UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMAZON.COM, INC.,
Petitioner,

v.

VIRTAMOVE CORP.,
Patent Owner.

IPR2025-00563
U.S. Patent No. 7,519,814

DECLARATION OF DARRELL D. E. LONG, Ph.D. IN SUPPORT OF PETITION FOR INTER PARTES REVIEW OF U.S. PATENT NO. 7,519,814

TABLE OF CONTENTS

I. BACKGROUND ---------- 1

A. Experience and Qualifications---------- 1

B. Materials Considered ---------- 6

II. APPLICABLE LEGAL STANDARDS ---------- 7

A. Claim Interpretation ---------- 8

B. Obviousness ---------- 8

III. PERSON OF ORDINARY SKILL IN THE ART ---------- 13

IV. TECHNOLOGY BACKGROUND ---------- 14

A. Containers Versus Virtual Machines ---------- 15

B. Containers Versus Shared Application Environment ---------- 15

C. Containers in the Prior Art ---------- 16

1. Gélinas' Linux VServer ---------- 16

2. Tucker's Solaris Zones ---------- 17

3. Osman's Zap Pods ---------- 18

V. THE '814 PATENT ---------- 19

A. Summary of the '814 Patent ---------- 19

B. Claim Construction ---------- 20

VI. CLAIMS 1-16 AND 31-34 OF THE '814 PATENT WOULD HAVE BEEN OBVIOUS---------- 22

A. Claims 1-15 and 31-34 Would Have Been Obvious in View of Osman. ---------- 22

1. Claim 1 ---------- 23

a. Limitation 1[pre][i]: "In a system having a plurality of servers" ---------- 23

b. Limitation 1[pre][ii]: "with operating systems that differ" ---------- 23

c. Limitation 1[pre][iii]: "operating in disparate computing environments" ---------- 24

d. Limitation 1[pre][iv]: "wherein each server includes a processor and an operating system including a kernel" ---------- 25

e. Limitation 1[pre][v]: "a set of associated local system files compatible with the processor" ---------- 25

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" ---------- 26

g. Limitation 1[pre][vii]: "wherein the applications are executed in a secure environment" ---------- 27

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" ---------- 27

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" ---------- 28

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"---------- 29

k. Limitation 1[a][iii]: "for use with a local kernel residing permanently on one of the servers"---------- 30

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" ---------- 30

m. Limitation 1[a][v]: "the containers of application software excluding a kernel"---------- 31

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" ---------- 31

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"---------- 32

p. Limitation 1[a][viii]: "wherein the application software cannot be shared between the plurality of secure containers of application software"---------- 33

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" ---------- 33

2. Claim 2 ---------- 34

3. Claim 3 ---------- 35

4. Claim 4 ---------- 35

5. Claim 5 ---------- 36

6. Claim 6 ---------- 36

7. Claim 7 ---------- 37

8. Claim 8 ---------- 37

9. Claim 9 ---------- 38

10. Claim 10 ---------- 39

11. Claim 11 ---------- 40

12. Claim 12 ---------- 40

13. Claim 13 ---------- 41

14. Claim 14 ---------- 42

a. Limitation 14[a][i]: "creating containers prior to said step of storing containers in memory, wherein containers are created by" ---------- 42

b. Limitation 14[a][ii]: "a) running an instance of a service on a server" ---------- 42

c. Limitation 14[a][iii]: "b) determining which files are being used" ---------- 42

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" ---------- 42

15. Claim 15 ---------- 43

a. Claim 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" ---------- 43

b. Claim 15[b]: "installing the container on a server" ---------- 43

c. Claim 15[c]: "testing the applications and files within the container." ---------- 44

16. Claim 31 ---------- 44

a. Limitation 31[pre]: "A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:" ---------- 44

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" ---------- 44

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" ---------- 44

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" ---------- 45

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" ---------- 45

f. Limitation 31[a][v]: "and associated system files for use in executing the one or more processes"---------- 46

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" ---------- 46

h. Limitation 31[a][vii]: "each container having its own execution file associated therewith for starting one or more applications"---------- 46

i. Limitation 31[a][viii]: "in operation, each container utilizing a kernel resident on the server"---------- 46

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"---------- 46

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"---------- 47

17. Claim 32 ---------- 47

18. Claim 33 ---------- 47

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"---------- 48

b. Limitation 33[a][ii]: "and for providing values alternate to values the kernel would have assigned in response to the system calls"---------- 48

c. Limitation 33[a][iii]: "so that the containers can run independently of one another without contention, in a secure manner"---------- 48

d. Limitation 33[a][iv]: "the values corresponding to at least one of the IP address, the host name and the Mac_Address."---------- 49

19. Claim 34 ---------- 49

a. Limitation 34[a]: "monitoring resource usage of applications executing;"---------- 49

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;"---------- 50

c. Limitation 34[c]: "comparing the monitored resource usage of the at least one respective application with the resource limits; and," ---------- 50

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."---------- 50

B. Claims 1, 2, 4, 6, 8-10, 13, and 16 Would Have Been Obvious in View of Tucker and Bandhole---------- 51

1. Claim 1 ---------- 53

a. Limitation 1[pre][i]: "In a system having a plurality of servers" ---------- 53

b. Limitation 1[pre][ii]: "with operating systems that differ"---------- 54

c. Limitation 1[pre][iii]: "operating in disparate computing environments"---------- 54

d. Limitation 1[pre][iv]: "wherein each server includes a processor and an operating system including a kernel" ---------- 55

e. Limitation 1[pre][v]: "a set of associated local system files compatible with the processor"---------- 56

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"---------- 56

g. Limitation 1[pre][vii]: "wherein the applications are executed in a secure environment"---------- 57

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"---------- 60

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" ---------- 62

p. Limitation 1[a][viii]: "wherein the application software cannot be shared between the plurality of secure containers of application software" ---------- 63

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" ---------- 64

2. Claim 2 ---------- 64

3. Claim 4 ---------- 65

4. Claim 6 ---------- 65

5. Claim 8 ---------- 65

6. Claim 9 ---------- 66

7. Claim 10 ---------- 67

8. Claim 13 ---------- 67

9. Claim 16 ---------- 68

C. Claims 1, 2, 4, 6, 8-10, 13-14, and 16 Would Have Been Obvious in View of Gélinas ---------- 69

1. Claim 1 ---------- 69

a. Limitation 1[pre][i]: "In a system having a plurality of servers" ---------- 69

b. Limitation 1[pre][ii]: "with operating systems that differ" ---------- 69

c. Limitation 1[pre][iii]: "operating in disparate computing environments" ---------- 70

d. Limitation 1[pre][iv]: "wherein each server includes a processor and an operating system including a kernel" ---------- 71

e. Limitation 1[pre][v]: "a set of associated local system files compatible with the processor" ---------- 72

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" ---------- 72

g. Limitation 1[pre][vii]: "wherein the applications are executed in a secure environment" ---------- 73

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" ---------- 73

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" ---------- 73

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" ---------- 74

k. Limitation 1[a][iii]: "for use with a local kernel residing permanently on one of the servers"---------- 75

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" ---------- 75

m. Limitation 1[a][v]: "the containers of application software excluding a kernel"---------- 76

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" ---------- 76

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"---------- 77

p. Limitation 1[a][viii]: "wherein the application software cannot be shared between the plurality of secure containers of application software"---------- 77

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" ---------- 78

2. Claim 2 ---------- 78

3. Claim 4 ---------- 79

4. Claim 6 ---------- 80

5. Claim 8 ---------- 80

6. Claim 9 ---------- 80

7. Claim 10 ---------- 81

8. Claim 13 ---------- 82

9. Claim 14 ---------- 82

a. Limitation 14[a][i]: "creating containers prior to said step of storing containers in memory, wherein containers are created by" ---------- 83

b. Limitation 14[a][ii]: "a) running an instance of a service on a server" ---------- 83

c. Limitation 14[a][iii]: "b) determining which files are being used" ---------- 83

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" ---------- 84

10. Claim 16 ---------- 84

VII. SECONDARY CONSIDERATIONS OF NONOBVIOUSNESS ---------- 85

VIII. CONCLUSION ---------- 85

I, Darrell D. E. Long, Ph.D., do hereby declare:

1. I am making this declaration at the request of Petitioner Amazon.com, Inc. ("Amazon"). I have been retained by Amazon as a technical expert in this matter.

2. I am being compensated for my work on this case. My compensation does not depend on the content of this Declaration or the outcome of these proceedings.

I. BACKGROUND

A. Experience and Qualifications

3. I am currently a Distinguished Professor emeritus of Engineering in the Jack Baskin School of Engineering at the University of California. I have been on the faculty of UC Santa Cruz since 1988. I am also currently a Distinguished Visiting Scholar at Santa Clara University.

4. I received a B.S. degree from San Diego State University in 1984, a M.S. from the University of California, San Diego, in 1986, and a Ph.D. from the University of California, San Diego in 1988, all in computer science.

5. My previous work experience includes serving as Director of the Center for Research in Systems and Storage, a National Science Foundation Industry/University Cooperative Research Center, and the founding Director of the Storage Systems Research Center at the University of California, Santa Cruz. Before that time, I was a research assistant at the University of California, San Diego, and a

lecturer in Mathematics at San Diego State University. After coming to the University of California, Santa Cruz, in 1988, I worked up through the ranks from Assistant to Associate to Full Professor, culminating in my appointment as Distinguished Professor (the highest rank at the university). All this experience is covered in my curriculum vitae (attached as Ex. 1027), which provides a more detailed recitation of my employment history and tenure at various jobs.

6. I have also held numerous positions at the University of California and various government agencies and laboratories. In particular, I served as the vice chair and later chair of the University of California Committee on Research Policy. I have served on the University of California President's Council on the National Laboratories and on the Science & Technology, National Security, and Intelligence committees for those laboratories. I also serve on the University of California Academic Council Special Committee on Laboratory Issues (ACSCOLI). I served for several years on the National Research Council's Standing Committee on Technology Insight-Gauge, Evaluate and Review (TIGER), on the Committee on Defense Intelligence Agency Technology Forecasts and Reviews, and on the National Research Council's Committee on Science and Technology for Defense Warning. I recently served on the Intelligence Science and Technology Experts Group (ISTEG) for the National Academies of Sciences, Engineering, and Medicine.

7. Additionally, I have held visiting faculty positions at the Université Paris–Dauphine (Paris IX), the Conservatoire National des Arts et Métiers, the Université Paris–Descartes (Paris V), Sorbonne Université, the University of Technology, Sydney, the Center for Communications Research, the United States Naval Postgraduate School and Professor ad Honorem de la Universidad Católica del Uruguay. I was a European Organization for Nuclear Research (CERN) Associate Member.

8. I am conversant with several computer languages (for example, C, C++, FORTRAN, Go, LISP, Pascal, Perl, Java and others), scripting languages (various Unix/Linux shells, JavaScript, Python, and Tcl), markup languages (HTML, XHTML, XML, and numerous XML applications), operating systems (Unix and Linux of all varieties; MacOS; NetWare; Windows NT; Windows 2000, XP, Vista, Windows 7, 8, and 10, and Windows Server 2003, 2008, 2012, and 2016), and network communication protocols (TCP/IP, IPv4, and IPv6, plus numerous older stacks including IPX/SPX, NetBIOS/NetBEUI, and AppleTalk).

9. I currently am a member of various professional societies, including the Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), the International Association for Cryptologic Research (IACR) and the American Association for the Advancement of Science (AAAS). I was elected a Fellow of the IEEE in 2006, and a Fellow of the AAAS in 2008. I served

on the IEEE Fellows Evaluation Committee in 2007, 2010, and from 2011 to 2014. From 2011 to 2012, I was a Member and Chair Emeritus for the IEEE Reynold B. Johnson Information Storage Systems Award Committee. From 2008 to 2011, I served as the Chair for the IEEE Reynold B. Johnson Information Storage Systems Award Committee and was a member of the IEEE Technical Field Awards Council. I am the inaugural Editor-in-Chief of the IEEE Letters of the Computer Society (LOCS), the Editor-in-Chief emeritus of the ACM Transactions on Storage (TOS). In 2002, I founded the Conference on File and Storage Technologies (FAST), one of the most prestigious venues in the computer data storage field.

10. I have co-authored numerous peer-reviewed journal articles in the space of secure computing, big data, computer systems, networking, and data storage. In particular, these peer-reviewed journals include *ACM Transactions on Storage*, *ACM/Baltzer Mobile Networks and Applications Journal*, *Cluster Computing Journal*, *Computer Networks*, *Computing Systems*, *Future Generation Computer Systems*, *IEEE Latin America Transactions*, *IEEE Micro*, *IEEE Transactions on Big Data*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Computers*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Magnetics*, *International Journal in Computer Simulation*, *Internet Computing*, *Journal of Software Engineering and Knowledge Engineering*, and *the Journal of the ACM*.

11. I am a co-inventor on eleven different patent applications that have been issued. These patents include, for example, U.S. Patent No. 5,889,992, titled "Predictive Event Tracking Method," U.S. Patent No. 6,405,315, titled "Decentralized Remotely Encrypted File System," and U.S. Patent No. 6,792,424, titled "System and Method for Managing Authentication and Coherency in a Storage Area Network."

12. I have written a broad range of other computing publications. For example, I was a co-author of a book titled *Avoiding Surprise in an Era of Global Technology*, which was peer-reviewed and published in 2005. In 1989, I co-authored another scholarly book with John L. Carroll titled *Theory of Finite Automata*. I have co-authored chapters of various books, including *Distributed Data & Structures 4*, *Multimedia Communications: Directions and Innovations*, *Network Systems Design*, and *Progress in Simulation II*.

13. Based on my education and experience, I am an expert in at least the field of computer operating systems and operating environments. I have used my education, years of experience in this field, and my understanding of the perspective of a person of ordinary skill in the art to form the opinions expressed in this declaration.

B. Materials Considered

14. In preparing this Declaration, I have considered the following materials:

Exhibit No.	Description
1001	U.S. Patent No. 7,519,814 ("the '814 patent")
1003	Steven Osman et al., *The Design and Implementation of Zap: A System for Migrating Computing Environments*, 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	*Virtual Private Servers and Security Contexts* ("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.
1017	Message to Linux Kernel Mailing List Regarding Linux VServer
1018	Slashdot post Regarding Linux VServer

Exhibit No.	Description
1019	Petitioner's Opening Claim Construction Brief in *VirtaMove, Corp. v. Amazon.com, Inc. et al.*, No. 7:24-cv-30-ADA-DTG (W.D. Tex.) (the "Litigation")
1020	Patent Owner's Sur-Reply Claim Construction Brief from the Litigation
1021	Excerpts from deposition of named inventor Donn Rochette
1022	J. Ball, "Managing Initscripts with Red Hat's chkconfig." Linux Journal, 2001.
1023	Kravetz, Mike, et al. "Enhancing Linux scheduler scalability." *Proceedings of the Ottawa Linux Symposium, Ottawa, CA*. 2001.

15. I have also relied on my education, training, and experience, and my knowledge of pertinent literature in the field of the '814 patent.

II. APPLICABLE LEGAL STANDARDS

16. I have been asked to provide my opinion as to whether the claims of the '814 patent would have been obvious to a person of ordinary skill in the art at the time of the alleged invention, in view of the prior art.

17. I am a computer scientist by training and profession. The opinions I am expressing in this report involve the application of my training, technical

knowledge and experience to the evaluation of certain prior art with respect to the '814 patent.

18. Although I have been involved as a technical expert in patent matters before, I am not an expert in patent law. Therefore, the attorneys from Knobbe, Martens, Olson & Bear, LLP have provided me with guidance as to the applicable patent law in this matter. The paragraphs below express my understanding of how I must apply current principles related to patent validity to my analysis.

A. Claim Interpretation

19. Unless expressly stated herein, I have applied the plain and ordinary meaning of the claim terms, which I understand is the meaning that a person of ordinary skill in the art would have given to the claim terms at the time of the invention, based on a review of the intrinsic evidence (patent specification and file history). I have been instructed that the time of the alleged invention here is around September 13, 2004, which I understand to be the filing date of the '814 patent. My conclusions would not change if the invention date were slightly earlier or slightly later (e.g., within a year).

B. Obviousness

20. It is my understanding that a claim is "obvious" if the claimed subject matter would have been obvious to a person of ordinary skill in the art at the time of the alleged invention. I understand that an obviousness analysis involves a number

of considerations. I understand that the following factors must be evaluated to determine whether a claim would have been obvious: (1) the scope and content of the prior art; (2) the differences, if any, between each claim of the ’814 patent and the prior art; (3) the level of ordinary skill in the art in September 2004; and (4) additional considerations, if any, that indicate that the invention was obvious or not obvious. I understand that these “additional considerations” are often referred to as “secondary considerations” of non-obviousness or obviousness.

21. I also understand that the frame of reference when evaluating obviousness is what a hypothetical person of ordinary skill in the pertinent art would have known in September 2004. I understand that the hypothetical person of ordinary skill is presumed to have knowledge of all pertinent prior art references.

22. I understand that a prior art reference may be a pertinent prior art reference (or “analogous art”) if it is in the same field of endeavor as the patent or if it is pertinent to the problem that the inventors were trying to solve. Here, all of the references relied on in my obviousness analysis below are in the same field of endeavor as the ’814 patent, e.g., computing systems and operating system architectures. The references are also pertinent to a particular problem the inventor was focused on, e.g., confining or isolating applications in containers.

23. It is my understanding that the law recognizes several rationales for combining references or modifying a reference to show obviousness of claimed subject matter. Some of these rationales include:

- combining prior art elements according to known methods to yield predictable results;
- simple substitution of one known element for another to obtain predictable results;
- a predictable use of prior art elements according to their established functions;
- using known techniques to improve similar devices (methods, or products) in the same way;
- applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
- choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success (in which case a claim would have been obvious to try);
- known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art; and

- some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

24. I understand that "secondary considerations" must be considered as part of the obviousness analysis when present. I further understand that the secondary considerations may include: (1) a long-felt but unmet need in the prior art that was satisfied by the claimed invention; (2) the failure of others; (3) skepticism by experts; (4) commercial success of a product covered by the patent; (5) unexpected results achieved by the claimed invention; (6) industry praise of the claimed invention; (7) deliberate copying of the invention; and (8) teaching away by others. I also understand that evidence of the independent and nearly simultaneous "invention" of the claimed subject matter by others is a secondary consideration supporting an obviousness determination and may support a conclusion that a claimed invention was within the knowledge of a person of ordinary skill as of September 2004. I am not aware of any evidence of secondary considerations that would suggest that the claims of the '814 patent would have been nonobvious in September 2004.

25. I understand that when assessing obviousness, using hindsight is impermissible; that is, what is known today or what was learned from the teachings of the patent should not be considered. The patent should not be used as a road map

for selecting and combining items of prior art. Rather, obviousness must be considered from the perspective of a person of ordinary skill at the time the alleged invention was made, which is September 2004 in this case.

26. I also understand that an obviousness analysis must consider the invention as a whole, as opposed to just a part or element of the invention. I understand this "as a whole" assessment to require showing that one of ordinary skill in the art at the time of invention, confronted by the same problems as the inventor and with no knowledge of the claimed invention, would have selected the elements from the prior art and combined them in the claimed manner.

27. I understand that when a person of ordinary skill in the art would be unable to ascertain the scope of a claim with reasonable certainty, the claim is indefinite. I also understand that a patentee's definition of a claim term can render a patent claim indefinite if an ambiguity in the definition would prevent a person of ordinary skill from ascertaining the boundaries of the claim with reasonable certainty. But even where the outer boundaries of a claim term are indefinite, I understand that it nonetheless may be possible to identify examples that satisfy the claim term. When such examples are available, they may be compared to the prior art to support a finding of obviousness, even though the claim is indefinite.

28. It is my understanding that something is "inherent in," and therefore taught by, the prior art, if it necessarily flows from the explicit disclosure of the prior

art. I understand that the fact a certain result or characteristic may or may not be present in the prior art is insufficient to establish inherency. However, if the result or characteristic is necessarily present based upon the explicit disclosure in the prior art, it is inherent in the prior art and is therefore disclosed.

III. PERSON OF ORDINARY SKILL IN THE ART

29. It is my understanding that when interpreting the claims of the ’814 patent and evaluating whether a claim would have been obvious, I must do so based on the perspective of a person of ordinary skill in the art at the relevant priority date. I have been instructed to assume, for the purposes of my opinions, that the relevant priority date of the ’814 patent is September 13, 2004.

30. I understand that in determining the level of ordinary skill in the art, several factors are considered. Those factors may include: (1) the type of problems encountered in the art; (2) prior art solutions to those problems; (3) the rapidity with which innovations are made; (4) the sophistication of the technology; and (5) the education level of active workers in the field. A person of ordinary skill in the art must have the capability of understanding the scientific and engineering principles applicable to the pertinent art.

31. Based on my review of the specification and claims of the ’814 patent, it is my opinion that a person of ordinary skill in the art would have had a minimum of a bachelor’s degree in computer engineering, computer science, software

engineering, or a similar field, and approximately two years of industry or academic experience in a related field. Work experience could substitute for formal education and additional formal education could substitute for work experience.

32. My conclusions below that the claims of the ’814 patent would have been obvious would remain the same even if the priority date, field of endeavor, or level of ordinary skill in the art were slightly different.

33. I meet the above definition of a person of ordinary skill in the art, and did so as of September 2004. Also, I have worked with persons of ordinary skill in the art in my professional and academic experiences, and I have an understanding of their skill level around September 2004.

IV. TECHNOLOGY BACKGROUND

34. Computer systems that confine applications to containers were well-known in the art before the ’814 patent’s filing date in 2004. For example, one container system was released in 2001 for use with the popular Linux operating system. (Ex. 1007 (Gélinas).) Another container system was announced in 2002 by Sun Microsystems, one of the world’s leading computer manufacturers at the time. (Ex. 1009 (Solaris 10 press release from Sun Microsystems).) A third container system was released later in 2002 by Columbia University researchers, who presented their container system at a major conference sponsored by Google, HP, Intel, and Microsoft. (Ex. 1003 (Osman).)

A. Containers Versus Virtual Machines

35. The ’814 patent does not discuss earlier container systems. Instead, the patent presents itself as an advance over “Virtual Machine technology, pioneered by VmWare” and released commercially in 1999. (Ex. 1001 (the ’814 patent), 1:51-56; Ex. 1010 (VMware article from Linux Journal).) 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 (the ’814 patent), 1:27-56.) However, the patent identifies a “key difference” between the Virtual Machine (“VM”) approach and the patent’s container-based approach. (Ex. 1001 (the ’814 patent), 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.)

B. Containers Versus Shared Application Environment

36. The ’814 patent acknowledged that, outside of VMs, multiple applications could share an operating system in an environment called “SoftGrid.” (Ex. 1001 (the ’814 patent), 2:4-12.) The patent distinguished SoftGrid by asserting 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

37. The ’814 patent does not explain how it differs from the container systems that preceded it. Several such systems are described below.

1. Gélinas’ Linux VServer

38. 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.) This system would eventually become known as Linux VServer. By 2002, Gélinas’s website described that his VServer system “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 (Gélinas), 1.)

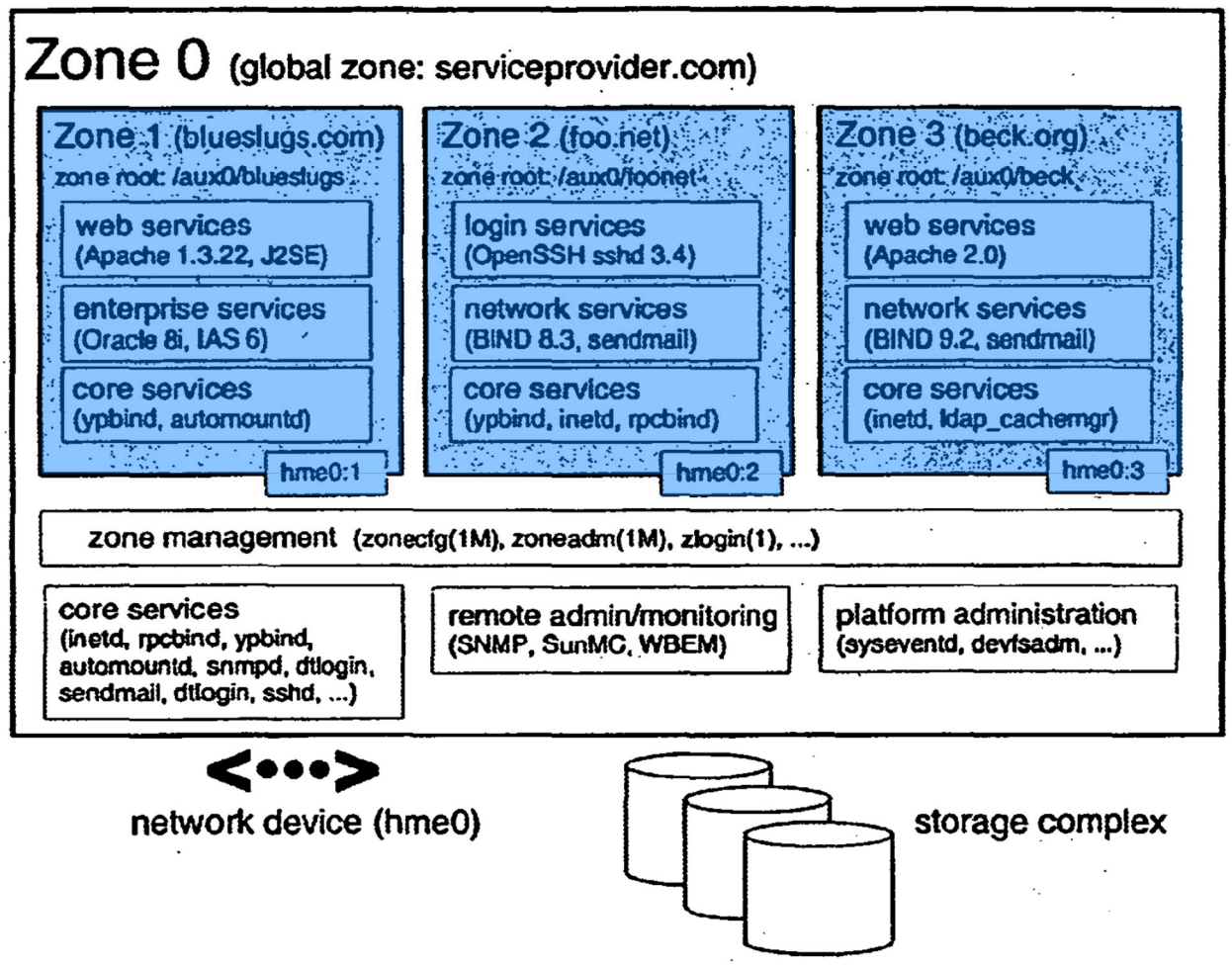
39. To provide isolation, VServer built on a command called “`chroot`,” which had been part of conventional operating systems for decades. The chroot command confined a particular application process to a limited view of the file system, locking it out of other areas of the file system. VServer went further by preventing processes from communicating with each other and isolating them from network resources. (Gélinas, 2-3.) VServer created secure containers that isolate and confine processes while allowing them to share one kernel.

2. Tucker's Solaris Zones

40. Sun Microsystems announced "containers" as part of Sun's Solaris 9 operating system release in 2002. (Ex. 1009, 2-3 (Solaris 9 press release from Sun Microsystems).) Sun's containers allowed "customers to run multiple applications on a single server, with fault, security and resource containment built-in." (*Id.*) The press release explained that "Solaris software containers will be delivered in phases, beginning with Solaris 9[.]" (*Id.*, 3.)

41. 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 (Tucker Provisional).) Sun's provisional explained that a "zone is an application container[.]" (Ex. 1009 (Solaris 9 press release from Sun Microsystems), 92.) That provisional eventually matured into an issued patent. (Ex. 1004 (Tucker).)

42. 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 (Tucker Provisional), 1-5.) Sun's provisional application 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 (Tucker Provisional), 3 (annotated).) Thus, Solaris Zones are also secure containers that isolate and confine processes while allowing the processes to share an underlying operating system.

3. Osman's Zap Pods

43. Containers were also the subject of academic research. In December 2002, researchers from Columbia University presented a paper on their system, called Zap, which was designed to allow groups of application processes to be easily moved from server to server. (*E.g.*, Ex. 1003 (Osman), 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.*)

V. THE '814 PATENT

A. Summary of the '814 Patent

44. The '814 patent describes a system of "associating applications with secure containers." (Ex. 1001 (the '814 patent), 7:4-15.) "Applications are segregated and execute with their own copies of files and with a unique identity." (*Id.*)

45. 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.)

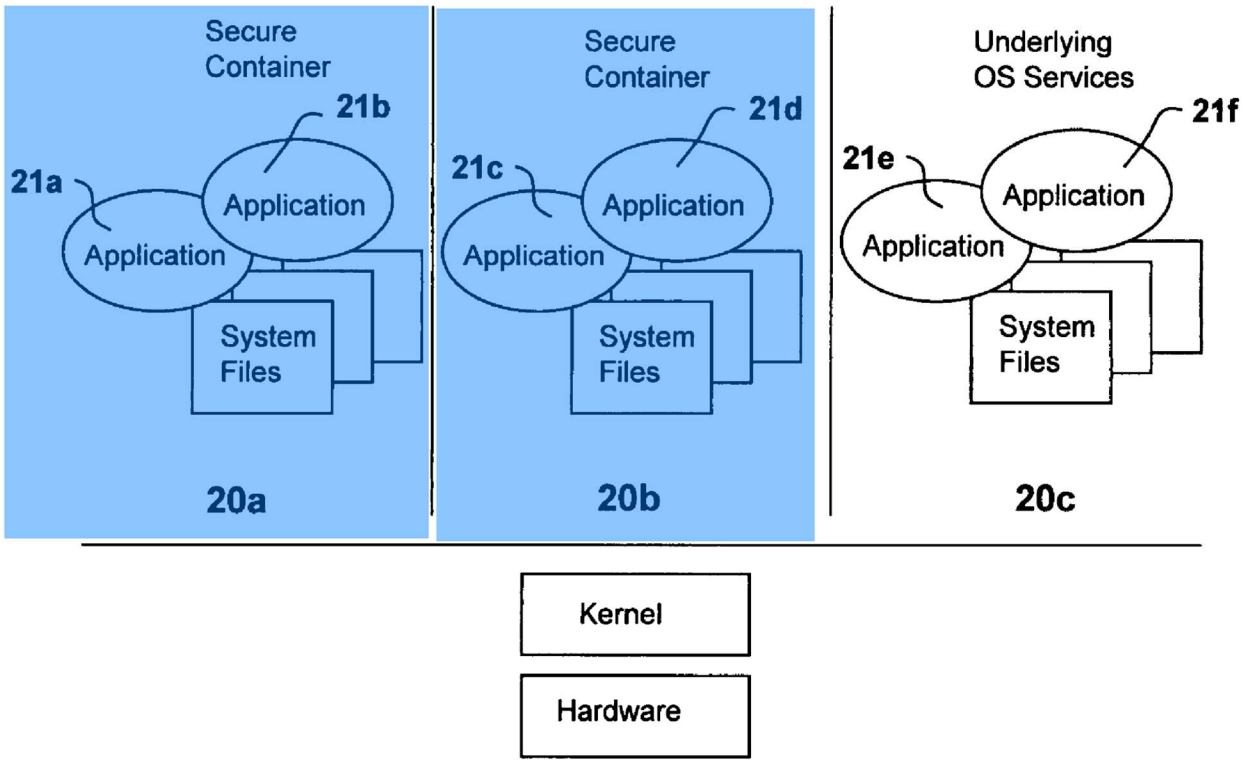


Figure 2

(Ex. 1001 (the '814 patent), Fig. 2.)

46. "[E]ach 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. Claim Construction

47. On January 26, 2024, the patent owner filed sued Amazon for infringement of the '814 patent in *VirtaMove Corp. v. Amazon.com, Inc.*, No. 7:24-cv-00030 (W.D. Tex.) ("the Litigation"). I understand that the parties proposed constructions for certain terms in the Litigation, but my opinion regarding the

obviousness of the claims does not depend on those constructions. The claims discussed below would have been obvious regardless.

48. The terms "container" and "system files," which are used in the claims, are defined in the patent's specification. (Ex. 1001 (the '814 patent), 2:16-3:19.) The prior art that I analyze below discloses the claims under the definitions in the specification and the plain meaning of the claims.

49. Amazon asserts in the Litigation that the phrase "disparate computing environments" recited in claim 1, is indefinite. (Ex. 1019 (Amazon's Claim Construction Brief), 3.) 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 (the '814 patent), 2:17-19.) The patent owner 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 (Patent Owner's Claim Construction Brief), 3.) In the patent owner's view, "disparate computing environments" would thus be limited to "environments run by … standalone computers." (*Id.*)

50. The patent owner 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." (Ex. 1020 (Patent Owner's Claim Construction Brief)

at 4.) The prior art here discloses "disparate computing environments" under the patent owner's interpretation. (¶¶ 57-59, 143-146, 175-178, discussed further below.)

51. Another potential interpretation of "disparate computing environments" was suggested by the patent's first named inventor, Donn Rochette, who explained 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 (Excerpts from Deposition Transcript of Donn Rochette), 73:18-74:18.) The prior art discloses "disparate computing environments" under Rochette's interpretation too. (¶¶ 57-59, 143-146, 175-178, discussed further below.)

52. Because the prior art discloses the claim limitations under any available interpretation, my opinion as to the obviousness of the claims would not be affected regardless of the interpretations of the above limitations.

VI. CLAIMS 1-16 AND 31-34 OF THE '814 PATENT WOULD HAVE BEEN OBVIOUS

A. Claims 1-15 and 31-34 Would Have Been Obvious in View of Osman.

53. Osman discloses or suggests each element of claims 1-15 and 31-34.

1. Claim 1

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

54. Claim 1 recites "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 (Osman), 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.

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

55. Claim 1 further recites "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 (Osman), 372.) The Linux kernel is the core of the Linux operating system, so a person of ordinary skill in the art would understand from this disclosure that Osman's system was used with Linux operating systems having minor kernel-version differences. Thus, Osman discloses this limitation.

56. Additionally, Osman discloses that its system lacks "dependencies on the host operating system." (*Id.*, 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 person of ordinary skill in the art would have found it obvious to use Osman's system with other operating systems in addition to Linux.

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

57. Claim 1 further recites "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 (Osman), 363.) This disclosure satisfies the claim limitation under the patent owner's interpretation, which requires "independently operable" computers. (¶¶ 49-52, above.)

58. Osman also discloses that pods running on different host machines have different network addresses. (Ex. 1003 (Osman), 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." (¶¶ 49-52, above.)

59. Thus, Osman discloses, or at least suggests, this limitation.

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

60. Claim 1 further recites "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 (Osman), 363.) A person of ordinary skill in the art would have understood that each such operating system included a kernel—which is the core of an operating system. (*see also* Ex. 1003 (Osman), 362 (Osman's system is "implemented … as a loadable kernel module").) Furthermore, Osman discloses implementing its system on servers "with a 933 MHz Intel Pentium-III" processor. (Ex. 1003 (Osman), 372.) Thus, Osman discloses this limitation.

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

61. Claim 1 further recites "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 (the '814 patent), 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 (Osman), 367 (describing example where "each pod maintains its own configuration" used to run

an "apache" web server).) These configuration files are in "private pod directories" that can be stored "locally on the host machine." (*Id.*) Thus, Osman discloses this limitation.

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"

62. Claim 1 further recites "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 (Osman), 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 (the '814 patent), 4:26-31.)

63. Further, applications in Osman's pods are secure because "[o]nly the host system administrator … or a user on the [access control list] are allowed to manipulate the pod[.]" (Ex. 1003 (Osman), 371.) Thus, Osman discloses this limitation.

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

64. Claim 1 further recites "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 (Osman), 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. (¶¶ 75-78, discussed further below.) Thus, Osman discloses this limitation.

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"

65. Claim 1 further recites "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 (Osman), 374 (listing tasks performed by Apache web server and eleven VNC applications).) For example, Osman discloses that its system may run a web server, which a person of ordinary skill in the art would have understood necessarily includes at least one object executable by the server's operating system

for performing a task related to hosting a web page. (*Id.*, 367, 373 (describing testing in which web server delivered "54 web pages").) Thus, Osman discloses this limitation.

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"

66. Claim 1 further recites "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 (the '814 patent), 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 (Osman), 367-68.) Osman's pods also satisfy the other requirements of containers that are recited later in the claims, as I explain below. (¶¶ 75-76, 91 (containers not sharing files), ¶ 69 (applications and system files in containers), ¶ 72 (no kernel in containers), ¶ 79 (execution file in containers), ¶ 70 (containers use server's kernel).)

67. 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." (Ex.

1003 (Osman), 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.)

68. Osman also discloses storing multiple pods together in memory accessible to one or more servers. (Ex. 1003 (Osman), 367-68 (pods stored on a "network file server" and on a local host's file system), 372, 374.) Thus, Osman discloses this limitation.

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"

69. Claim 1 further recites "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 (Osman), 374.) The VNC pod contained eleven applications, and the Apache pod contained one application. (*Id.*) Apache relies on system files as explained above. (¶ 61, above.) A person of ordinary skill in the art 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 (the '814 patent), 2:52-54), because shared libraries and

configuration files were routinely used in complex applications like the ones installed in the VNC pod. Accordingly, Osman discloses this limitation.

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

70. Claim 1 further recites "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 (Osman), 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.

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"

71. Claim 1 further recites "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 (Osman), 372.) A person of ordinary skill in the art 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. 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 (pre-migration) and the destination computer (post-migration). 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 they were migrated. Thus, Osman discloses, or at least suggests, this limitation.

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

72. Claim 1 further recites "the containers of application software excluding a kernel." As discussed above, Osman discloses that its system makes use of a local kernel. (¶ 70, above.) Further, Osman discloses that its system may "migrate pods across Linux kernels." (Ex. 1003 (Osman), 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 person of ordinary skill in the art would understand that the pod (container) does not include the kernel. Accordingly, Osman discloses this limitation.

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"

73. Claim 1 further recites "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 (Osman), 373.) As explained above, Apache uses system files to control its configuration. (¶ 61, above.) 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. 1003 (Osman), 367.) Thus, Osman discloses this limitation.

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"

74. Claim 1 further recites "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 (¶ 73, above.), the Apache configuration files inside the pod are copies or modified copies of the Apache configuration files

outside the pod. (Ex. 1003 (Osman), 367, 374.) Thus, Osman discloses this limitation.

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

75. Claim 1 further recites "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 (Osman), 367.) These private directories prevent applications running in different pods from interfering with each other. (*Id.*)

76. Osman also discloses that pods *may* be configured to share application software in a common location. (*Id.*) But absent such configuration, a person of ordinary skill in the art 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.*) Osman also discloses that processes in a pod are prevented from sharing memory with processes outside the pod. (Ex. 1003 (Osman), 364.) Thus, Osman discloses this limitation.

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"

77. Claim 1 further recites "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 that is different from the operating system's root file system. (Ex. 1003 (Osman), 367.) Thus, Osman discloses this limitation.

78. For the reasons above, claim 1 would have been obvious to a person of ordinary skill in the art in view of Osman.

2. Claim 2

79. Claim 2 depends on claim 1 and further recites "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 are created using a "`create_pod`" command whose options are specified in a configuration file, and those options include "what applications if any should be launched once the pod is created[.]" (Ex. 1003 (Osman), 371.) Thus, claim 2 would have been obvious in view of Osman.

3. Claim 3

80. Claim 3 depends on claim 2 and further recites "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 be used "with `/etc/init.d` for automatically starting up services like a web server" in a pod. (Ex. 1003 (Osman), 371.) In Linux, `init.d` was a directory used to store scripts that run when the system starts up. Scripts in `init.d` were executed in a specific order based on how they were named. For example, a script named "`S90mysql`" (to start a MySQL database application) would run before "`S95httpd`" (to start the Apache web server) because the number "90" comes before "95." (Ex. 1022 at 3 (emphasis added).) Thus, claim 3 would have been obvious in view of Osman.

4. Claim 4

81. Claim 4 depends on claim 1 and further recites "pre-identifying applications and system files required for association with the one or more containers prior to said storing step." Osman discloses claim 4's additional limitation because applications and system files associated with a pod are pre-identified when a pod is suspended on a first computer before being migrated and thus stored in memory a second computer. Specifically, when a pod is to be suspended, Osman's system saves "the information it needs to reconstruct the pod virtual file system, including a list of

all files opened by the processes within a pod." (Ex. 1003 (Osman), 368.) The "processes within a pod" include any applications running in the pod and the "files opened by the processes" include the system files that control the configuration of such applications. This information is identified on a first computer before the pod "migrates," and is therefore stored in memory on a second computer. (Ex. 1003 (Osman), 368.) Thus, claim 4 would have been obvious in view of Osman.

5. Claim 5

82. Claim 5 depends on claim 2 and further recites "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 (Osman), 367.) The configuration for each pod is stored in that pod's private directory and is therefore specific to that pod. (*Id.*) The '814 patent states that Apache configuration files are an example of system files. (Ex. 1001, 3:6-17.) Thus, claim 5 would have been obvious in view of Osman.

6. Claim 6

83. Claim 6 depends on claim 2 and further recites "assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address." Osman discloses

the additional limitation of claim 6 because each pod is assigned a "[p]er-pod IP address." (Ex. 1003 (Osman), 365; *id.*, 369.) Thus, claim 6 would have been obvious in view of Osman.

7. Claim 7

84. Claim 7 depends on claim 2 and further recites "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 (Osman), 372.) It would have been obvious to a person of ordinary skill in the art 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 a type of system file, according to the '814 patent. (Ex. 1001, 2:52-54.) Thus, claim 7 would have been obvious in view of Osman.

8. Claim 8

85. Claim 8 depends on claim 1 and further recites "wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers." Osman discloses claim 8's further limitation because its system "can use network file servers to support many pods running on many machines at the same time." (Ex. 1003 (Osman), 367.) The network file server

can store applications and other "common executables" used by the pods. (*Id.*) The network file server can also host the "private pod directory" for each pod, which stores system files (such as Apache configuration files). (*Id.*) The network file server (computer system) can also store all other information associated with a pod. (*Id.*, 365 (pods can be "checkpointed, suspended to secondary storage, and stored for future use"), 367 ("checkpointed data" can be stored on a network file server).) Thus, claim 8 would have been obvious in view of Osman.

9. Claim 9

86. Claim 9 depends on claim 2 and further recites "wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed." The claim contains a typographical error; it is missing a comma between "CPU" and "memory." (*Compare* Ex. 1001 (the '814 patent), claim 9 *with id.*, 8:47-51 ("CPU, memory, network bandwidth and disk usage"), Fig. 5 (listing "CPU" and "memory" separately).)

87. Osman discloses the additional limitation of claim 9 because, when a pod is migrated, its host saves "all process states, including memory, CPU registers, open file handles, etc." (Ex. 1003 (Osman), 365.) Accordingly, information relating to hardware resource usage, including at least CPU and memory usage, is associated

with a pod before the pod is restarted. And when the pod restarts, applications in it are executed. (Ex. 1003 (Osman), 371, 374.)

88. Additionally, Osman discloses "mounting various NFS mount points from a file server for pods … on the local machine." (Ex. 1003 (Osman), 372.) Based on this disclosure a person of ordinary skill in the art would understand that the system associates disk allocation information with a pod before starting the pod and executing applications.

89. Finally, Osman discloses experiments in which the disk allocation of particular pods was measured (e.g., "23 MB of image data") before such pods were started and applications executed. (Ex. 1003 (Osman), 374-375.)

90. Thus, claim 9 would have been obvious in view of Osman.

10. Claim 10

91. Claim 10 depends on claim 2 further recites "wherein in operation when an application residing within a container is executed, said application has no access to system files or applications in other containers or to system files within the operating system during execution thereof." Osman discloses the additional limitation of claim 10 because its system "uses the chroot call to set the [pod-specific] staging area as the root directory for the pod," which "prevent[s] processes with a pod from breaking out of their virtual file system environment." (Ex. 1003 (Osman), 367.) This call prevents processes running in the pod from accessing system files or

applications in other pods or outside of the pods (unless the administrator specifically configured the pods to allow such outside access). (¶¶ 75-76, above.) Thus, claim 10 would have been obvious in view of Osman.

11. Claim 11

92. Claim 11 depends on claim 2 and further recites that "containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location." Osman discloses claim 11's additional limitation because "various network-accessible directories that [a] pod is configured to access will be mounted from a network file server." (Ex. 1003 (Osman), 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, claim 11 would have been obvious in view of Osman.

12. Claim 12

93. Claim 12 depends on 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 claim 12's additional limitation by describing how a pod's virtual file system is staged when a pod is moved to a new host. (Ex. 1003

(Osman), 367-68.) The virtual file system merges "network-accessible directories that the pod is configured to access … from a network file server" with various files that are stored on the local host. (*Id.* (describing local files in "`/proc`" and "unlinked" local files).) The files stored on the local host are specified by parameters in the checkpointed state information, which I described above in connection with claim 11. (*Id.*, 365 (Table 1).) Thus, claim 12 would have been obvious in view of Osman.

13. Claim 13

94. Claim 13 depends on claim 1 and further recites "associating with a plurality of containers a stored history of when processes related to applications within the container are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application." Osman discloses the additional limitation of claim 13 because Osman's system maintains a "hash table" that contains identifiers for every process running in a pod. (Ex. 1003 (Osman), 365-66.) The system uses the information in this table to allocate "process resources" such as "shared memory." (*Id.*)

95. The hash table is also used to monitor the status of applications—for example, to determine whether an application is within a pod. (Ex. 1003 (Osman), 365-66.) By monitoring this information, the system prevents requests originating outside a pod from interfering with applications inside a pod. (*Id.*, 366 ("Outside of

the pod environment, the system call will reject any requests to physical resources which belong to pods").) Thus, claim 13 would have been obvious in view of Osman.

14. Claim 14

96. Claim 14 depends on claim 1. Osman discloses each additional limitation of claim 14.

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

97. Osman discloses pods running on a computer before they are suspended and stored, e.g., in network file storage. (Ex. 1003 (Osman), 365-371; ¶¶ 81, 85, above.) Such pods are created using a "`create_pod`" command. (*Id.* at 371.)

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

98. Osman discloses executing applications in a pod to provide a service. (¶¶ 62-63, above.) For example, Osman discloses running Apache. (*Id.*)

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

99. 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 (Osman), 368.) Apache is an example of an application running in a pod that is suspended. (*Id.*, 374.)

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"

100. Osman discloses running on copy of Apache outside a pod and a second copy inside a pod. (Ex. 1003 (Osman), 373; ¶¶ 73-74, above.) In this situation, a person of ordinary skill in the art 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.

101. Thus, Osman renders claim 14 obvious.

15. Claim 15

102. Claim 15 depends on claim 14 and recites three additional limitations.

a. Claim 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"

103. Osman discloses this limitation because each pod is assigned a "[p]er-pod IP address." (Ex. 1003 (Osman), 365; *id.*, 369.)

b. Claim 15[b]: "installing the container on a server"

104. Osman discloses installing containers on computers that act as servers. (Ex. 1003 (Osman) at 373 ("web server"); *see also* Ex. 1001, 12:5-7 ('814 patent stating that the terms "computing platform, server and computer are used interchangeably throughout this specification").)

c. Claim 15[c]: "testing the applications and files within the container."

105. Osman discloses testing the applications and files in the installed containers. (Ex. 1003 (Osman) at 373-75.) 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.*)

106. Thus, Osman renders claim 15 obvious.

16. Claim 31

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

107. Osman discloses computing systems performing tasks such as hosting web pages using a "number of worker processes," or hosting numerous applications (and thus numerous processes) in a "VNC thin-client computing user session." (Ex. 1003 (Osman), 374.)

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"

108. Osman discloses this limitation for the reasons I explained above in connection with Limitation 1[a][i].

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"

109. Osman discloses this limitation for the reasons I explained above in connection with Limitation 1[a][viii] and Limitation 1[a][ix]. Additionally, Osman 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 (Osman), 367.) This prevents files in a pod from being shared with other pods (unless the pods are specifically configured to allow such sharing).

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"

110. Osman discloses that each pod is assigned a "[p]er-pod IP address." (Ex. 1003 (Osman), 365; *id.*, 369.)

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"

111. Osman discloses executing eleven application programs related to a "VNC" service in a pod. (Ex. 1003 (Osman), 374.) Additionally, Osman discloses executing "Apache" (another application program) in a pod. (*Id.*) Each of these applications included one or more processes. (*Id.*, 374-75.) For example, Osman discloses that Apache used a "number of worker processes." (*Id.*, 374.)

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

112. Osman discloses this limitation for the reasons I explained above in connection with Limitation [1][pre][v] and Limitation 1[a][ii].

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"

113. Osman discloses this limitation for the reasons I explained above in connection with Limitation 1[a][vi] and Limitation 1[a][vii].

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

114. Osman discloses this limitation for the reasons I explained above in connection with Claim 2.

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

115. Osman discloses this limitation for the reasons I explained above in connection with Limitation 1[a][iii].

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"

116. Osman discloses this limitation for the reasons I explained above in connection with Limitation 1[a][v].

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"

117. Osman discloses a "loadable kernel module" that is used to "to intercept system calls as needed for virtualization[.]" (Ex. 1003 (Osman), 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.) Additionally, the kernel module intercepts system calls to control applications' access to network connections. (*Id.*, 370.)

118. Thus, Osman renders claim 31 obvious.

17. Claim 32

119. Claim 32 depends on 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 (Osman), 372.) This version of Linux (like all versions) included a scheduler for determining when each running process may utilize processing resources from a CPU. (Ex. 1023 (conference paper discussing Linux 2.4.x scheduler).) Thus, claim 32 would have been obvious in view of Osman.

18. Claim 33

120. Claim 33 depends on 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"

121. Osman discloses that its kernel module intercepts system calls from pods. (Ex. 1003 (Osman), 372.)

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

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

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

123. Osman's kernel module and system-call interception serve to "protect processes within a pod from dependencies on processes outside the pod." (Ex. 1003 (Osman), 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.)

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

124. 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 (Osman), 369-370.) The "addresses" here include "an IP address." (*Id.*)

125. Thus, Osman renders claim 33 obvious.

19. Claim 34

126. Claim 34 depends on claim 31 and recites that the "run time module" performs several additional steps. Osman's kernel module performs these additional steps as explained below.

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

127. Osman discloses monitoring resources such as memory, file systems, and networks by virtualizing those resources. (Ex. 1003 (Osman), 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.)

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;"

128. Osman's kernel module intercepts system calls from applications within a pod. (Ex. 1003 (Osman), 362, 365-66.) Because the system calls come from applications, they originate in user mode (where applications run). 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,"

129. Osman's kernel module compares the resources requested by an application (via system call) with limits on the set of resources available within a pod. (Ex. 1003 (Osman), 365 ("Operating system resources without corresponding pod virtual names are masked out of the respective pod's namespace."), 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."

130. Osman discloses that system calls which are not blocked are passed "on to the kernel for processing." (Ex. 1003 (Osman), 366.)

131. Thus, Osman renders claim 34 obvious.

B. Claims 1, 2, 4, 6, 8-10, 13, and 16 Would Have Been Obvious in View of Tucker and Bandhole

132. Tucker discloses "zones," which are application containers that Sun Microsystems implemented in its Solaris operating system. (¶¶ 40-42, above.) 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 (Tucker).)[1] But a person of ordinary skill in the art would have known at the time that Solaris servers were routinely networked with other servers to provide a wide array of services.

133. Bandhole describes a system in which servers and other computing resources are combined to provide various services. (Ex. 1006 (Bandhole).) Several examples in Bandhole involve a Solaris server combined with another server running a different operating system. (*Id.* ¶¶ [0005], [0034].)

134. 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." (Ex. 1006 (Bandhole) ¶ [0005].) A separate server using the Linux operating system runs "Apache web server software" as part of the same search

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

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.*)

135. A person of ordinary skill in the art would have been motivated to use Tucker's zones to implement the types of services described in Bandhole's background section. For example, a person of ordinary skill in the art would have found it obvious to run Bandhole's application server and database using two separate zones in accordance with Tucker. This approach would have been obvious for several reasons.

136. First, Tucker discloses that zones "limit the damage possible in the event of a security violation." (Ex. 1005 (Tucker), 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 person of ordinary skill in the art would have been motivated to use zones to run Bandhole's services to gain the security and deployment benefits that Tucker discloses.

137. 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). 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. 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.

138. Finally, because Bandhole's software was compatible with Solaris (Ex. 1006 (Bandhole) ¶ [0005]) and Tucker's zones ran on Solaris (¶¶ 40-42, above.), a person of ordinary skill in the art would have had a reasonable expectation of success in making the combination.

1. Claim 1

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

139. Claim 1 recites "In a system having a plurality of servers," and the '814 patent states that the terms "computing platform, server and computer are used interchangeably throughout this specification." (Ex. 1001 (the '814 patent), 12:5-7; 10:56-58 ("an existing server, for example a computer").)

140. 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 (Tucker), 2.) Thus, Tucker discloses this limitation.

141. 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 (Bandhole) ¶ [0005].)

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

142. Claim 1 further recites "with operating systems that differ." Bandhole's system includes Linux server and a Solaris server. (Ex. 1006 (Bandhole) ¶ [0005].) Linux and Solaris are different operating systems.

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

143. Claim 1 further recites "operating in disparate computing environments." Tucker and Bandhole disclose this limitation under both the patent owner's interpretation (requiring "independently operable" computers) and the inventor's interpretation (requiring computers with different network addresses). (¶¶ 49-52, above.)

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

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

146. 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 (Bandhole) ¶ [0005].) The Linux server and Solaris server are independently operable because they use distinct operating systems that do not control each other. Additionally, a person of ordinary skill in the art would understand that the servers use different network addresses to communicate with each other over the disclosed Ethernet LAN.

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

147. Claim 1 further recites, "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 (Tucker), 2.) Further, Tucker discloses that Solaris includes a kernel. (*E.g.*, *id.*, 4, 10, 21.)

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

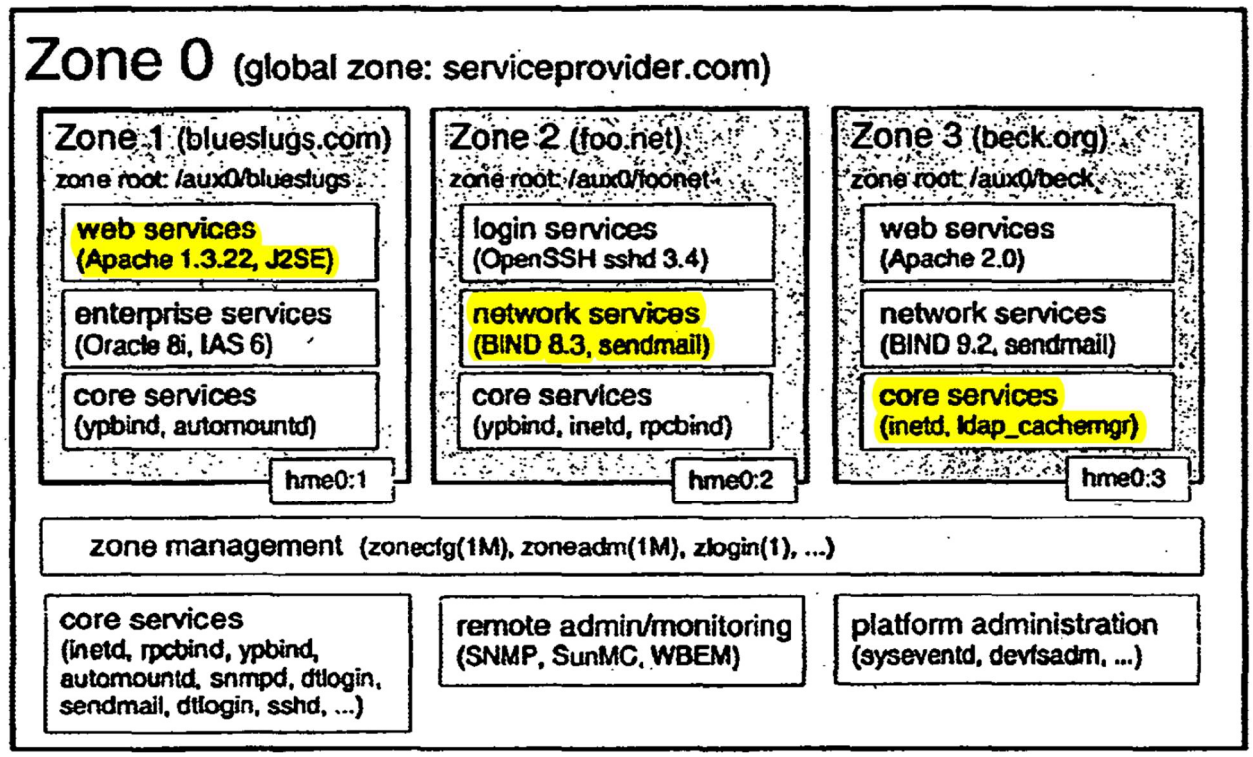
e. Limitation 1[pre][v]: "a set of associated local system files compatible with the processor"

149. Claim 1 further recites "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 (the '814 patent), 2:52-54.) Tucker discloses both. (Ex. 1005 (Tucker), 43-44, 71-72.) Additionally, the '814 patent admits that Apache, running on Linux, used both shared libraries and configuration files. (Ex. 1001 (the '814 patent), 2:55-3:19.) Bandhole discloses the use of Apache on Linux. (Ex. 1006 (Bandhole) ¶¶ [0005], [0034].) A person of ordinary skill in the art would have understood that these system files are compatible with the processor used to execute them.

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"

150. Claim 1 further recites "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 (Tucker), 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.)

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

151. Claim 1 further recites "wherein the applications are executed in a secure environment." Tucker 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 (Tucker), 1-2.) Thus, zones are a secure environment.

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"

152. Claim 1 further recites "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 (Tucker), 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.

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"

153. Claim 1 further recites "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 (Tucker), 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.*) 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 (Tucker), 10.) Finally, the zone is "running," meaning "processes have been started." (*Id.*) Thus, Tucker discloses (in at least the zone-installing step) storing a plurality of containers as this limitation requires.

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"

154. Claim 1 further recites "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 multiple system files (¶¶ 149-150, above). 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 (Tucker), 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 person of ordinary skill in the art 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. Solaris packages were the standard way of installing applications and their system files in the Solaris operating system.

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

155. Claim 1 further recites "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 (Tucker), 10-11, 4.) The kernel resides permanently on the servers because it remains in place as zones are added or removed. (Ex. 1005 (Tucker), 9-15.)

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"

156. Claim 1 further recites "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 (¶ 155, above.), a person of ordinary skill in the art 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.

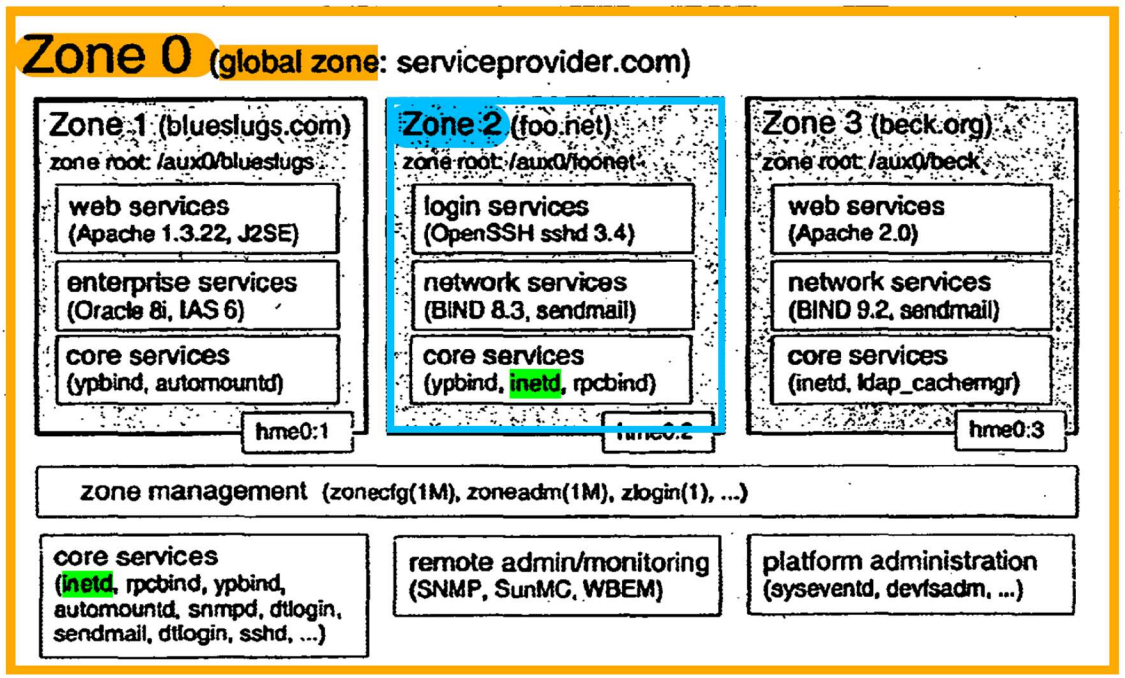
m. Limitation 1[a][v]: "the containers of application software excluding a kernel"

157. Claim 1 further recites "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 (Tucker), 11, 21.) The kernel also creates each of these zones. (*Id.*, 10.) Based on this disclosure, a person of ordinary

skill in the art would have understood that the kernel is outside of the non-global zones.

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"

158. Claim 1 further recites "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 (Tucker), 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.*, 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. 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.

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"

159. Claim 1 further recites "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. (¶ 158, above.) 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. Indeed, Tucker confirms that one zone

will sometimes duplicate system files from other zones leading to a "heavier disk footprint." (Ex. 1005 (Tucker), 72.)

160. Tucker also discloses that "different versions of the same application may be run without negative consequence in different zones." (Ex. 1005 (Tucker), 2.) Based on this disclosure, a person of ordinary skill in the art 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).

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

161. Claim 1 further recites, "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 (Tucker), 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.*)[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"

162. Claim 1 further recites, "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 (Tucker), 71; *see also id.*, 37; ¶ 161, above.)

163. For the reasons above, claim 1 would have been obvious to a person of ordinary skill in the art in view of Tucker and Bandhole.

2. Claim 2

164. Claim 2 recites "wherein each container has an execution file associated therewith for starting the one or more applications." Tucker discloses the further limitation of claim 2 because "[a]pplications that are installed into a zone and deliver start/stop scripts into `/etc/init.d` and one or more of the `/etc/rc?.d` directories will be started when the zone is booted." (Ex. 1005 (Tucker), 77; *see also id.*, 14.)

[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 (Tucker), 72.)

The "start/stop scripts" in each zone are execution files because they cause applications to execute.

3. Claim 4

165. Claim 4 recites "pre-identifying applications and system files required for association with the one or more containers prior to said storing step." Tucker discloses the further limitation of claim 4 because when a zone is installed using a particular command, the system "will prompt for certain options including … packages to install." (Ex. 1005 (Tucker), 74.) Accordingly, the server pre-identifies (e.g., by prompting the user) which applications and associated system files (e.g., in packages) are to be installed prior to the zone's storing. (*Id.*)

4. Claim 6

166. Claim 6 recites "assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address." Tucker discloses the further limitation of claim 6 because "[e]ach zone which requires network connectivity will have one or more dedicated IP addresses." (Ex. 1005 (Tucker), 44.)

5. Claim 8

167. Claim 8 recites "wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers." Tucker discloses the further limitation of claim 8 by describing a model

in which "applications can be freely moved between systems" by encapsulating the applications in zones. (Ex. 1005 (Tucker), 77; *id.*, 2 (zones "support rapid deployment (and redeployment) of applications, since the same application environment can be maintained on different physical machines").) Moving or redeploying a zone requires retrieving it from the computer system where it was previously deployed. It would have been obvious for the computer system where the zone was previously deployed to have "a plurality of secure containers" because Tucker discloses multiple non-global zones on the same system. (¶¶ 40-42, above.)

6. Claim 9

168. Claim 9 recites, "wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed." Tucker discloses the further limitation of claim 9 by way of "zone-wide limits [on] applicable resources," which limit "the total resource usage of all process entities within a zone." (Ex. 1005 (Tucker), 64.) For example, Tucker discloses that "Cpu shares" can be allocated to zones. (*Id.*, 64-65.) Further, Tucker discloses that "[t]he global administrator will be able to specify these limits in the zonecfg configuration file." (Ex. 1005 (Tucker), 64.) Because configuration occurs before a zone is running, the limits in this configuration file are associated with a zone before applications are executed within the zone.

7. Claim 10

169. Claim 10 "wherein in operation when an application residing within a container is executed, said application has no access to system files or applications in other containers or to system files within the operating system during execution thereof." Tucker discloses the further limitation of claim 10 because "[p]rocesses running within a zone will be limited to files and file systems that can be accessed from the restricted root." (Ex. 1005 (Tucker), 37; *see also id.*, 1 ("A zone is a 'sandbox' within which one or more applications can run without affecting or interacting with the rest of the system" and "isolation prevents processes running within a given zone from monitoring or affecting processes running in other zones"), 71 ("zones require a separate root file systems [sic] for isolation").)[3] Because Tucker discloses that each zone is limited to its own files, applications in the zone have no access to system files or applications in other zones or the operating system. (*Id.*)

8. Claim 13

170. Claim 13 depends on claim 1 and further recites "associating with a plurality of containers a stored history of when processes related to applications within the container are executed for at least one of, tracking statistics, resource

[3] 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 (Tucker), 72.)

allocation, and for monitoring the status of the application." Tucker discloses the further limitation of claim 13 via an "extended accounting subsystem" that "is virtualized to permit different accounting settings and files on a per-zone basis for process- and task-based accounting." (Ex. 1005 (Tucker), 64.) Records of processes "can be tagged with a zone id … allowing the global administrator to determine resource consumption on a per-zone basis." (*Id.*) Accordingly, Tucker discloses associating with zones a stored history of executed processes belonging to applications within the zone for the purpose of at least tracking statistics and resource allocation. (*Id.*)

9. Claim 16

171. Claim 16 depends on claim 1 and further recites a "step of creating containers prior to said step of storing containers in memory," which includes "using a skeleton set of system files as a container starting point and installing applications into that set of files." Tucker discloses claim 16's additional limitation. Specifically, 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 file system[.]" (Ex. 1005 (Tucker), 71.) This root file system "requires a `/etc` directory to hold standard configuration information for that zone." (*Id.*) A person of ordinary skill in the art would understand that the configuration information in the "`/etc` directory" is stored as files because directories hold files. Configuration files are

system files according to the '814 patent. (Ex. 1001 (the '814 patent), 2:52-54.) Thus, the configuration files in the zone's root file system constitute a skeleton set of system files used as a container starting point in accordance with the claim limitation. Further, applications are installed into that root file system using the Solaris packaging system. (Ex. 1005 (Tucker), 71-72.)

C. Claims 1, 2, 4, 6, 8-10, 13-14, and 16 Would Have Been Obvious in View of Gélinas

172. 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. (¶¶ 38-39, above.) Gélinas discloses or suggests each element of claims 1, 2, 4, 6, 8-10, 13-14, and 16.

1. Claim 1

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

173. Claim 1 recites, "In a system having a plurality of servers." Gélinas discloses that its containers (called "vservers") operate in a system with multiple physical servers. For example, it is "possible to move a vserver from one physical server to another." (Ex. 1007 (Gélinas), 30.)

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

174. Claim 1 further recites "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. 1007 (Gélinas), 29.) Each of these different versions of Linux is called a "distribution" (or "distro"). (*See* Ex. 1007 (Gélinas), 29.) Gélinas also recognizes that "you may want to create several vservers to tests various distributions." (*Id.*, 31; *see also id.* 27 (vserver code should "work on any recent distribution").) These various Linux distributions are operating systems that differ.

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

175. Claim 1 further recites "operating in disparate computing environments." Gélinas discloses this limitation under both the patent owner's interpretation (requiring "independently operable" computers) and the inventor's interpretation (requiring computers with different network addresses). (¶¶ 49-52, above.)

176. First, Gélinas discloses the implementation of containers on computers running Linux. (¶¶ 38-39, above.) Computers running Linux were independently operable because one computer running Linux could operate without being controlled by any other computer.

177. Second, Gélinas discloses that containers were being implemented on computers with different network addresses. For example, Gélinas discloses that each container "is assigned a host name and an IP number." (Ex. 1007 (Gélinas), 2.) "In

general, each [container] is tied to one IP [so] several servers may run the same services without conflict." (*Id.*, 16-18.)

178. 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 (Gélinas), 12.) For example, Gélinas' containers can operate in environments comprising "an IDE + uniprocessor server" or "SCSI + multiprocessor server." (Ex. 1007 (Gélinas), 30.)

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

179. Claim 1 further recites, "wherein each server includes a processor and an operating system including a kernel." Gélinas uses servers that include at least one processor. (Ex. 1007 (Gélinas), 30 ("uniprocessor" or "multiprocessor").) Further, the servers run a Linux operating system, which includes a kernel. (*Id.*, 12-13.)

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

180. Claim 1 further recites "a set of associated local system files compatible with the processor." "System files" include "shared libraries" and "configuration files." (Ex. 1001 (the '814 patent), 2:52-54.) Gélinas discloses both. (Ex. 1007 (Gélinas), 10 (shared libraries and configuration files); *id.*, 16, 24 (configuration files).) A person of ordinary skill in the art 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.

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

182. Finally, Gélinas discloses running Linux distributions such as "debian" or "redhat." (*Id.*, 30.) These distributions contained numerous shared libraries and configuration files compatible with the processor on which they ran.

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"

183. Claim 1 further recites "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 (Gélinas), 19, 11.) Each container acts as a "security box" that confines applications and exerts "complete control over their interaction with the rest of the computer and the network." (*Id.*, 1.)

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

184. Claim 1 further recites "wherein the applications are executed in a secure environment." Gélinas discloses that its containers "provide a higher level of security." (Ex. 1007 (Gélinas), 27.) For example, they can be used to "[r]un untrusted applications with complete control over their interaction with the rest of the computer and the network." (Ex. 1007 (Gélinas), 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"

185. Claim 1 further recites "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. (¶ 183, above.) The applications used to run such services included executable objects that perform tasks related to the services.

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"

186. Claim 1 further recites "the method comprising: storing in memory accessible to at least some of the servers a plurality of secure containers of application software." Gélinas discloses that installing a container "copies a linux installation inside a sub-directory." (Ex. 1007 (Gélinas), 9.) This sub-directory is stored on a disk, a type of memory. (*Id.*) Gélinas further discloses that one may "run several [containers] on the same box," which "you will certainly do." (Ex. 1007 (Gélinas), 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"

187. Claim 1 further recites "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 that its containers can run "mail servers, web servers, SQL servers," and the like. (Ex. 1007 (Gélinas), 11.) A person of ordinary skill in the art 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. For example, the '814 patent admits that "Linux Apache [a common web server] uses the following shared libraries … which are 'system' files," and then lists 21 libraries

and configuration files. (Ex. 1001 (the '814 patent), 2:55-3:19.) Gélinas discloses the use of Apache in its containers. (Ex. 1007 (Gélinas), 17, 21.)

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

188. Claim 1 further recites "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 (Gélinas), 26, 29.) Based on this disclosure, a person of ordinary skill in the art would understand that the containers use a local kernel on the underlying server.

189. Further, Gélinas discloses how to install a kernel but says nothing about moving or uninstalling it. (Ex. 1007 (Gélinas), 13-15.) Thus, it would have been obvious to a person of ordinary skill in the art that the kernel resides permanently on the server.

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"

190. Claim 1 further recites "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 (Gélinas), 11.) A person of ordinary skill in the art would understand that these services would be unable to run on the Linux operating system unless their associated system files were compatible with the kernel. Indeed, Gélinas discloses that "[o]nce a service [is] running in a vserver, it is talking directly to the kernel." (Ex. 1007 (Gélinas), 29.)

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

191. Claim 1 further recites "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 (Gélinas), 26.) Thus, the containers exclude a kernel.

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"

192. Claim 1 further recites "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 (Gélinas), 15-16, 29 ("A vserver is normally created from the root server.").) In such a 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 (Gélinas), 15.) Because processes in a container cannot access files outside the container, a person of ordinary skill in the art would understand that the copies in the container are utilized in place of the system files which remain on the underlying server.

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"

193. Claim 1 further recites "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 within Gélinas' containers are copies or modified copies of associated local system files that remain resident on the underlying server for the reasons explained above. (¶ 192, above.)

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

194. Claim 1 further recites "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

(Gélinas), 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 (Gélinas), 31; *see also id.* 9-10.) Unless an administrator specifically allows it, application software cannot be shared between the plurality of secure containers. (*Id.*)

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"

195. Claim 1 further recites "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 (Gélinas), 2.) "As such, [containers] can only see what is under their `/` directory" and not the operating system's full "`/`" directory. (Ex. 1007 (Gélinas), 31.) A container's "`/`" directory is its root, and that root is different from the operating system's root. (*Id.*)

196. For the reasons above, Gélinas renders claim 1 obvious.

2. Claim 2

197. Claim 2 recites "wherein each container has an execution file associated therewith for starting the one or more applications." Gélinas discloses the further

limitation of claim 2 by explaining that "[y]ou can setup a script" for "[e]xecuting tasks at vserver start/stop time." (Ex. 1007 (Gélinas), 20-21.) The script is called "`/etc/vservers/XX.sh` where XX is the name of the virtual server." (*Id.*) Based on the "`.sh`" extension of the script file, a person of ordinary skill in the art would understand that this is a shell script. Shell scripts were commonly used to start applications in Linux. (*Id.*) Thus, a person of ordinary skill in the art would have found it obvious to use `XX.sh` as the execution file for starting each container's applications. (Ex. 1007 (Gélinas), 20-21.)

3. Claim 4

198. Claim 4 recites "pre-identifying applications and system files required for association with the one or more containers prior to said storing step." Gélinas discloses the further limitation of claim 4 because vserver containers may be copied or moved "from one physical server to another." (Ex. 1007 (Gélinas), 30.) In this situation, the applications and system files in the container are identified on the source server before they are copied or moved to the destination. For example, all of the applications and system files would be located in the directory "`etc/vservers/XX`" (where XX is the name of the container to be copied). (Ex. 1007 (Gélinas), 30.) The command used to copy this directory—such as "`rsync`" (*id.*)—would identify the applications and system files in that directory when copying them. Such identification and copying would occur before the container was stored on the destination server.

4. Claim 6

199. Claim 6 recites "assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address." Gélinas discloses the further limitation of claim 6 because each container "is assigned a host name and an IP number." (Ex. 1007 (Gélinas), 2, 16-18.)

5. Claim 8

200. Claim 8 recites "wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers." Gélinas discloses the further limitation of claim 8 because it states that a physical server stores a plurality of containers on disk. (Ex. 1007 (Gélinas), 9-10.) When applications and system files are executed, a person of ordinary skill in the art would understand they are retrieved from the disk that stores the containers, which is part of a computer system (the physical server). (*Id.*)

6. Claim 9

201. Claim 9 recites "wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed." Gélinas discloses the further limitation of claim 9 because a container may have a flag associated with it that "unifies the

processes in a vserver from a scheduler view point" and thus "prevents a vserver from taking [too] much CPU resources." (Ex. 1007 (Gélinas), 17-18, 24.) A flag may also be set to limit a container to a maximum number of processes (*id.*, 23) or a particular set of networking capabilities (*id.*, 17). Thus, server information (e.g., a flag on the server) related to hardware resource usage (e.g., CPU or networking) may be associated with a container.

202. Further, the flag information is stored in the "`XX.conf`" file for a particular container. (Ex. 1007 (Gélinas), 16-17.) The `XX.conf` file is created before the container (and any applications within it) is started. (*See id.* ("`ONBOOT`" parameter in "`XX.conf`" controls how containers are started); *id.*, 30 (move "`XX.conf`" to a new physical server before starting container there).)

7. Claim 10

203. Claim 10 recites "wherein in operation when an application residing within a container is executed, said application has no access to system files or applications in other containers or to system files within the operating system during execution thereof." Gélinas discloses the further limitation of claim 10 because, as discussed above, containers can only see their own root directory and not what is outside of that directory. (Ex. 1007 (Gélinas), 31; ¶ 195, above.). Because each container is limited to its own files, applications within the container have no access

to system files or applications in other containers or within the underlying operating system. (*Id.*; *see also* Ex. 1007 (Gélinas), 2-3 (explaining isolation).)

8. Claim 13

204. Claim 13 recites "associating with a plurality of containers a stored history of when processes related to applications within the container are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application." Gélinas discloses the further limitation of claim 13 because its system can produce a report listing "the number of process in each active security context as well as the name of the vserver associated with this context." (Ex. 1007 (Gélinas), 14.) A person of ordinary skill in the art would have understood that such a report would be useful for tracking statistics. (*See id.* (command called "`/usr/sbin/vserver-stat`").) Further, Gélinas discloses another command that reports "the running status of every virtual server," useful for monitoring the status of applications in containers. (Ex. 1007 (Gélinas), 20.)

9. Claim 14

205. Claim 14 depends on claim 1. Gélinas discloses each further limitation of claim 14.

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

206. Gélinas discloses creating containers from system files on the underlying server. (Ex. 1007 (Gélinas) at 15-16; ¶ 192, above.) Further, Gélinas discloses that containers may be copied from a first server to a second server. (Ex. 1007 (Gélinas), 30.) In this situation, containers are created on the first server before being copied to the second server and stored in the second server's memory.

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

207. Gélinas discloses that a container on a physical server "runs normal services such as telnet, mail servers, web servers, SQL servers." (Ex. 1007 (Gélinas), 11.)

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

208. Gélinas discloses that files being used by a container "XX" are stored in a directory called "`/vservers/XX`." (Ex. 1007 (Gélinas), 15-16, 30.) These files are transferred when a container is copied from a first server to a second server. (*Id.*, 30.) Gélinas discloses that the files may be transferred using an "`rsync`" command, with the "`/vservers/XX`" directory supplied to the command as a parameter. (*Id.*) Thus, Gélinas discloses determining which files are being used (e.g., the files in "`/vservers/XX`") when a command is executed to copy a container. (¶ 198, above.)

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"

209. As explained above, Gélinas discloses copying a container from a first server to a second server. (Ex. 1007 (Gélinas), 30.) Because the contents of the copied container are stored on the second server, they do not overwrite the contents on the first server.

210. For the reasons above, Gélinas renders claim 14 obvious.

10. Claim 16

211. Claim 16 depends on claim 1 and further recites a "step of 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 the following directories from an existing Linux installation into the new container's file system: "`/sbin /bin /etc /usr /var /dev /lib`." (Ex. 1007 (Gélinas), 15.) A person of ordinary skill in the art would recognize these as standard directories that contained system files in a Linux installation.

212. A person of ordinary skill would also recognize that 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 after

it is created. (Ex. 1007 (Gélinas), 11.) Specifically, Gélinas discloses installing "standard packages (RPMs for example)." (*Id.*) "RPMs" are packages in the Red Hat Package Manager (RPM) format. 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. Thus, Gélinas renders claim 16 obvious.

VII. SECONDARY CONSIDERATIONS OF NONOBVIOUSNESS

213. I am not aware of any secondary considerations of nonobviousness. If Patent Owner identifies any alleged evidence of secondary considerations in the future, I reserve the right to respond to that information.

VIII. CONCLUSION

214. For the foregoing reasons, it is my opinion that claims 1-16 and 31-34 of the '814 patent would have been obvious to a person of ordinary skill in the art at the time of the alleged invention in view of the prior art discussed above.

215. I reserve the right to supplement my opinions in the future to address or respond to any issues that the Patent Owner may raise, as well as new information including, but not limited to, any claim constructions advanced by the Patent Owner or adopted by the Board in the Institution Decision, and respond to any alleged secondary considerations as they become available to me.

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

Executed on January 28th, 2025 at Soquel, California.

Darrell Long

Darrell D. E. Long, Ph.D.