I IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

VIRTAMOVE, CORP., Plaintiff, v. HEWLETT PACKARD ENTERPRISE COMPANY, Defendant.	Civil Action No. 2:24-CV-00093-JRG (Lead Case) **JURY TRIAL DEMANDED**
VIRTAMOVE, CORP., Plaintiff, v. INTERNATIONAL BUSINESS MACHINES CORP., Defendant.	Civil Action No. 2:24-CV-00064-JRG (Member Case) **JURY TRIAL DEMANDED**

DEFENDANT INTERNATIONAL BUSINESS MACHINES CORP.'S INVALIDITY CONTENTIONS

TABLE OF CONTENTS

Page

I. INTRODUCTION 1

II. ASSERTED PATENTS AND CLAIMS 7

III. PRIORITY DATES 8

IV. IDENTIFICATION OF PRIOR ART 15

V. GROUNDS FOR INVALIDITY UNDER 35 U.S.C. §§ 102, 103 26

VI. GROUNDS FOR INVALIDITY UNDER 35 U.S.C. § 112 98

VII. ADDITIONAL INVALIDITY CONTENTIONS 111

APPENDICES

Appendices A1-A23	Invalidity claim charts for USP 7,519,814
Appendices B1-B24	Invalidity claim charts for USP 7,784,058

Virtamove, Corp.,
Ex. 2008 IPR2025-00591
Page 3 of 115

I. INTRODUCTION

Pursuant to the Court's First Amended Docket Control Order in the above-captioned case (Dkt. 76), Defendant International Business Corp. ("IBM" or "Defendant") serves these Invalidity Contentions on Plaintiff VirtaMove, Corp. ("VirtaMove" or "Plaintiff") for U.S. Patent No. 7,519,814 (the "'814 Patent") and U.S. Patent No. 7,784,058 (the "'058 Patent") (collectively, the "Asserted Patents"). These Invalidity Contentions are based on IBM's current knowledge of the Asserted Patents and prior art, along with its understanding of VirtaMove's infringement allegations set forth in its July 1, 2024, Corrected Infringement Contentions Pursuant to Local Patent Rules 3-1 and 3-2 ("Infringement Contentions"). IBM's investigation of the prior art and other grounds of invalidity (*e.g.*, failure to comply with pre-AIA 35 U.S.C. § 112) is ongoing, and IBM expressly reserves the right to supplement these Invalidity Contentions as the case proceeds.

Nothing in these Invalidity Contentions is intended, nor should be construed, as a waiver of any claim construction argument or noninfringement position. IBM's statements herein (including the accompanying claim charts) reflect IBM's present understanding of the purported potential scope of the claims that VirtaMove appears to be advocating by way of its Infringement Contentions. They are not to be seen as any acquiescence to VirtaMove's interpretation of any claims. IBM disagrees that any such claim scope is proper. IBM reserves the right to supplement these contentions to address any supplemental infringement contentions. For purposes of these Invalidity Contentions, IBM identifies prior art references and provides element-by-element claim charts based on the apparent constructions of the Asserted Claims advanced by VirtaMove in its Infringement Contentions (which, for at least some limitations, contradict the plain language of the claim). IBM also reserves all rights to disclose additional invalidity contentions, including contentions relating to any additional Asserted Claims or theories VirtaMove is allowed to assert, to the extent VirtaMove is allowed to amend its contentions or otherwise attempts to correct the

deficiencies in its infringement contentions or supplements them in any way.

To the extent that these Invalidity Contentions rely on or otherwise embody particular constructions of terms or phrases in the Asserted Claims, IBM is not proposing any such constructions as proper constructions of those terms or phrases at this time. The Court established separate deadlines for the parties' proposed claim constructions, and IBM will disclose its proposed constructions accordingly. For purposes of these Invalidity Contentions, IBM may adopt alternative claim construction positions. In particular, portions of these Invalidity Contentions, including the claim charts attached as Appendices, may be based on the underlying claim constructions and/or interpretations as understood from VirtaMove's Infringement Contentions and/or VirtaMove's proposed claim constructions. IBM, however, does not concede that VirtaMove's apparent constructions are proper and expressly reserves the right to contest any such constructions. In addition, to the extent that these Invalidity Contentions rely on or otherwise embody a particular order in which the steps of method claims are performed, IBM does not necessarily propose that the method claims must be limited to such order, although IBM reserves the right to propose such an order. IBM reserves the right to supplement, modify, or otherwise amend these Invalidity Contentions, including based on the Court's claim construction ruling and/or arguments or positions taken during the claim construction process.

Throughout the attached Appendices, IBM provides examples of where references disclose subject matter recited in preambles, without regard to whether the preambles are properly considered to be limitations of the Asserted Claims. IBM reserves the right to argue, at the appropriate stage of this case, that the preambles are or are not limitations. Moreover, IBM reserves the right to argue that certain claim elements of the Asserted Claims do not in fact limit the scope of the Asserted Claims.

Nothing herein shall be interpreted as an admission that: (1) the Asserted Claims are infringed by any of IBM's products or instrumentalities, (2) any particular feature or aspect of any of the accused instrumentalities practices any limitation of the Asserted Claims, (3) there is 35 U.S.C. § 112 support for any limitation of the Asserted Claims, or (4) any of VirtaMove's proposed or implied constructions are supportable or proper.

Consistent with the Local Patent Rules and the Federal Rules of Civil Procedure, IBM reserves the right to amend these Invalidity Contentions. The information and documents that IBM produced along with its Subject Matter Eligibility Contentions and with these Invalidity Contentions are provisional and subject to further revision as follows. IBM expressly reserves the right to amend its disclosures and document production referenced herein should VirtaMove provide any information that it failed to provide in its Local Patent Rule 3-1 and 3-2 disclosures or should VirtaMove amend its disclosures in any way, whether explicitly (*i.e.*, via moving for leave to amend or supplement) or implicitly. Further, because discovery has only recently begun and because IBM has not yet completed its search for and analysis of relevant prior art, IBM reserves the right to amend the information provided herein. Such amendments include, for example, identifying and relying on additional references, should IBM's further search and analysis yield additional information or references, consistent with the Patent Rules or the Federal Rules of Civil Procedure. IBM reserves the right to supplement these contentions in light of any additional prior art of which VirtaMove is aware and did not disclose to IBM in discovery. Also, IBM has issued and may later issue subpoenas to third parties believed to have knowledge, documentation and/or corroborating evidence concerning some of the prior art listed herein and/or additional prior art. These third parties include, but are not limited to, the authors, employers of authors, inventors, assignees, or former or current employees of assignees, of the references

identified herein and/or in the Asserted Patents. For example, IBM (and/or co-defendant HPE) has issued subpoenas to potential prior artists including but not limited to individuals and entities responsible for the development of the Solaris Containers (Solaris Zones) system, the Softricity SystemGuard solution, the Thinstall Application Virtualization platform, the vMotion migration tool, and the Virtuozzo system. IBM (and/or co-defendant HPE) may issue additional subpoenas to third parties, including but not limited to individuals and other entities responsible for the development of the FreeBSD Jails system or the other systems identified above. IBM reserves the right to supplement these contentions in light of any newly discovered information produced by these or other companies from which IBM may seek discovery.

As set forth herein, several claim terms render the Asserted Claims indefinite and/or lacking written description. IBM's Invalidity Contentions under 35 U.S.C. §§ 101, 102, and/or 103 should not be taken as an admission that any of the terms are definite, are adequately described, or are enabled. IBM's identification of disclosure in the prior art and explanation of combining references is based on IBM's current understanding of the claims, to the extent that the terms used in those claims could be understood by a person of ordinary skill in the art at the time of the relevant claim's priority date or could be considered adequately described and enabled. IBM reserves the right to modify its disclosures based on any claim construction or interpretation by the Court, and/or based on VirtaMove's positions regarding claim construction, indefiniteness, and/or written description. IBM also reserves the right to challenge any of the claim terms herein under 35 U.S.C. § 112 beyond the grounds outlined herein or in its claim construction submissions or responses to interrogatories related to invalidity, including by arguing that they are indefinite, not supported by the written description, or not enabled. Nothing stated herein shall be construed as a waiver of any argument available under 35 U.S.C. §§ 101, 102, 103, and/or 112.

Moreover, IBM reserves the right to revise its ultimate contentions concerning the invalidity of the claims of the Asserted Patents based upon the Court's construction of the claims of the Asserted Patents, any findings as to the priority dates of the Asserted Claims, and/or positions that VirtaMove, IBM, or any expert witness may take concerning claim interpretation, infringement, and/or invalidity issues.

Prior art not included in this disclosure, whether known or not known to IBM, may become relevant. In particular, IBM is currently unaware of the extent, if any, to which VirtaMove will contend that limitations of the Asserted Claims are not disclosed in the prior art identified by IBM. To the extent that such an issue arises, IBM reserves the right to identify other references or other portions of the identified references that would have made the addition of the allegedly missing limitation to the disclosed device or method obvious or show that the allegedly missing limitation would have been known or readily apparent to one of ordinary skill in the art at the time of the invention in light of the disclosure of the prior art at issue.

IBM's claim charts in Appendices A-1 through A-23 and B-1 through B-24 cite to or reference particular teachings and disclosures of the prior art as applied to features of the Asserted Claims, but persons having ordinary skill in the art generally may view an item of prior art in the context of other publications, literature, products, and understanding. As such, the cited portions are only examples, and IBM reserves the right to rely on uncited portions of the prior art references and on other publications and expert testimony as aids in understanding and interpreting the cited portions, as providing context thereto, and as additional evidence that the prior art discloses a claim limitation. IBM further reserves the right to rely on uncited portions of the prior art references, other publications, and testimony to establish reasons for combining certain cited references that render the Asserted Claims obvious.

The references discussed in the claim charts in Appendices A-1 through A-23 and B-1 through B-24 may disclose the elements of the Asserted Claims explicitly and/or inherently, and/or they may be relied upon to show the State of the Art in the relevant time frame. The suggested obviousness combinations are provided in the alternative to IBM's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not by itself anticipatory.

Depending on the Court's construction of the claims of the Asserted Patents, and/or positions that VirtaMove, IBM, or any expert witness may take concerning claim interpretation, infringement, and/or invalidity issues, one or more of the charted prior art references may be of greater or lesser relevance and different combinations of these references may be implicated. Given this uncertainty, the charts may reflect alternative applications of the prior art against the Asserted Claims.

IBM's Invalidity Contentions are based at least in part on certain priority dates that VirtaMove alleges the Asserted Patents are entitled to in its Infringement Contentions served under P.R. 3-1.[1] IBM reserves the right to challenge any different priority date that VirtaMove later alleges is appropriate.

IBM reserves the right to assert invalidity under 35 U.S.C. §§ 101, 102(c), (d), or (f) to the extent that discovery or further investigation yields information forming the basis for such claims. IBM reserves the right to assert that the Asserted Patents are invalid under 35 U.S.C. § 102(f) in

[1] IBM understands that VirtaMove may seek leave to amend its P.R. 3-1 disclosures to allege that the Asserted Patents are entitled to the priority dates provided in VirtaMove's second supplemental response to Common Interrogatory No. 1, which VirtaMove served on October 4, 2024. VirtaMove did not disclose these contentions in its P.R. 3-1 disclosures and has not yet moved to amend its P.R. 3-1 disclosures as required. In the event that such a motion for leave is filed and is granted, IBM reserves the right to supplement these Invalidity Contentions in light of those different alleged priority dates.

the event IBM obtains evidence that the named inventors of the Asserted Patents did not invent the subject matter claimed in the Asserted Patents. Should IBM obtain such evidence, it will provide the name of the person(s) from whom and the circumstances under which the invention or any part of it was derived.

Pursuant to Local Patent Rules 3-3 and 3-4, IBM has provided disclosures and related documents pertaining only to the Asserted Claims as identified by VirtaMove in its Infringement Contentions. IBM reserves the right to modify, amend, or supplement these Invalidity Contentions to show the invalidity of any claims that the Court may allow VirtaMove to later assert, including additional and/or amended claims, resulting from a patent office proceeding (e.g., *ex parte* reexam, *inter partes* review, etc.), including but not limited to Reexamination Control No. 90/019,676. IBM further reserves the right to supplement its P.R. 3-4 document production should they later find additional, responsive documents.

II. ASSERTED PATENTS AND CLAIMS

VirtaMove asserts the following patents and claims in its Infringement Contentions against IBM (with independent claims identified in **bold**)[2]:

No.	Asserted Patents	Asserted Claims
1.	USP 7,519,814	**1**, 2, 6, 9, and 10
2.	USP 7,784,058	**1**-4 and 18

For the purposes of these contentions, IBM only addresses those claims specifically asserted by VirtaMove in its June 5, 2024, Infringement Contentions. IBM reserves the right to amend or supplement this disclosure as necessary in light of any changes or amendments made,

[2] The asserted patents and claims are collectively referenced throughout these contentions as the "Asserted Patents" and the "Asserted Claims."

for any reason, to VirtaMove's infringement theories, Infringement Contentions, or Asserted Claims.

III. PRIORITY DATES

The Plaintiff bears the burden of establishing the priority date to which each claim of the Asserted Patents is entitled. *See, e.g.*, *In re NTP, Inc.*, 654 F.3d 1268, 1276 (Fed. Cir. 2011) ("[F]or a patent's claim to be entitled to an earlier priority date, the patentee must *demonstrate* that the claims meet the requirements of 35 U.S.C. § 120."); *Natural Alternatives Int'l, Inc. v. Iancu*, 904 F.3d 1375,1380 (Fed. Cir. 2018) ("Accordingly, claims in a patent or patent application are not entitled to priority under § 120 at least until the patent owner *proves* entitlement to the PTO, the Board, or a federal court.").

The Asserted Patents were filed on or purport to claim priority as follows:

No.	Asserted Patents	VirtaMove's Alleged Conception Dates[3]	Alleged Priority Date on Face of Patent[4]
1.	USP 7,519,814	September 15, 2003, the filing date of provisional application No. 60/502,619.	October 20, 2003, the filing date of provisional application No. 60/512,103 and/or Sept. 15, 2003, the filing date of provisional application No. 60/502,619
2.	USP 7,784,058	September 22, 2003, the filing date of provisional application No. 60/504,213.	September 22, 2003, the filing date of provisional application No. 60/504,213

[3] *See supra* note 1.

[4] As set forth below in Section VI, none of the Asserted Claims of the '814 Patent or the '058 Patent is entitled to claim priority to the filing date of the respective provisional application(s). *See infra* Section VI.

IBM does not concede that the above priority dates are accurate and reserves the right to supplement these Invalidity Contentions should a different priority date apply for either Asserted Patent.

To the extent VirtaMove alleges that any prior art relied on in these Invalidity Contentions does not actually qualify as prior art to an Asserted Patents, IBM reserves the right to rebut those allegations (*e.g.*, by demonstrating an earlier critical date for the challenged prior art and/or a later priority date for the Asserted Patents and/or Asserted Claim).

A. The '814 Patent Priority Date

VirtaMove's infringement contentions allege that the Asserted Claims of the '814 Patent are entitled to a priority date of September 15, 2003. The '814 Patent was filed on September 13, 2004. The '814 Patent claims priority on its face to provisional application No. 60/512,103, filed on Oct. 20, 2003, and to provisional application No. 60/502,619, filed on September 15, 2003.

IBM understands that VirtaMove may seek leave to amend its Infringement Contentions to allege that the Asserted Patents are entitled to the priority dates provided in VirtaMove's second supplemental response to Common Interrogatory No. 1, served on October 4, 2024. In the event that such a motion for leave is filed and granted, IBM reserves the right to supplement these Invalidity Contentions in light of those different alleged priority dates (or any other attempt by VirtaMove to seek an earlier date of invention).

VirtaMove has not provided sufficient evidence to support its contention that the Asserted Claims of the '814 Patent are entitled to the priority dates of either (or both) of the provisional applications. By way of example, the provisional applications, either individually or collectively, fail to provide sufficient written description support for at least the following claim terms:

Claim Terms	Asserted Claims
"plurality of servers with operating systems that differ, operating in disparate computing environments"	Claim 1
"wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor"	Claim 1
"a method of providing at least some of the servers in the system with secure, executable, applications related to a service"	Claim 1
"wherein the applications are executed in a secure environment"	Claim 1
"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"	Claim 1
"a plurality of secure containers of application software, 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, for use with a local kernel residing permanently on one of the servers"	Claim 1
"wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel"	Claim 1
"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"	Claim 1
"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 servers"	Claim 1
"wherein the application software cannot be shared between the plurality of secure containers of application software"	Claim 1
"wherein each of the containers has a unique root file system that is different from an operating system's root file system"	Claim 1
"when an application residing within a container is executed, said application has no access … to system files within the operating system during execution thereof"	Claim 10

At least under VirtaMove's apparent interpretation and application of the Asserted Claims, the Provisional Application Nos. 60/512,103 and 60/502,619 do not provide sufficient written description support for the above claim terms. For example, in order to distinguish Claim 1 over the prior art during prosecution of the '814 Patent, VirtaMove amended Claim 1 to recite "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, and wherein the application software cannot be shared between the plurality of secure containers of application software, and wherein each of the containers has a unique root file system that is different from an operating system's root file system." '814 Patent Prosecution History, Sept. 3 2008 Applicant Arguments/Remarks Made in an Amendment. VirtaMove contended that "[s]upport for the amendments in claim 1 can be found within the specification," for example:

> *At paragraph [54] the following text is found:*
>
> *"In embodiments of the invention, all applications within a given container have their own common root file system exclusive thereto and unique from other containers and from the operating system's root file system."*
>
> *Figure 6 illustrates that system files from the underlying operating system are used independent of system files in the container. In the instant invention system files within the container are application specific and are distinct from and system files within the OS.*
>
> *Figure 9 shows that system files from the underlying operating system are used by operating system applications wherein system files from the container are used by applications associated with the container.*
>
> *At paragraph [91] the following text is recited:*
>
> *"Turning now to Figure 1, a system is shown having a plurality of servers 10a, 10b. Each server includes a processor and an independent operating system. Each operating system includes a kernel 12, hardware 13 in the form of a processor and a set of associated system files, not shown. Each server with an operating system installed is capable of executing a number of application programs. Unlike typical systems, containerized applications are shown on each server having application programs lla, llb and related system files llc. <u>These system files are used in place of system files such as library and configuration files present typically used in conjunction with the execution of applications.</u>" (underlining added)*
>
> *At paragraph [92], the following text is recited:*
>
> *"Figure 2 illustrates a system in accordance with the invention in which multiple applications 21a, 21b, 21c, 21d, 21e, and 21f can execute in a secure environment on a single server. This is made possible by associating applications with secure containers 20a, 20b and 20c. <u>Applications are segregated and execute with their own copies of files and with a unique network identity.</u> In this manner multiple applications may <u>contend for common resources and utilize different versions of system</u> files. Moreover, applications executing within a secure container can co-exist with <u>applications that are not associated with a container, but which are associated with the underlying operating system</u>." (underlining added)*

At paragraph [94] the following text is recited:

"The Secure application containers 20a, 20b and 20c are shown in Figure 2 in which each combines un-installed application files on a storage medium or network, application files installed on a computer, and system files. The container is an aggregate of all files required to successfully execute a set of software applications on a computing platform. In embodiments of the invention, containers are created by combining application files with system files."

Paragraph [107] clearly teaches the relationship between files and containers:

"It can be seen from Figure 6 that each application executing associated with a container 20a or 20b, or contained within a container 20a or 20b, is able to accesses files that are dedicated to the container. Applications with a container association, that is, applications contained within a container, <u>are not able to access the files provided with the underlying operating system of the compute platform 40 upon which they execute.</u> <u>Moreover, applications with a container 20a association are not able to access the files contained in another container 20b.</u>" (underlining added)

At paragraphs 112, 117, 121, 122, and 131, the following text is found which supports the amendments to Claim 1.

[112] In a first embodiment all files are exclusive. That <u>is, each container has a separate physical copy of the files required by applications associated with the container.</u> In future embodiments any files that is read-only will be shared between containers. (underlining added)

[117] Figure 3 illustrates container creation. The result of container creation is the collection of files required by applications associated with the container and the assembly of container specific configuration information. <u>Container files include both those files provided by an operating system distribution</u> and those files provided by an application install media or package file. (underlining added)

[121] Referring once again to Figure 3, applications may come from third party installation media on CDROM, package files 30, or as downloads from a web site, etc. <u>Once installed in the container the applications become unique to the container</u> in the manner described way that has been described heretofore. (underlining added)

[122] The second method employed to create a container file system uses an existing server, for example a computer or server already installed at a customer site. This is also shown in Figure 3. The applications of interest may have been installed on an existing server before the introduction of the software and data structures in accordance with this invention. <u>Files are copied from the existing server 32, including system files</u> and application specific files to network storage. Once the files are copied from the existing server a container is created from those files. (underlining added)

> *[131] In some embodiments of the invention, the method involves installing at least one of the pluralities of applications in a* <u>*container's own root file system.*</u> *(underlining added)*

'814 Patent Prosecution History, Sept. 3, 2008 Applicant Arguments/Remarks Made in an Amendment.

None of the above disclosures from the '814 Patent specification that VirtaMove contends provide written description support for the amendments to Claim 1 are found in Provisional Application Nos. 60/512,103 and 60/502,619. For this reason and the additional reasons set forth above, Claim 1, and Claims 2, 6, 9, and 10 which all depend from Claim 1, are not entitled to the benefit of the earlier priority dates of Provisional Application Nos. 60/512,103 and 60/502,619.

To the extent VirtaMove alleges that any prior art relied on in these Invalidity Contentions does not qualify as prior art to the '814 Patent, IBM reserves the right to rebut those allegations (*e.g.*, by demonstrating an earlier critical date for the challenged prior art and/or a later priority date for the '814 Patent and/or Asserted Claims) with additional evidence.

B. The '058 Patent Priority Date

VirtaMove's infringement contentions allege that the Asserted Claims of the '058 Patent are entitled to a priority date of September 22, 2003. The '058 Patent was filed on September 21, 2004. The '058 Patent claims priority on its face to Provisional Application No. 60/504,213, filed on September 22, 2003.

IBM understands that VirtaMove may seek leave to amend its Infringement Contentions to allege that the Asserted Patents are entitled to the priority dates provided in VirtaMove's second supplemental response to Common Interrogatory No. 1, served on October 4, 2024. In the event that such a motion for leave is filed and granted, IBM reserves the right to supplement these Invalidity Contentions in light of those different alleged priority dates (or any other attempt by VirtaMove to seek an earlier date of invention).

VirtaMove has not provided sufficient evidence to support its contention that the Asserted Claims of the ’058 Patent are entitled to the priority date of Provisional Application No. 60/504,213. By way of example, the provisional application does not provide sufficient written description support for at least the following claim terms:

Claim Terms	Asserted Claims
“a processor”	Claim 1
“when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications”	Claim 1
“without being shared with other of the plurality of software applications”	Claim 1
“wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE”	Claim 1
“wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.”	Claim 1
“wherein in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system.”	Claim 2

At least under VirtaMove’s apparent interpretation and application of the Asserted Claims, Provisional Application No. 60/504,213 does not provide sufficient written description support for the above claim terms. Thus Claim 1, as well as Claims 2, 3, 4, and 18 which all depend from Claim 1, are not entitled to the benefit of the earlier priority date of Provisional Application No. 60/504,213.

To the extent VirtaMove alleges that any prior art relied on in these Invalidity Contentions does not qualify as prior art to the ’058 Patent, IBM reserves reserve the right to rebut those allegations (*e.g.*, by demonstrating an earlier critical date for the challenged prior art and/or a later priority date for the ’058 Patent and/or Asserted Claims) with additional evidence.

IV. IDENTIFICATION OF PRIOR ART

Subject to IBM's reservations of rights, IBM identifies each item of prior art that anticipates or renders obvious one or more of the Asserted Claims, and/or that forms the basis of the knowledge of a person of ordinary skill in the art, in the sections that follow and/or in the claim chart Appendices to these Invalidity Contentions. To the extent any references identified below and/or in the claim charts is not identified as an item of prior art that anticipates or renders obvious an Asserted Claim, IBM intends to rely on such references as background and evidence of the state of the art at the time of the alleged inventions of the Asserted Patents. IBM reserves the right to rely on additional prior art that may be uncovered during its continuing investigation and discovery.

In addition, many of the prior art references have related patent applications, publications, or issued patents that contain the same or materially the same subject matter (*e.g.*, published U.S. patent applications and publications, issued U.S. patents, and foreign applications or issued patents). Any citation to or quotation from any of these applications, publications, or patents, therefore, should be understood as encompassing any parallel citation to the same subject matter in other related or corresponding patent applications, publications, or issued patents.

Also, to the extent not expressly stated herein, IBM incorporates by reference any and all prior art contained or identified in documents produced thus far by VirtaMove in this case, each reference identified in all Appendices to these Invalidity Contentions, and any and all materials regarding invalidity that should have been timely produced by VirtaMove months ago.

A. Prior Art Patents and Published Patent Applications

The table below identifies patents and patent publications now known to IBM that qualify as prior art to the Asserted Patents under at least 35 U.S.C. §§ 102(a), (b), (e), and/or (g). Each listed reference became prior art at least as early as the dates provided on the face of the document,

as indicated in the chart below, and/or as indicated in the corresponding Appendix. IBM reserves the right to rely upon earlier dates of publication or public availability to the extent such information is uncovered during discovery. IBM also reserves the right to rely on fact or expert witness testimony to establish the publication and/or public availability of any publications described below. Each Appendix identified by an "A" pertains to the '814 Patent, and each Appendix identified by a "B" pertains to the '058 Patent.

Table 1: Prior Art Patents and Published Patent Applications				
Patent Number	**Title**	**Date(s)**	**Short Name**	**Appendix**
U.S. 7,461,080	System Logging Within Operating System Partitions Using Log Device Nodes That Are Access Points to A Log Driver	Priority: 5/9/2003 Filed: 12/22/2003 Issued: 12/2/2008	Tucker '080	A-1
U.S. 7,437,556	Global Visibility Controls for Operating System Partitions	Priority: 5/9/2003 Filed: 1/21/2004 Issued: 10/14/2008	Tucker '556	A-2
U.S. 60/469,558	N/A	Filed: 5/9/2003	Tucker Provisional	A-3
U.S. 2003/0014466	System and Method for Management of Compartments in a Trusted Operating System	Filed: 6/29/2001 Pub: 1/16/2003	Berger '466	A-4 B-16
U.S. 2002/0194496	Multiple Trusted Computing Environments	Priority: 6/7/2002 Filed: 6/18/2002 Issued: 1/4/2011	Griffin	A-5
U.S. 2002/0124072	Virtual Computing Environment	Filed: 7/30/2001 Pub.: 9/5/2002	Tormasov	A-6
U.S. 2002/0174215	Operating System Abstraction and Protection Layer	Filed: 5/16/2001 Pub. 11/21/2002	Schaefer	A-7
U.S. 2003/0084436	System and Method for Installing Applications In A Trusted Environment	Filed: 10/30/2001 Pub.: 5/1/2003	Berger '436	A-16 B-17

Table 1: Prior Art Patents and Published Patent Applications				
Patent Number	**Title**	**Date(s)**	**Short Name**	**Appendix**
U.S. 7,962,950	System and Method for File System Mandatory Access Control	Filed: 6/29/2001 Pub.: 1/9/2003	Choo ’950	A-17 B-18
U.S. 7,673,308	Virtual OS computing environment	Priority: 10/18/2002 Filed: 10/18/2003 Issued: 3/2/2010	McMillan	A-20
U.S. 5,961,582	Distributed and portable execution environment	Filed: 10/25/1994 Issued: 10/5/1999	Gaines	A-21
U.S. 6,966,010	Application Container That Allows Concurrent Execution on Multiple Nodes in a Cluster	Filed: 8/31/2001 Issued: 11/15/2005	Curtis	A-22
U.S. 6,976,037	Method and Systems for DLL/COM Redirection	Priority: 3/27/2000 Filed: 6/27/2000 Issued: 12/13/2005	D’Souza	B-1
U.S. 6,529,985	Selective Interception of System Calls	Filed: 2/4/2000 Issued: 3/4/2003	Deianov	B-2
U.S. 2003/0177285	Operating system application programming interfaces and methods of using operating systems	Priority: 6/16/1999 Filed: 3/10/2003 Pub.: 9/18/2003	Hunt	B-21
U.S. 5,572,709	Using dynamically-linked libraries to add side effects to operations	Filed: 6/18/2993 Issued: 10/5/1996	Fowler	B-22
U.S. 8,032,625	Method and system for a network management framework with redundant failover methodology	Filed: 6/29/2001 Pub.: 1/9/2003 Issued: 10/4/2011	Benfield	B-23

B. Prior Art Non-Patent Publications

The table below identifies non-patent printed publications now known to IBM that qualify as prior art to the Asserted Patents under at least 35 U.S.C. §§ 102(a), (b) and/or (g). Each listed reference became prior art at least as early as the dates provided on the face of the document, as indicated in the chart below, and/or as indicated in the corresponding Appendix. IBM reserves the right to rely upon earlier dates of publication or public availability to the extent such information is uncovered during discovery. IBM also reserves the right to rely on fact or expert witness testimony to establish the publication and/or public availability of any publications described below. Each Appendix identified by an "A" pertains to the '814 Patent, and each Appendix identified by a "B" pertains to the '058 Patent.

Table 2: Prior Art Printed Publications			
Title	**Author/Publisher**	**Date**	**Appendix**
Virtual Private Servers and Security Contexts (VServer documentation) ("Gelinas")	Gelinas	2002	A-8
The Design and Implementation of Zap: A System for Migrating Computing Environments ("Osman")	Osman ACM SIGOPS Operating Systems Review 36, no. SI	Dec. 2002	A-9
Supporting Ubiquitous Computing with Stateless Consoles and Computation Caches ("Schmidt")	Schmidt	Aug. 2000	A-10
File Systems in User Space ("Eggert")	Eggert USENIX Winter 1993 Conf.	1993	B-3
Alpine: A User-Level Infrastructure for Network Protocol Development ("Ely Paper")	Ely USENIX Symposium on Internet Technologies and Systems (USITS 01)	2001	B-4
Shared Libraries in SunOS ("Gingell")	Gingell	1987	B-5
Linkers and Loaders ("Levine")	Levine	2001	B-6

Table 2: Prior Art Printed Publications			
Title	**Author/Publisher**	**Date**	**Appendix**
Protocol Service Decomposition for High-Performance Networking ("Maeda")	Maeda Proc. 14th ACM Symposium on Operating System Principles	1993	B-7
Implementing Network Protocols at User Level ("Thekkath")	Thekkath SIGCOMM Computer Communication Review	1993	B-8
Alpine: A User-Level Infrastructure for Network Protocol Development ("Ely Presentation")	Ely 3rd USENIX Symposium on Internet Technologies and Systems (USITS 01)	2001	B-13
HP C++ Programmer's Guide, Fourth Edition ("HP C++ Programmer's Guide")	HP	June 1996	A-18 B-19
HP C/HP-UX Reference Manual, Nineth Edition ("HP C Compiler Reference Manual")	HP	June 2000	A-19 B-20
hp secure OS software for Linux, Release Notes, Version 1.0	HP	Aug. 15, 2001	A-14 B-14
hp secure OS software for Linux, Administration Guide, Version 1.0	HP	Aug. 15, 2001	A-14 B-14
hp secure OS software for Linux, technical product brief	HP	2001	A-14 B-14
A Secure Linux Platform, in Proceedings of the 5th Annual Linux Showcase & Conference in Oakland, California	Nigel Edwards, Joubert Berger, Tse Huong Choo HP	Nov. 5-10, 2001	A-14 B-14
An Operating System Approach to Securing E-Services, Implementing Trusted Linux, an Ideal Platform for E-Services Application Hosting, 44 Communications of the ACM 2	Chris Dalton and Tse Huong Choo ACM	February 2001	A-14 B-14
HP Secure OS Software for Linux 1.0 (HP-LX 1.0), Product Reviews	Brian Wilson, Web Techniques, Vol. 7, Issue 1	Jan. 2002	A-14 B-14

Table 2: Prior Art Printed Publications			
Title	**Author/Publisher**	**Date**	**Appendix**
Absolute BSD: the Ultimate Guide to FreeBSD ("Absolute BSD)	M. Lucas, No Starch Press	2002	B-10
NetTap: an efficient and reliable PC-based platform for network programming ("Blott")	S. Blott, J. Brustoloni, C. Martin 2000 IEEE Third Conf. on Open Architectures and Network Programming Proceedings, pp. 13-22	Mar. 2000	Background[5]
Jails: Confirming the omnipotent root	P.H. Kamp, R.N. Watson Proceedings of the 2nd International SANE Conference, 15 pages, available at http://www.sane.nl/events/sane2000/papers/kamp.pdf	May 2000	Background[6]

C. Prior Art Systems On Sale and/or Publicly Used

The table below identifies systems and products that were publicly available and/or used, offered for sale, or sold and qualify as prior art to the Asserted Patents under 35 U.S.C. §§ 102(a), 102(b), and/or 102(g). Some or all of the supporting references for these systems, as cited below and in the corresponding Appendices, may also qualify as prior art publications under 35 U.S.C. §§ 102, such as manuals, patents, and other information intended to cover these products and systems, and may be used as invalidating prior art under 35 U.S.C. §§ 102 and 103. In addition, IBM incorporates by reference the individuals identified in its initial disclosures, and IBM also continues to investigate the systems and products and reserves the right to supplement the contentions and accompanying claim charts after further discovery and investigation. IBM also reserves the right to rely on fact or expert witness testimony to establish the public knowledge,

[5] *See*, *e.g.*, Non-Final Office Action dated October 24, 2024, U.S. Patent App. No. 90/019,676.

[6] *See*, *e.g.*, Non-Final Office Action dated October 24, 2024, U.S. Patent App. No. 90/019,676.

use, availability, sale, or offer for sale of any systems described below. Each Appendix identified by an "A" pertains to the '814 Patent, and each Appendix identified by a "B" pertains to the '058 Patent. Although additional information and prior art systems may be obtained through discovery, such prior art systems include at least the following:

Table 3: Prior Art Systems			
System/Item	**Date(s)**	**Person/Entity**	**Appendix**
HPE ServiceGuard and HP-UX	Mar. 2002	HPE	A-15 B-15
HPE Trusted Linux (a/k/a HP Secure OS)	Aug. 27, 2001	HPE	A-14 B-14
Linux VServer	2002	Jacques Gélinas	A-8
Thinstall	2001-2003	Thinstall	A-11 B-11
Virtuozzo	2002	SWsoft	A-12 B-12
User-Mode-Linux (UML)	1991-2000	Jeff Dike	B-9
FreeBSD	Mar. 14, 2000[7]	Freebsd.org	B-10
IBM Storage Tank	Jan. 23, 2002	Pease, IBM	A-13
Solaris Zones	2002[8]	Sun Microsystems	A-23 B-24

IBM's investigation into prior art systems is ongoing and discovery is only in its early stages. IBM will amend or supplement its disclosures of these systems, as its investigation and discovery continue, including third-party discovery. IBM also reserves the right to rely on fact or expert witness testimony to establish the public knowledge, use, availability, sale, or offer for sale of any systems described above and in the accompanying claim charts.

[7] https://web.archive.org/web/20000511211421/http://www.freebsd.org/releases/4.0R/announce.html; Kamp, Poul-Henning, and Robert NM Watson. "Jails: Confining the omnipotent root." In *Proceedings of the 2nd International SANE Conference*, vol. 43, p. 116. 2000.

[8] Conceived and diligently reduced to practice, as evidenced by Tucker '080, Tucker '556, Tucker Provisional, and Sun Microsystem's Solaris 10 Operating Environment software.

D. Admitted Prior Art

Statements made by the Applicants in the specification and/or during prosecution of the Asserted Patents and any related patents/applications, such as in the "Background of the Invention" section that describes the related art known to the Applicants, are admissions that can be relied upon for both anticipation and obviousness determinations, regardless of whether the admitted prior art would otherwise qualify as prior art under the statutory categories set forth in 35 U.S.C. § 102. See MPEP §§ 2129 and 608.01(c). To the extent VirtaMove contends that the Asserted Claims are not invalid as anticipated by and/or obvious in view of the prior art under 35 U.S.C. §§ 102 and 103, IBM reserves the right to rely on Applicants' "Admitted Prior Art" to demonstrate the invalidity of the Asserted Claims.

For example, the Asserted Patents and/or the alleged inventor(s) thereof admit that all of the following was prior art. IBM reserves the right to supplement the list of admitted prior art to refer to additional admissions:

- All references cited on the face of and/or in the prosecution history of the Asserted Patents, and the applicants' and/or inventors' admissions regarding such prior art.
- Admitted prior art referenced in inventor notes (VM_IBM_0001542-1845):
    - "ServerWorks LELP 3.0 chipset"; "AVAILANT" (VM_IBM_0001569).
    - "VMWARE"; "Oracle & Veritas link to Linux Kernel with kernel modules" (VM_IBM_0001575).
    - "TenSilica"; "Melanox" (VM_IBM_0001578).
    - "HURD"; "Linux or BSD kernel (monolithic)"; "GLIBC engine" (VM_IBM_0001580).
    - "Companies with Oss to check" (*see* list at VM_IBM_0001582).
    - "EGENERA" (VM_IBM_0001594).
    - "FREEBSD, netBSD, openBSD (Build on BSD)"; "Exokernel, HURD, VINO (experimental)"; "MACH, L4 + BSD components" (VM_IBM_0001598).

- “Abemco International”; “Created ‘Matting by Design’ engine patents on technology” (VM_IBM_0001614-15).
- “Novell” (VM_IBM_0001622).
- Systems referenced in diagram at VM_IBM_0001626.
- “What does PAN manager do? Why can’t we do that. Check EGENERA!” (VM_IBM_0001633).
- “PAN Manager”; “Enterprise Linux Forum” (VM_IBM_0001637).
- “How SPs do it today” (*see* systems described at VM_IBM_0001639-42).
- “Competitors: Red Hat Advanced Server; Ensine; Ejasent; Opsware?; VMware” (VM_IBM_0001654).
- “Competitors: 1) Display more servers 2) IBM VM Linux 3) SUN E10000” (VM_IBM_0001657).
- “SUN E10000 ‘Dynamic System Domain’ technology brings mainframe-style partitioning capabilities to Unix”; “A single SUN E10000 can be logically divided into multiple standalone servers”; “Each domain has one or more system boards logically isolated from the rest of the system”; “Each domain has its own instance of SOLARIS, own hostID, boot disk, network connection, memory and disk space”; “Investment Protection: Longer term compatibility of apps with HW/OS”; “64 processors/E10000 ((4 CPU’s/board) x 16 boards”; “1 → 16 dynamic system domains”; (VM_IBM_0001672).
- “Systems Management for Linux: RED HAT Command Centre (build on monitoring SW from NOC pulse [drive]. Web based GUI”; “Ximian v1.2 of Red Carpet Enterprise”; “Fivoli (IBM)” (VM_IBM_0001679).
- “NORTEL → SHASTA: Performance penalties for VXWORKS on LINUX!! READ on SHASTA” (VM_IBM_0001682).
- “VMware comparison” (*see* VM_IBM_0001692).
- “Apache”; “NFS – mount container” (VM_IBM_0001695).
- Systems referenced at VM_IBM_0001698.
- Systems referenced at VM_IBM_0001725.
- “SUN N1, HP, IBM utility computing” (VM_IBM_0001731).
- Linux and Red Hat Systems referenced at VM_IBM_0001753.

- "USER MODE LINUX" (VM_IBM_0001756).
- Systems referenced at VM_IBM_0001758-60.
- "[IBM] CLEARCASE" (VM_IBM_0001762).
- Systems referenced at VM_IBM_0001764.
- Systems referenced at VM_IBM_0001768.
- "LINUX RED HAT PRODUCT" (VM_IBM_0001772).
- "HP: PRM, WLM" (VM_IBM_0001798).
- "Itanuim (HP-UX running Oracle)"; Sun GRID containers E450's"; "Prototype a Windows solution using commercial products or something. Try running a windows image in an 'NFS' kind of mode!" (VM_IBM_0001803).
- Systems referenced at VM_IBM_0001806-07.
- "Solaris 8, 9, Spare, Frontnet application" (VM_IBM_0001810).
- "Verita cluster, SUN cluster, FLASH"; "Solaris: Apache, Tomcat, BEA, Oracle, Reportnet" (VM_IBM_0001820).
- "Solaris 10: Fire Engine TCP/IP stack"; "ZFS – Scalable 128 bit file system → bundled in with a volume mgr."; "now Solaris Containers (zones)"; "Some system calls are not permitted from a zone"; "zones use tagging with the probes structure"; "Leave /usr as a read only file system"; "60 Mb of disk space per zone" (VM_IBM_0001821).
- Systems referenced at VM_IBM_0001840-41.

- Admitted prior art referenced in deposition of Mr. Rochette:
  - Rochette (9/10/2024) Dep. Tr. at 27:6-28:20 (discussion of Sun Microsystem's Solaris operating system and support for shared libraries).
  - Rochette (9/10/2024) Dep. Tr. at 38:1-40:6 (discussion of Solaris zones).
  - Rochette (9/10/2024) Dep. Tr. at 41:11-42:6 (discussion of Solaris LD_Preload capabilities_.
  - Rochette (9/10/2024) Dep. Tr. at 44:14-22 (Discussion of 1987 article that describing interposition capabilities).

- Rochette (9/10/2024) Dep. Tr. at 51:1-5, 51:19-53:19, 54:15-55:13 (discussion of Sun Microsystem's Solaris Zones and related technologies known to software developers in Silicon Valley by 2002).

- Rochette (9/10/2024) Dep. Tr. at 57:15-58:14 (discussion of systems, including Solaris zones, known by software developers by 2002 in Silicon Valley with containers that could have their own root file system).

- Rochette (9/10/2024) Dep. Tr. at 60:10-61:7 (discussion of systems, known to software developers in Silicon Valley by 2002, that could run on top of a kernel that was residing outside of the caper and servers that could limit the resources used by a container).

- Rochette (9/10/2024) Dep. Tr. at 62:18-64:8 (discussion of systems, known to software developers in Silicon Valley by 2002, with servers that could monitor and log a container's resource usage, with containers that could have their own IP address separate from a host, and that could move software functionality into a shared library).

- Rochette (9/10/2024) Dep. Tr. at 76:8-89:13 (discussion of Solaris Zones functionality and capabilities).

- Admitted prior art referenced in Asserted Patents:

  - "The current state of the art, defines two architectural approaches to the migration of critical system elements from an operating system into an application context. In one architectural approach, a single server operating system places critical system elements in the same process. Despite the flexibility offered, the system elements continue to represent a centralized control point. In the other architectural approach, a multiple server operating system places critical system elements in separate processes. While offering even greater options this architecture has suffered performance and operational differences." '058 Patent, 1:35-45.

  - TCP/IP stack in Linux kernel and Berkeley Software Distribution. *See* '058 Patent, 5:41-49.

  - "Network services including TCP/IP, Bluetooth, ATM; or message passing protocols." '058 Patent, 6:9-10.

  - "There are many examples of this architecture in the prior art including SUN Solaris, IBM AIX, HP-UX and Linux." '058 Patent, 7:20-22.

  - "An example of the architecture described in FIG. 2a is the GNU Hurd operating system." '058 Patent, 7:27-29.

  - "An example of the architecture described in FIG. 2b is Apple's MAC OS X." '058 Patent, 7:61-62.

- "The current state of the art creates an environment where a collection of applications, each designed for a distinct function, must be separated with each application installed on an individual computer system." '814 Patent, 1:27-30.
- "Virtual Machine technology, pioneered by VmWare, offers the ability for multiple application/operating system images to effectively co-exist on a single compute platform." '814 Patent, 1:54-56.
- "A product offered by Softicity, called SoftGrid, offers what is described as Application Virtualization. This product provides a degree of separation of an application from the underlying operating system." '814 Patent, 2:4-7.
- "Linux Apache uses the following shared libraries, supplied by the OS distribution." '814 Patent, 2:55-56.
- "Applications such as Apache, MySql and PHP." '814 Patent, 4:28-29.
- "Linux, Windows and Unix systems and preferably programmed in C and C++." '814 Patent, 7:35-36.
- "Available software applications corresponding to software application icons ClearCase_Liserver, ClearCase_Registry & Samba2_clone." '814 Patent, 15:13-16.
- "Some example programming languages used in embodiments of the invention invlude, but are not limited to, C++, Java and a number of scripting languages including TCL/TK and PERL." '814 Patent, 17:15-18.

V. GROUNDS FOR INVALIDITY UNDER 35 U.S.C. §§ 102, 103

IBM contends that all Asserted Claims of the Asserted Patents are invalid under 35 U.S.C. §§ 101, 102, 103, and/or 112. The Appendices to these Invalidity Contentions contain claim charts for prior art references selected by IBM.[9] Specifically, the claim chart Appendices to these Invalidity Contentions set forth how the prior art identified by IBM anticipates either expressly or inherently, or renders obvious, each Asserted Claim. IBM cites exemplary relevant portions of the identified prior art references to provide notice of its theories of invalidity. However, other portions

[9] To the extent any of these charts cite a Figure or text related to a Figure, any corresponding text or Figure are incorporated by reference.

of he identified reference may additionally anticipate, either expressly or inherently, or render obvious one or more of the Asserted Claims. In addition, a person of ordinary skill in the art generally would view each prior art reference in the context of other publications, literature, products, systems, and understanding. As such, the cited portions are merely exemplary, and IBM reserves the right to rely on uncited portions of the prior art references and on other publications, expert testimony, and other evidence as aids in understanding and interpreting the cited portions, as providing context thereto, and as additional evidence that the prior art discloses a claim limitation or any of the Asserted Claims as whole. IBM further reserves the right to rely on uncited portions of the prior art references, other publications, and testimony, including expert testimony, to establish bases for combinations of certain cited references that render the Asserted Claims obvious. Further, as a general matter, all portions of each prior art reference are relied upon to support the disclosure of each patent claim limitation, as all portions provide general support. Supporting citations are still provided, but do not necessarily represent every location where a particular claim limitation may be found in the prior art reference.

Where IBM cites a particular drawing or figure in the accompanying charts, the citation encompasses the description of the drawing or figure, as well as any text associated with the drawing or figure (even if the associated text is not itself expressly cited). Similarly, where IBM cites particular text concerning a drawing or figure in the accompanying charts, the citation encompasses that drawing or figure as well (even if the associated drawing or figure is not expressly cited). IBM further incorporates by reference the prior art references cited during prosecution of the Asserted Patents, any patents or patent applications related to the Asserted Patents, and any patent or patent application sharing a common inventor with the Asserted Patents. This includes without limitation those identified in any other proceeding involving the Asserted

Patents or related U.S. or foreign patents. IBM additionally incorporates by reference all prior art produced or otherwise known to Plaintiff and all prior art known to the named inventors, any past or current owner, or any individual substantively involved in the prosecution of the Asserted Patents.

IBM contends that at least some of the systems and products disclosed in one or more of the prior art references identified here or in the attached Appendices are prior art under 35 U.S.C. §§ 102(a), (b), (e), (f) and/or (g), and/or 103. IBM's reference to any particular component, device, machine, or other product herein should also be interpreted as a reference to the product itself and any corresponding patents, publications, or product literature cited here or in the attached Appendices that relate to the cited component, device, machine, or other product.

IBM does not yet have complete information regarding the dates by which some of the cited products were publicly disclosed, used, sold, or offered for sale, the circumstances under which the research, design, and development activities were conducted, and the identities of the particular individuals involved in such activities through publicly available patents, publications, and product literature. IBM understands that HPE had identified persons as knowledgeable about Trusted Linux.

It is understood that a person of ordinary skill in the art reads a prior art reference as a whole and in the context of other publications, literature, and general knowledge in the field. To understand and interpret any specific statement or disclosure in a prior art reference, a person of ordinary skill in the art also relies upon other information including other publications and general scientific, engineering or other relevant knowledge. IBM reserves the right to rely upon the general scientific, engineering or other relevant knowledge in the field in interpreting the disclosure of the prior art references. In addition, IBM reserves the right to rely upon other publications and on

expert testimony to provide context and to aid in the understanding of the prior art references. IBM also reserves the right to rely upon other portions of the prior art references, other publications, and the testimony of experts to establish that the alleged inventions would have been obvious to a person of ordinary skill in the art, including the basis for modifying or combining the prior art references. Furthermore, IBM reserves the right to rely upon any admissions relating to prior art found in the intrinsic evidence for the Asserted Patents.

Certain pieces of identified prior art inherently disclose features of the Asserted Claims. IBM may rely on inherency to demonstrate the invalidity of the Asserted Claims. Moreover, certain prior art references and solutions may inherently anticipate certain features of the Asserted Claims as construed by Plaintiff. IBM may rely on other portions of the prior art, other documents, and expert testimony to establish the inherency of certain features of the prior art to invalidate the Asserted Claims. IBM also may rely on any reference identified in these invalidity contentions or any other reference to prove that the references are enabled or to explain the meaning of a term used in the solutions or any reference.

Whether a prior art reference anticipates or renders obvious the Asserted Claims may depend on claim construction. To the extent any limitation is deemed not to be met exactly by an item of prior art, IBM contends that the difference would have been obvious to a person of ordinary skill in the art and within the knowledge of one skilled in the art at the time of the alleged invention, such that the claimed invention would have been obvious both in light of the single reference alone and/or in light of combined references.

To the extent that prior art cited for a particular limitation discloses functionality that is the same or similar in some respect to the alleged functionality in the accused products as set forth in

VirtaMove's infringement contentions (notwithstanding the deficiencies therein), IBM does not concede that those limitations are in fact met by those accused functionalities.

Further, in the attached Appendices to these Invalidity Contentions, IBM has identified where the prior art references disclose subject matter recited in the preambles of the Asserted Claims, without regard to whether the preambles are properly considered to be limitations of the Asserted Claims. IBM does this only for completeness and reserves the right to argue that the preambles are or are not limiting during the claim construction proceedings in this case.

IBM's proposed obviousness combinations of the prior art references are separately identified throughout these Invalidity Contentions (including in Section V.B, below). Also, reasons to combine prior art references with each other or with other secondary references are provided below in detail. In addition to the exemplary prior art references, systems, and teachings identified herein and in the accompanying Appendices, the common knowledge of those of ordinary skill in the art also constitutes prior art, and may be relied on by IBM through testimony or declarations in combination with the identified prior art. The knowledge of a person of ordinary skill in the art at the time of the alleged invention would also include, for example, the known advantages and disadvantages of each of the limitations recited in the Asserted Claims. It would have been obvious to combine the references and teachings identified herein with one another based on the nature of the problem to be solved, the general knowledge of a person of ordinary skill in the art, the teachings of other art in existence at that time, and the recognition of similarities between the art and the claimed features. On these bases, each identified prior art item can be combined with others if a particular prior art item lacks or does not exactly disclose a limitation or feature of an Asserted Claim.

More particularly, and without limitation, all of the following recognized rationales support a finding of obviousness:

1. Combining prior art elements according to known methods to yield predictable results;

2. Simple substitution of one known element for another to obtain predictable results;

3. Use of known techniques to improve similar devices, methods, or products in the same way;

4. Applying a known technique to a known device, method, or product ready for improvement to yield predictable results;

5. "Obvious to try"—*i.e.*, choosing from a finite number of known, predictable solutions, with a reasonable expectation of success;

6. Known work in one field of endeavor prompting variations of it for use in either the same field or a different field based on design incentives or other market forces if the variations would have been predictable to a POSITA; and

7. Some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art items or to combine prior art teachings to arrive at the claimed invention.

These rationales would have motivated a person of ordinary skill in the art to combine the identified prior art items and teachings. As the Supreme Court held in *KSR Int'l Co. v. Teleflex, Inc.*, "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." 550 U.S. 398, 416 (2007). The Supreme Court further held that "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill in the art can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 418.

"[I]n many cases a person of ordinary skill will be able to fit the teachings of multiple

patents together like pieces of a puzzle." *Id.* at 420. For example, the Supreme Court held that it is sufficient that a combination of elements was "obvious to try," recognizing that "[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense." *Id.*

Patents like the Asserted Patents here are invalid, the Supreme Court explained, because "[g]ranting patent protection to advances that would occur in the ordinary course without real innovation retards progress and may, in the case of patents combining previously known elements, deprive prior inventions of their value or utility." *Id.* at 402. Here, each of the prior art items identified above is directed to the same field of endeavor as the Asserted Patents. Further, each prior art item was directed to the same problem or problems that the Asserted Patents purportedly addressed. The Asserted Claims therefore represent, at the most, mere "ordinary skill and common sense," not innovation. In other words, the alleged inventions of the Asserted Patents are, at most, obvious combinations or adaptations of well-known technology used in predictable ways.

IBM reserves the right to further streamline and reduce the number of anticipation and/or obviousness references relied on with respect to a given Asserted Claim and to exchange or otherwise modify the specific references relied upon for anticipation and within each obviousness combination for each asserted claim. Discovery is at an early stage, and VirtaMove has not provided sufficient contentions or documentation with respect to any pre-filing invention dates or with respect to claim limitations that are allegedly lacking or not obvious in the prior art.

IBM's reliance on each prior art reference identified throughout these Invalidity Contentions includes the reference itself, anything incorporated by the reference or described as

relevant technology by the reference, any system embodying the reference, and any testimony by those with knowledge of the reference, such as named authors and inventors. All such documents and information shall be considered one prior art reference to the extent they describe a single prior art system, technology or solution. Moreover, while some prior art may be charted separately, IBM reserves the right to show that combinations of individual charts describe a portion of single prior art system or solution. To the extent that VirtaMove argues that some limitation is not shown in the primary reference, IBM reserves the right to show that element would have been well known to one of ordinary skill in view of other documents describing the same technology or was admitted prior art of the named inventor. IBM reserves the right to revise, amend, and/or supplement the information provided herein, including by identifying and relying on additional references, based on developments in the case including, without limitation, based on changes in the priority date of any Asserted Claim, newly discovered prior art, depositions and document productions of prior art witnesses, claim construction determinations, challenges by VirtaMove to the authenticity or content of the prior art and positions taken by VirtaMove during the litigation. In particular, IBM reserves the right to supplement these Invalidity Contentions as discovery reveals further information about prior use and/or prior public knowledge of products, methods, or systems that IBM has reason to believe anticipate or render obvious one or more Asserted Claims. IBM further reserves the right to identify and rely on additional prior art in VirtaMove's possession, custody, or control that has not yet been produced or identified to IBM. For instance, as discussed above, IBM anticipates seeking discovery from several of the individuals and companies associated with the references below and reserve the right to rely on such discovery and/or supplement these contentions to the extent that discovery reveals additional facts and/or prior art bases. Discovery is ongoing, and IBM reserves the right to rely on additional facts and information as its

investigation continues. IBM reserves the right to assert that any of the Asserted Claims are invalid under 35 U.S.C. § 102(f) in the event IBM obtains evidence that any one or more of the named inventors did not invent (either alone or in conjunction with others) the subject matter claimed in the Asserted Claims.

IBM incorporates by reference all prior art and the invalidity grounds identified by Amazon.com, Inc., Amazon.com Services LLC, and Amazon Web Services, Inc. in their Invalidity Contentions in Case No. 7:24-cv-00030 (W.D. Tex.), Google LLC in its Invalidity Contentions in Case No. 7:24-cv-00033 (W.D. Tex.) or HPE in its Invalidity Contentions in Case No. 2:24-CV-00093 (E.D. Tex.). IBM also incorporates by reference all prior art, submissions, statements, or other materials relating to the reexamination request for the ’058 Patent filed by Unified Patents, LLC on September 23, 2024, which was granted on October 24, 2024, and assigned Reexamination Control No. 90/019,676. IBM reserves the right to rely on such prior art for anticipation, obviousness, and to provide context to or aid in establishing the state of the art and understanding the cited portions of the identified prior art or as showing reasons why one of skill in the art would have combined references.

A. Anticipation

1. Anticipation of USP 7,519,814

IBM contends that the Asserted Claims of the ’814 Patent are anticipated by the prior art, or are obvious in view of the prior art, as set forth in the table below. IBM provides Appendices A-1 through A-23, which contain claim charts demonstrating the invalidity of the Asserted Claims of the ’814 Patent.

’814 Patent Claims	Basis for Invalidity
Asserted Claims 1, 2, 6, 9, and 10	Each Asserted Claim is anticipated or rendered obvious by the following prior art, either alone or in combination with the knowledge of a person of ordinary skill in the art, the state of

	the art, admitted prior art, or in combination one or more other prior art references as described below in Section V.B: • Tucker ’080 (A-1) • Tucker ’556 (A-2) • Tucker Provisional (A-3) • Berger ’466 (A-4) • Griffin (A-5) • Tormasov (A-6) • Schaefer (A-7) • Gelinas/VServer (A-8) • Osman (A-9) • Schmidt (A-10) • Thinstall (A-11) • Virtuozzo (A-12) • IBM Storage Tank (A-13) • HPE Trusted Linux (A-14) • HPE ServiceGuard (A-15) • Berger ’436 (A-16) • Choo ’950 (A-17) • HP C++ Programmer’s Guide (A-18) • HP C Compiler Reference Manual (A-19) • McMillan (A-20) • Gaines (A-21) • Curtis (A-22) • Solaris Zones (A-23)

2. Anticipation of USP 7,784,058

IBM contends that the Asserted Claims of the ’058 Patent are anticipated by the prior art, or are obvious in view of the prior art, as set forth in the table below. IBM provides Appendices B-1 through B-24, which contain claim charts demonstrating the invalidity of the Asserted Claims of the ’058 Patent.

’058 Patent Claims	Basis for Invalidity
Asserted Claims 1, 2, 3, 4, and 18	Each Asserted Claim is anticipated or rendered obvious by the following prior art, either alone or in combination with the knowledge of a person of ordinary skill in the art, the state of the art, admitted prior art, or in combination one or more other prior art references as described below in Section V.B:

	• D'Souza (B-1) • Deianov (B-2) • Eggert (B-3) • Ely Paper (B-4) • Gingell (B-5) • Levine (B-6) • Maeda (B-7) • Thekkath (B-8) • User-Mode Linux (B-9) • FreeBSD (B-10) • Thinstall (B-11) • Virtuozzo (B-12) • Ely Presentation (B-13) • HPE Trusted Linux (B-14) • HPE ServiceGuard (B-15) • Berger '466 (B-16) • Berger '436 (B-17) • Choo '950 (B-18) • HP C++ Programmer's Guide (B-19) • HP C Compiler Reference Manual (B-20) • Hunt (B-21) • Fowler (B-22) • Benfield (B-23) • Solaris Zones (B-24)

B. Obviousness

IBM hereby expressly incorporates by reference any statements and reasons set forth by the Examiner during prosecution of the Asserted Patents and related patent applications as to why it would have been obvious to modify or combine references to achieve the limitations of the Asserted Claims. In addition, IBM provides additional motivations to combine as to specific exemplary combinations and elements, as discussed in the sections that follow. The Asserted Claims are anticipated and/or rendered obvious as set forth herein, in the accompanying Appendices, and as further discussed below. To the extent that any of the references charted in the Appendices are found not to anticipate the Asserted Claims, the foregoing references render the asserted claims obvious either alone or in combination with each other under § 103.

A massive body of relevant, analogous prior art existed as of the earliest filing dates of the Asserted Claims, as identified, for example, in §IV, *supra*. Taken together, this prior art discloses each and every limitation of every asserted claim. Example combinations of specific prior art references that IBM may rely on at trial to demonstrate obviousness of the Asserted Claims are set forth below. Such obviousness combinations are provided in the alternative to IBM's anticipation contentions and are not to be construed to suggest that any references included in the combinations is not by itself anticipatory or that any particular limitation is missing from a particular reference.

Given the scope and content of these prior art references, the lack of any substantial differences between these references and the Asserted Claims, and the level of ordinary skill in the art at the time of the alleged inventions, it would have been obvious to combine the specified references. *See KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398 (2007). Moreover, a person of ordinary skill in the art had a motivation to combine these references because—at a minimum—they all pertain to the subject of containment of computer applications and the sharing or isolation of resources.

One or more combinations of the prior art references identified above would have been obvious because those references would have been combined using: known methods to yield predictable results; known techniques in the same way; a simple substitution of one known, equivalent element for another to obtain predictable results; and/or teaching, suggestion, or motivation in the prior art generally. In addition, it would have been obvious to try combining the prior art references identified above because there were only a finite number of predictable solutions and/or because known work in one field of endeavor prompted variations based on predictable design incentives and/or market forces either in the same field or a different one. In addition, the combination of the prior art references identified above would have been obvious

because the combination represents the known potential options with a reasonable expectation of success.

Additional evidence that there would have been a motivation to combine the prior art references identified above includes the interrelated teachings of multiple prior art references; the effects of demands known to the design community or present in the marketplace; the existence of a known problem for which there was an obvious solution encompassed by the Asserted Claims; the existence of a known need or problem in the field of the endeavor at the time of the invention(s); and the background knowledge that would have been possessed by a person of ordinary skill in the art.

Thus, the motivation to combine the teachings of the prior art references disclosed herein is found in the references themselves and: (1) the nature of the problem being solved; (2) the express, implied and inherent teachings of the prior art; (3) the knowledge of persons of ordinary skill in the art; (4) the predictable results obtained in combining the different elements of the prior art; (5) the predictable results obtained in simple substitution of one known element for another; (6) the use of a known technique to improve similar devices, methods, or products in the same way; (7) the predictable results obtained in applying a known technique to a known device, method, or product ready for improvement; (8) the finite number of identified predictable solutions that had a reasonable expectation of success; and (9) known work in various technological fields that could be applied to the same or different technological fields based on design incentives or other market forces.

Any reference or combination of references that anticipates or makes obvious an asserted independent claim also makes obvious any asserted claim dependent on that independent claim because every element of each dependent claim was known by a person of ordinary skill in the art

at the time of the alleged invention, and it would have been obvious to combine those known elements with the independent claims at least as a matter of common sense and routine innovation. Accordingly, IBM contends that each Asserted Claim would have been obvious not only by the combinations explicitly defined in these contentions, but also by any combination of references that renders obvious an Asserted Claim.

Numerous prior art references, including those identified above, reflect common knowledge and the state of the prior art before the priority date claimed on the face of each asserted patent. For at least the reasons described above and below in the examples provided as well as in the claim chart appendices, it would have been obvious to a person of ordinary skill in the art to combine any of a number of prior art references, including any combination of those identified in Appendices A-1 to A-23 and B-1 to B-24, to meet the limitations of the Asserted Claims. As such, IBM's inclusion of exemplary combinations does not preclude IBM from identifying other invalidating combinations as appropriate.

1. Obviousness of USP 7,519,814

(i) Servers With Containers and Containment Mechanisms References

The Asserted Claims of the '814 Patent recite "a plurality of secure containers of application software" and further reference "each container," "the containers," and "a container." The Asserted claims also recite "each container comprising one or more of the executable applications and a set of associated system files" and "wherein the application software cannot be shared between the plurality of secure containers of application software" (claim 1), "each container has an execution file associated therewith for starting the one or more applications" (claim 2), and "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" (claim 10).

Nothing about the claimed, generic result of "secure containers" as recited in the above-referenced claims adds anything non-obvious over the prior art. *See, e.g.*, U.S. 6,966,010 1:67-2:2, 7:22-26 ("Since all the resources are in one container, a Package makes it easy for Cluster managers to monitor and manage the associated application," "Containers of information (whether they be Packages or in other formats) may be utilized in accordance with the present invention for other and/or additional purposes as provided for in the detailed description, drawing figures and claims."). The use of containers and containment mechanisms for computer applications was well known in the art before the alleged inventions of the '814 Patent. *See*, *e.g.*, *id.* To the extent any of the container claim elements is not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- Tucker '080 (A-1)
- Tucker '556 (A-2)
- Tucker Provisional (A-3)
- Berger '466 (A-4)
- Griffin (A-5)
- Tormasov (A-6)
- Schaefer (A-7)
- Gelinas/VServer (A-8)
- Osman (A-9)
- Schmidt (A-10)
- Thinstall (A-11)

- Virtuozzo (A-12)
- IBM Storage Tank (A-13)
- HPE Trusted Linux (A-14)
- HPE ServiceGuard (A-15)
- Berger ’436 (A-16)
- Choo ’950 (A-17)
- HP C++ Programmer’s Guide (A-18)
- HP C Compiler Reference Manual (A-19)
- McMillan (A-20)
- Gaines (A-21)
- Curtis (A-22)
- Solaris Zones (A-23)

A person of skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to improve the security of a system, such as by isolating applications from each other and from the host system, to improve portability of applications for migrations and/or redundancy, or to contain faults and intrusions of applications and thereby improve security, availability, and/or reliability, would seek to do so by implementing a containment mechanism such as a secure container.

Each of the references listed above, as demonstrated in the corresponding Appendices, discloses that containerization and containment was an obvious way to improve the security, portability, and resilience and reliability of a computer system. *See*, *e.g.*, Tucker ’080 at 3:59-4:3 (“In one embodiment, kernel 150 enforces the Zone boundaries. More specifically, kernel 150 ensures that processes 170 in one non-global Zone 140 are not able to access or affect processes,

file systems, and network interfaces of another Zone (non-global or global). In addition to enforcing the Zone boundaries, kernel 150 also provides a number of other services. These services include but are not limited to mapping the network interfaces 182 of the non-global Zones 140 to the physical network devices 120 of the computer system, and mapping the file systems 180 of the non-global Zones 140 to an overall file system and a physical storage 110 of the computer system."); Tucker '556 at 7:35-47; Tucker Provisional at 25 ("One of the basic tenets of the zone design is that no process running within a (non-global) zone, even one with super-user credentials (running with an effective user id of 0), is allowed to view or affect activity in other zones. This implies that any operation initiated from within a zone must have an effect that is local to that zone."); Berger '466 at [0035] ("As described above, containment is an effective security mechanism to implement within a system. As described in greater detail hereafter, containment functionality may be implemented within a system by utilizing compartments within the system. In general, compartments refer to groups of processes or threads which are limited to accessing certain subsets of system resources of a computer system. Thus, compartments are semi-isolated portions of a system. For example, an operating system for supporting a plurality of processes (e.g., applications) may be implemented on a system, wherein at least some of the processes are provided with a label or tag, each label or tag being indicative of a logically protected computing environment or "compartment." Each process having the same label or tag may belong to the same compartment. In certain implementations, containment functionality can be provided by mandatory protection of processes, files and network resources, with the principal concept being based on the compartment. Services and processes (e.g., applications) on the system may be run within separate compartments. Processes within each compartment may only have direct access to the resources in that compartment. Access to other resources, whether local or remote, may be

allowed only via well-controlled communication interfaces. Exemplary implementations of compartments within a system are described in further detail hereafter."); Griffin at [0031] ("Suitably, the computing environment 24 comprises a compartment. The actions or privileges within a compartment are constrained, particularly to restrict the ability of a process to execute methods and operations which have effect outside the compartment, such as methods that request network access or access to files outside of the compartment. Also, operation of the process within the compartment is performed with a high level of isolation from interference and prying by outside influences."); Tormasov at [0025], [0026]; Schaefer at [0013]; Gelinas (VServer) at 9; Osman at 1 ("A pod provides a group of processes with a private namespace that presents the process group with the same virtualized view of the system. This virtualized view associates virtual identifiers with operating system resources such as process identifiers and network addresses. This decouples processes in a pod from dependencies on the host operating system and from other processes in the system."); Schmidt at 18 ("A compute capsule is a private, portable, persistent computing environment that contains active processes."); Thinstall; Virtuozzo; IBM Storage Tank; HPE Trusted Linux; HPE ServiceGuard; Berger '436 at [0013] ("Compartment-based trusted operating system 100 comprises a plurality of compartments. Applications are installed and processes are run within these separate compartments."); Choo '950 at 1:39-49, 2:47-56, 2:32-46 ("Compartments refer to groups of processes or threads which are limited to accessing certain subsets of system resources of a computer system."); McMillan at 4:60-65 ("It is therefore convenient to create a virtual OS environment that contains a set of applications configured in a specific way (or a 'clean' virtual OS environment that contains no applications yet) and store it in a separate or central location to be used by multiple computers or the same computer at different times."); Gaines at 4:5-23 ("In a preferred embodiment, the system includes a host virtual

operating system, resident on a host computer and having a set of resources including process control, a file system, interprocess communications, and a set of device interfaces, overlaid on and distinguished from the host computer's actual resources."); Curtis at 1:67-2:2, 7:22-26 ("Since all the resources are in one container, a Package makes it easy for Cluster managers to monitor and manage the associated application," "Containers of information (whether they be Packages or in other formats) may be utilized in accordance with the present invention for other and/or additional purposes as provided for in the detailed description, drawing figures and claims."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 76:21-83:21 (discussing containers in Solaris Zones).

As these exemplary teachings demonstrate, improving existing application computing systems that are ready for improvement by implementing a containment mechanism was known in the art and would yield predictable results. Such improvement would be a combination of prior art references according to known methods to yield predictable results. A person of ordinary skill in the art would have had a reasonable expectation of success in making such modifications or combinations. Known techniques of computer programming and networking, such as those set forth in the HP C++ Programmer's Guide and the HP C Compiler Reference Manual, within the knowledge of a person of ordinary skill in the art would suffice to make any such combination and where, as here, such combinations are modifications implemented through software programming, i.e., predictable and well-developed fields, the results of making the combinations would be predictable. Moreover, as all such systems are within the same field of endeavor, the combination would have been obvious to a POSITA because it is simply the use of a known technique to improve similar devices or systems in the same way. Moreover, such a combination would be predictable because the combination is merely one of consolidating and/or interfacing disparate systems, and thus each system and its components continue to function in the combination as it

did separately. Moreover, such a combination would be predictable because each system and its components continue to function in the combination as it did separately.

In addition, a person of ordinary skill would have been motivated to combine or adapt known or familiar methods in the art, especially where market forces prompt such variations. For example, to the extent that market forces demanded security, portability, availability, reliability, or improved ability to migrate applications, as discussed above, a POSITA would have thought to combine or modify references that described known techniques for doing so because such techniques offered improvements to existing systems. Further, as described above and in the corresponding Appendices, each of the references listed above describes techniques that were known to offer such improvements, and, accordingly a person of ordinary skill would have been motivated to combine the references. The teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends in the field, and knowledge that the art is of special interest or importance in the field.

(ii) Different Operating System References

The Asserted Claims of the ’814 Patent generally recite a system having “a plurality of servers” with “operating systems that differ, operating in disparate computing environments,” and “different operating systems.” Nothing about these elements adds anything non-obvious over the prior art. Systems comprising a plurality of servers with operating systems that differ, operating in disparate computing environments, were well known in the art before the alleged invention of the Asserted Patents. To the extent these elements are not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and their corresponding appendices, such that the references in

combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- Tucker ’080 (A-1)
- Tucker ’556 (A-2)
- Tucker Provisional (A-3)
- Berger ’466 (A-4)
- Griffin (A-5)
- Tormasov (A-6)
- Schaefer (A-7)
- Gelinas/VServer (A-8)
- Osman (A-9)
- Schmidt (A-10)
- Thinstall (A-11)
- Virtuozzo (A-12)
- IBM Storage Tank (A-13)
- HPE Trusted Linux (A-14)
- HPE ServiceGuard (A-15)
- Berger ’436 (A-16)
- Choo ’950 (A-17)
- HP C++ Programmer’s Guide (A-18)
- HP C Compiler Reference Manual (A-19)
- McMillan (A-20)
- Gaines (A-21)
- Curtis (A-22)

- Solaris Zones (A-23)

A person of ordinary skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to improve compatibility and availability, to achieve distributed containment, or to improve the customer adoption of a solution, of the secure containers of application software described in the preceding section, would seek to do so by adapting and modifying the containment mechanism from one operating system for a different operating system.

Each of the references listed above demonstrates that providing e a containment mechanism that was supported across multiple operating systems or platforms was an obvious way to improve the compatibility, availability, adoptability, or ability to distribute a containment solution, and that a person of skill in the art would seek to do so by adapting the containment mechanism for use by multiple different operating systems to account for potential platforms utilized by customers or providers. *See*, *e.g.*, Trusted Linux; Tucker '080 at 2:50-60 ("For illustrative purposes, the OS is assumed to be SolarisTM manufactured by Sun Microsystems, Inc. of Santa Clara, Calif. However, the concepts taught herein may be applied to any OS, including but not limited to Unix, Linux, Microsoft Windows, MacOS, etc."); Tucker '556 at 6:17-27; Tucker Provisional at 1 ("Consolidation is already common in mainframe environments, where technology to support running multiple applications and even operating systems on the same hardware has been in development since the late 1960's."); Berger '466 at [0045] ("However, it will be apparent to a person skilled in the art that the principles described in conjunction with such trusted Linux OS may be applied to other types of Oss to achieve the same or similar effects."); Griffin at [0001] ("The present invention relates in general to a method for providing multiple

computing environments running on a single host computing platform, and relates to a method for verifying integrity of the computing environments."); Schaefer at [0005], [0026], [0032]; Gelinas (Vserver) at 29 ("For now the vserver project is a little redhat/mandrake aware, but some are using it with other distro. Once a service in running in a vserver, it is talking directly to the kernel. So a Debian vserver could running on a redhat server or the reverse."); Osman at 12; Schmidt at 1 ("As machines and networks continue to improve in both price and performance, users will enjoy fast, interactive access to a globally distributed set of virtually unlimited computing resources. In the traditional computing model, users had access to powerful desktop computers, and a great deal of personal computing was performed by local hardware. Today, we have already begun to see a trend toward moving personal computing off local desktop resources and onto remote servers. The worldwide web is a good example of the trend in this direction, as computing is increasingly being performed on a user's behalf by anonymous machines in distant corners of the internet."); Thinstall; Virtuozzo; IBM Storage Tank; HPE ServiceGuard; Berger '436 ("Graphical software installation tool 102 of the preferred embodiment of the present invention may be utilized on a computer system using any operating system, such as LINUX®, UNIX®, AIX®, HP-UX® and/or the like, now know or later developed. However, it is most advantageous when used in a computer system with a trusted operating system utilizing the concept of compartments to reduce the extent to which data stored on the computer system is compromised in case of attack by hackers."); Choo at 2:47-56 ("According to embodiments of the present invention, by utilizing compartments, the security of a computer system may be enhanced through mandatory access control. Mandatory access control refers to access control that a process cannot override."); McMillan at 4:21-25 ("Because all of the file system and registry calls are redirected, a copy of the base OS file system and registry must be created in the virtual OS environment file system and registry before any

applications may be executed in the virtual OS environment."); Gaines at 3:53-56 ("The virtual machine provides a layer of software, typically part of the operating system software, that simulates a computer that has been reserved for execution of the specific program."); Curtis at 1:18-24, 3:61-64, and 4:66-5:3 ("Cluster are often defined as parallel or distributed systems that consist of a collection of interrelated whole computers, that is utilized as a single, unified computing source. As such, it is commonly appreciated that Clusters commonly consist of computing devices (i.e., Nodes) and other peripheral devices to which access thereto may be controlled by particular Nodes." "[S]pecialized code is not needed nor are changes to the source code required in order to expand an application's capabilities or to implement a Cluster aware application on a Cluster." "It is to be appreciated, that for certain applications, the Package may be an extremely large segment of operating instructions, software codes and other information that may be implemented by each Node on the Cluster."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 50:19-51:5, 51:7-22, 75:8-11 (discussing Solaris Zones and servers running different operating systems in disparate computing environments).

As these exemplary teachings demonstrate, improving existing computer systems that implement containment mechanisms or containers by adding support for multiple different operating systems in the same way was known, desired, and would yield predictable results. Also, a person of ordinary skill in the art would understand that making such adaptions would involve the use of known techniques, *e.g.*, the implementation of containment for one operating system, to improve similar operating systems in the same way, such as different versions of the same Linux distribution, or Linux and Unix-based operating systems given the known relationship and common programming languages, e.g., C, utilized to implement those operating systems' kernels. *See* HP C++ Programmer's Guide; HP C Compiler Reference Manual.

Also, a person of ordinary skill would recognize the exemplary teachings of the references as providing an express suggestion or motivation to implement the containment mechanism on different operating systems, thereby leading a person of ordinary skill to modify other operating systems to arrive at the same containment mechanism supported by different operating systems. This is expressly seen, as one non-exhaustive example, with the numerous express statements of the desire to implement the Trusted Linux containment mechanism on different operating systems and the Tucker containment mechanism, as stated above. Market forces also prompt such variations and modifications, as indicated above as a variety of different platforms all access and utilize a common network such as the Internet.

A person of ordinary skill would have had a reasonable expectation of success in making the combination or modifications. For example, kernel modifications to one operating system to implement containment, such as Linux, are easily replicated across related but different operating systems, such as Unix, or even different versions of the Linux kernel. In fact, the implementation for one version of an operating system may be identical for a different version or release of that operating system, such as between different minor releases of the same operating system.

(iii) Container Containing Copies or Modified Copies of Kernel/OS System File References

The Asserted Claims of the ’814 Patent recite “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, 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.” The presence of associated system files within a container, and those files being used during execution of the containerized application rather than the same or similar files that may exist in the host’s location operating system files, does not add anything non-

obvious over the prior art. This was well-known in the art before the alleged inventions of the ’814 Patent. Likewise, the presence of the associated files within a container being so because those files were copied from the host operating system files also does not add anything non-obvious over the prior art. This was well-known and routine in the art before the alleged inventions of the ’814 Patent. To the extent these elements are not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- Tucker ’080 (A-1)
- Tucker ’556 (A-2)
- Tucker Provisional (A-3)
- Berger ’466 (A-4)
- Griffin (A-5)
- Tormasov (A-6)
- Schaefer (A-7)
- Gelinas/Vserver (A-8)
- Osman (A-9)
- Schmidt (A-10)
- Thinstall (A-11)
- Virtuozzo (A-12)
- IBM Storage Tank (A-13)
- HPE Trusted Linux (A-14)
- HPE ServiceGuard (A-15)

- Berger ’436 (A-16)
- Choo ’950 (A-17)
- HP C++ Programmer’s Guide (A-18)
- HP C Compiler Reference Manual (A-19)
- McMillan (A-20)
- Gaines (A-21)
- Curtis (A-22)
- Solaris Zones (A-23)

A person of skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to improve the security and stability of their applications, and to reduce competition between applications for system resources, isolate or contain applications from one another and preclude access to file system resources, and improve application availability, would seek to do so by creating and providing local copies within the container of any application dependencies from the host operating system/kernel for the containerized application to utilize.

Each of the references listed above, as demonstrated in the corresponding appendices, discloses that copying application dependencies into a container for use by the containerized application was an obvious way to improve the security, containment, isolation, and availability of the applications and services. *See*, *e.g.*, Tucker ’080 at Abstract (“A separate syslogd process executes in association with each zone. Each zone is associated with a separate virtual file system that contains a separate log file. Application processes executing in association with a zone can send messages to the log device node for that zone. The syslogd process executing in association

with a zone reads messages from that zone's log device and writes at least some of the messages into that zone's log file. Consequently, the system logging ability is maintained in all of the zones, while isolating the processes and data in each zone from the processes and data in each other non-global zone. Each of the zones is associated with a separate log device node. A separate syslogd process executes in association with each zone. Each zone is associated with a separate virtual file system that contains a separate log file. Application processes executing in association with a zone can send messages to the log device node for that zone. The syslogd process executing in association with a zone reads messages from that zone's log device and writes at least some of the messages into that zone's log file. Consequently, the system logging ability is maintained in all of the zones, while isolating the processes and data in 'ach zone from the processes and data in each other non-global zone."); Tucker '556 at 9:27-44 ("Further, non-global zones 140 provide a virtualized environment because they are capable of supporting the isolated execution of processes, such as a process A1 174-1(a), executing in non-global zone A 140(a) and a process B1 174-1(b) executing within non-global zone B 140(b). Both of non-global zones 140(a) and 140(b) are able to exist under a single kernel image 150."); Tucker Provisional at 2 ("Figure 1.1 shows a system with four zones. Zones 1, 2 and 3 are each running a disjoint workload in a sample consolidated environment. This example demonstrates that different versions of the same application may be run without negative consequence in different zones, to match the consolidation requirements. Each zone can provide an almost arbitrarily rich and customized set of services."); Berger '466 at [0102] ("Specifically, numerous approaches can be utilized to prevent processes associated with a compartment from accessing system resources."); Griffin at [0041] ("For the purposes of the present invention, the guest operating system 25 is preferably a replica of the host operating system, or at least necessary parts thereof."); Tormasov at [0025] ("Each virtual computing

environment includes a complete set of processes and files of an operating system that can be modified by the end user."); Schaefer at [0041] ("If an application tries to create a new file or open an existing file for writing on a user's local drive, the Operating System Guard must ensure that the file is actually created or modified in the redirected location."); Vserver; Osman at 7 ("The directory can be created on the network file server then mounted within the pod virtual file system as /private-pod. When a private pod directory is created, it may optionally be pre-populated with data from a template directory."); Schmidt at 33 ("Compute capsules embody a user's personal computing environment. As such, we would like them to be isolated from each other, capable of being customized, and managed by the underlying system."); Thinstall; Virtuozzo; IBM Storage Tank; HPE Trusted Linux; HPE ServiceGuard; Berger '436 at [0040] ("In step 156, supporting resources, such as libraries, configuration files, and/or the like, desirable to install application 120 in the selected compartment are automatically determined. The supporting resources may be determined, for example, by querying an executable file associated with application 120 itself."); Choo '950 at 5:4-12 ("In embodiments of the present invention, access to files by processes is restricted by rules based on process compartments."); HP C++ Programmer's Guide at 3-5, 3-8 ("The link-time template processing phase runs c++ptlink and retrieves information about the templates from the repository to automatically instantiate templates. c++ptlink may create additional object files in the repository."); HP C Compiler Reference Manual at 206 ("Generates shared library object code."); McMillan at 4:60-65 ("It is therefore convenient to create a virtual OS environment that contains a set of applications configured in a specific way (or a 'clean' virtual OS environment that contains no applications yet) and store it in a separate or central location to be used by multiple computers or the same computer at different times."); Gaines at 4:5-23 ("In a preferred embodiment, the system includes a host

virtual operating system, resident on a host computer and having a set of resources including process control, a file system, interprocess communications, and a set of device interfaces, overlaid on and distinguished from the host computer's actual resources."); Curtis at 1:50-59, 4:3-9 ("Cluster managers often utilize 'Packages', that contain all the resources that an application might need in order to run on a Node. These Package essentially provide the information needed by the fail-over Node in order to restart the application. Examples of information a Package may contain include information relating to the type of storage an application utilizes, the IP addresses that it uses for clients that connect to the application and other information that basically enables the application to run on the system." "The Custer management Software knows of the existence of the Package on the other Nodes of the Cluster, Such that whenever a fault occurs on one Node, other Nodes may be utilized to immediately implement the application without requiring a fail-over routine copying the needed information contained in the Package."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 76:21-83:21 (discussing Solaris Zones and container files).

As these exemplary teachings demonstrate, improving existing application computing systems that implement containerization mechanisms, including over different operating systems, that are ready for improvement by copying local operating system file dependencies of an application into the application's container was known in the art and would yield predictable results. Such improvement would be a combination of prior art references according to known methods to yield predictable results. For example, as application containment and isolation mechanisms are implemented, a person of skill in the art would understand that access to application dependencies must be ensured for the application to function, and creating and providing a copy to each application that uses the resource would be a basic, known technique for doing so, similar to how static libraries are copied and linked into an application by a C

compiler/linker. *See* HP C++ Programmer's Guide; HP C Compiler Reference Manual. Providing each application with its own copy of a system resource dependency may also be preferred over the other option of sharing access to such a system resource between containers because such sharing introduces the potential for security and/or performance issues as is known in the art regarding access to shared memory and shared resources. Furthermore, copying or preparing modified copies of system resource dependencies to include in the application container would have been obvious to try. The implementer would have the option of configuring access to a shared resource, or preparing multiple independent copies of any such resources and providing each application with its own copy. Furthermore, the teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends and best practices in the field, and knowledge that the art is of special interest or importance in the field.

A person of ordinary skill in the art would have had a reasonable expectation of success in making the copies and the modifications. For example, as indicated in the prior art identified above, preparing such copies or modified copies can be done utilizing standard programming techniques and commands. *See* HP C++ Programmer's Guide; HP C Compiler Reference Manual.

(iv) Unique Root File System for Container

The Asserted Claims of the '814 Patent recite "each of the containers has a unique root file system that is different from an operating system's root file system." Nothing about this element adds anything non-obvious over the prior art. Systems comprising containers that have unique root file systems that differ from the operating system of the host's root file system were well known in the art before the alleged invention of the Asserted Patents. To the extent these elements are not

explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and their corresponding appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- Tucker ’080 (A-1)
- Tucker ’556 (A-2)
- Tucker Provisional (A-3)
- Berger ’466 (A-4)
- Griffin (A-5)
- Tormasov (A-6)
- Schaefer (A-7)
- Gelinas/Vserver (A-8)
- Osman (A-9)
- Schmidt (A-10)
- Thinstall (A-11)
- Virtuozzo (A-12)
- IBM Storage Tank (A-13)
- HPE Trusted Linux (A-14)
- HPE ServiceGuard (A-15)
- Berger ’436 (A-16)
- Choo ’950 (A-17)
- HP C++ Programmer’s Guide (A-18)
- HP C Compiler Reference Manual (A-19)

- McMillan (A-20)
- Gaines (A-21)
- Curtis (A-22)
- Solaris Zones (A-23)

A person of ordinary skill in the art at the time of the alleged invention would have been motivated to implement a container with a unique root file system that differed from the root file system of the operating system for the reasons stated above. For example, a person of ordinary skill in the art desiring to improve the isolation, security, and containment would be motivated to implement file system protections such as mandatory access controls to restrict an application's ability to access certain files, particularly if those files or copies of those files may be utilized by other applications, programs, or services of the computing system.

Each of the references listed above, as shown in the corresponding appendices, discloses that providing a container with a unique root file system that differs from the operating systems' root file system was an obvious way to improve the containment and security of an application or hosted service. *See*, *e.g.*, Tucker '080 at 4:22-25 ("In doing so, the kernel 150 installs all of the necessary files into the Zone's root directory."); Tucker '556 at 10:3-18 ("Because root directory 290(a) is a zone root directory, it will appear to processes within the non-global zone A 140(a) to be the root directory."); Tucker Provisional at 11 ("Each zone has a path to its root directory relative to the global zone's root directory."); Berger '466 at [0041] ("Instead, filesystem protection is achieved by associating a different section of the main filesystem with each compartment. Each such section of the file system is a "chroot" of the main filesystem, and processes running within any compartment only have access to the section of filesystem which is associated with that compartment."); Griffin at [0037] ("Suitably, each compartment is allocated an individual section

of a file system of the computing platform. For example, the section is a chroot of the main file system. Processes running within a particular compartment only have access to that section of the file system."); Tormasov at [0018] ("virtual computing environment root file system is also never visible to other virtual computing environments running on the same computer."); Schaefer at [0019] ("In providing this functionality, each application is able to run in a private context within the system. To the application, it has its own private view of what the system looks like and its behavior. The present invention provides this by its inherent nature."); Vserver; Osman at 7 ("Minimally from a Unix-centric viewpoint, this set of directories would include /etc, /lib, /bin, and /usr. Each pod must also be configured with a /tmp that is private to the pod and is discussed further below. Zap then uses the chroot call to set the staging area as the root directory for the pod, thereby achieving file system virtualization with modest performance overhead."); Schmidt at 40 ("Each capsule maintains a private view of the global file system. This view is the primary mechanism for user customization, and it is similar to the per-process name space of Plan 9 [43]. When a capsule is created, a new file system view is synthesized by creating a root-level directory and mapping elements from the global file system into it."); Thinstall; Virtuozzo; IBM Storage Tank; HPE Trusted Linux; HPE ServiceGuard; Berger '436 at [0022] ("If desired, files may be further protected by gathering one or more files into a restricted file system for each compartment."); Choo '950 at 5:4-12 ("In embodiments of the present invention, access to files by processes is restricted by rules based on process compartments. Reference is now made to FIG. 4 that depicts an exemplary file system to which access is controlled by rules based on process compartments. This file system is organized according to a subdirectory structure. The highest component of this file system is the root directory (referred to as root 401)."); HP C++ Programmer's Guide at 3-9 ("Setting Your Path to the CC Command. The CC command is

normally installed in the directory /usr/bin (/opt/CC/bin for HP-UX 10.x C++ versions). So that you can use the CC command, you should ensure that your PATH environment variable includes this directory."); HP C Compiler Reference Manual at 271 ("libraries for threads"); McMillan at 4:21-25 ("Because all of the file system and registry calls are redirected, a copy of the base OS file system and registry must be created in the virtual OS environment file system and registry before any applications may be executed in the virtual OS environment."); Gaines at 8:19-32 (" Typically, the host computer 101 comprises a hierarchical file system 105, with files or other file system objects 107 organized in a tree-structured name space. Hierarchical file systems are known in the art of operating systems."); Curtis at 2:60-67, 4:66-5:3 ("[T]he present invention utilizes a Multi-Node Package (MNP) to concurrently run on multiple Nodes on a Cluster and thereby provide the desired operational performance without the downtime associated with fail-over operations. Instead of initiating a Package on a single Node, the present invention initiates a Package on every Node (or a desired percentage of Nodes) on a Cluster." "It is to be appreciated, that for certain applications, the Package may be an extremely large segment of operating instructions, software codes and other information that may be implemented by each Node on the Cluster."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 56:15-4 (discussing Solaris Zones containers having own root file system different than that of the underlying operating system).

As these exemplary teachings demonstrate, improving existing application computing systems that implement containment mechanisms by providing a container with a unique root file system that differs from the host operating system's root file system in the same way was known, desired, and would yield predictable results. A person of skill in the art would understand that making such modifications or implementing such features would involve the use of known techniques, such as the standard *chroot* command of the operating system. *See* HP C++

Programmer's Guide; HP C Compiler Reference Manual. Moreover, with reference to the copies of modified copies element discussed in the preceding section, restricting each container to a *chroot* jail would further motivate a person of ordinary skill in the art to copy or otherwise provide access to all application dependencies into the chroot environment as would be required to execute the application in the chroot environment. As this element can be satisfied by utilizing, for example, a standard and well-known command, such as *chroot*, a person of skill in the art would have had more than a reasonable expectation of success in doing so. Furthermore, the teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends and best practices in the field, and knowledge that the art is of special interest or importance in the field.

(v) Container Execution File References

Dependent claim 2 of the '814 Patent recites "wherein each container has an execution file associated therewith for starting the one or more applications." This element does not add anything non-obvious over the prior art. This was well-known in the art before the alleged invention of the '814 Patent. To the extent these elements are not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- Tucker '080 (A-1)
- Tucker '556 (A-2)

- Tucker Provisional (A-3)
- Berger ’466 (A-4)
- Griffin (A-5)
- Tormasov (A-6)
- Schaefer (A-7)
- Gelinas/Vserver (A-8)
- Osman (A-9)
- Schmidt (A-10)
- Thinstall (A-11)
- Virtuozzo (A-12)
- IBM Storage Tank (A-13)
- HPE Trusted Linux (A-14)
- HPE ServiceGuard (A-15)
- Berger ’436 (A-16)
- Choo ’950 (A-17)
- HP C++ Programmer’s Guide (A-18)
- HP C Compiler Reference Manual (A-19)
- McMillan (A-20)
- Gaines (A-21)
- Curtis (A-22)
- Solaris Zones (A-23)

A person of ordinary skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons

stated above. For example, a person of ordinary skill in the art desiring to improve the ease of use and convenience of a computer system with containerized applications would have sought to do so by implementing an execution file associated with the containers to automatically start applications contained therein or to at least facilitate the starting of applications contained therein.

Each of the references listed above, as demonstrated in the corresponding Appendices, discloses that providing an execution file associated with a container to start the applications contained therein was an obvious way to facilitate or automate the starting of the containerized applications. *See*, *e.g.*, Tucker ’080 at 4:26-42 (“To put an Installed Zone into the Ready state, a global administrator invokes another service of kernel 150, which causes the kernel 150 to start a Zoneadmd process 162 (there is a Zoneadmd process associated with each non-global Zone)”); Tucker ’556 at 8:8-25 (“To put an Installed zone into the Ready state, a global administrator invokes an operating system utility (in one embodiment, zoneadm(1 m) again), which causes a zoneadmd process 162 to be started (there is a zoneadmd process associated with each non-global zone).”); Tucker Provisional at 77 (“Applications 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.”); Berger ’466 at [0075] (“Create startup/shutdown scripts in compt/app 1”); Griffin at [0041]; Tormasov at [0025]; Schaefer at [0019], [0023], [0040] (“Many applications use data files within the application to store configuration entries or other application data. The present invention provides a virtual file system much like the virtual registry described above. Before the application starts, the present invention can load a list of file system changes, including files to hide and files to add to the virtual environment or files to redirect to another within the virtual environment.”); Vserver; Osman at 11 (“create_pod enables a user to create a new pod with various options that can be specified in a pod configuration file. These options include the network

configuration and file system configuration for the pod, what applications if any should be launched once the pod is created, and access control permissions for the pod."); Schmidt at 91 ("In addition, we added a new system program that allows a user to create a new capsule to execute a specified application. This is useful for creating capsules with arbitrary contents, not just ones that represent login sessions."); Thinstall; Virtuozzo; IBM Storage Tank; HPE ServiceGuard; Berger '436 at [0040] ("The executable file includes an area where all resources desirable to properly install the application are listed."); Choo '950 at 3:55-4:6 ("Command line utilities 304 may further include commands to execute a process in a specific compartment. For example, a command may be utilized to execute an HTTP web server application in WEB compartment 301. The command causes a thread to be created. The command also creates an entry in the thread registry of the kernel (not shown)."); McMillan at 4:21-32; Gaines at 5:20-25; Curtis at 1:50-51, 2:60-67("Cluster managers often utilize "Packages", that contain all the resources that an application might need in order to run on a Node." "[T]he present invention utilizes a Multi-Node Package (MNP) to concurrently run on multiple Nodes on a Cluster and thereby provide the desired operational performance without the downtime associated with fail-over operations. Instead of initiating a Package on a single Node, the present invention initiates a Package on every Node (or a desired percentage of Nodes) on a Cluster."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 76:21-83:21 (discussing starting container applications in Solaris Zones).

As these exemplary teachings demonstrate, improving existing container computer systems that are ready for improvement by providing an execution file associated with the container to start the container's application was known in the art and would yield predictable results. For example, a person of skill in the art would have understood that start scripts, which are files containing commands that that run to start an application, are a common and well-known tool utilized in the

industry for facilitating and automating starting of an application to improve ease of use of the application. Use of startup scripts is common in the prior art specifically with respect to starting up virtual machines, which are a containment mechanism, and these techniques could be utilized for other types of container mechanisms. Thus, this element is merely a combination of prior art elements according to known methods to yield predictable results. And a person of ordinary skill in the art would have a reasonable expectation of success in implementing a start script for a container because only routine programming techniques are required to do so and such scripts have been implemented for various application containment mechanisms as disclosed in the prior art. It is also therefore use of a known technique to improve similar features, such as containment mechanisms, in the same way, and would merely constitute applying a known technique to a known device or feature ready for improvement to yield predictable results.

(vi) Container Identifier References

Dependent claim 6 of the ’814 Patent recites “assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of an IP address, host name, and MAC address.” These elements do not add anything non-obvious over the prior art. This was well-known in the art before the alleged inventions of the ’814 Patent. To the extent these elements are not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition o the relevant teachings identified herein and/or charted, render the claim obvious.

- Tucker ’080 (A-1)
- Tucker ’556 (A-2)
- Tucker Provisional (A-3)

- Berger ’466 (A-4)
- Griffin (A-5)
- Tormasov (A-6)
- Schaefer (A-7)
- Gelinas/Vserver (A-8)
- Osman (A-9)
- Schmidt (A-10)
- Thinstall (A-11)
- Virtuozzo (A-12)
- IBM Storage Tank (A-13)
- HPE Trusted Linux (A-14)
- HPE ServiceGuard (A-15)
- Berger ’436 (A-16)
- Choo ’950 (A-17)
- HP C++ Programmer’s Guide (A-18)
- HP C Compiler Reference Manual (A-19)
- McMillan (A-20)
- Gaines (A-21)
- Curtis (A-22)
- Solaris Zones (A-23)

A person of skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to implement an organized

containerization mechanism in their system would seek to do so by assigning a unique identity to each container, where the identity includes one of an IP address, host name, and MAC address.

Each of the references listed above, as shown in the attached appendices, discloses that identifying a container using one or more of an IP address, host name, and MAC address was an obvious way to implement an organized containment mechanism. *See e.g.*, Tucker '080 at 4:25-43 ("In creating a non-global zone 140, a number of operations are performed, including but not limited to assigning a zone ID"); Tucker '556 at 7:35-47; Tucker Provisional at 1 ("For example, multiple applications running in different zones (but on the same system) can bind to the same network port using the distinct IP addresses associated with each zone."); Berger '466 at [0035] ("For example, an operating system for supporting a plurality of processes (e.g., applications) may be implemented on a system, wherein at least some of the processes are provided with a label or tag, each label or tag being indicative of a logically protected computing environment or 'compartment.'"); Griffin at [0057] ("hen it is desired to create or update a stored integrity metric for a particular computing environment, a result is reported to the trusted device driver 221 along with information identifying that particular computing environment, such as an arbitrary label. In one preferred embodiment a process ID of the virtual machine application 26 is used to identify the computing environment. In another embodiment each logical computing environment is supplied with a secret, e.g. a secret is supplied to the virtual machine application 26 by the trusted device driver 221, and then the secret is subsequently used to identify the computing environment."); Tormasov at [0018]; Schaefer at [0019]; Vserver; Osman at 5 ("Per-pod IP address"); Schmidt at 44 ("Our solution is to employ a technique much like mobile IP [41] in which a mobile host maintains a "home address" and a "care-of address." The home address is a well-known and static IP address, and the care-of address is the current IP address of the mobile

host on the network."); Thinstall; Virtuozzo; IBM Storage Tank; HPE Trusted Linux; HPE ServiceGuard; Berger '436 at [0027] ("If desired, port numbers 122 through which the compartments, for example compartments 140 through 145, communicate may be shown next to the corresponding communication rules 104."); Choo '950 at Figs. 1, 3, 4; HP C++ Programmer's Guide at 3-9 ("You should modify the command that sets the PATH variable in the appropriate shell script file, either .profile or .login, in your home directory."); HP C Compiler Reference Manual at 237 ("system model numbers of hosts where the source object files reside"); McMillan at 3:57-4:65; Gaines at 5:20-25 ("The host virtual operating system provides a safe environment for program execution, in that the owner or administrator of the host computer may use the virtual operating system to assure that the program to be executed is constrained."); Curtis at 1:50-59 ("Cluster managers often utilize "Packages", that contain all the resources that an application might need in order to run on a Node. These Package essentially provide the information needed by the fail-over Node in order to restart the application. Examples of information a Package may contain include information relating to the type of storage an application utilizes, the IP addresses that it uses for clients that connect to the application and other information that basically enables the application to run on the system"); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 62:2-18 (discussing capability of Solaris Zones containers to have their own IP address separate from a host).

As these exemplary teachings demonstrate, improving existing application computing systems that implement containers and containerization mechanisms, including over servers with different operating systems, that are ready for improvement by identifying the containers using well-known computer system identifiers was well known and would yield predictable results. Such improvement would be a combination of prior art references according to known methods to yield

predictable results. For example, the recited options of an IP address, host name, and MAC address as identifiers is a simple substitution of one known identifier for another to achieve predictable results. It also would be obvious to use an identifier such as a relocatable IP address or host name to facilitate or improve the portability of the container across different machines, for example for redundancy or failover purposes. Furthermore, utilizing an IP address, host name, and MAC address to identify a container would have been obvious to try because those three structures are common and well-known identifiers of computer system components in a networked environment. Furthermore, the teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends and best practices in the field, and knowledge that the art is of special interest or importance in the field.

A person of ordinary skill in the art would have had a reasonable expectation of success in making the combination or modification. For example, IP addresses are freely available and commonly used identifiers and would facilitate defining the routing table for the containerized application, host names are often user-defined unique names that identify a device or component on a network, and MAC addresses are also a common identifier of a device on a network. That is, each of the three claimed options are common, well-known identifiers of devices on a network and a person of skill in the art would have a reasonable expectation of success in utilizing those fields to identify a container on a network.

(vii) Container Hardware Resource Usage References

Dependent claim 9 of the ’814 Patent recites: “wherein sever 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." This element does not add anything non-obvious over the prior art. Associating hardware resource usage information, such as that identified in the claim, with computer applications and data structures, for example, prior to execution of an application was well-known in the art before the alleged invention of the '814 Patent. To the extent this element is not explicitly or inherently disclosed in a charted reference, such element is disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- Tucker '080 (A-1)
- Tucker '556 (A-2)
- Tucker Provisional (A-3)
- Berger '466 (A-4)
- Griffin (A-5)
- Tormasov (A-6)
- Schaefer (A-7)
- Gelinas/Vserver (A-8)
- Osman (A-9)
- Schmidt (A-10)
- Thinstall (A-11)
- Virtuozzo (A-12)
- IBM Storage Tank (A-13)
- HPE Trusted Linux (A-14)
- HPE ServiceGuard (A-15)

- Berger ’436 (A-16)
- Choo ’950 (A-17)
- HP C++ Programmer’s Guide (A-18)
- HP C Compiler Reference Manual (A-19)
- McMillan (A-20)
- Gaines (A-21)
- Curtis (A-22)
- Solaris Zones (A-23)

A person of ordinary skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to improve the efficiency, compatibility, availability, and uptime, of a computer system with containerized application environments, including in distributed configurations, would have sought to do so by implementing, determining, and using a pre-execution analysis and evaluation of the hardware and networking resource requirements of a container.

Each of the references listed above, as demonstrated in the corresponding Appendices, discloses that associating server information relating to hardware resource usage with a container prior to the application within the container being executed was an obvious way to improve the efficiency, compatibility, availability, and uptime of a computer system with containerized application environments, including in distributed configurations. *See*, *e.g.*, Tucker ’080 at 4:26-42 (“In creating a non-global Zone 140, a number of operations are performed, including but not limited to assigning a Zone ID. starting a Zsched process 164 (ZSched is a kernel process; however, it runs within the non-global Zone 140, and is used to track kernel resources associated with the

non-global Zone 140), mounting file systems 180, plumbing network interfaces 182, configuring devices, and setting resource controls."); Tucker '556 at 8:8-25; Tucker Provisional at 2 ("A zone does not require a dedicated CPU, physical device, or chunk of physical memory; those resources can either be multiplexed across a number of zones running within a single domain or system, or allocated on a per-zone basis using the resource management features available in the operating system[.]"), 64 ("In order to prevent processes in a zone from monopolizing the system, we will introduce zone-wide limits applicable resources, limiting the total resource usage of all process entities within a zone (regardless of project)."); Berger '466 at [0075]-[0080]; Griffin at [0068]-[0069]; Tormasov at [0023]-[0025]; Schaefer at [0044] ("Many applications use device drivers or other operating system level software to implement some of its functions such as hardware support or low level interactions directly with the operating system. In the present invention, the Operating System Guard will provide the capability of dynamically, and as possible privately, adding and removing these components to an application's virtual environment."); Vserver; Osman at 5 ("To migrate a pod, Zap first suspends the pod by stopping all processes in the pod, saving the virtualization mappings, and saving all process state, including memory CPU registers, open file handles, etc."); Schmidt at 55-56 ("Therefore, we created a measurement tool that interfaces directly with the kernel to collect detailed information on the activities of each process in the system, including bytes transmitted and received on the network, bytes read or written to disk or network files, processor utilization, physical memory occupancy, etc. All collected information is time-stamped with a high-resolution timer. This tool adds only negligible overhead to a running system, thereby allowing us to collect detailed statistics for each active job in the system."); Thinstall; Virtuozzo; IMB Storage Tank; HPE Trusted Linux; HPE ServiceGuard; Berger '436 at [0040]; Choo '950 at 2:32-46 ("Compartments refer to groups of processes or threads which are

limited to accessing certain subsets of system resources of a computer system. FIG. 1 depicts a block diagram example of compartments. This system includes two subsets of system resources (resource 1 and resource 2). This system also includes three compartments (designated compartments A, B, and C). Compartment A is only permitted to access the system resources associated with resource 1. Compartment C is only permitted to access the system resources associated with resource 2. Compartment B is permitted to access the system resources associated with both resource 1 and resource 2. As an example, if a process is designated as belonging to compartment A, the process would be allowed to access resource 1 but would be prevented from accessing resource 2."); HP C++ Programmer's Guide at 3-22 ("+DA*architecture*"); HP C Compiler Reference Manual at 238 (Pragmas), 271 ("libraries for threads"); McMillan at 3:1-4 ("The applications running within the virtual OS environments still share the base OS attributes such as networking information, user login rights, services, hardware information, and the Windows clipboard."); Gaines at 6:47-55 ("The resource filters 146 include a process control filter 151 for controlling access to the resources 122 of the host computer 101 for process control, including the set of processes 124 and the set of actions the operating system 103 permits with regard to processes 124. Typically, these actions include creating and destroying processes 124, modifying the address space 125, the allotment of processor time 126, or the permissions 127 assigned to a process 124, and interprocess communication."); Curtis at 6:59-7:7 ("Further, if the Package is a GMNP, it will do the same thing as a SMP, but in a "load balancing" embodiment. When load balancing, the Cluster manager utilizes the specifications for the Package to determine how to transfer the application. For example, such load balancing may indicate that that Package should run one at least "x" number of Nodes or at least "y"% of the Nodes in the Cluster. As such, the cluster manager may try to start another instance of the package on a Node where an instance

of that package is not currently running, or, if the load balance thresholds are being met, the Cluster manager may not start another instance of the Package. A starting of another instance of the Package on another Node, however, is not a failover because any running instances on Nodes that did not fail will continue to run while the new instance is started on the new Node."); Solaris Zones.

As these exemplary teachings demonstrate, improving existing application container computing systems that are ready for improvement by implementing and determining hardware resource usage information associated with a structure such as an application container was known in the art and would yield predictable results. For example, a container as disclosed in the prior art is implemented as a logical computer data structure and a person of ordinary skill in the art would have understood that the underlying operating systems disclosed in the prior art automatically associates the claimed hardware resource usage information, including at least memory and disk allocation information, with any file, data structure, or application or program, including a container, installed on the system, prior to execution of the file or even opening or running the file or program. A person of skill in the art would have understood at least that such operating systems evaluate necessary memory and disk allocation information prior to even installing an application or container on a system before execution. Thus, this claim recites a mere combination of prior art references according to known methods to yield predictable results. Given the ubiquity of this feature, as described for example above, a person of ordinary skill in the art would have had a reasonable expectation of success in implementing and utilizing this feature simply using known operating system features. Moreover, as all such systems are within the same field of endeavor, the combination would have been obvious to a person of ordinary skill in the art because it is simply the use of a known technique to improve similar devices or systems in the same way. And

the combination was predictable because each component or features continues to function in the combination as it did separately. This was a known technique in the prior art, and the known work in the prior art related to operating systems automatically associating hardware resource usage information with programs, files, and structures on a computer system would naturally be applied to an application container because a container is simply a logical data structure.

2. Obviousness of USP 7,784,058[10]

(i) User Mode Shared Library References

The Asserted Claims of the ’058 Patent recite “a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode” and “wherein some of the SLSCEs stored in the shared library…are accessible to some of the plurality of software applications.” This element does not add anything non-obvious over the prior art. A user mode shared library including “critical system elements” well-known in the art before the alleged inventions of the ’058 Patent. To the extent this element is not explicitly or inherently disclosed in a charted reference, such element is disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- D'Souza (B-1)
- Deianov (B-2)
- Eggert (B-3)

[10] In light of the obviousness of the ’058 Patent based upon known solutions in the field prior to the priority date of the ’058 Patent, the United States Patent and Trademark Office recently granted a request for *ex parte* reexamination of the ’058 Patent due to a substantial new question of patentability affecting claims 1 and 12. *See* Non-Final Office Action dated October 24, 2024, U.S. Patent App. No. 90/019,676.

- Ely Paper (B-4)
- Gingell (B-5)
- Levine (B-6)
- Maeda (B-7)
- Thekkath (B-8)
- User-mode Linux (B-9)
- FreeBSD (B-10)
- Thinstall (B-11)
- Virtuozzo (B-12)
- Ely Presentation (B-13)
- Trusted Linux (B-14)
- HPE ServiceGuard (B-15)
- Berger ’466 (B-16)
- Berger ’436 (B-17)
- Choo (B-18)
- HP C++ Programmer's Guide (B-19)
- HP C Compiler (B-20)
- Hunt (B-21)
- Fowler (B-22)
- Benfield (B-23)
- Solaris Zones (B-24)

A person of skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated

above. For example, a person of ordinary skill in the art desiring to improve system efficiency, including memory and disk efficiency, such as by reducing redundancy or copying of files to several system locations, and/or to reduce load time and simplify the update process by providing a single source to update, would seek to do so by utilizing shared libraries in user space.

Each of the references listed above discloses that providing a shared library in user space was an obvious way to improve efficiency, reduce redundancy or copying, reduce load time, and/or simplify the update process to a single source. *See e.g.*, D'Souza at 4:30-43 ("A DLL (dynamic-link library) is a feature of the MICROSOFT WINDOWS family of operating systems and the OS/2 operating system that allows executable routines, generally servicing a specific function or set of functions—to be stored separately as files with DLL extensions and to be loaded only when needed by the program that calls it."); Deianov at 10:7-17 ("User level threads do not utilize operating system resources, or provide bona fide threads of execution. Instead, user level threads simulate multiple threads of execution with a set of localized library functions."); Eggert at 237 ("By manipulating an environment variable [LD_PRELOAD], a user can tell the dynamic linker to preload IFS1 before linking the standard C library"); Ely Paper at 1 ("Moving the networking stack into a user-level library for development gains the following advantages over developing protocols in the kernel[.]"); Gingell at 112 ("The design and implementation of a shared libraries facility for Sun's implementation of the UNIX operating system (SunOS) is described. Shared libraries extend the resource utilization benefits obtained from sharing code between processes running the same program to processes running different programs by sharing the libraries common to them."); Levine at 39-40 ("When a program is in use in several different address spaces, the operating system can usually load the program at the same place in each of the address spaces in which it appears."); Maeda at 1 ("More importantly, our user-level protocol

libraries achieve performance (both throughput and latency) that is comparable to, and in some cases better than, well-tuned kernel-based implementations."); Thekkath at 67 ("In our design, protocol functionality is provided to an application by three interacting components—a protocol library that is linked into the application, a registry server that runs as a privileged process, and a network I/O module that is co-located with the network device driver."); User-mode Linux; FreeBSD; Thinstall; Virtuozzo; Ely Presentation at 5 ("Alpine runs a FreeBSD 3.3 stack in a user-space library[.]"); Trusted Linux; HPE ServiceGuard; Berger '466 at Abstract ("A computer-readable medium is also disclosed that comprises a library of software functions for managing compartment(s) implemented by an operating system. Such library includes at least one command-line utility executable to manipulate the compartment(s)."); Berger '436 at [0040] ("In step 156, supporting resources, such as libraries, configuration files, and/or the like, desirable to install application 120 in the selected compartment are automatically determined."); Choo at 2:7-15 ("In one embodiment, the present invention is related to a computer system including compartments implemented on an operating system. A database contains access rules with said access rules defining which compartments are authorized to access particular file resources. A kernel module receives a system call to access a file from a user space application belonging to a compartment."); HP C++ Programmer's Guide at 3-5, 3-8, 3-44 (discussing linking, including automatically, of shared libraries); HP C Compiler at 195 ("C Library Functions"), 209 ("dynamic" linking of libraries), 237 ("Compiling in Networked Environments"), 272 ("Shared libraries startup"), 273 ("Parallel runtime library") Hunt at [0060] ("When an application is built, it links against a set of dynamic linked libraries or (DLLs). The DLLs contain code that corresponds to the particular calls that an application makes."); Benfield at 6:1-10 ("Once an executable is spawned, daemon 226 has no further interaction with it. Each executable is linked with application

runtime library 228, which handles all further communication with the gateway."); Fowler at Abstract ("A backup file system which can be used without modification of application programs, the operating system, or the hardware. The backup file system is implemented by means of a dynamically-linkable replacement library and user-level processes on a primary computer system and a backup computer system. The dynamically-linkable replacement library has the same interfaces as a standard dynamically linkable library of file operations."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. At 83:22-88:8 (discussing Solaris Zones use of shared libraries.).

As these exemplary teachings demonstrate, improving existing computing systems and applications that are ready for improvement by implementing and utilizing shared libraries, as opposed to static or archive libraries, for example, was known in the art and would yield predictable results. Utilizing or linking to shared libraries rather than or in addition to archive or static libraries would simply be a combination of prior art elements according to known methods to yield predictable results. Moreover, as the use of both shared and static libraries is well-known and routine and conventional as of the date of the alleged inventions, utilizing a shared library over a static library is a simple substitution of one known element for another to obtain predictable results—shared resources as opposed to independent or separate ones. It also would be obvious to try because there are generally speaking only two primary choices for linking application libraries, static linking of archive libraries or dynamic linking of shared libraries, and a person of skill in the art would have been well aware of each of these options and their predictable results. In addition, a person of ordinary skill would have been motivated to combine or adapt known or familiar methods in the art, especially where market forces prompt such variations, such as a trend to more efficiently utilize memory and disk space and resources. The teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or

implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends in the field, and knowledge that the art is of special interest or importance in the field.

A person of ordinary skill in the art would have had a reasonable expectation of success in implementing shared libraries in user space because, for example, there are ample resources available to the person of skill in the art that disclose at length the usage and implementation choices and options for both static and shared libraries. *See* HP C++ Programmers Guide; HP C Compiler Reference Manual.

(ii) User Mode Shared Library Elements Are Functional Replicas of Kernel Elements References

The Asserted Claims of the ’058 Patent recite “wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs” and “wherein OSCSEs corresponding to and capable of performing the same function as SLCSEs remain in the operating system kernel” and “wherein SLCSEs are not copies of OSCSEs.” Dependent claim 4 of the ’058 Patent also recites: “wherein the one or more SLCSEs provided to one of the plurality of software applications having exclusive use thereof, use system calls to access services in the operating system kernel.” These elements do not add anything non-obvious over the prior art. This was well-known in the art before the alleged inventions of the ’058 Patent. To the extent these elements are not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- D'Souza (B-1)

- Deianov (B-2)
- Eggert (B-3)
- Ely Paper (B-4)
- Gingell (B-5)
- Levine (B-6)
- Maeda (B-7)
- Thekkath (B-8)
- User-mode Linux (B-9)
- FreeBSD (B-10)
- Thinstall (B-11)
- Virtuozzo (B-12)
- Ely Presentation (B-13)
- Trusted Linux (B-14)
- HPE ServiceGuard (B-15)
- Berger ’466 (B-16)
- Berger ’436 (B-17)
- Choo (B-18)
- HP C++ Programmer's Guide (B-19)
- HP C Compiler (B-20)
- Hunt (B-21)
- Fowler (B-22)
- Benfield (B-23)
- Solaris Zones (B-24)

A person of skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to have a controlled, secure, and safe way for an application to request services from the kernel would seek to do so by implementing a user space shared library that contained elements that are functional replicas of kernel elements that perform the same or similar functions as elements within the user space shared library, and that utilize system calls to access services or functions of the operating system kernel.

Each of the references listed above, as demonstrated in the corresponding appendices, discloses that using user space shared libraries that contain elements that are functional replicas of kernel elements that perform the same or similar functions, and that utilize system calls to kernel services, was an obvious way to request services from the kernel in a controlled, secure, and safe way. *See e.g.*, D'Souza at 5:8-17 ("The present invention contemplates that when an application is deployed that utilizes a local version of a shared component rather than a global version of the same component, the executable application code and the isolated component(s) be installed into a same directory."); Deianov at 1:64-2:2 ("In order to intercept a system call, a pointer in an interrupt vector table to a system call is replaced with a pointer to alternative object code to be executed instead of the system call. Then, when the system call is made, the alternative object code will execute instead. The alternative object code is known as a system call wrapper."); Eggert at 231-33; Ely Paper at 3 ("At the application interface to the networking stack, we exported the same networking API as the kernel. This involved writing a pseudo-system call layer that replaced the traditional networking system calls with our own. We also provide some support code that emulates kernel functions upon which the networking stack relies. These mostly include synchronization functions and functions manipulating kernel data structures."); Gingell at

114 (discussing statically and dynamically libraries linked to an application); Levine at 187 ("Program libraries date back to the earliest days of computing, since programmers quickly realized that they could save a lot of time and effort by reusing chunks of program code by reusing blocks of program code."); Maeda at 12 ("Our system … facilitates the integration of protocols into a complete operating system environment."); Thekkath at 67 ("In our design, protocol functionality is provided to an application by three interacting components--a protocol library that is linked into the application, ... The library contains the code that implements the communication protocol."); User-mode Linux ("UML is a fully-functional Linux kernel. It runs its own scheduler and VM system, relying on the host kernel only for the equivalent of hardware support"); FreeBSD; Thinstall; Virtuozzo; Ely Presentation at 3 ("Alpine moves the networking stack into user-space for development [and] changes are easily moved back to the kernel[.] Works for any transport protocol[.]"); Trusted Linux; HPE ServiceGuard; Berger '466 at [0102] ("Specifically, numerous approaches can be utilized to prevent processes associated with a compartment from accessing system resources. For example, access control can be implemented at the user-level via several techniques."); Berger '436 at [0040] ("In step 156, supporting resources, such as libraries, configuration files, and/or the like, desirable to install application 120 in the selected compartment are automatically determined. The supporting resources may be determined, for example, by querying an executable file associated with application 120 itself."); Choo at 5:4-12 ("In embodiments of the present invention, access to files by processes is restricted by rules based on process compartments. Reference is now made to FIG. 4 that depicts an exemplary file system to which access is controlled by rules based on process compartments."); HP C++ Programmer's Guide at 3-5, 3-8, 3-44 (discussing linking, including automatically, of shared libraries); HP C Compiler at 195 ("C Library Functions"), 209 ("dynamic" linking of libraries), 237 ("Compiling

in Networked Environments"), 272 ("Shared libraries startup"), 273 ("Parallel runtime library"); Hunt at [0060] ("contain [functions] that correspond[] to the particular calls that an application makes."); Benfield at 8:13-22 ("The representation of resources within DKS allows for the dynamic management and use of those resources by applications. DKS does not impose any particular representation, but it does provide an object-oriented structure for applications to model resources. The use of object technology allows models to present a unified appearance to management applications and hide the differences among the underlying physical or logical resources. Logical and physical resources can be modeled as separate objects and related to each other using relationship attributes."); Fowler at 6:37-50 ("The dynamically-linked libraries and user-level namespace just described may be employed to make a user-level backup file system which automatically backs up only selected ones of the files modified by an application process running an application program. FIG. 5 shows such a user-level backup file system 501."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 83:22-88:8 (discussing Solaris Zones use of shared libraries and elements therein).

As these exemplary teachings demonstrate, improving existing computing systems by implementing a user space library with functional replica elements of kernel elements and system calls to kernel services was known in the art and would yield predictable results. Such an improvement would be a combination of prior art references according to known methods to yield predictable results, as this use of user mode libraries is a standard, basic feature of operating systems. A person of ordinary skill in the art would have been motivated to use such shared libraries because, for example, certain standard compiler and linker programs and commands automatically include such shared libraries when building an application, and such shared libraries elements are known to use system calls to the operating system kernel. *See* HP C++ Programmer's

Guide; HP C Compiler Reference Manual. Indeed, this express teaching, suggestion, or motivation in the prior art to automatically include such shared application libraries when building an application would have led a person of ordinary skill in the art to utilize such libraries to arrive at the claimed invention. Moreover, as all such systems are within the same field of endeavor, the combination would have been obvious to a POSITA because it is simply the use of a known technique to improve similar devices or systems in the same way.

A person of ordinary skill in the art would have had a reasonable expectation of success in utilizing shared libraries that contain functional replicas of kernel elements and system calls to kernel resources because doing so was a standard and well-documented practice and instructions were readily available. HP C++ Programmer's Guide; HP C Compiler Reference Manual. Further, as described above and in the corresponding Appendices, each of the references listed above describes techniques that were known to offer such improvements, and, accordingly a person of ordinary skill would have been motivated to combine the references. The teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends in the field, and knowledge that the art is of special interest or importance in the field.

(iii) Multiple Instances of Shared Library Elements Running Simultaneously References

The Asserted Claims of the ’058 Patent recite “where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing the same function,” “wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications

for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously," and "multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system." These elements do not add anything non-obvious over the prior art. This was well-known in the art before the alleged inventions of the '058 Patent. To the extent these elements are not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- D'Souza (B-1)
- Deianov (B-2)
- Eggert (B-3)
- Ely Paper (B-4)
- Gingell (B-5)
- Levine (B-6)
- Maeda (B-7)
- Thekkath (B-8)
- User-mode Linux (B-9)
- FreeBSD (B-10)
- Thinstall (B-11)
- Virtuozzo (B-12)
- Ely Presentation (B-13)
- Trusted Linux (B-14)

- HPE ServiceGuard (B-15)
- Berger ’466 (B-16)
- Berger ’436 (B-17)
- Choo (B-18)
- HP C++ Programmer's Guide (B-19)
- HP C Compiler (B-20)\
- Hunt (B-21)
- Fowler (B-22)
- Benfield (B-23)
- Solaris Zones (B-24)

A person of skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to implement an application containment mechanism, for example, including file system mandatory access control measures such as implementing a chroot environment or jail, to achieve the benefits and improvements of containment as discussed herein, would have sought to do so by having multiple instances of shared library elements running simultaneously, such as running for different containerized applications at the same time within their respective chroot jails rather than in a shared memory segment. A person of ordinary skill in the art would have understood that running an application in a chrooted environment prevents the application from accessing the standard /lib/ or /user/lib/ library repositories of shared libraries in the host operating system, which would motivate and lead a person of ordinary skill in the art to copy any such shared library dependencies into the chrooted environment. *See* Trusted Linux (“A skeleton start and stop script should be included that will set

the default routes for you and any additional default files/commands that are needed. This could be accomplished by creating a directory (/etc/secskel) that contains all the scripts, commands, and files that **need to be copied** into the init directories," "When creating the chrooted environment, a skeleton start and stop script should be included that will set the default routes for you and any additional default files/commands that are needed. This could be accomplished by creating directory (/etc/secskel) that contains all the scripts, commands, and files that need to be copied into the chrooted environment."); *see also* Jensen, *Protection Wrappers A Simple and Portable Sandbox for Untrusted Applications* (1998), available at https://dl.acm.org/doi/pdf/10.1145/319195.319211 ("everything the program needs to run (configuration files, dynamic libraries, …) has to be copied into the sandbox before the program is run").

Each of the references listed above, as demonstrated in the corresponding Appendices, discloses that running multiple instances of shared library elements simultaneously within respective separate application containers was an obvious way to improve containment of applications in the system. *See e.g.,* D'Souza at 2:49-54 ("Side-by-side sharing allows multiple versions of the same component to run at the same time in different processes. Applications can then use a specific version of a component for which they were designed and tested, even if another application requires a different version of the same component."); Deianov at 7:50-58 ("The loading of selected processes 107 and non-selected processes with two different loader programs is possible because multitasking operating systems such as UNIX allow the use of multiple loader programs. The decision as to which processes 107 are to be loaded with the modified loader program 107 can be made by a system administrator, or by a user."); Eggert at 229 ("The recent rise of dynamic linking provides a new way for user to develop their own file systems; by

interposing a layer of user code between a program and the system call interface, a user can alter or extend a file system's behavior."); Ely Paper at 7 ("Alpine can be used with unmodified application binaries by exploiting dynamic linking, which delays the binding of function calls until the application executes."); Gingell at 124; Levine at 211-12 ("Once the linker's own initializations are done, it finds the names of the libraries required by the program. The program's program header has a pointer to the dynamic segment that contains dynamic linking information for the program."); Maeda at 1 ("More importantly, our user-level protocol libraries achieve performance (both throughput and latency) that is comparable to, and in some cases better than, well-tuned kernel-based implementations."); Thekkath at 64 ("However, considerations of code maintenance, ease of debugging, customization, and the simultaneous existence of multiple protocols argue for separating the implementations into more manageable user-level libraries of protocols. This paper describes the design and implementation of transport protocols as user-level libraries."); User-mode Linux; FreeBSD; Thinstall; Virtuozzo; Ely Presentation at 6-8; Trusted Linux; HPE ServiceGuard; Berger '466 at [0041] ("Instead, filesystem protection is achieved by associating a different section of the main filesystem with each compartment. Each such section of the file system is a "chroot" of the main filesystem, and processes running within any compartment only have access to the section of filesystem which is associated with that compartment."); Berger '436 at [0013] ("Compartment-based trusted operating system 100 comprises a plurality of compartments. Applications are installed and processes are run within these separate compartments."); Choo at 3:31-46 ("System 300 of FIG. 3 depicts another exemplary system that utilizes compartments to provide containment. System 300 includes a plurality of compartments."); HP C++ Programmer's Guide at 3-5, 3-8, 3-44 (discussing linking, including automatically, of shared libraries); HP C Compiler at 195 ("C Library Functions"), 209 ("dynamic"

linking of libraries), 237 ("Compiling in Networked Environments"), 272 ("Shared libraries startup"), 273 ("Parallel runtime library"); Hunt at [0068], [0070]; Benfield at 8:23-38 ("Using multiple objects has the burden of creating multiple object identifiers (OIDs) because each object instance has its own OID. However, a first order object can represent the entire resource and contain references to all of the constituent parts."); Fowler at 6:37-50 ("The dynamically-linked libraries and user-level namespace just described may be employed to make a user-level backup file system which automatically backs up only selected ones of the files modified by an application process running an application program. FIG. 5 shows such a user-level backup file system 501. System 501 is implemented in two computer systems: primary system 511, upon which application process 503 runs, and backup system 513, upon which backup copies of the files modified by application process 503 are maintained."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 83:22-88:8 (discussing Solaris Zones use of shared libraries and elements therein).

As these exemplary teachings demonstrate, improving existing application computing systems that are ready for improvement by implementing a containment mechanism was known in the art and would yield predictable results of having multiple instances of shared library elements executing simultaneously in separate containers. Such improvement would be a combination of prior art references according to known methods to yield predictable results. A person of ordinary skill in the art would have had a reasonable expectation of success in making such modifications or combinations. Known techniques of computer programming and networking, such as those set forth in the HP C++ Programmer's Guide and the HP C Compiler Reference Manual, within the knowledge of a person of ordinary skill in the art would suffice to make any such combination and where, as here, such combinations are modifications implemented through software programming, i.e., predictable and well-developed fields, the results of making

the combinations would be predictable. Moreover, as all such systems are within the same field of endeavor, the combination would have been obvious to a POSITA because it is simply the use of a known technique to improve similar devices or systems in the same way. Moreover, such a combination would be predictable because the combination is merely one of consolidating and/or interfacing disparate systems, and thus each system and its components continue to function in the combination as it did separately.

In addition, a person of ordinary skill would have been motivated to combine or adapt known or familiar methods in the art, especially where market forces prompt such variations. For example, to the extent that market forces demanded security, portability, availability, reliability, or improved ability to migrate applications, as discussed above, a POSITA would have thought to combine or modify references that described known techniques for doing so because such techniques offered improvements to existing systems. Further, as described above and in the corresponding Appendices, each of the references listed above describes techniques that were known to offer such improvements, and, accordingly a person of ordinary skill would have been motivated to combine the references. The teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends in the field, and knowledge that the art is of special interest or importance in the field.

(iv) Shared Library Element Run in Application Context and Form Part of Application References

The Asserted Claims of the ’058 Patent recite “when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of Software applications,” and “an instance of a SLCSE provided to at least a first of the

plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications." These elements do not add anything non-obvious over the prior art. This was well-known in the art before the alleged inventions of the '058 Patent. To the extent these elements are not explicitly or inherently disclosed in a charted reference, such elements are disclosed and/or taught by each of the prior art references identified in the non-exhaustive list below and the corresponding Appendices, such that the references in combination, in addition to the relevant teachings identified herein and/or charted, render the claim obvious.

- D'Souza (B-1)
- Deianov (B-2)
- Eggert (B-3)
- Ely Paper (B-4)
- Gingell (B-5)
- Levine (B-6)
- Maeda (B-7)
- Thekkath (B-8)
- User-mode Linux (B-9)
- FreeBSD (B-10)
- Thinstall (B-11)
- Virtuozzo (B-12)
- Ely Presentation (B-13)
- Trusted Linux (B-14)
- HPE ServiceGuard (B-15)
- Berger '466 (B-16)

- Berger ’436 (B-17)
- Choo (B-18)
- HP C++ Programmer's Guide (B-19)
- HP C Compiler (B-20)
- Hunt (B-21)
- Fowler (B-22)
- Benfield (B-23)
- Solaris Zones (B-24)

A person of skill in the art at the time of the alleged invention would have been motivated to combine the teachings of these references with the charted references for the reasons stated above. For example, a person of ordinary skill in the art desiring to use shared libraries for the reasons stated above and desiring to use containment for the reasons stated above, would have sought to do so by utilizing standard shared library linking command and functions within the containerized environment to efficiently implement the containerized environment using readily available structures, functionality, and commands.

Each of the references listed above, as demonstrated in the corresponding Appendices, discloses standard shared library execution associated with an application in a containerized environment which results in a shared library element running in the context of the application’s container. *See e.g.,* D'Souza at 4:44-46 (“The component object model (COM) is a software architecture that allows components made by different software vendors to be combined into a variety of applications.”); Deianov at 1:64-2:2 (“In order to intercept a system call, a pointer in an interrupt vector table to a system call is replaced with a pointer to alternative object code to be executed instead of the system call. Then, when the system call is made, the alternative object code

will execute instead. The alternative object code is known as a system call wrapper."); Eggert at 231 ("IFS0 extends the UNIX tree-structured pathnames by adding intensional pathnames, which are expressions evaluable with the shell. Intensional pathnames can evaluate to either file contents, or to names of other files"); Ely Paper at 3 ("Figure 2: In Alpine the unmodified networking code is placed in a user-level library. An application is not modified because it interfaces with the library using the traditional socket system API."); Gingell at 124 ("When the program is executed, crt0 will discover the __DYNAMIC structure ld left behind and map in the execution-time linker, ld.so. ld.so will map in the appropriate version of libc.so, allocate any uninitialized commons required by the program, and cause unresolved procedure references in both hello and libc.so to call ld.so."); Levine at 188 ("When the program is loaded, startup code finds those libraries and maps them into the program's address space before the program starts (Figure 9.1)."); Maeda at 2 ("Source-level compatibility with existing protocol clients. We are willing to recompile or relink existing protocol clients against our new implementation, but we (as we expect most others) are unwilling to modify these clients. Consequently, our operating system and protocol interface is syntactically and semantically compatible with existing interfaces."); Thekkath at 67 ("In our design, protocol functionality is provided to an application by three interacting components--a protocol library that is linked into the application"); User-mode Linux; FreeBSD; Thinstall; Virtuozzo; Ely Presentation at 8 ("Applications use Alpine's sockets by either linking with libAlpine.a before libc.a [or] LD_PRELOAD=libAlpine.so."); Trusted Linux; HPE ServiceGuard; Berger '466 at [0001] ("For instance, an application may invoke an OS routine (e.g., via a system call) to save a particular file, and the OS may interact with the basic input/output system (BIOS), dynamic link libraries, drivers, and/or other components of the computer system to properly save the particular file."); Berger '436 at [0040] ("In step 156, supporting resources, such as libraries,

configuration files, and/or the like, desirable to install application 120 in the selected compartment are automatically determined."); Choo at 3:31-46 ("System 300 of FIG. 3 depicts another exemplary system that utilizes compartments to provide containment. System 300 includes a plurality of compartments. In this example, WEB compartment 301, FTP compartment 302, and SYSTEM compartment 303 are shown. Each compartment is associated with various executing processes or threads."); HP C++ Programmer's Guide at 3-5, 3-8, 3-44 (discussing linking, including automatically, of shared libraries); HP C Compiler at 195 ("C Library Functions"), 209 ("dynamic" linking of libraries), 237 ("Compiling in Networked Environments"), 272 ("Shared libraries startup"), 273 ("Parallel runtime library"); ; Hunt at [0060] ("the operating system loads [a DLL] into the address space for the application."); Benfield at 8:13-22 ("The representation of resources within DKS allows for the dynamic management and use of those resources by applications. DKS does not impose any particular representation, but it does provide an object-oriented structure for applications to model resources. The use of object technology allows models to present a unified appearance to management applications and hide the differences among the underlying physical or logical resources. Logical and physical resources can be modeled as separate objects and related to each other using relationship attributes."); Fowler at 6:37-50 ("The dynamically-linked libraries and user-level namespace just described may be employed to make a user-level backup file system which automatically backs up only selected ones of the files modified by an application process running an application program. FIG. 5 shows such a user-level backup file system 501. System 501 is implemented in two computer systems: primary system 511, upon which application process 503 runs, and backup system 513, upon which backup copies of the files modified by application process 503 are maintained. Primary system 511 and backup system 513 are connected by a communications medium by means of which

messages from processes running on primary system 511 can be sent to processes running on backup system 513."); Solaris Zones, *see, e.g.*, Rochette Dep. Tr. at 83:22-88:8 (discussing Solaris Zones use of shared libraries and elements therein).

As these exemplary teachings disclose, utilizing standard shared library linking and execution functions to implement a containerized environment and system was known in the art and would yield predictable results. Such improvement would be a combination of prior art references according to known methods to yield predictable results. The shared library linking and execution commands operate the same in an outside of a containerized environment, which contributes to the person of ordinary skill in the art's reasonable expectation of success. Known techniques of library linking and loading, such as those set forth in the HP C++ Programmer's Guide and the HP C Compiler Reference Manual, within the knowledge of a person of ordinary skill in the art would suffice to make any such combination and where, as here, such combinations are modifications implemented through software programming, i.e., predictable and well-developed fields, the results of making the combinations would be predictable. Moreover, as all such systems are within the same field of endeavor, the combination would have been obvious to a POSITA because it is simply the use of a known technique to improve similar devices or systems in the same way. Moreover, such a combination would be predictable because each system and its components continue to function in the combination as it did separately.

In addition, a person of ordinary skill would have been motivated to combine or adapt known or familiar methods in the art, especially where market forces prompt such variations. For example, to the extent that market forces demanded security, portability, availability, reliability, or improved ability to migrate applications, as discussed above, a person of ordinary skill in the art would have thought to combine or modify references that described known techniques for doing

so because such techniques offered improvements to existing systems and existing functionality for linking and loading shared libraries can be utilized for containerized and non-containerized environments. Further, as described above and in the corresponding Appendices, each of the references listed above describes techniques that were known to offer such improvements, and, accordingly a person of ordinary skill would have been motivated to combine the references. The teaching, suggestion, or motivation to modify or combine the references in the manner claimed can be found in the explicit and/or implicit teachings for the references listed above, as identified in the corresponding Appendices, and the prior art as a whole, the general knowledge of those skilled in the art, including the knowledge of trends in the field, and knowledge that the art is of special interest or importance in the field.

C. Knowledge of One of Ordinary Skill in the Art

To the extent that VirtaMove contends that any particular feature or element of the Asserted Claims is a novel aspect of the claimed subject matter, IBM reserves the right to illustrate that the particular feature was widely known, understood, and implemented by those of ordinary skill in the art at the time of the alleged invention, and that it would have been obvious to combine and/or modify the prior art identified throughout these Invalidity Contentions with the knowledge of one of ordinary skill in the art. IBM reserves the right to illustrate the knowledge of one of ordinary skill in the art using any of the prior art references identified throughout these Invalidity Contentions, expert testimony, learned treatises, and/or testimony of prior art fact witnesses.

D. No Secondary Considerations

As of the date these Invalidity Contentions were served, VirtaMove has not identified any secondary considerations of non-obviousness. IBM reserves all rights to respond to any secondary considerations of non-obviousness raised by VirtaMove, including by updating, modifying, and/or adding to these Invalidity Contentions.

IBM contends that there are no secondary considerations of non-obviousness evidencing the validity of any of the Asserted Claims. Secondary considerations of non-obviousness, also referred to as objective indicia of non-obviousness, "can include copying, long felt but unsolved need, failure of others, commercial success, unexpected results created by the claimed invention, unexpected properties of the claimed invention, licenses showing industry respect for the invention, awards or other industry praise for the invention, and skepticism of skilled artisans before the invention." *Power Integrations, Inc. v. Fairchild Semiconductor Int'l, Inc.*, 711 F.3d 1348, 1368 (Fed. Cir. 2013). "A nexus between the merits of the claimed invention and evidence of secondary considerations is required in order for the evidence to be given substantial weight in an obviousness decision." *Ruiz v. A.B. Chance Co.*, 234 F.3d 654, 668 (Fed. Cir. 2000). Even when a nexus is found to exist, secondary considerations of non-obviousness "simply cannot overcome [a] strong prima facie showing of obviousness." *Sundance, Inc. v. DeMonte Fabricating Ltd.*, 550 F.3d 1356, 1368 (Fed. Cir. 2008).

VirtaMove has not identified or proven any objective indicia or secondary considerations of non-obviousness supporting the validity of any Asserted Claim. VirtaMove also has not established a nexus between any of the Asserted Claims and any alleged objective indicia or secondary considerations. IBM reserves the right to supplement these invalidity contentions to respond to any such evidence should VirtaMove be permitted to raise them in the future. Further, while discovery in this case is ongoing and IBM's investigation continues (including via fact and expert discovery), to the extent VirtaMove alleges that one or more Asserted Claims are not obvious based on secondary considerations, IBM contends such allegations are without merit.

VI. GROUNDS FOR INVALIDITY UNDER 35 U.S.C. § 112

IBM alleges that the Asserted Claims are invalid under 35 U.S.C. § 112 for being indefinite (Section 112, ¶ 2) and for failing to satisfy the written description and enablement requirements

(Section 112, ¶ 1). IBM reserves all rights to amend its Invalidity Contentions under 35 U.S.C. § 112, including after the Asserted Claims are ultimately construed by the Court, in response to any interpretation of the Asserted Claims embodied in VirtaMove's infringement positions, and/or to account for any changes in the law concerning invalidity under 35 U.S.C. § 112. IBM also reserves the right to provide additional explanation and/or argument for its Invalidity Contentions under Section 112, including, for example, based on expert testimony.

IBM incorporates by reference any other terms identified by Amazon.com, Inc., Amazon.com Services LLC, and Amazon Web Services, Inc. in their Invalidity Contentions in Case No. 7:24-cv-00030 (W.D. Tex.), Google LLC in its Invalidity Contentions in Case No. 7:24-cv-00033 (W.D. Tex.) or HPE in its Invalidity Contentions in Case No. 2:24-CV-00093 (E.D. Tex.).

These contentions shall not be construed as an admission that any claim construction advanced by IBM in this case is in any way inconsistent, flawed or erroneous. Nor should these contentions prevent IBM from advancing claim construction and/or non-infringement positions in lieu of, or in addition to, invalidity positions. Further, these contentions shall not be construed as an admission of or acquiescence to VirtaMove's purported construction of the claim language or of other positions advanced by VirtaMove during the course of this litigation.

A. USP 7,519,814

1. Indefiniteness

The Supreme Court has held that patent claims need to provide "reasonable certainty . . . about the scope of the invention" in order to avoid invalidity based on indefiniteness. *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2123 (2014). The Asserted Claims of the '814 Patent are invalid as indefinite under Section 112, ¶ 2 for failing to particularly point out and distinctly claim the subject matter which the Applicants regarded as the alleged invention

particularly as VirtaMove is asserting the claims for purposes of infringement. For example, as demonstrated either individually or collectively by the claim elements addressed below, the Asserted Claims fail to inform those skilled in the art about the scope of the invention with reasonable certainty, particularly as VirtaMove is asserting the claims for purposes of infringement, rendering the Asserted Claims (and any claims depending therefrom) invalid as indefinite:

Claim Terms	Asserted Claims
"operating systems that differ"	Claim 1
"disparate computing environments"	Claim 1
"a kernel a set of associated local system files compatible with the processor"	Claim 1
"secure … applications"	Claim 1
"applications related to a service"	Claim 1
"the applications each include an object executable by at least some of the different operating systems for performing a task related to the service"	Claim 1
"task related to the service"	Claim 1
"secure containers of application software"	Claim 1
"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, for use with a local kernel residing permanently on one of the servers"	Claim 1
"residing permanently"	Claim 1
"secure environment"	Claim 1
"wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel"	Claim 1
"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"	Claim 1
"copies or modified copies of the associated local system files that remain resident on the server"	Claim 1

"application software cannot be shared"	Claim 1
"an operating system's root file system"	Claim 1
"wherein the application software cannot be shared between the plurality of secure containers of application software"	Claim 1
"wherein each of the containers has a unique root file system that is different from an operating system's root file system"	Claim 1
"wherein each container has an execution file associated therewith for starting the one or more applications"	Claim 2
"unique associated identity"	Claim 6
"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"	Claim 9
"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"	Claim 10

The above limitations fail to set forth the scope of the alleged invention with reasonable certainty, particularly as VirtaMove is asserting the claims for purposes of infringement. Accordingly, all Asserted Claims are invalid as indefinite.

In addition, all Asserted Claims of the '814 Patent are invalid as indefinite because they mix statutory classes of subject matter and thus a person of ordinary skill in the art would not be able to ascertain when infringement occurs. *See IPXL Holdings, L.L.C. v. Amazon.com, Inc.*, 430 F.3d 1377, 1384 (Fed. Cir. 2005); *Rembrandt Data Techs., LP v. AOL, LLC*, 641 F.3d 1331, 1339 (Fed. Cir. 2011); *In re Katz Interactive Call Processing Patent Litigation*, 639 F.3d 1303, 1318 (Fed. Cir. 2011).

2. Lack of Written Description

The Asserted Claims of the '814 Patent are also invalid for failure to comply with the written description requirement under Section 112, ¶ 1. For example, the '814 Patent does not contain

written description support at least for the following claim terms, either individually or collectively, particularly as VirtaMove is asserting the claims for purposes of infringement, rendering the Asserted Claims in which they appear (and any claims depending therefrom) invalid for lack of written description:

Claim Terms	Asserted Claims
"operating systems that differ"	Claim 1
"disparate computing environments"	Claim 1
"a kernel a set of associated local system files compatible with the processor"	Claim 1
"secure … applications"	Claim 1
"applications related to a service"	Claim 1
"the applications each include an object executable by at least some of the different operating systems for performing a task related to the service"	Claim 1
"task related to the service"	Claim 1
"secure containers of application software"	Claim 1
"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, for use with a local kernel residing permanently on one of the servers"	Claim 1
"residing permanently"	Claim 1
"secure environment"	Claim 1
"wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel"	Claim 1
"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"	Claim 1
"copies or modified copies of the associated local system files that remain resident on the server"	Claim 1
"application software cannot be shared"	Claim 1
"an operating system's root file system"	Claim 1
"wherein the application software cannot be shared between the plurality of secure containers of application software"	Claim 1

"wherein each of the containers has a unique root file system that is different from an operating system's root file system"	Claim 1
"wherein each container has an execution file associated therewith for starting the one or more applications"	Claim 2
"unique associated identity"	Claim 6
"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"	Claim 9
"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"	Claim 10

At least under VirtaMove's apparent interpretation and application of the Asserted Claims, the '814 Patent does not provide sufficient written description support for the above claim terms. For example, at least under VirtaMove's apparent interpretation and application of the Asserted Claims, the specification does not adequately describe a "plurality of servers with operating systems that differ, operating in disparate computing environments." Moreover, the specification does not provide written description support for the full scope of the claim to the extent the scope of the claims is interpreted to cover the accused embodiments, which are not described in the specification.

These limitations are required by all Asserted Claims. Accordingly, all Asserted Claims are invalid for lack of written description. IBM also incorporates its contentions regarding Provisional Application No. 60/512,103 and Provisional Application No. 60/502,619 from Section III above.

3. Lack of Enablement

The Asserted Claims of the '814 Patent are also invalid for failure to comply with the enablement requirement under Section 112, ¶ 1. For example, the '814 Patent fails to provide an enabling disclosure with respect to the following claim terms, either individually or collectively,

particularly as VirtaMove is asserting the claims for purposes of infringement, rendering the Asserted Claims in which they appear (and any claims depending therefrom) invalid for lack of enablement:

Claim Terms	Asserted Claims
"operating systems that differ"	Claim 1
"disparate computing environments"	Claim 1
"a kernel a set of associated local system files compatible with the processor"	Claim 1
"secure … applications"	Claim 1
"applications related to a service"	Claim 1
"the applications each include an object executable by at least some of the different operating systems for performing a task related to the service"	Claim 1
"task related to the service"	Claim 1
"secure containers of application software"	Claim 1
"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, for use with a local kernel residing permanently on one of the servers"	Claim 1
"residing permanently"	Claim 1
"secure environment"	Claim 1
"wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel"	Claim 1
"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"	Claim 1
"copies or modified copies of the associated local system files that remain resident on the server"	Claim 1
"application software cannot be shared"	Claim 1
"an operating system's root file system"	Claim 1
"wherein the application software cannot be shared between the plurality of secure containers of application software"	Claim 1

"wherein each of the containers has a unique root file system that is different from an operating system's root file system"	Claim 1
"wherein each container has an execution file associated therewith for starting the one or more applications"	Claim 2
"unique associated identity"	Claim 6
"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"	Claim 9
"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"	Claim 10

At least under VirtaMove's apparent interpretation and application of the Asserted Claims, the '814 Patent does not enable the above claim terms. For example, at least under VirtaMove's apparent interpretation and application of the Asserted Claims, the specification does not adequately enable a person of ordinary skill in the art to "plurality of servers with operating systems that differ, operating in disparate computing environments." Moreover, the specification does not provide enablement for the full scope of the claim to the extent the scope of the claims is interpreted to cover the accused embodiments, which are not described or enabled in the specification. These limitations are required by all Asserted Claims. Accordingly, all Asserted Claims are invalid for lack of enablement.

B. USP 7,784,058

1. Indefiniteness

The Supreme Court has held that patent claims need to provide "reasonable certainty . . . about the scope of the invention" in order to avoid invalidity based on indefiniteness. *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2123 (2014). The Asserted Claims of the '058 Patent are invalid as indefinite under Section 112, ¶ 2 for failing to particularly point out and distinctly claim the subject matter which the Applicants regarded as the alleged invention,

particularly as VirtaMove is asserting the claims for purposes of infringement. For example, as demonstrated either individually or collectively by the claim elements addressed below, the Asserted Claims fail to inform those skilled in the art about the scope of the invention with reasonable certainty, particularly as VirtaMove is asserting the claims for purposes of infringement, rendering the Asserted Claims (and any claims depending therefrom) invalid as indefinite:

Claim Terms	Asserted Claims
"critical system elements"	Claim 1
"an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor"	Claim 1
"a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of Software applications in user mode"	Claim 1
"functional replicas of OSCSEs"	Claim 1
"wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs"	Claim 1
"wherein some of the SLCSEs stored in the shared library…are accessible to some of the plurality of software applications"	Claim 1
"forms a part of the one or more of the plurality of software applications"	Claim 1
"when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications"	Claim 1
"wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications"	Claim 1
"without being shared with other of the plurality of software applications"	Claim 1
"where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function"	Claim 1
"a predetermined function"	Claim 1

"wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE"	Claim 1
"wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously"	Claim 1
"wherein in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system"	Claim 2
"wherein OSCSEs corresponding to and capable of performing the same function as SLCSEs remain in the operating system kernel"	Claim 3
"wherein the one or more SLCSEs provided to one of the plurality of software applications having exclusive use thereof, use system calls to access services in the operating system kernel"	Claim 4
"wherein SLCSEs are not copies or modified copies of OSCSEs"	Claim 18

The above limitations fail to set forth the scope of the alleged invention with reasonable certainty, particularly as VirtaMove is asserting the claims for purposes of infringement. The scope of these terms is not reasonably certain. These limitations are required by all Asserted Claims. Accordingly, all Asserted Claims are invalid as indefinite.

In addition, all Asserted Claims of the '058 Patent are invalid as indefinite because they mix statutory classes of subject matter and thus a person of ordinary skill in the art would not be able to ascertain when infringement occurs. *See IPXL Holdings, L.L.C. v. Amazon.com, Inc.*, 430 F.3d 1377, 1384 (Fed. Cir. 2005); *Rembrandt Data Techs., LP v. AOL, LLC*, 641 F.3d 1331, 1339 (Fed. Cir. 2011); *In re Katz Interactive Call Processing Patent Litigation*, 639 F.3d 1303, 1318 (Fed. Cir. 2011).

2. Lack of Written Description

The Asserted Claims of the '058 Patent are also invalid for failure to comply with the written description requirement under Section 112, ¶ 1. For example, the '058 Patent does not contain written description support at least for the following claim terms, either individually or

collectively, particularly as VirtaMove is asserting the claims for purposes of infringement, rendering the Asserted Claims in which they appear (and any claims depending therefrom) invalid for lack of written description:

Claim Terms	Asserted Claims
"critical system elements"	Claim 1
"an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor"	Claim 1
"a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of Software applications in user mode"	Claim 1
"functional replicas of OSCSEs"	Claim 1
"wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs"	Claim 1
"wherein some of the SLCSEs stored in the shared library…are accessible to some of the plurality of software applications"	Claim 1
"forms a part of the one or more of the plurality of software applications"	Claim 1
"when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications"	Claim 1
"wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications"	Claim 1
"without being shared with other of the plurality of software applications"	Claim 1
"where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function"	Claim 1
"a predetermined function"	Claim 1
"wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE"	Claim 1
"wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously"	Claim 1

"wherein in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system"	Claim 2
"wherein OSCSEs corresponding to and capable of performing the same function as SLCSEs remain in the operating system kernel"	Claim 3
"wherein the one or more SLCSEs provided to one of the plurality of software applications having exclusive use thereof, use system calls to access services in the operating system kernel"	Claim 4
"wherein SLCSEs are not copies or modified copies of OSCSEs"	Claim 18

At least under VirtaMove's apparent interpretation and application of the Asserted Claims, the '058 Patent does not provide sufficient written description support for the above claim terms. For example, at least under VirtaMove's apparent interpretation and application of the Asserted Claims, the specification does not adequately describe the phrase "when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications." Moreover, the specification does not provide written description support for the full scope of the claim to the extent the scope of the claims is interpreted to cover the accused embodiments, which are not described in the specification.

These limitations are required by all Asserted Claims. Accordingly, all Asserted Claims are invalid for lack of written description. IBM also incorporates its contentions regarding Provisional Application No. 60/504,213 from Section III above.

3. Lack of Enablement

The Asserted Claims of the '058 Patent are also invalid for failure to comply with the enablement requirement under Section 112, ¶ 1. For example, the '058 Patent fails to provide an enabling disclosure with respect to the following claim terms, either individually or collectively, particularly as VirtaMove is asserting the claims for purposes of infringement, rendering the Asserted Claims in which they appear (and any claims depending therefrom) invalid for lack of enablement:

Claim Terms	Asserted Claims
"critical system elements"	Claim 1
"an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor"	Claim 1
"a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of Software applications in user mode"	Claim 1
"functional replicas of OSCSEs"	Claim 1
"wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs"	Claim 1
"wherein some of the SLCSEs stored in the shared library…are accessible to some of the plurality of software applications"	Claim 1
"forms a part of the one or more of the plurality of software applications"	Claim 1
"when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications"	Claim 1
"wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications"	Claim 1
"without being shared with other of the plurality of software applications"	Claim 1
"where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function"	Claim 1
"a predetermined function"	Claim 1
"wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE"	Claim 1
"wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously"	Claim 1
"wherein in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system"	Claim 2
"wherein OSCSEs corresponding to and capable of performing the same function as SLCSEs remain in the operating system kernel"	Claim 3

"wherein the one or more SLCSEs provided to one of the plurality of software applications having exclusive use thereof, use system calls to access services in the operating system kernel"	Claim 4
"wherein SLCSEs are not copies or modified copies of OSCSEs"	Claim 18

At least under VirtaMove's apparent interpretation and application of the Asserted Claims, the '058 Patent does not enable the above claim terms. For example, at least under VirtaMove's apparent interpretation and application of the Asserted Claims, the specification does not adequately enable a person of ordinary skill in the art to "when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications." Moreover, the specification does not provide enablement for the full scope of the claim to the extent the scope of the claims is interpreted to cover the accused embodiments, which are not described or enabled in the specification. These limitations are required by all Asserted Claims. Accordingly, all Asserted Claims are invalid for lack of enablement.

VII. ADDITIONAL INVALIDITY CONTENTIONS

Precise identification of all of the bases upon which the Asserted Claims are invalid and/or unenforceable, including inequitable conduct, improper inventorship, and the judicially-created non-statutory obviousness-type double patenting doctrine, are likely to be revealed only after further developments in the case, including fact and expert discovery. IBM expressly reserves the right to amend or supplement these Invalidity Contentions to address any additional invalidity arguments that become apparent in view of any relevant facts and information revealed during the course of discovery.

Dated: October 30, 2024 By: */s/ Kyle Calhoun*

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on October 30, 2024, all counsel of record who are deemed to have consented to electronic service are being served with a copy of this document via electronic mail.

/s/ Kyle Calhoun