 31[a][iii]: “each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address”**

Osman discloses that each pod is assigned a “[p]er-pod IP address.” (Ex. 1003, 365, 369; Ex. 1002 ¶110.)

- e. **Limitation 31[a][iv]: “wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes”**

Osman discloses executing eleven application programs related to a “VNC” service in a pod. (Ex. 1003, 374; Ex. 1002 ¶111.) Additionally, Osman discloses executing “Apache” (another application program) in a pod. (*Id.*) Each of these

applications included one or more processes. (Ex. 1002 ¶111.) For example, Apache used a “number of worker processes.” (Ex. 1003, 374.)

f. Limitation 31[a][v]: “and associated system files for use in executing the one or more processes”

Osman discloses this limitation as explained above for Limitations [1][pre][v] and 1[a][ii]. (*Supra*, §VI.A.1.e; §VI.A.1.j; Ex. 1002 ¶112.)

g. Limitation 31[a][vi]: “wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system”

Osman discloses this limitation as explained above for Limitations 1[a][vi]-[vii]. (*Supra*, §VI.A.1.n; §VI.A.1.o; Ex. 1002 ¶113.)

h. Limitation 31[a][vii]: “each container having its own execution file associated therewith for starting one or more applications”

Osman discloses this limitation as explained above for the limitation of claim 2. (*Supra*, §VI.A.1.r; Ex. 1002 ¶114.)

i. Limitation 31[a][viii]: “in operation, each container utilizing a kernel resident on the server”

Osman discloses this limitation as explained above for Limitation 1[a][iii]. (*Supra*, §VI.A.1.k; Ex. 1002 ¶115.)

- j. Limitation 31[a][ix]: “wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel”**

Osman discloses this limitation as explained above for Limitation 1[a][v].

(*Supra*, §VI.A.1.m; Ex. 1002 ¶116.)

- k. Limitation 31[b]: “a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications”**

Osman discloses a “loadable kernel module” that is used to “to intercept system calls as needed for virtualization[.]” (Ex. 1003, 362; *id.*, 372, 375.) By intercepting system calls, the kernel module can limit an application’s access to resources outside of the pod in which the application runs. (*Id.*, 365-66; Ex. 1002 ¶117.) Additionally, the kernel module intercepts system calls to control applications’ access to network connections. (Ex. 1003, 370.)

Thus, Osman renders claim 31 obvious. (Ex. 1002 ¶118.)

8. Claim 32

Claim 32 depends from claim 31 and further recites “a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.” Osman discloses running pods on “the Linux 2.4.10 kernel.” (Ex. 1003, 372.) This Linux kernel included a scheduler for determining when each running process would receive processing resources from a CPU.

(Ex. 1002 ¶119; *see also* Ex. 1023 (publication discussing Linux scheduler).) Thus, Osman renders claim 32 obvious. (Ex. 1002 ¶119.)

9. Claim 33

Claim 33 depends from claim 32.

- a. **Limitation 33[a][i]: “the run time module includes an intercepting module associated with the plurality of containers for intercepting system calls from any of the plurality of containers”**

Osman’s kernel module intercepts system calls from pods. (Ex. 1003, 372; Ex. 1002 ¶121.)

- b. **Limitation 33[a][ii]: “and for providing values alternate to values the kernel would have assigned in response to the system calls”**

Osman’s kernel module provides alternate values in response to system calls. (Ex. 1003, 372, 364-66; Ex. 1002 ¶122.)

- c. **Limitation 33[a][iii]: “so that the containers can run independently of one another without contention, in a secure manner”**

Osman’s kernel module and system-call interception serve to “protect processes within a pod from dependencies on processes outside the pod.” (Ex. 1003, 364-65.) This protection keeps pods independent from one another and avoids contention by ensuring that “there are no resource naming conflicts for processes in different pods.” (*Id.*) Osman’s system-call interception also keeps pods secure by

ensuring that processes outside pods “do not attempt to manipulate processes inside pods.” (*Id.*, 374.) Thus, Osman discloses this limitation. (Ex. 1002 ¶123.)

d. Limitation 33[a][iv]: “the values corresponding to at least one of the IP address, the host name and the Mac_Address.”

Osman discloses “intercepting system calls for connection setup requests from [an] application to [a] transport protocol and replacing relevant physical addresses with virtual addresses.” (Ex. 1003, 369-370.) The “addresses” here include “an IP address.” (*Id.*)

Thus, Osman renders claim 33 obvious. (Ex. 1002 ¶125.)

10. Claim 34

Claim 34 depends from claim 31 and recites that the “run time module” performs additional steps. Osman’s kernel module performs these additional steps as explained below. (Ex. 1002 ¶126.)

a. Limitation 34[a]: “monitoring resource usage of applications executing;”

Osman discloses monitoring resources such as memory, file systems, and networks by virtualizing those resources. (Ex. 1003, 364-65.) Each resource is given a virtual name in the context of a pod, and applications within a pod use that virtual name to access the resource. (*Id.*, 365.) This virtualization is performed by Osman’s kernel module. (*Id.*, 362, 372.) The kernel module monitors requests for resources

to ensure that applications within a pod do not attempt to use resources belonging to other pods. (*Id.*, 366, 372; Ex. 1002 ¶127.)

- b. Limitation 34[b]: “intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;”**

Osman’s kernel module intercepts system calls from applications in a pod. (Ex. 1003, 362, 365-66.) The system calls originate in user mode because they come from applications. (Ex. 1002 ¶128.) System calls are intercepted on their way to the kernel. (*Id.*, 366.)

- c. Limitation 34[c]: “comparing the monitored resource usage of the at least one respective application with the resource limits; and,”**

Osman’s kernel module compares the resources that an application requests via system call with limits on the set of resources available in a pod. (Ex. 1003, 365-66; Ex. 1002 ¶129.) The module blocks requests for resources that are not available in the pod. (Ex. 1003 at 366 (“limit the successful calls to those that use valid identifiers ... created within the pod”).)

- d. Limitation 34[d]: “forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.”**

Osman discloses that system calls which are not blocked based on the resource limits are passed “on to the kernel for processing.” (Ex. 1003, 366.)

Thus, Osman renders claim 34 obvious. (Ex. 1002 ¶131.)

B. Ground 2: Claim 16 is Unpatentable as Obvious in View of Tucker and Bandhole.

Tucker discloses “zones,” which are application containers that Sun Microsystems implemented in its Solaris operating system. (*Supra*, §I.C.2.) Tucker focuses on how zones work on a single server, with limited discussion of how that single server might be used with other servers. (Ex. 1005.)¹ But a POSITA would have known at the time that Solaris servers were routinely networked with other servers to provide a wide array of services. (Ex. 1002 ¶132.)

Bandhole describes a system in which servers and other computing resources are combined to provide various services. (Ex. 1006.) Several examples in Citations to the Tucker Provisional (Ex. 1005) are also citations to Tucker, which incorporates the entire Tucker Provisional by reference (Ex. 1004, 1:6-10). Bandhole involve a Solaris server combined with another server running a different operating system. (*Id.* ¶¶ [0005], [0034].)

In one of Bandhole’s examples, the Solaris server runs both “custom application server software” and “Oracle database software” to “provide a search service on the Web.” (*Id.* ¶ [0005].) A separate server using the Linux operating

¹ Citations to the Tucker Provisional (Ex. 1005) are also citations to Tucker, which incorporates the entire Tucker Provisional by reference (Ex. 1004, 1:6-10).

system runs “Apache web server software” as part of the same search service. (*Id.*) This example appears in Bandhole’s background section, indicating that architectures like this were well known before Bandhole’s 2002 filing date. (*Id.*)

A POSITA would have been motivated to use Tucker’s zones to implement the types of services described in Bandhole’s background section. (Ex. 1002 ¶135.) For example, a POSITA would have found it obvious to run Bandhole’s application server and database using two separate zones in accordance with Tucker. (*Id.* ¶135.) This approach would have been obvious for several reasons.

First, Tucker discloses that zones “limit [] the damage possible in the event of a security violation.” (Ex. 1005, 1.) Zones “allow the deployment of multiple applications on the same machine,” even when those applications have differing requirements. (*Id.*) Zones also “can be useful to support rapid deployment (and redeployment) of applications.” (*Id.*, 2.) A POSITA would have been motivated to use zones to run Bandhole’s services to gain the security and deployment benefits that Tucker discloses. (Ex. 1002 ¶136.)

Second, the combination is a simple addition of one known element (Tucker’s zones) to another known element (Bandhole’s software) to obtain predictable results (software running in zones). (Ex. 1002 ¶137.) Moreover, the combination uses a known technique (zones on Solaris) to improve a similar device and method (Bandhole’s Solaris server and search service) in the same way. (*Id.*) The

combination also applies a known technique (zones on Solaris) to a known device and method (Bandhole's Solaris server and search service) that is ready for improvement and yields predictable results. (*Id.*)

Finally, because Bandhole's software was compatible with Solaris (Ex. 1006 ¶[0005]) and Tucker's zones ran on Solaris (*supra*, §I.C.2.), a POSITA would have had a reasonable expectation of success in making the combination. (Ex. 1002 ¶138.)

1. Claim 16

Claim 16 depends from claim 1. Thus, the limitations of claim 1 are addressed below, along with the added limitation of claim 16.

a. Limitation 1[pre][i]: “In a system having a plurality of servers”

The '814 patent states that the terms “computing platform, server and computer are used interchangeably throughout this specification.” (Ex. 1001, 12:5-7; *see also, e.g., id.* at 10:56-58 (“an existing server, for example a computer”).)

Tucker discloses that “the same application environment can be maintained on different physical machines” (plural), to support rapid deployment of applications across such machines. (Ex. 1005, 2.) Thus, Tucker discloses this limitation. (Ex. 1002 ¶140.)

Bandhole also discloses a plurality of servers: “a Linux server running Apache web server software” and “a Solaris server running a custom application server software and Oracle database software.” (Ex. 1006 ¶[0005]; Ex. 1002 ¶141.)

b. Limitation 1[pre][ii]: “with operating systems that differ”

Bandhole’s system includes Linux server and a Solaris server. (Ex. 1006 ¶¶0005.) Linux and Solaris are different operating systems. (Ex. 1002 ¶142.)

c. Limitation 1[pre][iii]: “operating in disparate computing environments”

Tucker and Bandhole disclose this limitation under both VirtaMove’s interpretation (requiring “independently operable” computers) and the inventor’s interpretation (requiring computers with different network addresses). (*Supra*, §V.B.)

First, Tucker discloses implementing zones on computers running Solaris. (Ex. 1005, 1-3.) Computers running Solaris were independently operable because one computer running Solaris could operate without being controlled by any other computer. (Ex. 1002 ¶144.) Tucker shows an example of this in Figure 1.1, where zones run on a single Solaris computer operating independently. (Ex. 1005, 1-3.)

Second, Tucker discloses implementing zones on computers with different network addresses. (Ex. 1002 ¶145.) For example, Tucker discloses that a system running zones would have its own “primary IP address.” (Ex. 1005, 1-2.) Further, different zones within the system could have “distinct IP addresses.” (*Id.*)

Additionally, Bandhole discloses independently operable computers with different network addresses. For example, Bandhole discloses one server running

Linux and another running Solaris, communicating with each other over an “Ethernet LAN.” (Ex. 1006 ¶ [0005].) The Linux server and Solaris server are independently operable because they use distinct operating systems that do not control each other. (Ex. 1002 ¶146.) Additionally, a POSITA would understand that the servers use different network addresses because they communicate with each other over the disclosed Ethernet LAN. (*Id.*)

d. Limitation 1[pre][iv]: “wherein each server includes a processor and an operating system including a kernel”

Tucker discloses that a server running zones on the Solaris operating system uses one processor or multiple processors. (Ex. 1005, 2.) Further, Tucker discloses that Solaris includes a kernel. (*E.g., id.*, 4, 10, 21.)

Bandhole’s servers run either Solaris or Linux, both of which are operating systems that include a kernel. (Ex. 1006 ¶ [0005]; Ex. 1002 ¶148.) Thus, Bandhole and Tucker each disclose this limitation. (Ex. 1002 ¶148.)

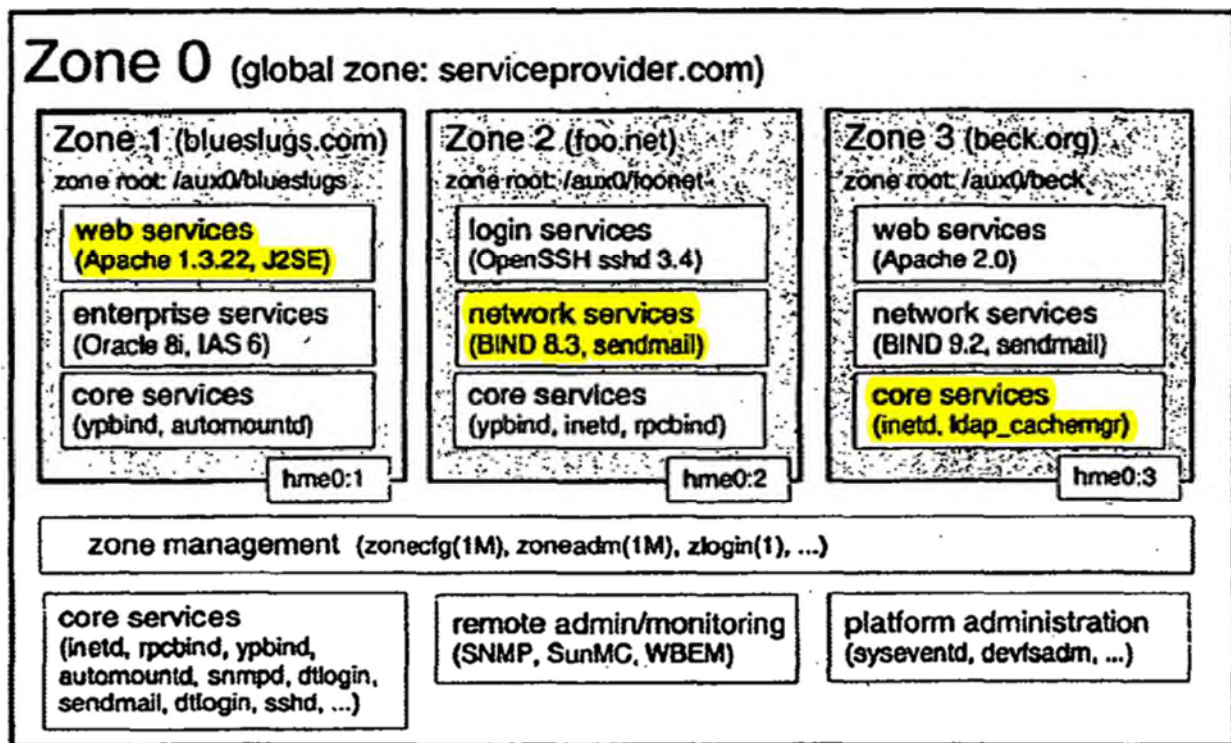
e. Limitation 1[pre][v]: “a set of associated local system files compatible with the processor”

The ’814 patent defines system files as including “shared libraries” and “configuration files.” (Ex. 1001, 2:52-54.) Tucker discloses both. (Ex. 1005, 43-44, 71-72.) Additionally, the ’814 patent admits that Apache running on Linux used both shared libraries and configuration files. (Ex. 1001, 2:55-3:19.) Bandhole discloses the use of Apache on Linux. (Ex. 1006 ¶¶ [0005], [0034].) A POSITA

would have understood that these system files are compatible with the processor used to execute them. (Ex. 1002 ¶149.)

f. **Limitation 1[pre][vi]: “a method of providing at least some of the servers in the system with secure, executable applications related to a service”**

Tucker discloses that its zones feature provides a way in which “one or more applications can run without affecting or interacting with the rest of the system,” thereby providing “Security” and “Isolation” benefits. (Ex. 1005, 1.) Tucker further discloses that such applications may relate to services, including web services, enterprise services, login services, network services, and the like:



(Id., 3; Ex. 1002 ¶150.)

g. Limitation 1[pre][vii]: “wherein the applications are executed in a secure environment”

Tucker also discloses that “applications running in distinct zones cannot observe the network traffic of the other” and “cannot access each other’s on-disk data.” (Ex. 1005, 1-2.) Thus, zones are a secure environment. (Ex. 1002 ¶151.)

h. Limitation 1[pre][viii]: “wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service”

Tucker discloses many applications that run in zones, including an Apache web server, Oracle database, OpenSSH login, and email server. (Ex. 1005, 3, Fig. 1.1.) These applications each included at least one object executable by the operating system for performing a task related to the types of services listed in Tucker’s Figure 1.1. (Ex. 1002 ¶152.)

i. Limitation 1[a][i] “the method comprising: storing in memory accessible to at least some of the servers a plurality of secure containers of application software”

Tucker discloses a four-step process for installing and configuring zones. (Ex. 1005, 9-10.) First, the zone is “configured,” meaning “a zone’s configuration has been completely specified and committed to stable storage.” (*Id.*, 9.) Second, the zone is “installed,” meaning “a zone’s configuration has been laid out on the system: packages [e.g., application software] have been installed under the zone’s root path.” (*Id.*; Ex. 1002 ¶153.) Third, the zone is “ready,” meaning “the kernel

has created the zone, ... file systems have been mounted, ... but no processes have been started.” (Ex. 1005, 10.) Finally, the zone is “running,” meaning “processes have been started.” (*Id.*) Thus, Tucker discloses (in at least the zone-installing step) storing a container as this limitation requires. (Ex. 1002 ¶153; *supra*, §I.C.2 (Tucker’s non-global zones are containers).) Tucker also discloses that a plurality of zones are stored together. (*E.g., id.*, 2-3 (three non-global zones).)

j. Limitation 1[a][ii]: “each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications”

Tucker discloses that its zones run various applications and use various system files. (*Supra* §VI.B.1.f, §VI.B.1.e.) Tucker also discloses an approach in which “all packages required for a zone that make up a particular metacluster will be installed as well as any other packages selected by the global administrator.” (Ex. 1005, 72.) This approach “will provide the maximum configurability” because “all of the required and any selected optional Solaris packages” are installed within the zone. (*Id.*) A POSITA would understand that, in this arrangement, the applications and system files used by a zone are installed into the zone as part of the required Solaris packages. (Ex. 1002 ¶154.) Solaris packages were the standard way of installing applications and their system files in the Solaris operating system. (*Id.*)

k. Limitation 1[a][iii]: “for use with a local kernel residing permanently on one of the servers”

Tucker discloses that zones are created and managed by the kernel. (Ex. 1005, 10-11, 4; Ex. 1002 ¶155.) The kernel resides permanently on the servers because it remains in place as zones are added or removed. (Ex. 1005, 9-15; Ex. 1002 ¶155.)

l. Limitation 1[a][iv]: “wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems”

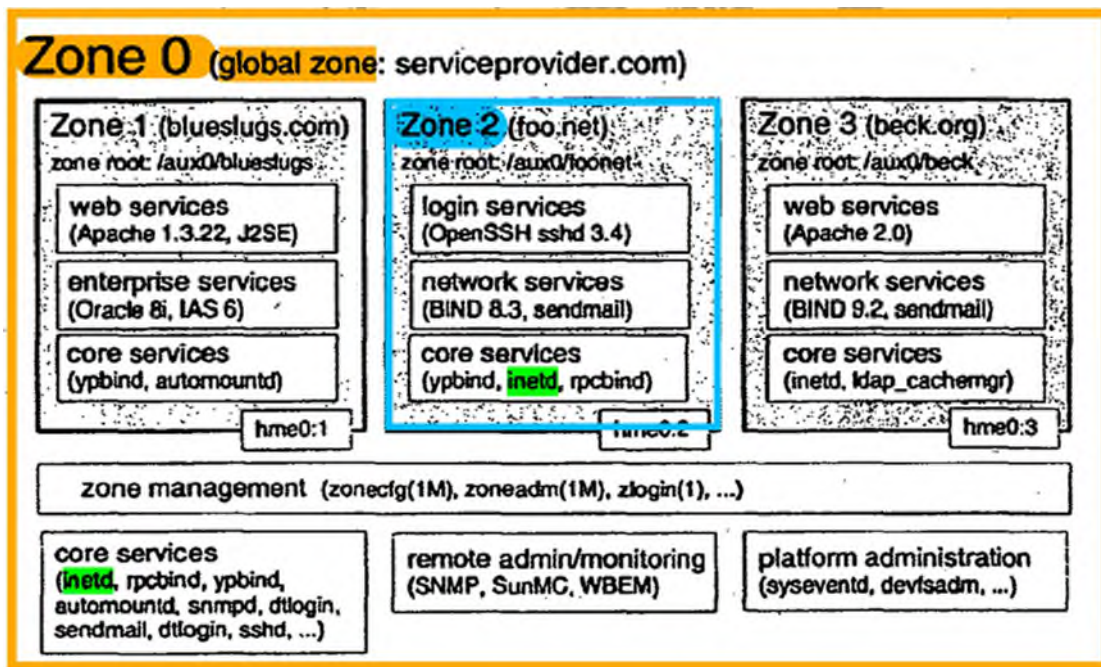
Because applications installed in zones are executed by the local kernel (*supra* §VI.B.1.k; Ex. 1005, 4, 74.), a POSITA would have understood that the associated system files (e.g., libraries and configuration files required to execute the applications) are likewise compatible with the local kernel. (Ex. 1002 ¶156.)

m. Limitation 1[a][v]: “the containers of application software excluding a kernel”

Tucker discloses that the operating system’s single kernel keeps track of multiple non-global zones. (Ex. 1005, 11, 21.) The kernel also creates each of these zones. (*Id.*, 10.) Based on this disclosure, a POSITA would have understood that the kernel is outside of the non-global zones. (Ex. 1002 ¶157.)

- n. **Limitation 1[a][vi]:** “wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server”

Tucker discloses that various applications in non-global zones (containers) may also be executed in the global zone (underlying server). For example, Tucker discloses that the service “**inetd**” may be executed in both a **non-global zone** (container) and a **global zone** (underlying server):



(Ex. 1005, 3.) Tucker further discloses that the “inetd” service uses a configuration file (system file), and different configuration files may be used for inetd in each zone. (*Id.* at 43-44 (“each zone can, for example, run its own inetd(1M) with a full configuration file”).) Thus, inetd in the non-global zone uses the system file from its own zone in place of the inetd system file in the global zone. (Ex. 1002 ¶158.)

The same is true for other applications like “ypbind” and “sendmail,” which use system files and are installed in both the global and non-global zones. (*Id.*)

- o. Limitation 1[a][vii]: “wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server”**

As discussed above, Tucker discloses that the underlying server and one or more non-global zones may run the same application, utilizing the non-global zone’s associated system files in place of the server’s associated local system files. (*Supra* §VI.B.1.n.) Because the same application is running in both locations, at least some of the associated system files (configuration files or shared libraries) are copies or modified copies of the associated local system files on the underlying server. (Ex. 1002 ¶159.) Indeed, Tucker confirms that one zone will sometimes duplicate system files from other zones leading to a “heavier disk footprint.” (Ex. 1005, 72.)

Tucker also discloses that “different versions of the same application may be run without negative consequence in different zones.” (Ex. 1005, 2.) Based on this disclosure, a POSITA would find it obvious that at least some system files in the non-global zone (e.g., files associated with a newer application version) would be modified copies of system files in the global zone (e.g., files associated with an older application version). (Ex. 1002 ¶160.)

- p. Limitation 1[a][viii]: “wherein the application software cannot be shared between the plurality of secure containers of application software”**

Tucker discloses that “[v]irtualization of storage in a zone is achieved via a restricted root.... Processes running within a zone will be limited to files and file systems that can be accessed from the restricted root.” (Ex. 1005, 37; *id.*, 71 (“zones require a separate root file systems [sic] for isolation”).) Thus, application software located in one zone cannot be shared with other zones, which will be unable to access that application’s files. (*Id.*; Ex. 1002 ¶161.)²

- q. Limitation 1[a][ix]: “wherein each of the containers has a unique root file system that is different from an operating system’s root file system”**

Tucker discloses that “zones require a separate root file systems [sic] for isolation.” (Ex. 1005, 71; *see also id.*, 37; *supra*, §VI.B.1.p; Ex. 1002 ¶162.)

- r. Limitation 16: “creating containers prior to said step of storing containers in memory, wherein a step of creating containers includes: using a skeleton set of system files as a container starting point and installing applications into that set of files.”**

Tucker discloses “lay[ing] down a root file system for [a] non-global zone in such a way that the standard Solaris packaging tools can be used to administer the

² Zones may be able to share files if they are specifically configured to do so, but absent such configuration no sharing can occur. (Ex. 1005, 72.)

file system[.]” (Ex. 1005, 71.) This root file system “requires a /etc directory to hold standard configuration information for that zone.” (*Id.*) A POSITA would understand that the configuration information in the “/etc directory” is stored in the form of files because directories hold files. (Ex. 1002 ¶171.) Configuration files are system files according to the ’814 patent. (Ex. 1001, 2:52-54.) Thus, the configuration files in the zone’s root file system are a set of system files. (Ex. 1002 ¶171.)

Tucker also discloses that applications are installed into the root file system using the Solaris packaging system. (Ex. 1005, 71-72.) Thus, Tucker discloses using a skeleton set of system files (in the zone’s root file system) as a container starting point and installing applications into that set of files (using Solaris packages). (Ex. 1002 ¶171.)

For the reasons above, claim 16 would have been obvious to a POSITA in view of Tucker and Bandhole. (Ex. 1002 ¶¶139-163, 171.)

C. Ground 3: Claim 16 is Unpatentable as Obvious in View of Gélinas.

Gélinas introduces Linux VServer, which provides secure containers that isolate and confine processes within the containers while allowing them to share the kernel of the underlying server. (*Supra*, §I.C.1.)

1. Claim 16

Claim 16 depends from claim 1. Thus, the limitations of claims 1 are addressed below, along with the added limitation of claim 16.

a. Limitation 1[pre][i]: “In a system having a plurality of servers”

Gélinas discloses that its containers (called “vservers”) operate in a system with multiple physical servers. (Ex. 1002 ¶173.) For example, it is “possible to move a vserver from one physical server to another.” (Ex. 1007, 30.)

b. Limitation 1[pre][ii]: “with operating systems that differ”

Gélinas discloses that its containers can run on differing versions of the Linux operating system, such as Debian and Redhat. (Ex. 1002 ¶174; Ex. 1007, 29.) Each of these different versions of Linux is called a “distribution” (or “distro”). (See Ex. 1007, 29; Ex. 1002 ¶174.) Gélinas also recognizes that “you may want to create several vservers to tests various distributions.” (Ex. 1007, 31; *id.* 27 (vserver code should “work on any recent distribution”).) These various Linux distributions are operating systems that differ. (Ex. 1002 ¶174.)

c. Limitation 1[pre][iii]: “operating in disparate computing environments”

Gélinas discloses this limitation under both VirtaMove’s interpretation (requiring “independently operable” computers) and the inventor’s interpretation (requiring computers with different network addresses). (*Supra*, §V.B.)

First, Gélinas discloses implementing containers on computers running Linux. (*Supra*, §I.C.1.) Linux computers were independently operable because one Linux computer could operate without being controlled by any other computer. (Ex. 1002 ¶176.)

Second, Gélinas discloses implementing containers on computers with different network addresses. For example, Gélinas discloses that each container “is assigned a host name and an IP number.” (Ex. 1007, 2.) Each container “is tied to one IP [so] several servers may run the same services without conflict.” (*Id.*, 16-18.)

To the extent “disparate computing environments” requires some other type of difference, Gélinas would have rendered such an environment obvious. Gélinas discloses that “[m]ost hardware issues,” such as disk configuration, network configuration, processor type, and number of processors, “are irrelevant for the virtual server installation”—meaning that Gélinas’s system can operate in environments with a variety of different computer configurations. (Ex. 1007, 12; Ex. 1002 ¶178.) For example, Gélinas’ containers can operate in environments comprising “an IDE + uniprocessor server” or “SCSI + multiprocessor server.” (Ex. 1007, 30.)

d. Limitation 1[pre][iv]: “wherein each server includes a processor and an operating system including a kernel”

Gélinas uses servers with processors. (Ex. 1007, 30 (“uniprocessor” or “multiprocessor”).) Further, the servers run Linux, which includes a kernel. (*Id.*, 12-13; Ex. 1002 ¶179.)

e. Limitation 1[pre][v]: “a set of associated local system files compatible with the processor”

“System files” include “shared libraries” and “configuration files.” (Ex. 1001, 2:52-54.) Gélinas discloses both. (Ex. 1007, 10 (shared libraries and configuration files); *id.*, 16, 24 (configuration files).) A POSITA would understand that the disclosed system files were compatible with the processor on which Linux was running; otherwise the applications that use the files would not work. (Ex. 1002 ¶180.)

Additionally, the ’814 patent admits that Apache used shared libraries and configuration files. (Ex. 1001, 2:55-3:20.) Gélinas discloses Apache on its servers. (Ex. 1007, 17, 21.)

Finally, Gélinas discloses running Linux distributions such as “debian” or “redhat.” (*Id.*, 29.) These distributions contained numerous shared libraries and configuration files compatible with processors on which they ran. (Ex. 1002 ¶182; *see also* Ex. 1001, 2:52-56 (system files supplied by Linux distribution).)

- f. **Limitation 1[pre][vi]: “a method of providing at least some of the servers in the system with secure, executable applications related to a service”**

Gélinas’ containers “run pretty much any services,” including “telnet, mail servers, web servers, [and] SQL servers.” (Ex. 1007, 19, 11.) Each container acts as a “security box” that confines applications running in it. (*Id.*, 1; Ex. 1002 ¶183.)

- g. **Limitation 1[pre][vii]: “wherein the applications are executed in a secure environment”**

Gélinas’ containers “provide a higher level of security.” (Ex. 1007, 27; Ex. 1002 ¶184.) They can “[r]un un-trusted applications with complete control over their interaction with the rest of the computer and the network.” (Ex. 1007, 1.)

- h. **Limitation 1[pre][viii]: “wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service”**

As discussed above, Gélinas’ containers ran mail servers, web servers, and other services. (*Supra* §VI.C.1.f.) The applications used to run such services included executable objects that perform tasks related to the services. (Ex. 1002 ¶185.)

- i. **Limitation 1[a][i] “the method comprising: storing in memory accessible to at least some of the servers a plurality of secure containers of application software”**

Gélinas discloses VServer virtual servers, which are containers. (*Supra*, §I.C.1.) Installing such a container “copies a linux installation inside a

sub-directory.” (Ex. 1007, 9.) This sub-directory is stored on a disk, which is a type of memory. (*Id.*; Ex. 1002 ¶186.) Gélinas further discloses that one may “run several [containers] on the same box,” which “you will certainly do.” (Ex. 1007, 9.)

j. Limitation 1[a][ii]: “each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications”

As discussed above, Gélinas discloses’ containers run “mail servers, web servers, SQL servers,” and the like. (Ex. 1007, 11.) A POSITA would have understood that such services comprise one or more executable applications and a set of associated system files because libraries and configuration files were needed to execute such applications. (Ex. 1002 ¶187.) For example, the ’814 patent admits that “Linux Apache [a common web server application] uses the following shared libraries ... which are ‘system’ files,” and then lists 21 libraries and configuration files. (Ex. 1001, 2:55-3:19.) Gélinas discloses Apache in its containers. (Ex. 1007, 17, 21; Ex. 1002 ¶187.)

k. Limitation 1[a][iii]: “for use with a local kernel residing permanently on one of the servers”

Gélinas discloses that its containers are “sharing the same kernel” and “do not need kernel packages,” which saves disk space and promotes efficiency. (Ex. 1007, 26, 29.) Based on this disclosure, a POSITA would understand that the containers make use of a local kernel on the underlying server. (Ex. 1002 ¶188.)

Further, Gélinas discloses how to install a kernel (outside of the containers) but says nothing about moving or uninstalling it. (Ex. 1007, 13-15.) Thus, it would have been obvious to a POSITA that the kernel resides permanently on the server. (Ex. 1002 ¶189.)

l. Limitation 1[a][iv]: “wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems”

Gélinas discloses that its containers operate “like a normal Linux server” and run “normal services such as telnet, mail servers, web servers, SQL servers.” (Ex. 1007, 11.) A POSITA would understand that these services would be unable to run on Linux unless their associated system files were compatible with the kernel. (Ex. 1002 ¶190.) Indeed, Gélinas discloses that “[o]nce a service [is] running in a vserver, it is talking directly to the kernel.” (Ex. 1007, 29; *see also id.*, 16-18.)

m. Limitation 1[a][v]: “the containers of application software excluding a kernel”

Gélinas contrasts its approach, in which the containers are “sharing the same kernel,” with a virtual machine approach in which each machine “is running its own kernel.” (Ex. 1007, 26.) Thus, the containers exclude a kernel. (Ex. 1002 ¶191.)

- n. **Limitation 1[a][vi]: “wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server”**

Gélinas discloses that a container may be created by “copy[ing] some parts of the current server.” (Ex. 1007, 15-16, 29 (“A vserver is normally created from the root server.”).) In this case, system files resident on the server are copied to the container using the command “cp -ax /sbin /bin /etc /usr /var /dev /lib” (or similar). (Ex. 1007, 15; *see* Ex. 1001, 2:55-3:16 (admitting that at least /bin, /usr, /etc, and /lib directories hold system files on Linux).) Because processes in a container cannot access files outside the container, a POSITA would understand that the copies in the container are utilized in place of the system files which remain on the underlying server. (Ex. 1002 ¶192 (explaining contents of copied directories).)

- o. **Limitation 1[a][vii]: “wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server”**

System files in Gélinas’ containers are copies of associated local system files that remain resident on the underlying server for the reasons explained above. (*Supra* §VI.C.1.n; Ex. 1002 ¶193.)

p. Limitation 1[a][viii]: “wherein the application software cannot be shared between the plurality of secure containers of application software”

Gélinas discloses five ways in which containers are isolated from each other. (Ex. 1007, 2-3.) This isolation prevents applications in one container from interfering with applications in other containers. (*Id.*, 1-3.) Although an administrator can configure containers to share files, such configuration is optional. (Ex. 1007, 31; *see also id.* 9-10.) Unless an administrator specifically allows it, application software cannot be shared between the containers. (*Id.*; Ex. 1002 ¶194.)

q. Limitation 1[a][ix]: “wherein each of the containers has a unique root file system that is different from an operating system’s root file system”

Gélinas discloses that “[t]he vserver is trapped into a sub-directory of the main server and can’t escape. This is done by the standard chroot() system call found on all Unix and Linux boxes.” (Ex. 1007, 2.) “As such, [containers] can only see what is under their / directory” and not the operating system’s full “/” directory. (Ex. 1007, 31.) A container’s “/” directory is its root, and that root is different from the operating system’s root. (*Id.*; Ex. 1002 ¶195.)

- r. **Limitation 16: “creating containers prior to said step of storing containers in memory, wherein a step of creating containers includes: using a skeleton set of system files as a container starting point and installing applications into that set of files.”**

Gélinas discloses creating a new container by copying directories from an existing Linux installation into the new container’s file system: “/sbin /bin /etc /usr /var /dev /lib.” (Ex. 1007, 15.) A POSITA would recognize these as standard directories that contained system files in a Linux installation. (Ex. 1002 ¶211.)

The copied directories would provide “a skeleton set of system files” used as a “container starting point” because Gélinas discloses installing additional packages into the container once it is created. (Ex. 1007, 11; Ex. 1002 ¶212.) Specifically, Gélinas discloses installing “standard packages (RPMs for example).” (*Id.*) “RPMs” are packages in the Red Hat Package Manager (RPM) format. (Ex. 1002 ¶212.) Installing such packages in a container would cause applications in the packages to be installed into the standard directories that contain the system files discussed above. (*Id.*) Thus, Gélinas renders claim 16 obvious. (*Id.*)

VII. SECONDARY CONSIDERATIONS OF NONOBVIOUSNESS

Secondary considerations should be considered but do not control the obviousness conclusion. *Newell Cos. v. Kenney Mfg. Co.*, 864 F.2d 757, 768 (Fed. Cir. 1988). Where, as here, a strong *prima facie* obviousness showing exists, even relevant secondary considerations may not dislodge the obviousness conclusion.

Leap-frog Enters. v. Fisher-Price, Inc., 485 F.3d 1157, 1162 (Fed. Cir. 2007).

Petitioner is aware of no evidence supporting a claim for secondary considerations.

VIII. DISCRETIONARY DENIAL IS UNWARRANTED

Pursuant to and in reliance on Acting Director Coke M. Stewart's March 26, 2025, Memorandum regarding Interim Processes for PTAB Workload Management, Petitioner understands that discretionary denial issues, if any, will be raised in a separate brief to be filed by Patent Owner. If Patent Owner files such a brief, Petitioner intends to respond in an opposition brief consistent with Acting Director Coke M. Stewart's March 26, 2025, Memorandum regarding Interim Processes for PTAB Workload Management. Accordingly, Petitioner will not address discretionary denial issues in this Petition other than to note that (a) Petitioner is filing concurrently with this Petition a Motion for Joinder with Amazon's IPR2025-00566 for the '814 patent, and, if joined, Petitioner would be taking an understudy role and the Board's finite resources would not be impacted; (b) Microsoft filed on April 18, 2025, along with its Petition for Inter Partes Review of the '814 patent that is "substantively identical to" Amazon's IPR2025-00566, a Motion for Joinder to the same Amazon's IPR2025-00566, and Microsoft likewise stated there that it will, if joined, accept an understudy role. *See Microsoft Corp. v. VirtaMove, Corp.*, IPR2025-00852, Paper 3 (PTAB Apr. 18, 2025).

IX. MANDATORY NOTICES

Petitioner Oracle Corporation (“Oracle”) is the Real Party-in-Interest. No other parties exercised or could have exercised control over this Petition. No other parties funded or directed this Petition. *See* Office Patent Trial Practice Guide, 77 Fed. Reg. 48759-60.

A. Related Matters

1. United States Patent & Trademark Office

The application from which U.S. Patent No. 7,519,814 issued claims priority to two provisional applications: No. 60/502,619, filed September 15, 2003 and No. 60/512,103, filed October 20, 2003.

The following U.S. patent applications claim the benefit of priority to U.S. Patent 7,519,814:

(i) U.S. Patent Application 11/432,843 (U.S. Patent No. 7,757,291), filed May 12, 2006;

(ii) U.S. Patent Application 11/380,285 (U.S. Patent No. 7,774,762), filed April 26, 2006;

(iii) U.S. Patent Application 12/075,842 filed March 13, 2008.

2. USPTO Patent Trial and Appeal Board

Concurrently with the present petition, Petitioner is also filing IPR2025-00964, IPR2025-00965, and IPR2025-01001, challenging U.S. Patent No. 7,519,814 (the “’814 patent”).

Petitioner is also filing IPR2025-00966, IPR2025-00981, and IPR2025-00982, challenging U.S. Patent No. 7,784,058 (the “’058 patent”), which is also asserted in *VirtaMove, Corp. v. Oracle Corporation*, Case No. 7:24-cv-00339-ADA, listed below.

Other *inter partes* review proceedings filed against the ’814 or ’058 patents include:

(i) *Google LLC v. VirtaMove, Corp.*, Case No. IPR2025-00487 (PTAB, filed January 31, 2025);

(ii) *Google LLC v. VirtaMove, Corp.*, Case No. IPR2025-00488 (PTAB, filed January 31, 2025);

(iii) *Google LLC v. VirtaMove, Corp.*, Case No. IPR2025-00489 (PTAB, filed January 31, 2025);

(iv) *Google LLC v. VirtaMove, Corp.*, Case No. IPR2025-00490 (PTAB, filed January 30, 2025);

(v) *Amazon.com, Inc. v. VirtaMove, Corp.*, Case No. IPR2025-00561 (PTAB, filed January 31, 2025);

(vi) *Amazon.com, Inc. v. VirtaMove, Corp.*, Case No. IPR2025-00563
(PTAB, filed January 31, 2025);

(vii) *Amazon.com, Inc. v. VirtaMove, Corp.*, Case No. IPR2025-00566
(PTAB, filed January 31, 2025);

(viii) *International Business Machines Corp. v. VirtaMove, Corp.*, Case No.
IPR2025-00591 (PTAB, filed February 6, 2025);

(ix) *International Business Machines Corp. v. VirtaMove, Corp.*, Case No.
IPR2025-00599 (PTAB, filed February 7, 2025);

(x) *Microsoft Corp. v. VirtaMove, Corp.*, Case No. IPR2025-00849
(PTAB, filed April 18, 2025);

(xi) *Microsoft Corp. v. VirtaMove, Corp.*, Case No. IPR2025-00850
(PTAB, filed April 18, 2025);

(xii) *Microsoft Corp. v. VirtaMove, Corp.*, Case No. IPR2025-00851
(PTAB, filed April 18, 2025);

(xiii) *Microsoft Corp. v. VirtaMove, Corp.*, Case No. IPR2025-00852
(PTAB, filed April 18, 2025);

(xiv) *Microsoft Corp. v. VirtaMove, Corp.*, Case No. IPR2025-00853
(PTAB, filed April 18, 2025);

(xv) *Microsoft Corp. v. VirtaMove, Corp.*, Case No. IPR2025-00854
(PTAB, filed April 18, 2025);

(xvi) *Microsoft Corp. v. VirtaMove, Corp.*, Case No. IPR2025-00855 (PTAB, filed April 18, 2025).

3. U.S. District Court for the Eastern District of Texas

(i) *VirtaMove, Corp. v. Hewlett Packard Enterprise Company*, Case No. 2:24-cv-00093;

(ii) *VirtaMove, Corp. v. International Business Machines Corp.*, Case No. 2:24-cv-00064.

4. U.S. District Court for the Western District of Texas

(i) *VirtaMove, Corp. v. Google LLC*, Case No. 7:24-cv-00033 (transferred to Northern District of California per Order dated May 7, 2025, *see* Ex. 1028);

(ii) *VirtaMove, Corp. v. Amazon.com, Inc. et al.*, Case No. 7:24-cv-00030 (pending transfer to Northern District of California per Order dated February 19, 2025, *see* Docket Entry No. 94);

(iii) *VirtaMove, Corp. v. Microsoft Corp.*, Case No. 7:24-cv-00338;

(iv) *VirtaMove, Corp. v. Oracle Corp.*, Case No. 7:24-cv-00339.

5. U.S. District Court for the Northern District of California

(i) *Red Hat, Inc. v. VirtaMove, Corp.*, Case No. 5:24-cv-04740;

(ii) *VirtaMove, Corp. v. Google LLC*, Case No. 5:25-cv-00860.

B. Counsel, Service, and Fee Information

Lead Counsel	Back-Up Counsel
<p>Bas de Blank Registration No. 74,930 (M2BPTABDocket@orrick.com)</p> <p>Postal & Hand-Delivery Address: Orrick, Herrington & Sutcliffe LLP 1000 Marsh Road Menlo Park, CA 94025 Tel: 650-614-7400 Fax: 650-614-7401</p> <p><i>Attorney for Petitioner Oracle Corporation</i></p>	<p>Parth Sagdeo Registration No. 71,275 (PTABDocketP2S7@orrick.com)</p> <p>Jared Bobrow <i>Pro Hac Vice</i> to be submitted (PTABDocketJ3B3@orrick.com)</p> <p>Postal & Hand-Delivery Address: Orrick, Herrington & Sutcliffe LLP 1000 Marsh Road Menlo Park, CA 94025 Tel: 650-614-7400 Fax: 650-614-7401</p> <p>Shane Anderson <i>Pro Hac Vice</i> to be submitted (PTABDocket9SA@orrick.com)</p> <p>Postal & Hand-Delivery Address: Orrick, Herrington & Sutcliffe LLP 355 S. Grand Ave. Ste. 2700 Los Angeles, CA 90071 Tel: 213-629-2020 Fax: 213-612-2499</p> <p><i>Attorneys for Petitioner Oracle Corporation</i></p>

Petitioner consents to service by electronic mail at the following addresses:

M2BPTABDocket@orrick.com, PTABDocketJ3B3@orrick.com,
PTABDocket9SA@orrick.com; PTABDocketP2S7@orrick.com, and oracle-

virtamove_ohs@orrick.com. Pursuant to 37 C.F.R. §42.10(b), Petitioner attaches a Power of Attorney.

C. Payment of Fees (37 C.F.R. §42.103)

The fee set forth in 37 C.F.R. §42.15(a) for this petition has been paid. Any additional fees due in connection with this petition may be charged to the deposit account of Orrick, Herrington, & Sutcliffe LLP: 15-0665.

D. Grounds for Standing (37 C.F.R. §42.104(a))

Oracle certifies that the '814 patent is available for IPR and that Oracle is not barred or estopped from requesting IPR. This petition is being filed within one year of service of the district court Complaint

X. CONCLUSION

Petitioner respectfully requests cancellation of the challenged claims.

Respectfully submitted,

ORRICK, HERRINGTON & SUTCLIFFE LLP

Dated: May 16, 2025

/Bas de Blank/

Bas de Blank

Orrick, Herrington & Sutcliffe LLP

Lead Counsel for Petitioner

Registration No. 74,930

1000 Marsh Road

Menlo Park, CA 94025

Tel: 650-614-7400

Fax: 650-614-7401

Email: m2bptabdocket@orrick.com

Attorney for Petitioner Oracle Corporation

CERTIFICATE OF COMPLIANCE

The undersigned certifies that the foregoing PETITION FOR *INTER PARTES* REVIEW complies with the type volume limitation in 37 C.F.R. § 42.24(a). According to the utilized word-processing system's word count, the petition—excluding the caption, table of contents, table of exhibits, mandatory notices, certificate of word count, and certificate of service—contains 13,379 words.

Dated: May 16, 2025

By: */Bas de Blank/*

Bas de Blank
Orrick, Herrington & Sutcliffe LLP
Lead Counsel for Petitioner
Registration No. 74,930
1000 Marsh Road
Menlo Park, CA 94025
Tel: 650-614-7400
Fax: 650-614-7401
Email: m2bptabdocket@orrick.com

*Attorney for Petitioner Oracle
Corporation*

CERTIFICATE OF SERVICE

The undersigned hereby confirms that the foregoing Petition for *Inter Partes* Review and associated documents and exhibits were caused to be served on May 16, 2025 via overnight courier upon the following counsel of record for Patent Owner:

Allen, Dyer, Doppelt + Gilchrist, PA
1135 East State Road 434, Suite 3001
Winter Springs, FL 32708

Copies of this Petition and accompanying documents and exhibits were also served via electronic mail on Patent Owner's counsel of record for related PTAB proceedings and in the related district court litigation – Russ, August & Kabat:

Reza Mirzaie (rmirzaie@raklaw.com)
Marc A. Fenster (mfenster@raklaw.com)
Neil Rubin (nrubin@raklaw.com)
James A. Milkey (jmilkey@raklaw.com)
Qi (Peter) Tong (ptong@raklaw.com)
Jacob R. Buczko (jbuczko@raklaw.com)
Christian W. Conkle (cconkle@raklaw.com)
Jefferson Cummings (jcummings@raklaw.com)
Daniel B Kolko (dkolko@raklaw.com)
Jonathan Ma (jma@raklaw.com)
Mackenzie Paladino (mpaladino@raklaw.com)
James S. Tsuei (jtsuei@raklaw.com)

/Karen Johnson/

Karen Johnson