Paper No. __

UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

GOOGLE LLC,
Petitioner,

v.

VIRTAMOVE, CORP.,
Patent Owner.

_____

Case No. IPR2025-00487
Patent No. 7,519,814

_____

PETITION FOR INTER PARTES REVIEW
UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.1 et seq

TABLE OF CONTENTS

MANDATORY NOTICES xvi

A. Real Party-In-Interest xvi

B. Related Matters xvi

1. United States Patent & Trademark Office xvi

2. USPTO Patent Trial and Appeal Board xvii

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

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

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

C. Counsel and Service Information - § 42.8(b)(3) and (4) xviii

I. STANDING 1

II. GROUNDS 1

III. THE '814 PATENT 1

A. Specification 1

B. Person of Ordinary Skill in the Art ("POSA") 3

C. Prosecution History 4

D. Challenged Claims 4

IV. CLAIM INTERPRETATION 4

V. GROUND 1: Claims 1-4, 7-11, 14, AND 16-30 ARE UNPATENTABLE OVER Blaser-Calder 5

A. Blaser (EX1005) 5

B. Calder (EX1006) 7

C. Blaser-Calder Combination 9

D. Mapping to Challenged Claims 14

1. Claim 1 14

a. [1PREA] 14

i. "1. In a system having a plurality of servers with operating systems that differ" 14

ii. "operating in disparate computing environments" 15

iii. "wherein each server includes a processor and an operating system" 16

iv. "[OS] including a kernel [and] a set of associated local system files compatible with the processor" 17

b. [1PREB] 20

i. "a method of providing at least some of the servers in the system with secure, executable, applications" 20

ii. "[applications] related to a service" 21

iii. "wherein the applications are executed in a secure environment" 22

iv. "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" 23

c. [1A] 25

i. "storing in memory accessible to at least some of the servers a plurality of secure containers of application software" 25

(1) Blaser-Calder's layers (containers) are "stor[ed] in memory accessible to…the servers" 25

(2) Blaser-Calder's layers are "containers" 27

(3) Blaser-Calder's layers are "secure containers of application software" 29

ii. "each container comprising one or more of the executable applications" 32

iii. "and a set of associated system files required to execute the one or more applications" 34

iv. "for use with a local kernel residing permanently on one of the servers" 35

d. [1B] "wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems," 36

e. [1C] "the containers of application software excluding a kernel," 37

f. [1D] "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," 39

g. [1E] "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,"....40

h. [1F] "and wherein the application software cannot be shared between the plurality of secure containers of application software," 41

i. [1G] "and wherein each of the containers has a unique root file system that is different from an operating system's root file system."....43

2. Claim 2: "[C]laim 1, wherein each container has an execution file associated therewith for starting the one or more applications."....46

3. Claim 3: "[C]laim 2['s]…execution file includes instructions related to an order in which executable applications within will be executed." 47

4. Claim 4: "[C]laim 1…pre-identifying applications and system files required for association with the one or more containers prior to said storing step" [1A]....48

5. Claim 7: "[C]laim 2…modifying at least some of the system files to define container specific mount points associated with the container." 49

6. Claim 8: "[C]laim 1['s]…applications and associated system files are retrieved from a computer system having a plurality of secure containers."....50

7. Claim 9: "[C]laim 2, 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." 51

8. Claim 10: "[C]laim 2, 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." 52

9. Claim 11: "[C]laim 2['s]…containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location." 54

10. Claim 14 54

    a. [14PRE] "[C]laim 1…creating containers prior to said step of storing containers in memory" 54

    b. [14A] "wherein containers are created by: a) running an instance of a service on a server;" 55

    c. [14B] "b) determining which files are being used; and," 56

    d. [14C] "c) copying applications and associated system files to memory without overwriting the associated system files so as to provide a second instance of the applications and associated system files." 56

11. Claim 16 57

    a. [16A] "[C]laim 1…creating containers prior to said step of storing containers in memory," 57

    b. [16B] "wherein a step of creating containers includes using a skeleton set of system files as a container starting point and installing applications into that set of files." 57

12. Claim 17 58

    a. [17A] "[C]laim 1…installing a service on a target server selected from one of the plurality of servers," 58

    b. [17B]: "…using a graphical user interface [GUI], associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server." 59

13. Claim 18: Claim 17's "target server and" GUI "are at remote locations." 60

14. Claim 19: Claim 18's GUI "is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server." 60
15. Claim 20: "Claim 19," where [17B]'s "associating" step "includes…relatively moving the unique icon representing the service to the unique icon representing a server." 61
16. Claim 21: "[C]laim 20…starting a distributed software application." 61
17. Claim 22: "[C]laims 20…updating a console on the selected target server with information indicating that the service is resident on the selected target server."....62
18. Claim 23: "[C]laim 17…testing to determine if the selected target server is a valid computing platform, prior to causing the applications to be distributed to, and installed on the target server."....62
19. Claim 24: "[C]laim 17…creating a user account for the service."....62
20. Claim 25: "[C]laim 17…installing files specific to the selected application on the selected server."....63
21. Claim 26: "[C]laim 17…setting file access permissions to allow a user to access the one of the applications to be distributed."....63
22. Claim 27 63
    a. [27A] "[C]laim 1…de-installing a service from a server, comprising:"....63
    b. [27B] "displaying the icon representing the service; [and] displaying the icon representing the server on which the service is installed; and" 64
    c. [27C] "utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected service from the server on which it was installed." 64
23. Claim 28: "[C]laim 27…separating icon representing the service from the icon representing the server."....64
24. Claim 29: "[C]laim 27…testing whether the selected server is a valid computing platform for de-installation of the service."....65

25. Claim 30: "[C]laim 27…copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation." 65

VI. GROUND 2: Claims 5-6, 12-13, 15, AND 31-34 Are Unpatatentbale OVER Blaser-Calder-Schmidt.....66

A. Schmidt-449 (EX1007) 66

B. Blaser-Calder-Schmidt 67

C. Mapping to Challenged Claims 68

1. Claim 5: "[C]laim 2…modifying at least some of the associated system files in plural containers to provide an association with a container specific identity assigned to a particular container." 68

2. Claim 6: "[C]laim 2…assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address." 69

3. Claim 12 75

4. Claim 13 75

5. Claim 15 71

a. [15A] "[C]laim 14…assigning an identity to the containers including at least one of a unique IP address, a unique Mac address and an estimated resource allocation;" 71

b. [15B] "installing the container on a server; and," 71

c. [15C] "testing the applications and files within the container." 71

6. Claim 31 72

a. Previously Addressed Limitations 72

b. [31I] "a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications." 73

7. Claim 32 78

8. Claim 33: "[C]laim 32['s]…run time module includes an intercepting module associated with the…containers for intercepting system calls from any of the…containers and for

providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of" [31C]'s unique identifiers....74

9. Claim 34 75

a. [34A]: "[C]laim 31['s]…run time module performs: monitoring resource usage of applications executing;"....75

b. [34B]: "intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;" 75

c. [34C]: "comparing the monitored resource usage…with the resource limits; and, forwarding the system calls to a kernel on the basis of the comparison…."....76

VII. DISCRETIONARY DENIAL IS UNWARRANTED 76

A. §314(a)....76

1. Stay Potential....77

2. Trial Timing 77

3. Litigation Investment 77

4. Issue Overlap....77

5. Litigation Defendants....78

6. Other Circumstances 78

B. §325(d)....78

VIII. CONCLUSION....78

IX. APPENDIX: CLAIM LISTING....80

TABLE OF AUTHORITIES

CASES

Apple Inc. v. Fintiv, Inc.,
IPR2020-00019, Paper 11 (Mar. 20, 2020) 76

BMW of North America, LLC v. Michigan Motor Techs., LLC,
IPR2023-01224, Paper 15 (Feb. 15, 2024) 77

Google LLC v. Security First Innovations, LLC,
IPR2024-00215, Paper 15 (May 23, 2024) 5

Markforged Inc. v. Continuous Composites Inc.,
IPR2022-00679, Paper 7 (Oct. 25, 2022) 77, 78

PLR Worldwide Sales Ltd. v. Flip Phone Games, Inc.,
IPR2024-00209, Paper 9 (May 10, 2024) 15, 44

Protect Animals With Satellites LLC v. OnPoint Sys., LLC,
IPR2021-01483, Paper 11 (Mar. 4, 2022) 77

Sand Revolution II, LLC v. Cont'l Intermodal Group–Trucking,
IPR2019-01393, Paper 24 (June 16, 2020) 77

REGULATIONS

37 C.F.R. §42.100(b) 4

37 C.F.R. §42.104(a) 1

37 C.F.R. 42.104(b)(3) 5

STATUTES

35 U.S.C. §102(b) 1

35 U.S.C. §102(e) 1

35 U.S.C. §103 1

35 U.S.C. §282(b) 4

35 U.S.C. §314(a) 76

35 U.S.C. §325(d) 78

OTHER AUTHORITIES

Director's Interim Procedure for Discretionary Denials (June 21, 2022) 76, 78

APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 7,519,814 ("'814 patent")
1002	Prosecution History of U.S. Patent No. 7,519,814
1003	Declaration of Samrat Bhattacharjee, Ph.D. ("Bhattacharjee")
1004	Curriculum Vitae of Samrat Bhattacharjee, Ph.D.
1005	U.S. Patent No. 7,117,495 ("Blaser")
1006	U.S. Patent Application Publication No. 20020066022 ("Calder")
1007	U.S. Patent No. 6,931,449 ("Schmidt-449")
1008	U.S. Patent Application Publication No. 20020095479 ("Schmidt-479")
1009	U.S. Patent Application Publication No. 20020133529 ("Schmidt-529")
1010	U.S. Patent Application Publication No. 20020124072 ("Tormasov")
1011	U.S. Patent Application Publication No. 20010009425
1012	U.S. Patent Application Publication No. 20010018708
1013	U.S. Patent Application Publication No. 20010047472
1014	U.S. Patent Application Publication No. 20010056572
1015	U.S. Patent Application Publication No. 20010029605
1016	U.S. Patent Application Publication No. 20020023158
1017	U.S. Patent Application Publication No. 20020052727
1018	U.S. Patent Application Publication No. 20020143795
1019	U.S. Patent Application Publication No. 20020156612
1020	U.S. Patent Application Publication No. 20020156877
1021	U.S. Patent Application Publication No. 20020174215
1022	U.S. Patent Application Publication No. 20020188718
1023	U.S. Patent Application Publication No. 20020194394
1024	U.S. Patent Application Publication No. 20020194488
1025	U.S. Patent Application Publication No. 20030009408
1026	U.S. Patent Application Publication No. 20030014381
1027	U.S. Patent Application Publication No. 20030014466
1028	U.S. Patent Application Publication No. 20030018717
1029	U.S. Patent Application Publication No. 20030023839
1030	U.S. Patent Application Publication No. 20030041173
1031	U.S. Patent Application Publication No. 20030093688
1032	U.S. Patent Application Publication No. 20030097464
1033	U.S. Patent No. 6,467,052
1034	U.S. Patent Application Publication No. 20030140179
1035	U.S. Patent Application Publication No. 20030154221
1036	U.S. Patent No. 7,159,184

Exhibit	Description
1037	U.S. Patent No. 7,103,745
1038	U.S. Patent No. 6,263,440
1039	U.S. Patent Application Publication No. 20040190534
1040	U.S. Patent No. 7,356,771
1041	U.S. Patent No. 5,903,753
1042	U.S. Patent No. 4,685,125
1043	U.S. Patent No. 5,406,644
1044	U.S. Patent No. 5,421,009
1045	U.S. Patent No. 5,568,630
1046	U.S. Patent No. 5,640,562
1047	U.S. Patent No. 5,657,221
1048	U.S. Patent No. 5,675,831
1049	U.S. Patent No. 5,742,829
1050	U.S. Patent No. 4,742,450
1051	U.S. Patent No. 5,761,669
1052	U.S. Patent No. 5,784,555
1053	U.S. Patent No. 5,835,765
1054	U.S. Patent No. 6,044,465
1055	U.S. Patent No. 6,122,744
1056	U.S. Patent No. 6,195,650
1057	U.S. Patent No. 6,321,323
1058	U.S. Patent No. 6,363,409
1059	U.S. Patent No. 6,453,470
1060	U.S. Patent No. 6,567,767
1061	U.S. Patent No. 6,732,359
1062	U.S. Patent No. 6,874,148
1063	U.S. Patent No. 6,985,937
1064	U.S. Patent No. 7,140,015
1065	U.S. Patent No. 7,185,192
1066	U.S. Patent No. 7,454,440
1067	U.S. Patent No. 8,209,680
1068	U.S. Patent No. 5,926,636
1069	U.S. Patent No. 5,218,530
1070	U.S. Patent No. 6,134,593
1071	U.S. Patent No. 6,918,038
1072	U.S. Patent Application Publication No. 20040034623
1073	U.S. Patent No. 7,461,148
1074	U.S. Patent No. 7,136,800

Exhibit	Description
1075	PCT Application Publication No. WO2002056156
1076	Kamp et al., "Jails: Confining the omnipotent root," Proceedings of the 2nd International SANE Conference (Vol. 43, p. 116) (2000)
1077	Prevelakis et al., "Sandboxing Applications," USENIX Annual Technical Conference, FREENIX Track (pp. 119-126) (2001)
1078	Plaintiff VirtaMove Corp.'s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sept. 06, 2024)
1079	Chart re: '814 Patent accompanying Plaintiff VirtaMove Corp.'s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sept. 6, 2024)
1080	First Amended Complaint for Patent Infringement Against Google LLC in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (May 21, 2024)
1081	Scheduling Order in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (June 17, 2024) (ECF 34)
1082	Federal Court Management Statistics–Profiles, U.S. District Courts–Combined Civil and Criminal (September 2024)
1083	U.S. Patent Application Publication No. 20020095500 ("Schmidt-500")
1084	U.S. Patent Application Publication No. 20050169073
1085	U.S. Provisional Application No. 60/533,388
1086	U.S. Patent No. 5,412,808
1087	Order Granting Defendant Google LLC's Motion to Transfer Venue to the Northern District of California in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Jan. 22, 2025)
1088	Microsoft's Computer Dictionary (5th ed. 2002) (excerpt)
1089	U.S. Patent No. 6,381,742
1090	U.S. Patent Application Publication No. 20030035430
1091	U.S. Patent No. 6,477,624
1092	U.S. Patent No. 5,931,947
1093	U.S. Patent No. 6,148,335
1094	U.S. Patent No. 6,944,790
1095	U.S. Patent No. 6,014,135
1096	U.S. Patent No. 6,282,660
1097	U.S. Patent Application Publication No. 20040153709
1098	U.S. Patent Application Publication No. 20050050389
1099	U.S. Patent Application Publication No. 20040225952

Exhibit	Description
1100	Google LLC's Proposed Claim Terms for Construction in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1101	Plaintiff's Disclosure of Proposed Claim Constructions in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1102	Google LLC's Opening Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 24, 2024)
1103	Plaintiff's Responsive Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 12, 2024)
1104	Google LLC's Reply Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 26, 2024)
1105	Plaintiff's Sur-Reply Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 13, 2024)
1106	Joint Claim Construction Statement in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 18, 2024)
1107	U.S. Provisional Application No. 60/502,619
1108	U.S. Patent Application Publication No. 20050188268
1109	U.S. Patent Application Publication No. 20020140743
1110	U.S. Patent Application Publication No. 20020143906 ("Tormasov-906")
1111	U.S. Patent No. 7,076,633 ("Tormasov-633")
1112	U.S. Patent Application Publication No. 20060089950 ("Tormasov-950")
1113	U.S. Patent No. 7,222,132 ("Tormasov-132")
1114	U.S. Patent Application Publication No. 20020147815 ("Tormasov-815")
1115	U.S. Patent No. 7,209,973 ("Tormasov-973")
1116	U.S. Patent Application Publication No. 20020138629 ("Schmidt-629")
1117	U.S. Patent No. 6,944,860 ("Schmidt-860")
1118	U.S. Patent Application Publication No. 20020174265 ("Schmidt-265")
1119	U.S. Patent Application Publication No. 20020116659 ("Tormasov-659")
1120	Bach, Maurice. 1987. Design of the Unix Operating System by Marice J Bach, 1 edition (Feb. 27, 1987) Prentice Hall; ISBN: 0132017997.
1121	Crowley, Charles. 1997. Operating Systems: a design-oriented approach. Irwin. 1997. ISBN 0-256-15151-2.

Exhibit	Description
1122	Eckel, George and Chris Hare. 1995. Building a Linux Internet Server, Chris Hare", G. Eckel, New Riders Publishing; ISBN: 1562055259.
1123	RESERVED
1124	RESERVED
1125	RESERVED
1126	RESERVED
1127	U.S. Patent No. 6,854,009
1128	U.S. Patent No. 7,080,356
1129	U.S. Patent No. 5,870,539
1130	U.S. Patent No. 6,883,028
1131	U.S. Patent No. 7,209,959
1132	U.S. Patent Application Publication No. 20020065835
1133	The Wayback Machine archive of USENIX Association – Home Page (2001-11-14)
1134	The Wayback Machine archive of USENIX – Publications – Proceedings (2001-11-15)
1135	The Wayback Machine archive of USENIX – 2001 USENIX Annual Technical Conference (2001-11-11)
1136	The Wayback Machine archive of USENIX – 2001 USENIX Annual Technical Conference (2001-11-11)
1137	The Wayback Machine archive of 2001 FREENIX Track Technical Program – Abstract (2002-01-12)
1138	The Wayback Machine archive of "Sandboxing Applications" (2003-03-29)
1139	The Wayback Machine archive of "Sandboxing Applications" (2004-01-19)
1140	The Wayback Machine archive of PDF of "Sandboxing Applications" (2004-01-19)
1141	The Wayback Machine archive of NLUUG: UNIX User Group – The Netherlands (2001-07-22)
1142	The Wayback Machine archive of NLUUG: Previous Events (2001-08-03)
1143	The Wayback Machine archive of Second International SANE Conference (2001-08-12)
1144	The Wayback Machine archive of Second International SANE Conference (2001-06-20)
1145	The Wayback Machine archive of "Jails: Confining the omnipotent root." (2000-09-02)

Exhibit	Description
1146	The Wayback Machine archive of Linux Journal (1999-05-08)

MANDATORY NOTICES

A. Real Party-In-Interest

Petitioner Google LLC is the Real Party-in-Interest.[1]

B. Related Matters

1. United States Patent & Trademark Office

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

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

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

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

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

---

[1] Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

2. USPTO Patent Trial and Appeal Board

Concurrently with the present petition, Petitioner is filing IPR2025-00488, also challenging U.S. Patent No. 7,519,814.

Petitioner is also filing IPR2025-00489 and IPR2025-00490 challenging U.S. Patent No. 7,784,058, which is also asserted in *VirtaMove, Corp. v. Google LLC*, Case No. 7:24-cv-00033, listed below.

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

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

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

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

(i) *VirtaMove, Corp. v. Google LLC*, Case No. 7:24-cv-00033 (pending transfer to Northern District of California per Order dated January 22, 2025, *see* EX1087);

(ii) *VirtaMove, Corp. v. Amazon.com, Inc. et al*, Case No. 7:24-cv-00030 (pending transfer to Northern District of California per Order dated January 22, 2025, *see* Docket Entry No. 87);

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

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

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

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

C. Counsel and Service Information - § 42.8(b)(3) and (4)

Lead Counsel	Elisabeth H. Hunt, Reg. No. 67,336
Backup Counsel	Gregory S. Nieberg, Reg. No. 57,063 Anant K. Saraswat, Reg. No. 76,050
Service Information	E-mail: EHunt-PTAB@wolfgreenfield.com GNieberg-PTAB@wolfgreenfield.com ASaraswat-PTAB@wolfgreenfield.com Post and hand delivery: Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206 Telephone: 617-646-8000 Facsimile: 617-646-8646

A power of attorney is submitted with the Petition. Counsel for Petitioner consents to service of all documents via electronic mail.

Petitioner requests *inter partes* review and cancellation of claims 1-34 of the ’814 patent (EX1001).

I. STANDING

Petitioner certifies that the ’814 patent is available for *inter partes* review and that Petitioner is not barred or estopped from requesting *inter partes* review of the challenged claims. 37 C.F.R. §42.104(a).

II. GROUNDS

Ground Number and Reference(s)		Claims	Basis
1	Blaser-Calder	1-4, 7-11, 14, 16-30	§103
2	Blaser-Calder-Schmidt	5-6, 12-13, 15, 31-34	§103

Each reference above is prior art under (at least) pre-AIA §102(e) to the ’814 patent’s earliest alleged priority date; Calder is also prior art under §102(b).

These references were not of record during prosecution of the ’814 patent. EX1003 (“Bhattacharjee”), [0029]-[0031], [0001]-[0014].

III. THE ’814 PATENT

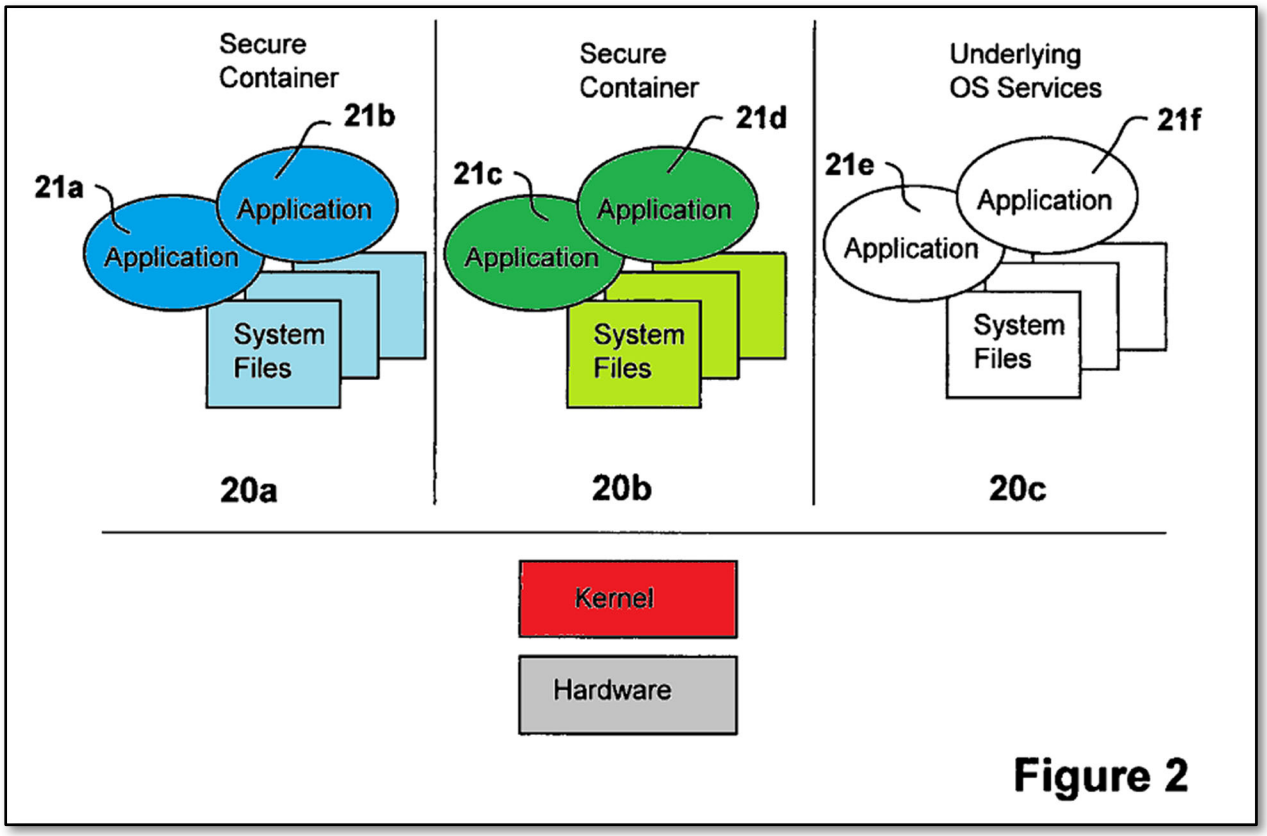
A. Specification

Computer Operating Systems (“OSs”) typically provide system files (like configuration files and libraries) used by applications. EX1001, 2:52-3:19. Configuration files identify, e.g., the computer’s IP address, available font(s), etc. Libraries provide pre-existing code for common functions (e.g., opening files). Problems can arise, however, when one application hogs shared computing

"resources" (e.g., the processor or "network port numbers") or updates a library shared by multiple applications in a way that makes it incompatible with another application. EX1001, 1:31-36. Bhattacharjee, [0032]-[0033].

The patent says one known solution to these issues was to "separate[]" applications by installing them on "individual computer system[s]" (EX1001, 1:27-30), but that can quickly become costly. An alternative prior-art solution is "Virtual Machine [VM] technology," where one computer can run multiple VMs, each simulating a different machine with its own hardware, OS, etc. EX1001, 1:51-61. The patent alleges, however, that VMs "impose[] significant performance overhead" from executing multiple full OSs. EX1001, 1:62-65. The patent says another known technique "separat[ed]…application[s] from the underlying operating system" but allegedly did "not isolate applications into distinct environments." EX1001, 2:4-12. Bhattacharjee, [0034].

Figure 2 (color added below) illustrates the patent's alleged invention: having "a single server" executing sets of "[a]pplications" (blue and green) that "are segregated" into respective "secure containers" 20a-20b, each container having respective system files (light blue and light green). EX1001, 7:4-15, 9:53-57.



Containers share the computer's hardware (grey) and OS kernel (red) (which controls access to the computer's resources). EX1001, FIG. 2, 1:56-61. Each container can also have its own unique identifier (e.g., IP address). EX1001, 3:55-57. Bhattacharjee, [0035].

B. Person of Ordinary Skill in the Art ("POSA")

A POSA as of the '814 patent's earliest claimed priority date would have had at least a bachelor's degree in computer science, computer engineering, or a related field, with three years of academic and/or industry experience in the area of "management and deployment of server applications." EX1001, 1:15-16. More education may substitute for less experience. Bhattacharjee, [0036]-[0038].

C. Prosecution History

The original application's claims were found anticipated by EX1089. EX1002, 238-271. The applicants responded by adding Limitations [**1E**]-[**1G**] and the word "remain" to Limitation [**1D**]. EX1002, 287-288, 291; *see infra* §IX (Claim Listing). The Examiner subsequently allowed the claims, stating that the prior art of record failed to disclose application software "not being shareable between" secure containers, and "unique root file systems different from an operating system's root file system." EX1002, 341-344. Bhattacharjee, [0039].

D. Challenged Claims

The '814 patent's two independent claims (1, 31) each recite a system having secure containers of applications comprising associated system files for executing applications. The containers do not include an OS kernel, but instead use a server's underlying OS's kernel.

The dependent claims recite additional, routine prior-art functionality. Bhattacharjee, [0040]-[0042].

IV. CLAIM INTERPRETATION

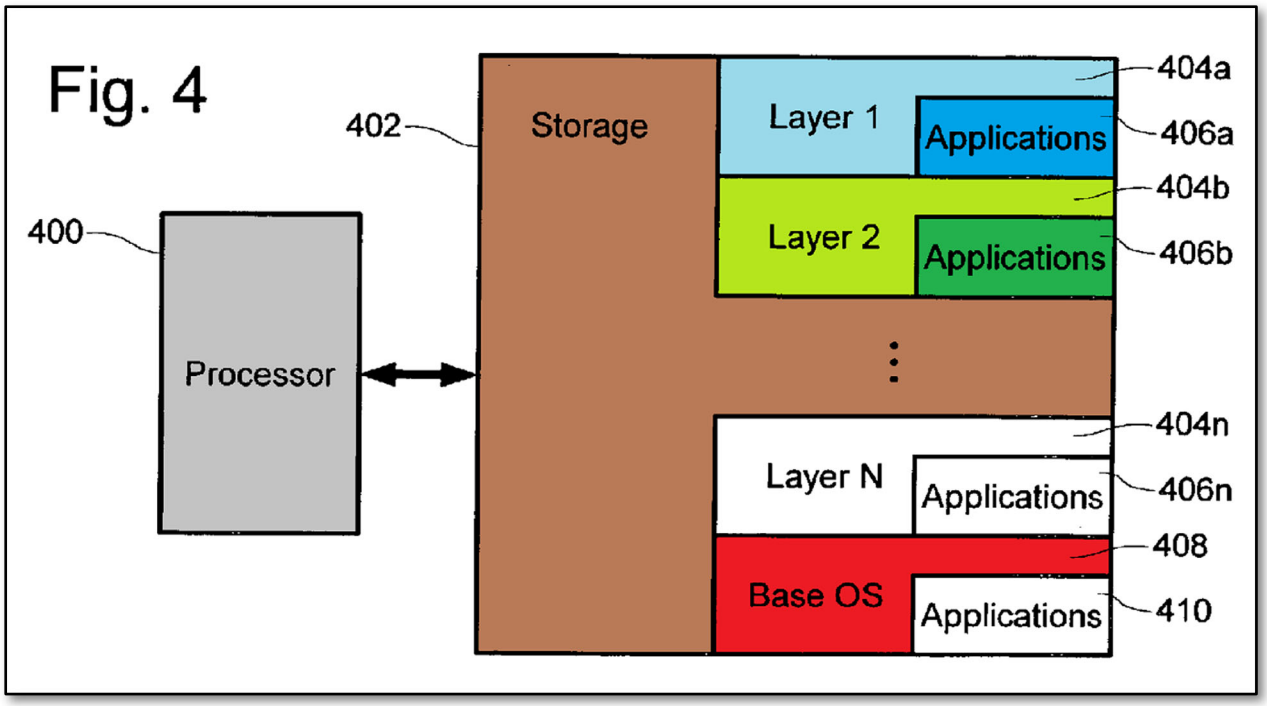
Claim terms are construed herein using the standard used in civil actions under 35 U.S.C. §282(b), in accordance with the ordinary and customary meaning as understood by POSAs and the patent's prosecution history. 37 C.F.R. §42.100(b). In the concurrent district-court proceeding, Petitioner "Google" and Patent Owner "VirtaMove" have briefed competing constructions of various claim

terms. EX1100-EX1101. As discussed further within the Grounds, this Petition presents alternative mappings of the prior art under both parties' proposed constructions of disputed terms, thus demonstrating unpatentability regardless of which proposed construction is correct. *Google LLC v. Security First Innovations, LLC*, IPR2024-00215, Paper 15, 6 (May 23, 2024) (finding this approach "complies with" Rule 42.104(b)(3)). Bhattacharjee, [0045]-[0061].

V. GROUND 1: CLAIMS 1-4, 7-11, 14, AND 16-30 ARE UNPATENTABLE OVER BLASER-CALDER

A. Blaser (EX1005)

Like the '814 patent (*supra* §III.A), Blaser concerns reducing "application conflicts" that may arise on a computer from conflicting "shared libraries and system configuration" (e.g., applications requiring "different versions" of a single shared library). Blaser, 1:14-31. Figure 4 (annotated below) illustrates Blaser's software-based solution (4:51-59): layers of "application files and directories [that] are shadowed or overlaid over the regular operating system file system" (5:20-26). Bhattacharjee, [0062]-[0063].



As illustrated above, Blaser's computing device (e.g., a "server[]") (3:16-27) has a "processor" (gray) and "storage devices" (brown) (4:51-59) that store a "base operating system" (red) and several "layers" – e.g., Layer 1 (blue) and Layer 2 (green) – each "having [respective] applications." Each layer also contains software "libraries" and configuration files (i.e., system files). Blaser, 3:42-58, 4:7-27, 5:18-32, 6:62-7:23. The shared Base OS (red) "forms a platform with which applications can be run." Blaser, 4:7-27. Bhattacharjee, [0064]-[0065] (citing EX1046, 1:25-26; EX1103, 12).

Each layer is "managed as a unit for the purposes of exporting[and] importing." Blaser, 3:59-4:6. Thus, "vendors can provide 'pre-installed' applications as layers on" storage "media" and applications can be transferred

"between similar computing systems" "by moving the layer containing the application." Blaser, 6:62-7:23. Different layers "may use conflicting configuration or library files" because layers "provide a barrier between applications." Blaser, 3:42-58. Thus, "[a] layer [] provides a convenient container to limit access to an application." Blaser, 6:62-7:23. Bhattacharjee, [0066]-[0067].

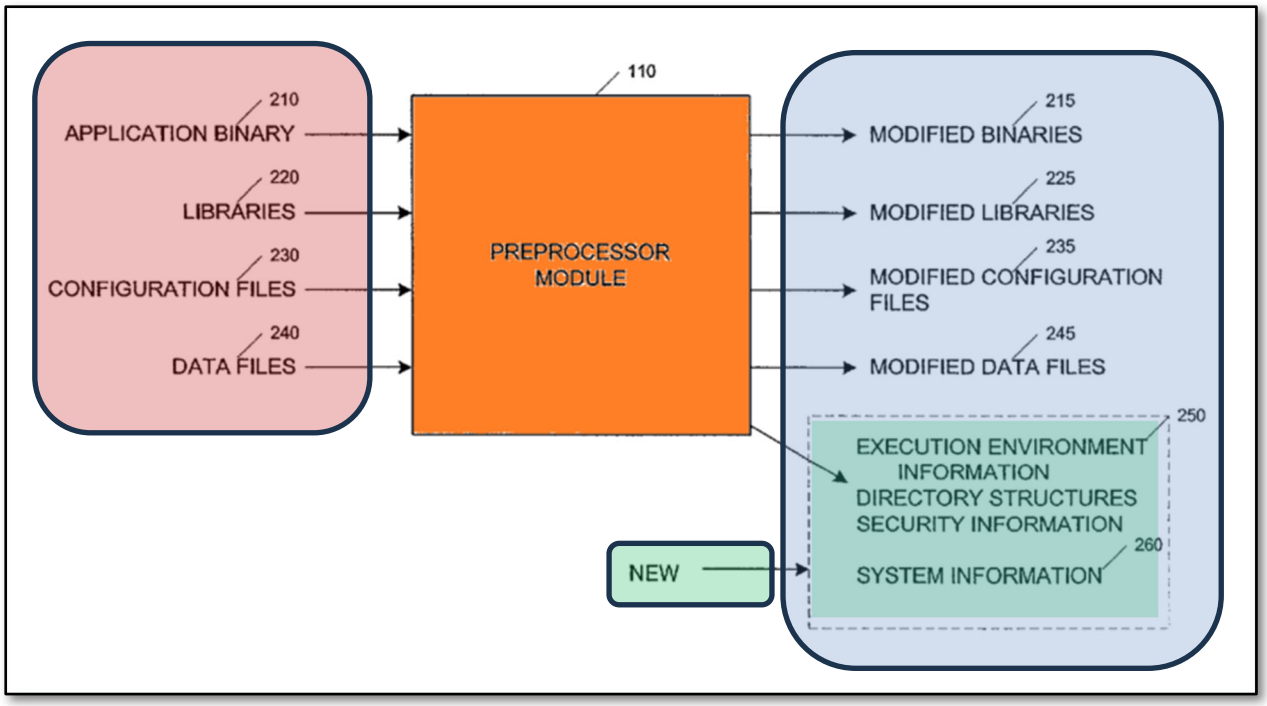
B. Calder (EX1006)

Calder teaches techniques for modifying "an application package to be executed safely, securely, and transparently on a remote machine." Calder, [0073]. Conventional "packages" for installing applications typically include application binaries ("also called application program[s]," e.g., executable software files for starting and running the application) and system files (e.g., libraries and configuration files) for the application to use. Calder, [0076], [0082]; EX1014, [0026]-[0028] ("Typically, one software **package** includes all the files…required for a local setup procedure [for]…the application…including the Dynamic Link **Libraries** (DLL)" and "**executable** file[s]"); Blaser, 1:14-19. Some conventional libraries include functionality for making "system calls," which request services/functionality provided by the OS's kernel (e.g., opening a file). EX1120, 19-20. Because different OSs may "provide different mechanisms for implementing system calls," an application package designed for one OS (its

"native" OS) might not work with another (non-native) OS. EX1097, [0082]. Bhattacharjee, [0068]-[0071].

To overcome that potential restriction, Calder teaches modifying an "application package" by adding "system libraries…that translate" "system calls" native to one OS (e.g., "Windows 2000") to work with a different OS (e.g., "Linux"). Calder, [0101]. Bhattacharjee, [0072].

Calder's Figure 2 (color added below) shows a software-implemented "preprocessing module" (orange) processing an application package's original files (pink) to create modified files (blue) (e.g., modified libraries and configuration files) that "allow for the application [] to execute and communicate on any non-native platforms" (i.e., to execute on OSs that are not the application's "native," originally intended, OS). Calder, [0101]. Bhattacharjee, [0073]-[0074]; EX1054, 4:37-41 ("'non-native' server…run[s] a" "different" "operating system").



C. Blaser-Calder Combination

POSAs had reason to install Blaser's layers in a system having multiple servers running different OSs, with each layer being executable on more than one different OS, based on Calder's teachings to make applications executable on both native and non-native OSs. Bhattacharjee, [0075].

Blaser's "layering system" provides for "transfer of applications between similar computing systems." Blaser 6:62-7:23; 8:48-9:4 ("a layer [can] be moved from one Windows system to another"); *supra* §V.A. Blaser's "computing systems" include "servers" (Blaser, 3:16-27) and Blaser's layers are compatible with various "Windows operating system[s]" ("Windows 95, 98, NT, 2000, and

XP" (Blaser, 13:43-57)) and "operating systems other than Windows" (Blaser, 3:28-36). Bhattacharjee, [0076]-[0082].

Blaser says "multiple versions of a software product are [often] installed" in a network because "an older version…supports a function desired but not supported in a newer version" or because a "software product tester[]" wants to "verify" "multiple development versions." Blaser, 13:29-41. POSAs understood that different "versions" of software typically included versions of the same application designed for different OSs. Bhattacharjee, [0083].

Blaser also discusses an example system in "a company ha[ving] 500 computers" using different "layers defined for different types of users." Blaser, 13:1-15. For example, a "secretarial layer contains…secretarial applications" (e.g., "word processing") and an "engineering layer contains…engineering related applications" (e.g., "CAD tools"). Blaser, 13:1-15. POSA knew that such large corporate systems often had "computers spread across a wide geographical area that are networked together," with groups of users "interconnected" to "**one or more server computers**" that "run[] **different network operating systems**." EX1049, 1:12-14, 1:41-52. Bhattacharjee, [0084]-[0085].

POSAs also knew it was "important…that an application be usable with several **different operating systems**." EX1058, 1:14-21. Bhattacharjee, [0086] (citing also EX1057, 1:10-14 ("[s]oftware developers often strive to tailor or 'port'

their applications to a variety of computing platforms" (e.g., "variable[]… operating system[s]")); EX1014, [0029]-[0030]).

For example, "in a distributed system of a large corporation having many end users, there are usually a number of **computers with different operating systems**" running the applications. EX1049, 1:41-58. POSAs knew that typically, in those large systems, application "software…[is] centrally hosted by a Server and used by a number of Clients." EX1020, [0004]-[0005]; Bhattacharjee, [0087] (citing also EX1051, 2:35-37).

And in those "larger networks, it is common…for the servers to be running **different network operating systems**." EX1051, 2:35-37; Bhattacharjee, [0088]-[0098] (citing also EX1054, 1:14-35, 4:48-50; EX1066, 1:17-37; EX1068, 1:19-27; EX1071, 2:4-14; EX1016, [0002]; EX1053, 1:36-41; EX1049, 1:53-58; EX1012, [0002]-[0005]; EX1093, 1:66-2:25; EX1127, 4:39-65; EX1109, [0002]-[0003]).

Thus, Blaser discloses or at minimum motivates using Blaser's layers in a system having multiple servers running different OSs (e.g., in a large organization's network), with each layer executable on at least some of those different OSs, and POSAs had independent reason based on their background knowledge to implement Blaser's layers that way. Bhattacharjee, [0098].

During prosecution, the '814 patent's applicants stated that different versions of OSs in the same OS-family (i.e., OSs with the same tradename, like

Unix-based OSs "Solaris 8" and "Solaris 10") are OSs that differ. EX1002, 300-301. POSAs knew that applications native to one OS (e.g., Windows 95) were often compatible with other OSs in the same family (e.g., Windows 98). Bhattacharjee, [0077]-[0082] (citing EX1017, [0019]-[0021]; EX1128, 1:51-57, 3:7-30; EX1050, 2:55-38). Thus, POSAs reasonably expected each Blaser layer to be successfully executable on (at least) different OSs in the same OS family. Bhattacharjee, [0082].

As discussed above (§V.B), Calder teaches how to modify an "application package" (e.g., its libraries and configuration files) so "the application" can "execute and communicate on any non-native platforms," such as allowing Windows-native applications to work with a Unix-based OS. Calder, [0101]. As discussed above, Blaser's layers include similar system files for installing applications on OSs similar to the conventional OSs Calder discusses. Thus, POSAs would have been motivated and reasonably expected success to apply Calder's teachings to make any suitable application package (including Blaser's application layers) executable on multiple, different OSs (including different OS-families), which POSAs had reason to do, as discussed above. Bhattacharjee, [0099].

In this Blaser-Calder combination, the system comprises multiple servers, at least some running different OSs (including in different OS families), and each

application layer used in the system comprises, *inter alia*, (1) the files that Blaser includes in layers (*supra* §V.B; e.g., application(s) and system files required to execute the application(s) on, at least, the application's native OS), and (2) modified system-file copies (e.g., modified libraries and configuration files) to allow the layer (and its application(s)) to also execute on non-native OSs/OS-families, based on Calder's teachings (*supra* §V.B). For example, a native Windows 2000 layer would have "system libraries added" to execute software that "translate[s] Windows 2000 system calls to Linux system calls." Calder, [0101]. Bhattacharjee, [0100]-[0102].

POSAs would have reasonably expected success in the combination. As discussed above, Blaser expressly discloses using application layers in large corporate networks (13:1-15), for servers running Windows- and non-Windows-based OSs (3:28-41). Calder's techniques similarly allow one application package to be executable on multiple OSs. Calder, [0101]. Thus, it would have required only ordinary programming skill, based on Calder's teachings, to make a Blaser layer executable on multiple servers running different OSs. Bhattacharjee, [0103]-[0106].

D. Mapping to Challenged Claims

1. Claim 1

Claim 1's preamble, which the parties agree is limiting (EX1106, 6), is addressed below in two parts: [**1PREA**]'s "system" and [**1PREB**]'s "method."

a. [1PREA]

i. "1. In a system having a plurality of servers with operating systems that differ"

Blaser's "layering system" provides for "transfer of applications between similar computing systems [plural]" *in a system* (e.g., corporate network). Blaser 6:62-7:23, 8:48-9:4. As discussed above (§V.A), Blaser's "computing systems" include "**servers**" (3:16-27) *with OSs that differ* (e.g., different versions of "**Windows**" (13:43-57) and "operating systems **other than Windows**" (3:28-36)). Blaser's Figure 4 shows **physical** hardware-based servers, thus meeting the parties' joint construction of "*servers*" as "physical servers." EX1106, 6; *compare* EX1001, FIG. 2. Thus, Blaser uses conventional OSs, including an example OS in the '814 patent (e.g., Windows). *See* EX1001, 7:34-38. Bhattacharjee, [0107]-[0113].

EX1088's definition of "*operating system*" – which is Google's construction (EX1106, 4) – is "software that controls the allocation and usage of hardware resources such as memory, central processing unit (CPU) time, disk space, and

peripheral devices" – and identifies "Windows 98, Windows NT…" as examples. EX1088, 378.

Thus, conventional prior art OSs that Blaser uses meet Google's construction and VirtaMove's "ordinary meaning" construction of "*operating system*" (EX1106, 4) and Blaser's *system has a plurality of servers with operating systems that differ*. Bhattacharjee, [0114]-[0120] (citing EX1121, 2-4).

Moreover, as discussed above (§V.C), Blaser-Calder's *system* also *has a plurality of servers with operating systems that differ* by OS family (e.g., Windows- and Unix-based servers). Bhattacharjee, [0121] (citing Calder, [0083]).

ii. "operating in disparate computing environments"

VirtaMove construes "*disparate computing environments*" as "[e]nvironments run by standalone or unrelated computers," based on the "definition[]" in the specification. EX1103, 8; EX1001, 2:16-19. While Google explains that it is unclear what "unrelated" means (EX1102, 5-7), Blaser-Calder's servers are (at least) "stand-alone" computers as discussed below, so the Board should find Blaser-Calder meets [**1PREA**] even if indefinite. *PLR Worldwide Sales Ltd. v. Flip Phone Games, Inc.*, IPR2024-00209, Paper 9, 39 (May 10, 2024) ("even if full scope of" term "is indefinite because the specification does not provide a sufficient boundary…that does not prevent us from making a determination that a teaching is well within that boundary").

Blaser-Calder's servers (*supra* §V.D.1.a.i) are stand-alone computers (at least) because they are separate computer machines. Blaser-Calder's servers also operate independently of each other, which is what VirtaMove says makes accused computers "standalone." EX1103, 9. Bhattacharjee, [0122]-[0125] (citing Blaser, FIG. 4, 4:40-59).

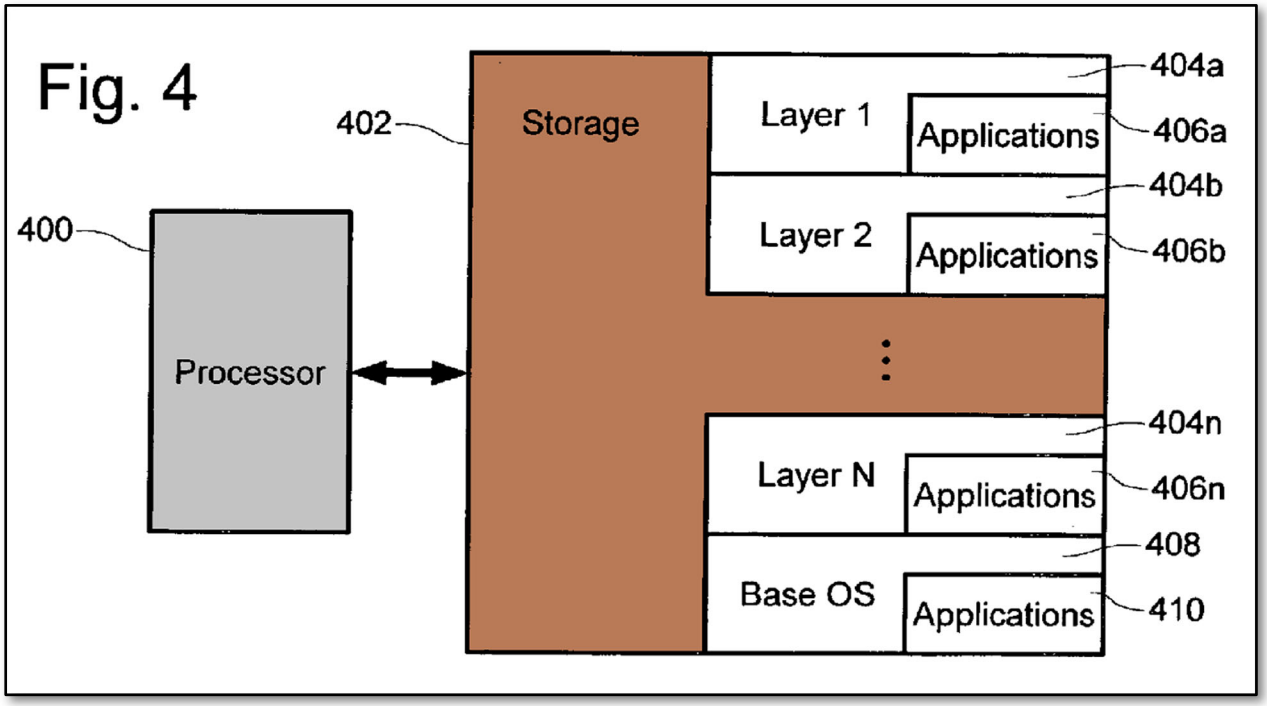
Additionally, during prosecution, the Examiner found that [**1PREA**] was met by a "network environment" (EX1002, 239-240) having a server and client (EX1089, FIG. 2, 6:28-35). VirtaMove did not disagree. EX1002, 286-333. Bhattacharjee, [0126]-[0127]. Blaser-Calder's system likewise has servers and client computers in a network environment. *See supra* §V.C.

iii. "wherein each server includes a processor and an operating system"

Blaser's Figure 4 (highlighted below) shows that each "computing device" – a *server* in Blaser-Calder – "includes a ***processor*** [gray]" and "a base ***operating system*** [red]." Blaser, 4:51-59. *See also* Calder, [0083] (computer's "standard architecture" has an "operating system"); *supra* §V.D.1.a (demonstrating that Blaser-Calder's OSs meet each party's construction of "*operating system*").

Blaser uses "a conventional" computer "having a CPU." Blaser, 3:15-27. POSAs understood the processor/CPU for a conventional computer is a physical device and thus meets the parties' joint construction of "*processor*" as "physical computer processor." EX1106, 6. Blaser says its CPU may "have peripheral

devices attached" to it (Blaser, 4:50-59), reinforcing that it is a physical processor. Bhattacharjee, [0128]-[0130].



iv. "[OS] including a kernel [and][2] a set of associated local system files compatible with the processor"

The conventional prior-art OSs that Blaser supports – Windows-based (13:43-57) and "other than Windows" (3:28-36) – include *system files* and a conventional local *kernel*. Bhattacharjee, [0137]-[0147] (citing EX1029, [0027] (the "minimum…**OS components include[] a kernel**…**and system files**"); EX1023, [0023] (Windows has "***system files* which form the *kernel* part** of the ***operating system***"), [0027]; EX1011, [0032]; EX1067, 9:12-20; EX1064, 1:53-62;

[2] The parties agree "and" should be inserted. EX1106, 6.

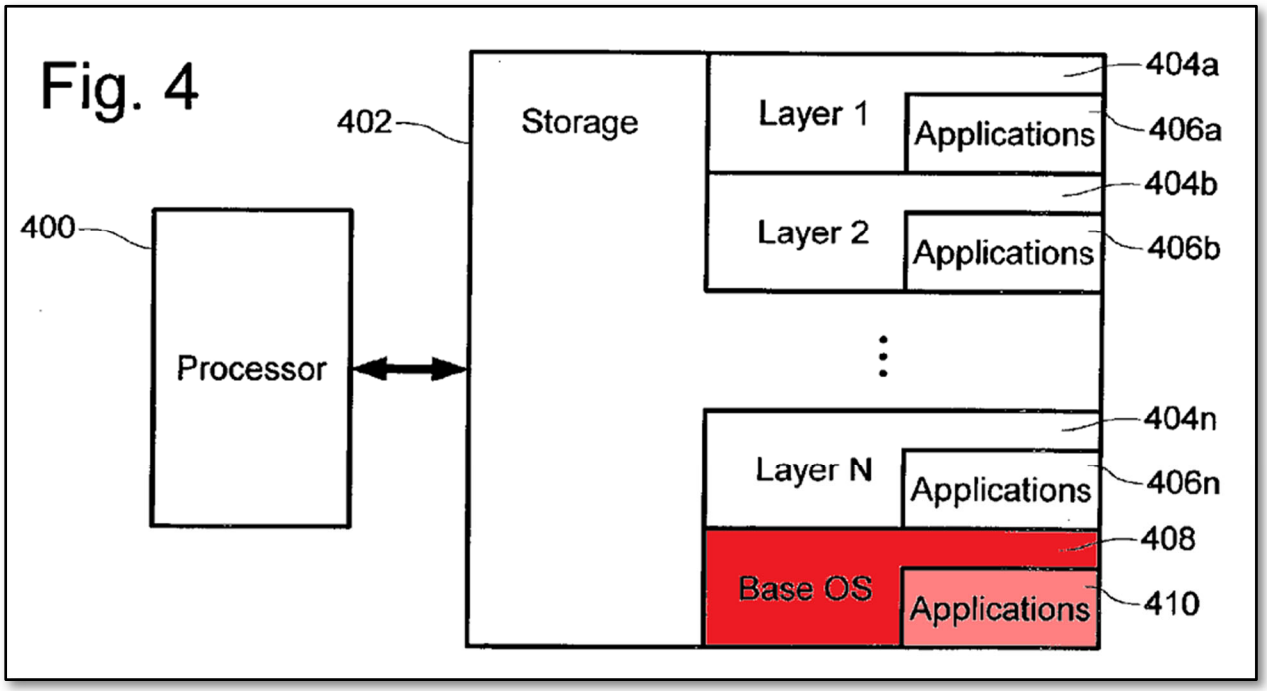
EX1062, 3:37-40; EX1094, 3:34-63; EX1046, 1:25-26 (the "***kernel*** is the core of the operating system"); EX1103, 12 (VirtaMove arguing an OS "by definition, includes a ***kernel***")).

EX1088's definition of "*kernel*" – which is Google's construction (EX1106, 4) is "[t]he core of an operating system—the portion of the system that manages memory, files, and peripheral devices; maintains the time and date; launches applications; and allocates system resources." EX1088, 300. Thus, Blaser-Calder's conventional OS *kernels* meet Google's construction and VirtaMove's "ordinary meaning" construction. EX1106, 4. Bhattacharjee, [0136]-[0148].

The '814 patent defines "***system files***" as "files provided within an operating system and which are available to applications as shared libraries and configuration files." EX1001, 2:52-54, 2:55-3:19. Thus, the patent relies on conventional *system files* typically provided with conventional prior-art OSs, like libraries and configuration files. EX1001, 2:52-3:19. Bhattacharjee, [0131].

In Blaser, a layer is "overlaid over the regular operating system file system" that exists in the server's Base OS. Blaser, 5:18-32. That "regular ***operating system file*** system" includes the files that are provided with Blaser's Base *OS*, which include the same conventional system files the '814 patent identifies as examples: "**[s]hared libraries** (such as DLLs)" **and** "system accessible **configuration** (such as registry entries)." Blaser, 5:18-32, 3:42-58 ("an

application layer" includes "**application configuration stored to operating system files**"). Blaser's Figure 4 (highlighted below) shows applications 410 executable outside of layers. Bhattacharjee, [0132]-[0133] (citing EX1108, [0003]).



Those applications use *local system files* that are separate from the system files inside layers. Bhattacharjee, [0134]; *see* EX1001, 7:12-15 ("applications executing within a secure container can co-exist with applications that are not associated with a container, but which are associated with the underlying operating system"). Thus, Blaser's server's Base OS has and uses *local system files.* Bhattacharjee, [0135].

POASs understood that Blaser's Base OS's *kernel* and *set of associated local system files* are *compatible with the* server's *processor* because they are used by the processor, which would not occur if they were incompatible. Bhattacharjee, [0136], [0149]-[0151] (citing EX1067, 35:48-54 ("the active ***kernel* must be compatible with the computer's *processor***"); EX1045, 16:60-62 ("***kernel*** programs tend not to be compatible from ***processor*** to ***processor***"); EX1043, 1:17-20 ("computer programs written for" a ***processor*** "are **frequently unsuitable for use with** computers based on **a different *processor***")).

b. [1PREB]

i. "a method of providing at least some of the servers in the system with secure, executable, applications"

Blaser's method "***provide[s]*** an installation for an ***application***" that is "installed to the destination system" (Blaser-Calder's *servers*), "but isolated in a layer." Blaser, 10:25-42; *see also* Calder, [0076] ("application package" includes "an application binary (also called application program)"). Applications in Blaser's layers are "***Secure Applications***" (at least) because "[a]pplications can be **protected** from unauthorized access." Blaser, 11:20-38. For example, an application that is "isolated in a layer" "is **protected** from [*secured* against] corruption from other applications or meddling." Blaser, 10:25-42. Bhattacharjee, [0152]-[0153].

Blaser says an "application 200 is **running** [i.e., *executing*] on a layered computing system." Blaser, 4:27-50, col. 20, page 17 ("FSLActivate" function identifies "***[a]pplications*** in the layer that are specified **to be run** on system startup…are started."); Calder, [0073], [0082]. Bhattacharjee, [0154].

Thus, Blaser-Calder's method *provides at least some of the servers in the system with secure, executable, applications*. Bhattacharjee, [0155]-[0156].

ii. "[applications] related to a service"

The '814 patent says conventional website-related "applications such as Apache, MySql and PHP" can "support[] a single ***service***, such as a web based human resource or customer management type of ***service***." EX1001, 4:26-31. The patent identifies additional conventional "***services***" like "CRM (Customer Relation Management) tools, Accounting, and Inventory." EX1001, 7:18-21. Bhattacharjee, [0157]-[0158].

Blaser discloses application "layers defined for different types of users, for example, secretaries, engineers and accountants." Blaser, 13:1-15. "The secretarial layer contains…secretarial applications" and "[t]he accounting layer includes accounting software." *Id.* Each layer provides a conventional *service* because each layer comprises one or more applications (e.g., secretarial applications), and POSAs understood that those applications that are for different types of users are specialized, software-based functionality provided by Blaser-

Calder's network servers, meeting Google's construction of "*service*" (EX1106, 4), and VirtaMove's "ordinary meaning" construction (*id.*). Further, Blaser's layer services (e.g., accounting) are like the patent's example services discussed above. Thus, each layer's applications are *related to a service* (e.g., accounting service). Bhattacharjee, [0159]-[0162].

iii. "wherein the applications are executed in a secure environment"

Blaser's layers provide *secure environments* for *executing* "***Secure*** Applications" because the applications in layers are "**protected** from unauthorized access." Blaser, 11:20-38, 13:25-28 ("layers…represent different ***environments*** on a system"). Layers "**protect** application files from viewing and copying" because "[a]ny [executing] application of the computing system desiring to open files within the layer" must supply "an authentication key or token." Blaser, 11:21-38. Layers are further secured by "be[ing] **encrypted**" (Blaser, 11:31-34) and "authenticat[ing] a user before enabling a layer" (Blaser, 11:59-62). Bhattacharjee, [0163].

Based on Calder, POSAs would have been additionally motivated and reasonably expected success to "intercept[]" "system calls made by the ***application***" packages installed to Blaser-Calder's layers to restrict "access [to] approved files," thereby protecting the servers and "the contents of the ***application*** package." Calder, [0087], [0006] ("**protect[ing] the contents of the project** from

improper tampering" and "**protect the non-secure machine** from improper tampering by the project"), [0073] ("application package" is "**executed safely, securely,** and transparently on a remote machine"), [0100] (using "security information" to "**protect[] both the client computer** [], **as well as the contents of the application**"). Bhattacharjee, [0164]-[0167].

For either reason, Blaser-Calder's *applications are executed in a secure environment*. Bhattacharjee, [0168]-[0169]. *See also infra* §V.D.1.c.i.(3) (demonstrating that layers are ***secure*** *containers*).

iv. "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"

The '814 patent says "**[a]n icon is an *object*** supported by a GUI [graphical user interface]" that "**associates an image with an *object*** that can be operated on through a GUI." EX1001, 14:21-23. For example, "the GUI support[s] the ability to **select an *object*, move it and initiate an operation**," e.g., "copy[ing] and transfer[ring] of files," "when the **object** is placed on a specific destination." EX1001, 14:4-8. Example "***tasks***" that use icon objects include "[i]nstall[ing] application specific files" and "[s]tarting a first application." EX1001, 15:35-56. Thus, the patent discloses conventional uses of application icons (e.g., clicking an icon object to activate an application's executable object). Bhattacharjee, [0170]-[0175] (citing EX1047, 1:21-67, 1:43-51; EX1059, 1:34-38).

Blaser-Calder uses two or more conventional prior-art GUI-based OSs (e.g., Windows-based OSs and Linux-based OSs), whose *applications each include an object* (e.g., icons and application files and executables) *executable* by the *different OSs* in the system, thus meeting Google's construction (EX1106, 4) and VirtaMove's "ordinary meaning" construction (*id*.) of "*at least some of the different*" OSs. Blaser, 3:28-36, 13:25-28 ("layers that represent different environments" include their own "icons" for executing applications). Bhattacharjee, [0176]-[0179] (citing EX1121, 43).

For example, when "layer A contain[s] **B.EXE**" (an executable file) and "**B.EXE** executes, it results in process C running." Blaser, 15:6-36. In GUI-based OSs, an .EXE file is an executable object that is conventionally started via its corresponding icon. Bhattacharjee, [0180]-[0182] (citing EX1047, 1:39-45 (GUI's "[e]xecution of an application program is initiated by selecting its corresponding icon"); EX1059, 1:34-39; EX1095, 8:4-13; EX1096, 4:7-10). Thus, the icon included and associated with, for example, a word-processing application is for *performing a task related to the* secretarial *service* in a layer. Blaser, 13:1-15. Bhattacharjee, [0180]-[0183].

Moreover, the "process [] running" in response to executing an application's object(s) (Blaser, 15:27-36) is itself a *task related to the service* because "[a] 'task' is also a process" that "the operating system creates" "[w]henever a program is

executed" (EX1030, [0002]). Bhattacharjee, [0184]-[0189] (citing also EX1042, 1:5-20; EX1061, 1:13-16; EX1064, 2:12-23; EX1052, 4:28-31).

In Blaser-Calder (*supra* §V.C), each layer is configured with system files (e.g., libraries and configuration files, *see* Blaser, 5:18-32, Calder, [0082]) that allow the application's object(s) discussed above "to execute and communicate on **any non-native [or native] platforms**" (including different OSs) (Calder, [0101]). Bhattacharjee, [0190]-[0192].

Thus, Blaser-Calder meets [**1PREB**]. Bhattacharjee, [0193]-[0194].

c. [1A]

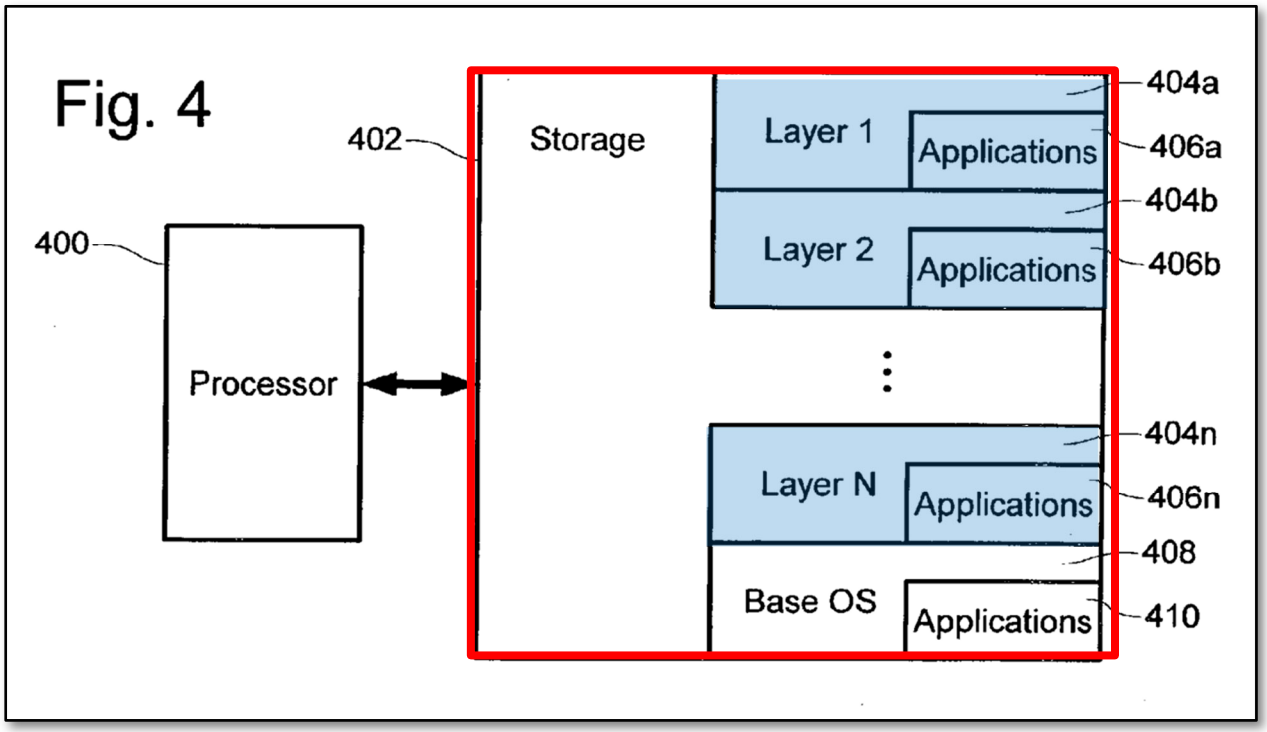
i. "storing in memory accessible to at least some of the servers a plurality of secure containers of application software"

(1) Blaser-Calder's layers (containers) are "stor[ed] in memory accessible to…the servers"[3]

Blaser's Figure 4 (color added below) shows layers 1-N (blue) *stored* on the *memory* of storage 402 (red) which is *accessible to* the Blaser-Calder *server* containing processor 400. Blaser, 4:51-59 ("layers 404a-n are also contained on storage 402"), 3:16-27, 3:59-4:6, 6:62-7:23 (layers can be "stored on separate disk partitions or remote file systems"), 7:36-47 ("each layer spanning one or more

[3] *See infra* §V.D.1.c.i.(2) (VCE-capsules are *containers*), §V.D.1.c.i.(3) (VCE-capsules are *secure containers of application software*).

fixed disks" of storage). This meets VirtaMove's construction, which only requires the claimed *memory* to be *accessible to* one *server*. EX1103, 12; EX1106, 4; Bhattacharjee, [0195]-[0199].



Google's construction of "*memory accessible to at least some of the servers*" is "memory that at least two or more of the servers can read from or write to." EX1106, 4. Blaser-Calder meets that construction because Blaser's layers "may be **stored** [written] to a network server and retrieved [read from] and **stored** as required to any computer of a computer farm" (e.g., a collection of Blaser-Calder servers). Blaser, 12:36-44. Thus, layers (*containers*) are *stored on memory accessible to* at least two *servers* (i.e., the network storage device(s)' memory which is also accessible to the servers that retrieve layers from it). Bhattacharjee,

[0200]-[0202]. *See infra* §V.D.1.c.i.(3) (demonstrating that layers are *secure containers of application software*).

(2) Blaser-Calder's layers are "containers"

Blaser's layer "provides a convenient ***container*** to limit access to an application." Blaser, 6:62-7:23, 9:18-34 ("captured data is held separate from the regular OS…in a data file, hard disk partition, or some other ***container***"). Bhattacharjee, [0203]-[0204].

The parties' joint construction of "*container*" (EX1106, 6) is addressed in two parts ([**1**]-[**2**]) below.

> **[1]**: *"An aggregate of files required to successfully execute a set of software applications on a computing platform."* EX1106, 6.

In Blaser, each layer includes *a set of software applications* (e.g., accounting applications) and "the persistent ***files*** and configuration ***required to*** **operate [i.e., *execute*] an *application***" in the layer. Blaser, 7:36-47, 4:28-50 ("***application*** 200 is **running** [*executing*] on a layered computing system"). As discussed above (§V.C), each Blaser-Calder layer includes modified "application binar[ies]," "libraries," and "configuration ***files***" that are *required to successfully execute a set of software applications* in the layer *on a computing platform*, even platforms using OSs non-native to the applications. Calder, [0082], [0102]. Bhattacharjee, [0205]-[0209].

> **[2]**: *"Each container for use on a server is mutually exclusive of the other containers, such that read/write files within a container cannot be shared with other containers."* EX1106, 6.

Blaser "**isolat[es]** application files and configuration in a layer" "**to provide a barrier between applications** which may use conflicting configuration or library files." Blaser, 3:42-58. Applications are "stored individually in layers" so "**interactions** between application files **may no longer occur**." Blaser, 6:62-7:23. Each layer has its own "**read-writable portion**" that can be "**protected from access by other users**." Blaser, 7:48-58. Moreover, "layers may be provided **accessible only to an individual user**," where "a writable layer is provided for each user, **providing data protection and isolation between users**." Blaser, 7:59-67. Thus, read/write files associated with[4] one user's layer cannot be shared with another user's layer.[5] Another mechanism by which Blaser prevents *read/write files within a* layer (*container*) from *being shared with other* layers

---

[4] The parties agree that "within a container" in the joint construction of "*container*" means "associated with a container." EX1106, 6; EX1001, 2:29-34.

[5] For the same reason, data within each application set within a layer are insulated from effects of other application sets, which meets each party's construction of *secure containers of application software*. *See infra* §V.D.1.c.1.(3).

(*containers*) is by providing each layer its own root file system, as discussed below (§V.D.1.g). Blaser, 29:11-28. Bhattacharjee, [0210].

POSAs also had motivation and reasonably expected success to prevent layers from sharing read/write files to prevent applications in other layers from corrupting those files based on Calder, which discloses "**protecting the contents** of the application package 115 **from user access**" ([0087]) to beneficially prevent "improper[] modifying or accessing data" ([0088]). Conventional techniques – like chroot (change-root) – existed for that reason, and thus required only ordinary skill to implement in Blaser-Calder. Bhattacharjee, [0211]-[0216] (citing EX1027, [0042]; EX1035, [0002]-[0003]; EX1072, [0049]-[0053]); Bhattacharjee, [0217]-[0236].

(3) Blaser-Calder's layers are "secure containers of application software"

Google's construction of "*secure containers of application software*" – which quotes the '814 patent (EX1001, 2:43-48) – is "environments where each application set appears to have individual control of some critical system resources and/or where data within each application set is insulated from effects of other application sets." EX1106, 5. VirtaMove's construction replaces "environments" with "containers." *Id.*

An "application set" includes "a set of one or more software applications that support a specific service." EX1001, 7:39-41.

As discussed above (§V.A), Blaser layers provide *environments* for, e.g., a "secretarial layer contain[ing]…secretarial applications," which are an application set that supports the secretarial service. Blaser, 13:1-15, 13:25-28 ("layers...represent different ***environments***"). Bhattacharjee, [0237]-[0240].

As discussed above (§V.D.1.c.i.(2), "container"-construction part [**2**]), each layer (and the application set it includes) is an *environment* insulated from effects of application sets in other layers because applications are "**isolated** in a layer" and "**protected from corruption** from other applications or meddling." Blaser, 10:25-42. "[A]pplication layers…**are isolated** from other applications" to "provide **a barrier** between applications." Blaser, 3:42-58. When "applications are stored individually in layers, interactions between application files may no longer occur." Blaser, 6:62-7:23. Blaser's layering technique thus provides "***Secure Applications***" because "[a]pplications can be **protected** from unauthorized access through the…layered system." Blaser, 11:20-38. As discussed above (§V.D.1.b.iii), Blaser-Calder's applications are also modified based on Calder to prevent corruption from other applications. Calder, [0079], [0087], [0100]. Bhattacharjee, [0241].

Thus, application sets in Blaser-Calder's layers are in environments that are insulated from effects of other application sets, which is one way Blaser-Calder

meets each party's construction of "*secure containers*…." EX1106, 5. Bhattacharjee, [0242].

Furthermore, exemplary "resource[s]" in the '814 patent "include[] CPU time, memory, file activity and network bandwidth." EX1001, 10:15-17. Blaser says because changes made by a layer's applications are only "recorded to a layer and not to the underlying file systems and OS resources" users can "**use the computer in an unrestricted fashion**." Blaser, 12:8-20. Because layers are unrestricted, their application sets appear to have control of some critical system resources (like the memory and hard disk), but do not actually control them, because when a layer is deleted, any changes recorded within the layer are "wiped" and the computer is restored to the state it was in before the layer was installed. Blaser, 12:8-20. Similarly, as discussed below (§V.D.1.i), layers cannot access the server's root directory, reinforcing that layers lack actual control of that critical system resource. Blaser, 29:11-28; Calder, [0138], [0241]. Bhattacharjee, [0243]-[0246] (citing EX1072, [0049]-[0053]).

Blaser's layers also "intercept **key file system and registry calls** and manipulate the results **to create the appearance** that virtual files and registry settings contained in the layer definitions exist in the real file system and real registry." Blaser, 14:60-15:3. The key file system and registry calls – which a layer *appears* to have control of – are and/or manage critical system resources.

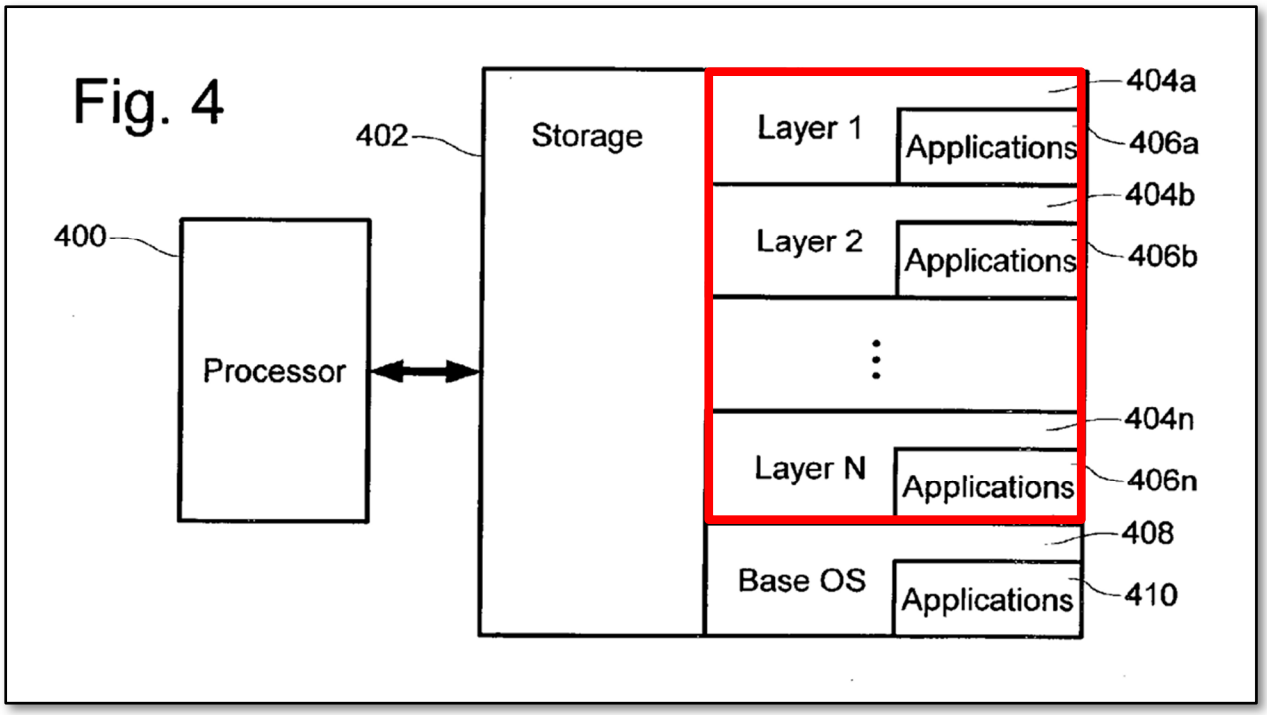
Bhattacharjee, [0247]-[0248] (citing EX1026, [0004] ("non-file system resources" include the "registry")).

Additionally, based on Calder, application "resource requests [are] virtualized" in Blaser-Calder (§V.C) and appear to "control[] resources," including "memory usage," "network bandwidth," "and disk usage." Calder, [0126], [0130] (explaining that the resources actually used are "transparen[t]" to and "hidden from the application").

Thus, Blaser-Calder layer application sets appear to have individual control of some critical system resources, which is another way Blaser-Calder meets each party's construction of "*secure containers of application software*." EX1106, 5. Bhattacharjee, [0249]-[0251].

ii. "each container comprising one or more of the executable applications"

As discussed below, Blaser's layers "***each*** hav[e] ***applications***" (4:51-59) that are "running" (i.e., *executable*) in the layer (4:27-29). Blaser, 5:18-32, 13:1-15 ("layer…contains…***applications***"), 15:6-36 (when an application in a layer "***executes***, it results in [a] process [] running"). As illustrated in Blaser's Figure 4 (annotated below), "***application[s]*** [406a-406n are] stored in a layer." Blaser, 5:18-32.



Each layer stores the associated "files and directory structure of the application's installation," including *system files* like "[s]hared **libraries** (such as DLLs), [**and**] system accessible **configuration** (such as registry entries)." Blaser, 5:18-32. Thus, each Blaser-Calder layer comprises *application programs* and *associated system files* (e.g., libraries **and** configuration files) *for use in executing* the applications. Bhattacharjee, [0252]-[0255].

As discussed above (§V.C), based on Calder, each Blaser-Calder layer comprises *associated system files for use in executing* the applications on both native and non-native OSs. *See* Calder, [0082]-[0083], [0101]. Bhattacharjee, [0256]-[0257].

iii. "and a set of associated system files required to execute the one or more applications"

The '814 patent defines "***system files***" as "files provided within an operating system and which are available to applications as shared libraries and configuration files." EX1001, 2:52-54. Thus, the patent recognizes (as POSAs knew) that conventional prior-art OSs typically provided *system files*. Bhattacharjee, [0258].

In Blaser, each "layer may be defined to be a set of file system and registry changes" (4:60-5:2), which includes "the ***files*** and directory structure of the application's installation" (5:19-32). These "***files*** and directories" are "shadowed or overlaid over the regular operating system file system." Blaser, 5:19-32. Files copied into layers include "**shared libraries** (DLLs)" (Blaser, 6:62-7:23), "persistent ***files*** **and configuration** ***required to*** **operate** ***an application***" (Blaser, 7:36-47), and copies of "registry changes" (Blaser, 4:60-5:2), which meet the '814 patent's definition of "*system files*." *See* EX1001, 2:55-3:19 (exemplary system files include libraries and configuration files); Blaser, 9:18-34 (layers "capture" "all of the application[']s shared DLLs, registry entries, and .ini files"), 10:50-59 ("software installation of an application layer" is recorded to the layer's "readable-only portion" so the user cannot "damage the application installation"). Bhattacharjee, [0259]-[0260].

As discussed above (§V.C), based on Calder, Blaser-Calder's layers also include modified *system files* (e.g., modified libraries and configuration files) that

are *required to execute the applications* on non-native OSs. Calder, [0077], [0082], [0094], [0101]. For example, "***system*** libraries are added…that translate Windows 2000 system calls to Linux system calls." Calder, [0101]. Bhattacharjee, [0261]-[0262] (citing EX1061, 1:13-16, 2:53-58 ("Typically, when" an OS "instantiate[s] an application," the OS will "locate[]" "and then process[]" a "configuration file[] for the application")).

iv. "for use with a local kernel residing permanently on one of the servers"

In Blaser, the "base operating system [] forms a platform with which applications can be run" in layers. Blaser, 4:7-27, FIGs. 1-4. When applications request access to the Base OS, "the layering system software" "determine[s] whether the accesses should be permitted to continue to the base operating system [], or should be redirected" within the layer. Blaser, 4:7-27. As discussed above (§V.D.1.a.iv), the conventional OSs *resident on* Blaser-Calder's *servers* (e.g., Windows-based, Unix-based, etc.) each have a *local kernel* that is exclusively *used* to obtain resources/services provided by the kernel that are needed to execute applications (*e.g.*, open files, use the underlying hardware). *See* Calder, [0150] (POSAs knew applications can "execute an interrupt call on the [computer's] operating system kernel"). Bhattacharjee, [0263]-[0265] (citing EX1097, [0051]).

POSAs understood *local kernel residing* in the server's memory (Blaser, 4:51-59, FIG. 4) is not (or conventionally would not be) lost when power is

removed from the server, and thus is stored in non-volatile memory, which meets Google's construction of a "*local kernel…*" (EX1106, 4) and VirtaMove's "ordinary meaning" construction (*id.*). Bhattacharjee, [0266]-[0269] (citing EX1091, 1:23-38).

d. [1B] "wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems,"

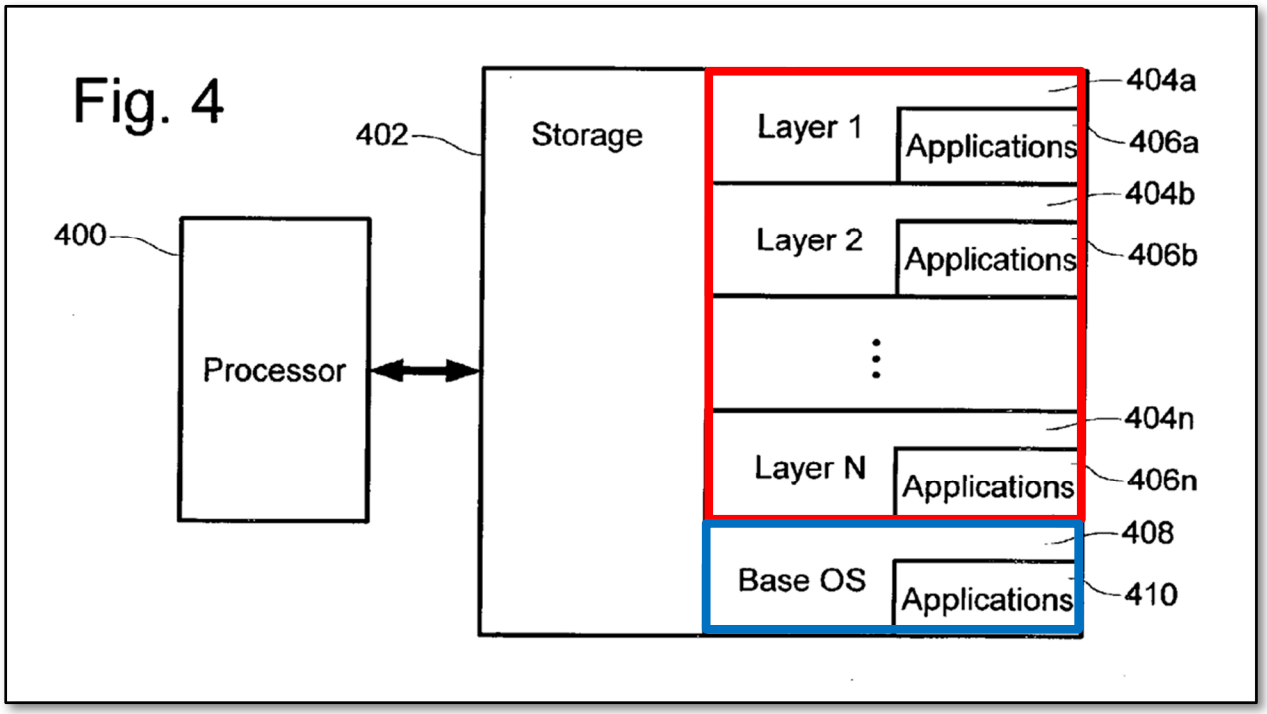
As discussed above (§§V.D.1.c.iii-iv), each Blaser-Calder layer (container) comprises a *set of associated system files* that are *for use with* the server's *local kernel*. Blaser, 4:60-5:2; Calder, [0077], [0082], [0094], [0101] (preparing modified *system files* so an application can run on "any non-native" OS). Because each server's Base OS and its *local kernel* provide the "platform" for layers (Blaser, 4:7-27, FIG. 1), each layer's *associated system files are compatible with the local kernel of* the Base OS, which would not occur if they were incompatible. Bhattacharjee, [0270]-[0272] (citing EX1086, 2:53-55 ("the *operating system* is commonly called the system *kernel* [], or just the *kernel*"); EX1064, 1:53-62 (a "traditional operating system" has a "kernel")). Bhattacharjee, [0270]-[0272]

As discussed above (§V.D.1.a.i), each Blaser-Calder layer is compatible with at least two different OSs; thus, *the set of associated system files* in each layer is compatible with *the plurality of different* OSs (and their *local kernels*) under

each party's construction of "*at least some*…," which would not occur if they were incompatible. EX1106, 4. Bhattacharjee, [0273]-[0275].

e. [1C] "the containers of application software excluding a kernel,"

Each layer has "***applications***" and contains "a set of file system and registry changes" for use in executing the applications. Blaser, 4:51-63, 13:1-15 ("layer…contains…***applications***"). As Blaser's Figure 4 (colored below) shows, layers and their ***application software*** (red) are outside of and "overlaid on top of an operating system" (the Base OS and its ***kernel***) (blue). Blaser, 2:50-58. The "base operating system [] forms a platform with which ***applications*** can be run and files can be accessed in file systems." Blaser, 4:7-27.



When a layer is initially "captured" (i.e., created for future use), "changes made by the installation procedure **do not affect the base system**" because "captured data is **held separate from the regular OS**." Blaser, 9:5-34. After a layer is subsequently installed on another computer, changes are recorded to the installed layer but "**not to the** underlying file systems and **OS** resources." Blaser, 12:8-20. Bhattacharjee, [0276]-[0278].

Thus, layers *exclude* and cannot affect the Base OS's kernel. Bhattacharjee, [0279], [0222]-[0227] (citing EX1024, [0005]-[0006]).

Moreover, "each layer is a separate and individual entity within **the** host OS" (Blaser, 5:18-32), reinforcing that each computing device with layers has a single OS whose kernel is shared by all layers on the device. Bhattacharjee, [0280]-[0281].

Furthermore, based on Calder's teachings, a Blaser-Calder layer (§V.C) include "system libraries [that] are added…that translate Windows 2000 system calls to Linux system calls." Calder, [0101]. "[W]hen a program runs under the Windows operating system," for example, "it accesses the operating system via the Windows API," with "[o]nly the WIN32 API calls [] allowed to access the operating system." Calder, [0096]. POSAs knew that the WIN32 API is how applications access a 32-bit Windows kernel. Bhattacharjee, [0282] (citing EX1048, 6:7-8 ("Win32 ***kernel***")). Because layers access the OS kernel via an

API, that reinforces to POSAs that the layers themselves *exclude* the *kernel*. Bhattacharjee, [0282]-[0283].

f. [1D] "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,"

As discussed above (§V.D.1.a.iv), the *associated system files within* each Blaser-Calder layer (*container*) are *stored in memory* and include "captures" of "all of the applications shared DLLs, registry entries, and .ini files" that are needed to execute the layer's application(s). Blaser, 9:18-34. The "[a]pplication files that would be placed on file systems managed by the OS are also redirected into the layer." Blaser, 9:18-34. "All of the captured data is held [i.e., *stored*] separate from the regular OS either locally or remotely in a data file, hard disk partition, or some other container." Blaser, 9:18-34. Bhattacharjee, [0284]-[0285].

Similarly, based on Calder (*supra* §V.C), Blaser-Calder's layers include modified *system files* – e.g., "libraries" and "configuration files" (Calder, [0082]) – that *are utilized* to allow the layer's application(s) to execute on both native and non-native OSs. Bhattacharjee, [0286].

As discussed above (§V.D.1.iv), the Base OS *that remains resident on* a Blaser-Calder *server* comprises *associated local system files* that are used, e.g., to execute applications that are not in layers. Blaser, FIG. 4, 3:42-58, 4:51-59. But as discussed above (§V.D.1.c.iii), the *associated system files within* a layer are

utilized in place of those *local system files* when executing an application in a layer. Bhattacharjee, [0287]-[0288].

Thus, *system files within* Blaser-Calder layers (containers) *are utilized in place of the associated local system files that remain resident on the server*.

g. [1E] "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,"

Blaser's "layers may be stacked on top of each other, with **the real file system at the bottom of the stack.**" Blaser, 8:6-18. The real file system at the bottom is the Base OS's file system, which includes its *associated local system files that remain resident on the server*. *Supra* §V.D.1.f. "[F]iles of the same name and location" can "exist in multiple layers, or in the base file system," and "rules can be provided whereby the layered system can determine which file to present to an application." Blaser, 8:6-18. POSAs understood that means a layer's *system files are utilized in place of system files that remain resident on the server* in its base OS (e.g., when a layer and the "real file system at the bottom of the stack" have *system files* with the same name and location, the layer's *system files* are utilized). Bhattacharjee, [0289].

A layer's *system files* are created "through a 'capture' operation" that "intercept[s] operations" involving "files and configuration such as a registry," such that "changes made by the installation procedure **do not affect the base**

system but are rather recorded to the new layer." Blaser, 9:5-17. "[A]ll operations by any application to create, ***modify*** or delete files are entered into [the] layer," so "all of the applications shared DLLs, registry entries, and .ini files that would be directed to the Windows [local] system directories **become trapped in the capture layer**." Blaser, 9:18-34. Blaser also uses a "file system ***copy***" to create layers (Blaser, 9:64-67), which copies the system files in the file system used to create a layer. Bhattacharjee, [0290]-[0292]. Thus, the *system files* in a Blaser-Calder layer are *copies or modified copies of the associated local system files*.

Moreover, based on Calder's teachings (*supra* §V.C), Blaser-Calder's layers include "***modified***" *system files* ("***modified*** libraries [and], ***modified*** configuration files," [0082]) that are based on "***cop[ying]*** specific information **from the existing system registry**" ([0121]).

Thus, with Blaser-Calder's layers, the *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*. Bhattacharjee, [0293]-[0295].

h. [1F] "and wherein the application software cannot be shared between the plurality of secure containers of application software,"

Blaser-Calder's "application layers" (*containers*) "**are isolated from other applications** on a computer" by "provid[ing] **a barrier between applications**

which may use **conflicting** configuration or library files." Blaser, 3:42-58. Blaser's teachings that layers can have conflicting *system files* (like configuration and library files) and that deleting a layer does not disturb other layers (Blaser, 3:42-58) reinforce that each layer's application(s) are not shared. "[E]ach layer is **a separate and <u>individual</u> entity** within the host OS" (Blaser, 5:18-32) and applications in layers are "**protected from corruption** from other ***applications*** or meddling" (Blaser, 10:25-42). For example, for "***applications***…stored individually in layers, **interactions between *application*** files **may no longer occur** due to conflicting shared libraries (DLLs), as each ***application*** 'sees' only it's [sic] own installed libraries first, followed by libraries in the base operating system." Blaser, 6:62-7:23. "**A second *application*** installed to the computer…**cannot view the *application* layer files** because it does not possess the correct authentication key." Blaser, 11:39-58. Bhattacharjee, [0296]-[0297].

Moreover, as discussed further below (§V.D.1.i), each layer has its own, unique root directory to which applications are installed, which further teaches or at least renders obvious that applications cannot be shared across layers. For example, POSAs knew chroot was commonly used to create a "sandbox" environment where applications have "access only to the constrained environment and **cannot corrupt software applications outside it**, i.e. beyond the sandbox boundary." EX1075, 2:9-15; Bhattacharjee, [0298].

Thus, Blaser-Calder's *application software* within a layer (*container*) *cannot be shared between the plurality of secure containers of application software.* Bhattacharjee, [0299].

i. [1G] "and wherein each of the containers has a unique root file system that is different from an operating system's root file system."

In Blaser, each layer's file system is "overlaid on the normal file system." Blaser, 29:11-28. An "fslrdr directory" is created, and "[u]nder the fslrdr directory[,] **directories that correspond to *each* layer** are maintained." Blaser, 29:11-28. For example, if layer "TEST" has "the directory 'C:\fslrdr\TEST\c\XYZCorp,'" the layer's application(s) do not know that "XYZCorp" is a subfolder of "fslrdr\TEST\c" because "the directory 'c:\XYZCorp' **appears** on the C: drive to all applications running under that layer." Blaser, 29:11-28. Bhattacharjee, [0300] (citing Calder, [0241] ("root directory C:")).

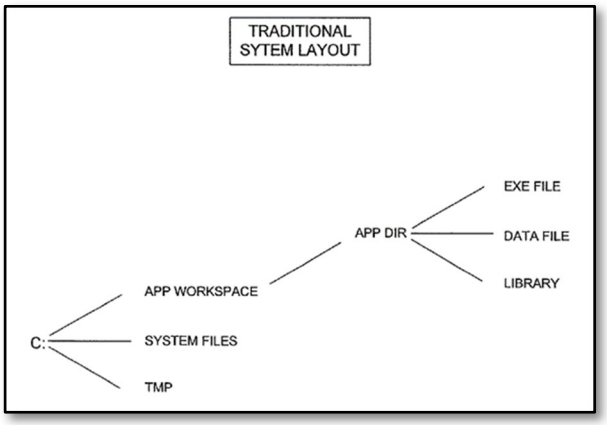
Thus, based on Blaser, each layer *has a unique root file system* (e.g., "C:\fslrdr\TEST") *that is different from* the server's *operating system's root file system* – the *root file system* under "c:\," meeting VirtaMove's interpretation of [**1G**]. *See* EX1103, 8 ("Realistically if a container on a particular server does not share a root file system with the OS *on that server*, it will not share a root file

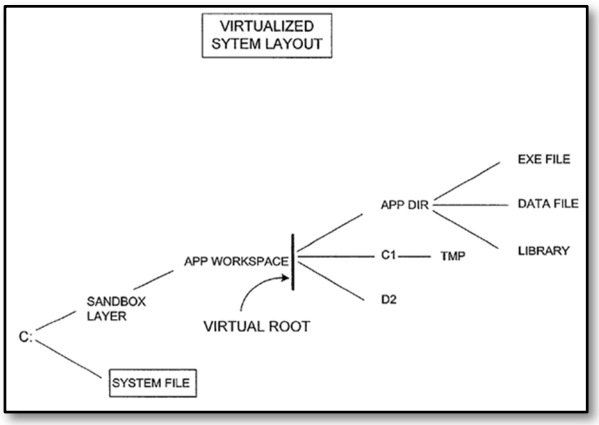
system with ***any*** operating system (in the entire world).") (emphasis original); Bhattacharjee, [0301].

Google explained in district court that [**1G**] is indefinite because it recites "*different from **an** operating system*" without indicating **which** OS the *container*'s *root file system* must be "*different from*." EX1102, 14. The Board need not resolve that dispute (*see PLR*, IPR2024-00209, Paper 9, 39), because it would have been obvious to POSAs that some modifications users may make to their own layer's *unique root file system* would make it *different from* the *root file system* of **every** OS in the world, which would meet [**1G**] regardless of the uncertain boundaries of what OS (or set of OSs) the claim is referring to. For example, Blaser discloses that users may make "**changes**…to that user's layers" (12:27-31) and can "**install[]**" applications (12:31-34), which POSAs understood can change the layer's *file system* to be different from every OS's root file system in the world. Bhattacharjee, [0302]-[0304].

Additionally, POSAs understood the *unique root file system* assigned to each Blaser layer provides a chroot (changed root) file system, which meets [**1G**] under VirtaMove's apparent claim scope in the district-court litigation, which alleges that a *container* assigned to a "**chroot**" file system meets [**1G**]. EX1079, 40-51 (emphasis original). Bhattacharjee, [0305]-[0306].

POSAs were also motivated to chroot each layer based on Calder. Calder's Figure 46 (below left) shows a "traditional system layout"; Figure 47 (below right) shows a "virtualized system layout":





In the traditional layout, applications can "access…the ***root*** directory C:" and its "three folders": "APP WORKSPACE," "SYSTEM FILES," and "TMP." Calder, [0241]. In Calder's "virtualized system layout," each application's "sandbox directory" has "a **virtual *root* directory**." Calder, [0241]. When a Calder-modified application "request[s]" "to access[] the subdirectory 'C:\TMP,'" it is "transparently" redirected to "the sandbox directory C:\SANDBOX_LAYER\AP[P]_WORKSPACE\C1\TMP" under the *unique root directory* "C:\SANDBOX_LAYER_AP[P]_WORKSPACE." Calder, [0241]. POSAs had motivation to sandbox Blaser-Calder's layers in the same way Calder sandboxes applications with a changed *root*, to support "safely" and "securely" "execut[ing]" applications "on a remote machine." Calder, [0073]. Bhattacharjee, [0307]-[0312].

2. Claim 2: "[C]laim 1, wherein each container has an execution file associated therewith for starting the one or more applications."

As discussed above (§V.D.1.b.iv), each Blaser-Calder layer has "icons," where each icon is or is *associated with* an "*execution file…for starting the one or more applications*" (e.g., an application's .EXE file). Blaser, 13:25-28, 15:6-36. Bhattacharjee, [0313]-[0315] (citing EX1047, 1:21-67; EX1059, 1:36-38; EX1070, 4:50-56), [0170]-[0193]. The .EXE files (*execution files*) are for ***applications*** like "word processors." Blaser, 3:42-58. *See also* Calder, [0077] ("prepares a software package for ***execution*** on…computers"). Bhattacharjee, [0315]-[0316] (citing EX1069, 11:35-39).

Additionally, Blaser's "FSLLIB32.DLL runtime library" "provides an API [application program interface] that may be used by other ***applications*** to manage the layered system," "includ[ing] functions to" "**activate** and deactivate layers" and "**enable** and disable layers." Blaser, 20:19-36. The function "FSLActivate (PTCHAR fslName)" "[v]alidates the fslName against defined layers" – i.e., the defined layers being the layers available in the system – and "[c]ommunicates with FSLX driver…to **activate** the layer[/layer group]" identified by "fslNAME." *See* Blaser, col. 20 (table's first function); *see also* ("FSLAddLayerToGroup (PTCHAR fslName, PTCHAR, groupName": "Verifies that both the specified layer [fslName] and group [groupName] are defined."). The variable "fslName"

identifies "[a]pplications in the layer that are specified to be run on system startup…[, which] are **started**." *See* Blaser, col. 20 (table's first listed function). Thus, FSLActivate is an *execution file* that *starts* a layer/group of layers, which further causes *execution* of *execution file(s)* in the layer for starting startup *application(s)* within the identified layer(s), as claimed. Bhattacharjee, [0317]-[0318].

VirtaMove alleges claim 2 is met by an "image configuration" that "includes information such as application arguments, environments, etc." EX1079, 51-55. Blaser similarly discloses a "layer definition" that "include[s] layer properties and settings" (Blaser, 4:60-5:2), which POSAs understood would be interpreted by an *execution file associated* with the layer *for starting applications*. Bhattacharjee, [0319]-[0322] (citing EX1084, [0145]-[0153]).

Blaser-Calder's layers meet claim 2 in any of the above ways. Bhattacharjee, [0323].

3. Claim 3: "[C]laim 2['s]…execution file includes instructions related to an order in which executable applications within will be executed."

As discussed above (§V.D.2), Blaser's function "FSLActivate" (an *execution file*) "activate[s] the layer" identified to FSLActivate and "[a]pplications in the layer that are specified to be run on **system startup** (in **win.ini**, **registry**, **startup folder**, etc.) are started." Blaser, col. 20 (table's first function).

Those various files for "startup" each *include* (or obviously would include) *instructions related to an order in which executable applications will be executed*. Bhattacharjee, [0324]-[0330] (citing EX1041, 2:8-17; EX1027, [0070]-[0073]; EX1013, [0003]; EX1021, [0020]).

4. Claim 4: "[C]laim 1…pre-identifying applications and system files required for association with the one or more containers prior to said storing step" [1A].

Blaser's "bundling of an application and user files into a layer **provides a package** that may be…transported conveniently" to "provide 'pre-installed' applications as layers on CD-ROM or other media." Blaser, 6:62-7:23. Calder's teachings reinforce that Blaser-Calder's layers contain *pre-identified* applications and associated system files ([0082]) for transmission to and installation on computers ([0073]). Bhattacharjee, [0331].

Because applications and files within (i.e., *associated with*) a layer are stored in a transportable package that exists **before** installing the layer and its applications on a particular computer, the *applications and system files required for association with the one or more containers* are *pre-identified prior to* [**1A**]'s *storing step* (*supra* §V.D.1.c). Bhattacharjee, [0332]-[0336] (citing EX1084, [0145]-[0150]).

5. Claim 7: "[C]laim 2…modifying at least some of the system files to define container specific mount points associated with the container."

The '814 patent mentions "***mount points***" without explanation. EX1001, 11:17-18. POSAs knew "[a] ***mount point*** simply allows a drive volume to be ***mounted*** at some ***point*** within an existing directory tree structure." EX1056, 2:37-52; Bhattacharjee, [0337] (citing also EX1018, [0044]).

Blaser discloses a "layer becom[ing] ***mounted*** (or enabled)" by being "shadowed or overlaid over the regular operating system file system." Blaser, 5:18-32. "[A]n fslrdr directory at the root of the volume" (the host's root) "contains file system information for each of the defined layers." Blaser, 29:11-28. As discussed above (§V.D.1.i), the example layer "TEST" has the *specific mount point* "C:\fslrdr\TEST\" (Blaser, 29:11-28), which is identified in a *modified system file* – e.g., the registry (configuration) "key called fslrdr," which keeps "all registry information contained in each layer." Blaser, 14:43-59. *See also* Calder, FIG. 47, [0135] (mounting applications to a "sandbox directory" with mount point "C:\SANDBOX_LAYER\AP[P]_WORKSPACE\"). Bhattacharjee, [0338]-[0341].

Thus, Blaser-Calder meets claim 7.

6. Claim 8: "[C]laim 1['s]…applications and associated system files are retrieved from a computer system having a plurality of secure containers."

Each "layer may be **stored to a network server [*a computer system*] and *retrieved* and stored** as required to any computer." Blaser, 12:36-44. Thus, layers (and the *application(s) and associated system files* stored therein) *are retrieved from a computer system* (network server) *having a plurality of* layers (*secure containers*). Bhattacharjee, [0342]-[0343].

Calder similarly discloses "at least **one *server* that transmits application packages to the member computers**." Calder, [0075]. That would have also motivated (and provided reasonable expectation of success in) having Blaser-Calder's servers *retrieve* layers *from a computing system* (a server) *having a plurality of secure containers*. Bhattacharjee, [0344]-[0349] (citing EX1084, [0162]-[0166]; EX1014, [0005]-[0006], [0015]; EX1015, [0004]).

Blaser-Calder also meets claim 8 because Blaser's Figure 4 shows a *computer system* storing layers 1-N. Blaser, 4:51-59. Accessed layers are thus *retrieved from* that *computer system having a plurality of secure containers*. Bhattacharjee, [0350].

7. Claim 9: "[C]laim 2, 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."

Calder teaches to modify an application package to manage its use of computer resources, like "***memory usage***," "***network bandwidth used***," and "***disk usage***." Calder, [0125]-[0126]. The "grant[ed] allocation" of resources is "predicated" (i.e., pre-assigned) using "heuristics" (e.g., "amount of virtual memory currently being consumed"). Calder, [0126].

Thus, Calder teaches allocating to applications *hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation*" that "*is associated with at least some of the containers prior to the applications within the containers being executed*. Bhattacharjee, [0351]-[0352].

POSAs had motivation to apply resource-allocation techniques, like Calder's, to Blaser-Calder's layers and allocate resources "*prior to the applications within the containers being executed*" because layers are isolated from each other and thus unaware of competition for hardware resources, so POSAs would want to ensure layers had sufficient resources to execute their applications by allocating resources before executing applications. *See* Blaser, 3:42-58. Bhattacharjee, [0353]-[0361] (citing EX1132, [0009]-[0011]; EX1036, 1:16-26; EX1063, 1:52-

2:62; EX1027, [0176]; EX1037, 6:29-38; EX1073, 1:48-54; EX1074, 3:66-4:22; EX1076, 13; EX1083, [0035]-[0040]; EX1094, 2:5-21).

POSAs would have reasonably expected success because, as the cited corroboration demonstrates, pre-allocating resources to applications/processes was conventional by the early-2000s. Bhattacharjee, [0362]-[0363].

8. Claim 10: "[C]laim 2, 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."

Applications in layers "are **isolated from other *applications*** on a computer" "to **provide a barrier between *applications*** which may use conflicting configuration or library files." Blaser, 3:42-58. "[E]ach layer is a separate and individual entity within the host OS." Blaser, 5:18-32. Additionally, "layers may be provided **accessible only to** an individual user," which "provid[es] data protection and isolation between users" (Blaser, 7:59-67) and thus isolates the system files and applications in another user's layer. Thus, an application "isolated in a layer" "is **protected** from corruption **from other applications** or meddling." Blaser, 10:25-42, 5:3-17 (layers "prevent[] accidental data modification, loss, or meddling" between applications). Bhattacharjee, [0364].

Moreover, as discussed above (§V.D.1.i), each layer has a unique root file system and thus the layer's *system files* (in the layer's root or a

subfolder/subdirectory thereof) *cannot* be *accessed* by applications in other layers. Blaser, 29:11-28. Bhattacharjee, [0365].

When Blaser creates ("captures") a layer, "all operations by any application to create, modify or delete files are entered into a layer" and "[a]pplication files that would be placed on file systems managed by the OS are also redirected into the layer," so that "[a]ll of the captured data is **held separate from the regular OS**." Blaser, 9:18-34. Requests to access local system files are redirected into the layer because a layer's application *has no access to system files within the OS*. Bhattacharjee, [0365].

Thus, applications executed within Blaser-Calder's layers *have no access to system files or applications in other containers* (layers) *or to system files within the operating system during execution thereof*. Bhattacharjee, [0366].

POSAs also had motivation to prevent applications in layers from *accessing system files within the OS during execution* to protect the Base OS's *system files* from being corrupted by layers. EX1015 discloses one known technique for doing so: assigning a "'namespace'" to "**isolate** software packages from one another even if the same component is used by multiple packages" ([0080]), such that "any files and components installed in that namespace are visible to the application while files and components installed in other **namespaces** are not" ([0086]). Bhattacharjee, [0367]-[0372] (citing also EX1076, 5; EX1077, 120-121). *See*

EX1079, 64 (VirtaMove arguing "namespaces" "isolat[e] processes in an environment").

9. Claim 11: "[C]laim 2['s]…containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location."

Blaser-Calder's layers (*containers*) *include* "***files*** located to…***network drives***" (Blaser, 7:36-47) (i.e., *files stored in network file storage*). Blaser, 12:36-44 ("layer may be ***stored*** to a ***network*** server and retrieved…as required"), 12:55-67; Calder, [0139] (disclosing "***files***…***stored*** remotely on separate machine"). Bhattacharjee, [0373]-[0374].

Blaser's "manager application" manages *parameters forming descriptors of* layers (*containers*), such as "view[ing] information about a layer" and "edit[ing] certain layer information." Blaser, 8:19-41, 20:19-38. That information is (or obviously would be) *stored in a separate location* (e.g., to facilitate management by a system administrator). Bhattacharjee, [0375]-[0377].

10. Claim 14

a. [14PRE] "[C]laim 1…creating containers prior to said step of storing containers in memory"

As discussed above (§V.D.4), Blaser's "layer provides **a package** that may be…transported conveniently" so "application vendors can provide **'pre-installed' applications as layers** on CD-ROM or other media." Blaser, 6:62-7:23. A "packaged" layer is thus *created prior to said step of storing containers in memory*

of the destination server computer. Bhattacharjee, [0378]-[0379]. *See* Calder, [0073] (disclosing pre-packaged applications "transferred" "from a server to the client").

b. [14A] "wherein containers are created by: a) running an instance of a service on a server;"

Files in a layer are identified by "captur[ing]" how an application's installation process uses/changes the file system and system files. Blaser, 9:5-39, 10:1-24. Capturing begins by "execut[ing] an application installation program." Blaser, 9:18-34. Executing the application's installation program to create layers meets *running an instance of* the *service* provided by the application. Bhattacharjee, [0380]-[0382] (citing EX1040, 1:14-25; EX1099, [0104]).

POSAs also had motivation to *create* layers by *running an instance of a service on a server*: e.g., running programs for an accounting service (Blaser, 13:1-23), which, as explained below (§V.D.10.c, [**14B**]) would identify the system files these programs required when executing, and to test the layers+applications to ensure their compatibility with various server hardware-OS combinations in the system. Bhattacharjee, [0383]-[0387] (citing EX1129, 1:11-20; EX1031, [0064]-[0068]; EX1020, [0007]).

c. [14B] "b) determining which files are being used; and,"

Blaser discloses that "[d]uring the install, all of the applications shared DLLs, registry entries, and .ini files that would be directed to the Windows system directories become trapped in the capture layer" and "[a]pplication files that would be placed on file systems managed by the OS are also redirected into the layer." Blaser, 9:18-34, 10:1-24. That teaches/suggests *determining which files are being used* by the application. Bhattacharjee, [0388].

As discussed above (§V.D.10.b, [**14A**]), POSAs also had additional motivation to create layers by first "running [*using*] an instance of a service on a server" to *determine* the system files *being used* (and thus needed) to execute the layer's application(s). Bhattacharjee, [0389]-[0391] (citing EX1077, 120-122 ("Sandboxing Applications," similar to creating Blaser-Calder's layers, by "[r]unning the application" to identify *files being used*)).

d. [14C] "c) copying applications and associated system files to memory without overwriting the associated system files so as to provide a second instance of the applications and associated system files."

Blaser's captured files and applications are "trapped in the capture layer" and "redirected into the layer" rather than "placed on file systems managed by the OS." Blaser, 9:18-34. "[T]he captured data is **held separate from** the regular OS" (e.g., "in a data file, hard disk partition, or some other container"). Blaser, 9:18-

34. As discussed above (§V.D.1.g, [**1E**]), each layer's system files are copies of system files that remain in the Base OS.

Thus, Blaser's capturing process includes *copying applications and associated system files to memory without overwriting the associated system files*. Bhattacharjee, [0392]-[0393].

Pre-existing or subsequent installation of an application *provides a second instance of the applications and associated system files*. *See* Blaser, 6:62-7:23 ("Different versions of an application may be stored as layers on a single computer."), 13:29-41 ("**multiple versions of a software product are installed on a computer**, each isolated in a layer"). Bhattacharjee, [0394]-[0397]. *See* EX1079, 67 (VirtaMove arguing claim met by storing "different instance[s]" of alleged *containers*).

11. Claim 16

a. [16A] "[C]laim 1…creating containers prior to said step of storing containers in memory,"

See §V.D.10.b ([**14A**]). Bhattacharjee, [0398].

b. [16B] "wherein a step of creating containers includes using a skeleton set of system files as a container starting point and installing applications into that set of files."

A "***skeleton***" file system is "the OS provided ***system files*** before an ***application*** is ***installed***." EX1001, 10:41-47.

As discussed above (§V.D.1.e, §V.D.10), Blaser "'capture[s]" (i.e., creates) layers (*containers*) by copying various *system files* that would otherwise "be placed on file systems managed by the OS." Blaser, 9:18-34. Bhattacharjee, [0399]-[0400].

Blaser seeks to avoid "application conflicts with the host operating system (OS) and other applications." Blaser, 1:14-31. Thus, POSAs had motivation to use *a skeleton set of system files* to serve *as a container starting point* when "capturing" a layer's applications so those *skeleton files* would be *installed into that set of files* to avoid creating conflicts with the base OS's local system files. Bhattacharjee, [0401]-[0404] (citing EX1077, 119-122).

12. Claim 17

a. [17A] "[C]laim 1…installing a service on a target server selected from one of the plurality of servers,"

Blaser-Calder's *plurality of servers* (*supra* §V.D.1.a.i) "*install*[, e.g.,] the accounting type layer" (with accounting-*service* applications) when "a computer is to be transferred from engineering to accounting." Blaser, 13:16-23. Thus, Blaser-Calder meets *installing a service on a target server*. Bhattacharjee, [0405].

POSAs also had motivation and reasonably expected success to *install services* so they were available to users (e.g., so the accounting-group server could provide accounting *services*), which was conventional, prior-art server

functionality. Bhattacharjee, [0406]-[0409] (citing EX1014, [0015]; EX1020, [0069]-[0078]).

b. [17B]: "…using a graphical user interface [GUI], associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server."

"For ease of configuring and managing a layering system," Blaser provides "a manager application may be provided" that "permits an administrator or user to control the presentation of applications and data on a system" by providing "facilities for importing and exporting layers" between systems. Blaser, 8:19-41. Using GUIs to configure servers was conventional and provided convenience (e.g., allowing drag-and-drop operations using icons), and Blaser's "layer manager application" "permit[s] control and configuration of the layer[]…through a management API and library." Blaser, 4:7-27. Thus, POSAs had motivation and reasonably expected success to use a GUI with Blaser's manager application (accessed via Blaser's API) to *associate a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server*. Bhattacharjee, [0410]-[0418] (citing EX1014, [0030]-[0031]; EX1020, [0024], [0046]; EX1047, 1:39-67; EX1109, [0002]-[0005]).

13. Claim 18: Claim 17's "target server and" GUI "are at remote locations."

POSAs had motivation and reasonably expected success to allow system administrators at remote locations (e.g., corporate/IT headquarters) to configure target servers using a GUI, rather than require administrators at each target server. Bhattacharjee, [0419]-[0423] (citing EX1014, [0030]; EX1020, [0024], [0046], [0108]; EX1109, [0002]-[0005]).

14. Claim 19: Claim 18's GUI "is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server."

The '814 patent says "[a] computer system with a single instance of a fully functional operating system installed is referred to as a ***computing platform***." EX1001, 2:20-22.

If *different computing platforms* include different physical computers, then Blaser-Calder meets claim 19 the same way it meets claim 18 (§V.D.18). Bhattacharjee, [0424]-[0425].

If claim 19 requires computers with different OSs and/or hardware, POSAs knew large organizations typically had multiple OS-families and/or OS-versions on their servers, which often had different hardware configurations. Bhattacharjee, [0426]-[0428] (citing EX1020, [0024], [0046]); *supra* §V.C (Blaser-Calder servers had different OSs).

15. Claim 20: "Claim 19," where [17B]'s "associating" step "includes…relatively moving the unique icon representing the service to the unique icon representing a server."

POSAs would have found it obvious to implement Blaser-Calder's associating of server with server icons (*supra* §V.D.12.b ([**17B**])) to include *relatively moving the unique icon representing the service to the unique icon representing a server* because that was a conventional, convenient, drag-and-drop function for GUIs. Bhattacharjee, [0429]-[0433] (citing EX1014, [0031]; EX1047, 1:39-67; EX1109, [0048]).

16. Claim 21: "[C]laim 20…starting a distributed software application."

Conventional *distributed software applications* are "spread over several computers that are connected via a network." EX1025, [0038]. Blaser's layers support *distributed software applications* because a layer can "span[] one or more fixed disks," including those on "network drives." Blaser, 7:36-46. Bhattacharjee, [0434]-[0435]. POSAs also had motivation to include a *distributed software application* because it was a common, beneficial implementation of applications. Calder's techniques "relate[] to ***distributed* computing**" ([0002]), with files "stored remotely **on separate machines**," where calls to a "kernel to perform [a] file operation" are "communicated over the network [] to another…server" ([0139]). Bhattacharjee, [0436]-[0440] (citing EX1025, [0038]; EX1028, [0005]; EX1014, [0046]).

17. Claim 22: "[C]laims 20…updating a console on the selected target server with information indicating that the service is resident on the selected target server."

"**Computing devices typically include a *console***" (e.g., "to control the computing device manually"). EX1022, [0004]. Thus, POSAs would have found it obvious for Blaser-Calder to *update a console on the selected target server with information indicating that the service is resident on the selected target server* to configure the server's console to recognize and properly execute its services. Bhattacharjee, [0441]-[0444] (citing EX1022, [0004]; EX1032, [0025], [0032]-[0033])

18. Claim 23: "[C]laim 17…testing to determine if the selected target server is a valid computing platform, prior to causing the applications to be distributed to, and installed on the target server."

POSAs would have found it obvious to *test to determine if the selected target server is a valid computing platform,* before *causing the applications to be distributed to, and installed on the target server*, to avoid wasting time and/or and potentially damaging Blaser-Calder servers by attempting to install incompatible layers. Bhattacharjee, [0445]-[0449] (citing EX1044, 2:25-32; EX1031, [0064]-[0068]; EX1020, [0007]).

19. Claim 24: "[C]laim 17…creating a user account for the service."

Blaser "provide[s] a multi-***user*** environment" that allows "an administrator to designate layers accessible to **individual *users*** and…to automatically **enable**

layers on ***user login***." Blaser, 7:59-67. Each user has a "user's login name." Blaser, 8:54-67. That teaches or suggests *creating a user account* (login name) *for the service* the user may access. Bhattacharjee, [0450]-[0453] (citing EX1032, [0041]-[0042]).

20. Claim 25: "[C]laim 17…installing files specific to the selected application on the selected server."

In Blaser, "***install[ing]***…**an *application***" (and *files specific* thereto) to *the selected server* "can be as simple as enabling…a containing layer." Blaser, 7:24-35. Bhattacharjee, [0454]-[0455].

21. Claim 26: "[C]laim 17…setting file access permissions to allow a user to access the one of the applications to be distributed."

As discussed above (§V.D.19), Blaser's "administrator [] designate[s] layers accessible to individual users" (Blaser, 7:59-67), which POSAs understood would conventionally include *setting the file access permissions to allow an individual user to access the one of the applications to be distributed* and installed in the layer. Bhattacharjee, [0456]-[0458] (citing EX1065, 1:33-56).

22. Claim 27

a. [27A] "[C]laim 1…de-installing a service from a server, comprising:"

Blaser provides for "***de-installation* of an application**" (and thus its corresponding *service*) by "disabling a containing layer" (Blaser, 7:24-35). *See*

Blaser, 13:16-23 ("**remov[ing]** the engineering type layer"). Bhattacharjee, [0459]-[0460].

b. [27B] "displaying the icon representing the service; [and] displaying the icon representing the server on which the service is installed; and"

As discussed above (§V.D.12.b), POSAs had reason to install a service by dragging and dropping an associated icon into an icon/folder representing the target server and thus would also *display* those icons to provide a user-friendly way to identify which services are installed on which servers. Bhattacharjee, [0461]-[0465] (citing EX1047, 1:39-67; EX1109, [0048]).

c. [27C] "utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected service from the server on which it was installed."

POSAs knew a conventional administrator feature was using icons as a user-friendly way to de-install services from target servers. Bhattacharjee, [0466]-[0470] (citing EX1095, 8:4-13; EX1109, [0048]).

23. Claim 28: "[C]laim 27…separating icon representing the service from the icon representing the server."

As discussed above (§§V.D.2.a-b), POSAs would want distinct *icons* for de-installing services and thus would also *separate* those *icons* as a conventional, user-friendly, GUI-based way to de-install services from Blaser-Calder's servers. Bhattacharjee, [0471]-[0473] (citing EX1109, [0048]).

24. Claim 29: "[C]laim 27…testing whether the selected server is a valid computing platform for de-installation of the service."

As discussed above (§V.D.18), POSAs would *test whether the selected server is a valid platform for installing the service* and thus would also *test whether the selected server is a valid computing platform for de-installation of the service* to prevent corrupting the server by improper de-installation. Bhattacharjee, [0474]-[0475].

25. Claim 30: "[C]laim 27…copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation."

Blaser says it was known to "record[] a series of changes (or 'diffs') to a buffer" so "a system can be restored back to a point in time." Blaser, 2:12-25. Blaser similarly says that as layers use application(s) "it may be desired to record changes to the virtual file system into the writable portion of a layer." Blaser, 10:50-59. Blaser further discloses that a user's "***changes* are recorded to that user's layer**" so the layer can be "**retained for future use**." Blaser, 12:21-35. For example, "**a user layer may be *stored* to a network server and retrieved and *stored* as required**." Blaser, 12:36-44. The above disclosures teach/suggest *copying data file changes specific to the service*. Bhattacharjee, [0476]-[0477].

POSAs would have found it obvious to store those changes *back to a storage medium from which the data file changes originated prior to installation* to

allow any data file changes to be retrievable for later user. Bhattacharjee, [0478]-[0479].

VI. GROUND 2: CLAIMS 5-6, 12-13, 15, AND 31-34 ARE UNPATATENTBALE OVER BLASER-CALDER-SCHMIDT

A. Schmidt-449 (EX1007)

Schmidt-449 says "[m]oving (or migrating) [] programs between servers is desirable" (e.g., when a "remote computer" "becomes busy or is off-line for repair or upgrades" or to access a "different machine")). Schmidt-449, 2:15-28. However, when a "user migrates to another machine the IP address changes," so "packets received prior to migration [are] lost" and "packets remaining in the transfer will not reach the user." Schmidt-449, 2:39-54. Bhattacharjee, [0480]-[0481].

Schmidt-449's solution is to provide software "capsules" with "a unique locator, such as an IP address" that allows the capsule (and the applications it contains) to "be moved to a different machine having potentially a different operating system or on [a] different network," while "maintain[ing] the open network connections [the capsule] had" before migrating. Schmidt-449, 2:57-65. Each capsule "comprises one or more processes and their associated system environment," including "state information relating to exactly what the processes are doing," including system files like "configuration settings, working directories

and files…[and] installed software." Schmidt-449, 4:14-26. Bhattacharjee, [0482].

B. Blaser-Calder-Schmidt

POSAs would have been motivated and reasonably expected success to assign unique capsule IDs, as taught by Schmidt-449, to Blaser-Calder layers.

Blaser's layers can be "accessible to all users or some users" or "only to an individual user" (Blaser, 7:59-67) and "defined for different types of users, for example, secretaries, engineers and accountants" (Blaser, 13:1-15). One Blaser server can have "multiple layers…active at the same time." Blaser, 15:27-36. Bhattacharjee, [0483]-[0484]. Additionally, Blaser provides layers for "software product testers" that "verify software functionality against multiple development versions." Blaser, 13:29-41. Like Schmidt-449 moving capsules between computers (§VI.A), Blaser moves layers between computers (6:62-7:23).

Schmidt-449 assigns each application capsule "a unique locator" so it "may be moved to a different machine having potentially a different operating system or on different network" (and thus a different server) "and maintain the open network connections it had prior to the migration." Schmidt-449 2:57-65. Bhattacharjee, [0484]-[0485].

Thus, POSAs had motivation to assign Blaser-Calder's layers unique locators like Schmidt-449 assigns to capsules to beneficially facilitate

(i) exchanging data between applications and/or users isolated in different layers, (ii) moving layers to different machines (including with different OSs) and/or different networks, and/or (iii) testing/running software that beneficially provided multiple, unique IP addresses on one server (e.g., so different users/companies could have their respective websites, with different IP addresses, hosted by a shared physical server). Bhattacharjee, [0486].

In the resulting "Blaser-Calder-Schmidt" combination, each layer is assigned one or more unique locators (e.g., IP address). POSAs would have reasonably expected success because unique locators were conventional, Schmidt-449's capsules (like applications in Blaser-Schmidt's layers) are OS-independent, and Schmidt-449 says its techniques are "implement[able] in any type of computer system or programming or processing environment" (11:43-46). Bhattacharjee, [0487]-[0488].

C. Mapping to Challenged Claims

1. Claim 5: "[C]laim 2…modifying at least some of the associated system files in plural containers to provide an association with a container specific identity assigned to a particular container."

Blaser-Calder-Schmidt's layers (*supra* §VI.B) have *a container specific identity assigned to [that] particular container* (e.g., an IP address). Schmidt-449 2:57-65. POSAs had reason to *modify at least some of the associated system files in plural containers to provide an association* with the unique locators so a layer's

application could identify/communicate with an application isolated in another layer (e.g., to exchange data over IP-based networks). Bhattacharjee, [0489]-[0492].

2. Claim 6: "[C]laim 2…assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address."

Blaser-Calder-Schmidt (*supra* §VI.B) *assigns a unique associated identity to each of a plurality of the containers* (layers) like Schmidt's exemplary "IP address" (Schmidt-449, 2:57-65). POSAs would have been motivated and reasonably expected success to also assign additional conventional unique locators – like a "host name" and "MAC address" – to allow communication protocols that used those identities to address the layer (e.g., MAC address for Ethernet networks; hostname for remote access). Bhattacharjee, [0493]-[0499] (citing EX1039, [0002] ("**hostname is a unique name by which a computing device may be identified** on a network…[and] simplify access to computing devices"); EX1019, [0012], [0023] ("**assign[ing] virtual MAC addresses to each processor node**" so "processing resources may be deployed rapidly and easily through software via configuration commands"); EX1130, 2:6-14; EX1131, 1:27-56).

3. Claim 12

As noted above (§V.D.6, claim 8), Blaser-Calder's layer (*container*) files can be retrieved from storage of a "network server." Blaser, 12:36-44. The retrieved

layer "provide[s] files and directories **overlaid on** the existing fixed disk file systems." Blaser, 12:36-44. POSAs understood that overlaying is *merging* those *files stored in network storage* with that existing file system. POSAs also had motivation to further merge those *files with the parameters* discussed above for claim 11 (§V.D.9), *to affect* claim 1's step of storing (§V.D.1.c.i). For example, it would have been obvious to merge layer *files* with the names of those layers and locations of where to store those layers on the network, which are examples of *parameters forming descriptors of containers* (§V.D.9). Bhattacharjee, [0500]-[0503].

4. Claim 13

As discussed above (§V.C), Blaser-Calder's exemplary OSs include prior art Windows-based and conventional non-Windows OSs; those OS included tools for measuring/tracking processes related to an application, including tracking statistics for resource allocation and monitoring the status of application(s). Bhattacharjee, [0504]-[0510] (citing, for Windows-based OSs, EX1033, 2:62-3:18, 34:30-39, 36:53-65; EX1055, 5:35-39; for Unix-based OSs, EX1060, 5:57-61; EX1094, 2:5-21, 4:20-30; EX1120, 450; for generic OSs, EX1121, G-14).

Thus, POSAs had motivation (with reasonable expectation of success) to use those (or other conventional tracking) tools in Blaser-Calder-Schmidt to track and store the history of processes in layers for the same reasons as in conventional

computer systems (e.g., projecting server-resource needs for purchasing/assigning additional equipment). Bhattacharjee, [0511]-[0514] (citing EX1093, 1:66-3:6; EX1121, G-5).

5. Claim 15

a. [15A] "[C]laim 14…assigning an identity to the containers including at least one of a unique IP address, a unique Mac address and an estimated resource allocation;"

See §V.D.10 (claim 14), §V.E.2 (claim 6). Bhattacharjee, [0515].

b. [15B] "installing the container on a server; and,"

Blaser discloses "'pre-installed' applications" provided as "layers on CD-ROM or other media" (Blaser, 6:62-7:23), which are for *installing the container* (layer) *on a* Blaser-Calder *server* (*supra* §V.A.1.a.i). Bhattacharjee, [0516].

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

Blaser's "'pre-installed' applications" on CD/media are "**pre-*tested***." Blaser, 6:62-7:23. For example, "software product ***testers***" use layers to "verify software functionality against multiple development versions." Blaser, 13:29-41. Thus, Blaser meets *testing the applications and files within the* layer (*container*). Bhattacharjee, [0517]-[0520] (citing EX1129, 1:11-20).

6. Claim 31

a. Previously Addressed Limitations

Blaser-Calder-Schmidt meets the following limitations of claim 31 for the reasons discussed in the corresponding section(s) above. Bhattacharjee, [0521]-[0537].

Limitation:	**Addressed in Section:**
[**31PRE**]	§§V.D.1.a-b ([**1PREA**]-[**1PREB**]), §V.D.1.c.ii ([**1A**])
[**31A**]	§V.D.1.c ([**1A**])
[**31B**]	§V.D.1.c.i.(2) ([**1A**], "container"-construction part [**2**])
[**31C**]	§VI.C.2 (claim 6)
[**31D**]	§V.D.1.c.ii ([**1A**])
[**31E**]	§V.D.1.c ([**1A**]), §§V.D.1.f-g ([**1D**]-[**1E**])
[**31F**]	§V.D.1.c ([**1A**]), §V.D.2 (claim 2)
[**31G**]	§V.D.1.c.iv ([**1A**])
[**31H**]	§V.D.1.c.iv ([**1A**]), §V.D.1.e ([**1C**])

b. [31I] "a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications."

Blaser says for Windows-based OSs, "a layering system is formed by adding several files to the stock Windows operating system," "including a **runtime library** FSLLIB32.DLL" "and an FSLX driver." Blaser, 13:43-57. "[T]he FSLLIB32.DLL runtime library provides an API…to manage the layered system and communicate with the FSLX driver, and further provides system management function implementations." Blaser, 20:19-36. The FSLX driver "operate[s] to intercept key file system and registry calls and manipulate the results to create the appearance that virtual files and registry settings contained in the layer definitions exist in the real file system and real registry." Blaser, 14:60-15:1. For example, "[w]hen requests come that access virtual files or virtual registry settings, these requests are redirected by the FSLX driver to the proper locations in the layer." Blaser, 14:60-15:1. Bhattacharjee, [0538]-[0539].

Blaser's FSLLIB32.DLL "runtime library" is thus a *run time module for monitoring system calls from applications associated one or more containers* (e.g., by "intercept[ing] key file system and registry calls") and are *for providing control of the one or more applications* (e.g., by "redirect[ing] [access requests] to the proper locations in the layer"). Bhattacharjee, [0540]-[0541].

Moreover, Blaser-Calder-Schmidt incorporates Calder's translating of *system calls* for use on non-native OSs. Calder, [0101]; *supra* §V.C. Calder says run-time software module(s) (e.g., an "interception ***module***") will *monitor system calls from the applications* to translate them, and thereby *provide control of the one or more applications*. Calder, [0086]-[0087]. That is another reason Blaser-Calder-Schmidt meets [**31I**]. Bhattacharjee, [0452]-[0544].

7. Claim 32

As discussed above (§V.D.7, claim 9), POSAs had reason to pre-allocate system resources to layers and thus had motivation to include *a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources*, as a conventional way to assign server resources to applications. Bhattacharjee, [0545]-[0549] (citing EX1120, 452; EX1034, [0003]; EX1094, 2:5-21).

8. Claim 33: "[C]laim 32['s]…run time module includes an intercepting module associated with the…containers for intercepting system calls from any of the…containers and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of" [31C]'s unique identifiers.

As discussed above (§VI.C.7, claim 32), Blaser-Calder-Schmidt meets a "*run time module*" that "*intercept[s] system calls*" from reaching the host's kernel and redirects them into the layer. Blaser, 14:60-15:1; Calder, [0139]. As discussed

above (§VI.C.2, claim 6), POSAs would assign each layer *at least one of the IP address, the host name and the Mac_Address*. POSAs knew a conventional way to implement those unique identifiers was for the *run time module* that *intercepts system calls from any of the plurality of containers* to replace the host's identifier with the layer's identifier (or vice-versa) to ensure the layers *can run independently of one another without contention* as Blaser teaches. *Supra* §V.D.1.c.i.(3). For example, if a layer made a system call to request the server's IP address, Blaser-Calder-Schmidt's run-time module would provide the layer's unique IP address. Bhattacharjee, [0550]-[0552].

9. Claim 34

a. [34A]: "[C]laim 31['s]…run time module performs: monitoring resource usage of applications executing;"

See §V.D.7 (claim 9), §VI.C.4 (claim 13). Bhattacharjee, [0553].

b. [34B]: "intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;"

POSAs knew that applications typically run in *user mode* and make *system calls to kernel mode from user mode* (e.g., to access the server's hardware). Bhattacharjee, [0554] (citing EX1120, 15). As discussed above (§VII.A.4.a.i.(4)), applications in Blaser-Calder-Schmidt's layers share a *kernel*, and *system calls* are *intercepted* and redirected into the layer (e.g., if a requested system file is within

the layer) or passed to the *kernel* (e.g., if server hardware is required). Blaser, 14:60-15:1; Calder, [0074]-[0076]. Bhattacharjee, [0555]-[0557].

c. [34C]: "comparing the monitored resource usage…with the resource limits; and, forwarding the system calls to a kernel on the basis of the comparison…."

POSAs would have found it obvious to *compare the monitored resource usage of the at least one respective application with the resource limits* discussed *supra* §VI.C.7 (claim 32) and only *forward the system calls to a kernel on the basis of the comparison* so that allocated resource limits would not be violated if the system call were passed to the kernel. Bhattacharjee, [0558]-[0559] (citing EX1132, [0009]-[0011]).

VII. DISCRETIONARY DENIAL IS UNWARRANTED

A. §314(a)

"[T]he PTAB will not deny institution based on *Fintiv* if," as here, "there is compelling evidence of unpatentability." Director's Interim Procedure for Discretionary Denials, 5 (June 21, 2022) ("Interim Procedure") (discussing *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential)). That should conclude the *Fintiv* analysis.

If considered, the *Fintiv* factors weigh against denial.

1. Stay Potential

This factor is currently neutral or favors institution. *Sand Revolution II, LLC v. Cont'l Intermodal Group–Trucking*, IPR2019-01393, Paper 24, 7 (June 16, 2020) (informative). The district-court proceeding is currently temporarily stayed pending transfer. EX1087, 4-5.

2. Trial Timing

Petitioner's motion to transfer the district-court proceeding to the Northern District of California was recently granted. EX1087. That district's median time-to-trial is 47.9 months (EX1082, 66), and no post-transfer trial schedule has yet been set. Factor 2 therefore weighs against denial. *BMW of North America, LLC v. Michigan Motor Techs., LLC*, IPR2023-01224, Paper 15, 11 (Feb. 15, 2024).

3. Litigation Investment

There has been little investment in invalidity-related issues in the district court litigation. A *Markman* hearing was canceled before the transfer decision EX1087, and "a significant portion of work remains to be done" including "fact and expert discovery and dispositive motions." *Protect Animals With Satellites LLC v. OnPoint Sys., LLC*, IPR2021-01483, Paper 11, 14-15 (Mar. 4, 2022).

4. Issue Overlap

This Petition challenges all 34 claims, whereas only eight are asserted in district court. EX1079. This weighs against denial. *Markforged Inc. v. Continuous Composites Inc.*, IPR2022-00679, Paper 7, 32-33 (Oct. 25, 2022).

5. Litigation Defendants

This factor is at worst neutral. *Id.*, 33.

6. Other Circumstances

The Petition's "strong showing on the merits" weighs against denial. Interim Procedure, 3-5.

B. §325(d)

Blaser, Calder, and Schmidt-449 were not before the Office during prosecution, so denial is unwarranted.

VIII. CONCLUSION

The Board should institute review and cancel claims 1-34.

Dated: 2025-01-31

Respectfully submitted,
Google LLC

/Elisabeth Hunt/_____
Elisabeth Hunt, Reg. #67,336

IX. APPENDIX: CLAIM LISTING

The following claim listing assigns element labels (*e.g.*, [**1PREA**], [**1PREB**], [**1A**], etc.) to certain claims for convenience of reference.

Claim 1
[**1PREA**] In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor,
[**1PREB**] a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications are executed in a secure environment, 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, the method comprising:
[**1A**] storing in memory accessible to at least some of the servers 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;
[**1B**] wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems,
[**1C**] the containers of application software excluding a kernel,
[**1D**] 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,
[**1E**] 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,
[**1F**] and wherein the application software cannot be shared between the plurality of secure containers of application software,
[**1G**] and wherein each of the containers has a unique root file system that is different from an operating system's root file system.

Claim 2
A method as defined in claim 1, wherein each container has an execution file associated therewith for starting the one or more applications.
Claim 3
A method as defined in claim 2, wherein the execution file includes instructions related to an order in which executable applications within will be executed.
Claim 4
A method as defined in claim 1 further comprising the step of pre-identifying applications and system files required for association with the one or more containers prior to said storing step.
Claim 5
A method as defined in claim 2, further comprising the step of modifying at least some of the associated system files in plural containers to provide an association with a container specific identity assigned to a particular container.
Claim 6
A method as defined in claim 2, comprising the step of assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address.
Claim 7
A method as defined in claim 2 further comprising the step of modifying at least some of the system files to define container specific mount points associated with the container.
Claim 8
A method as defined in claim 1, wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers.
Claim 9
A method as defined in claim 2, 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 10
A method as defined in claim 2, 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 11
A method as defined in claim 2, wherein containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location.
Claim 12
A method as defined in claim 11, further comprising the step of merging the files stored in network storage with the parameters to affect the step of storing in claim 1.
Claim 13
A method as defined in claim 1 further comprising the step of associating with a plurality of containers a stored history of when processes related to applications within the container are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application.
Claim 14
[**14PRE**] A method as defined in claim 1 comprising the step of creating containers prior to said step of storing containers in memory,
[**14A**] wherein containers are created by: a) running an instance of a service on a server;
[**14B**] b) determining which files are being used; and,
[**14C**] c) copying applications and associated system files to memory without overwriting the associated system files so as to provide a second instance of the applications and associated system files.
Claim 15
A method as defined in claim 14 comprising the steps of: assigning an identity to the containers including at least one of a unique IP address, a unique Mac address and an estimated resource allocation; installing the container on a server; and, testing the applications and files within the container.

Claim 16
[**16A**] A method as defined in claim 1 comprising the step of creating containers prior to said step of storing containers in memory,
[**16B**] wherein a step of creating containers includes: using a skeleton set of system files as a container starting point and installing applications into that set of files.
Claim 17
[**17A**] A method as defined in claim 1 further comprising installing a service on a target server selected from one of the plurality of servers,
[**17B**] wherein installing the service includes: using a graphical user interface, associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server.
Claim 18
A method as defined in claim 17 wherein the target server and the graphical user interface are at remote locations.
Claim 19
A method as defined in claim 18, wherein the graphical user interface is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server.
Claim 20
A method as defined in claim 19, wherein the step of associating includes the step of relatively moving the unique icon representing the service to the unique icon representing a server.
Claim 21
A method as defined in claim 20 further comprising starting a distributed software application.
Claim 22
A method according to anyone of claims 20 further comprising updating a console on the selected target server with information indicating that the service is resident on the selected target server.

Claim 23
A method claim 17, further comprising, the step of testing to determine if the selected target server is a valid computing platform, prior to causing the applications to be distributed to, and installed on the target server.
Claim 24
A method according to claim 17 further comprising creating a user account for the service.
Claim 25
A method as defined in claim 17, further comprising the step of installing files specific to the selected application on the selected server.
Claim 26
A method according to claim 17 further comprising the steps of setting file access permissions to allow a user to access the one of the applications to be distributed.
Claim 27
[**27A**] A method as defined in claim 1, further comprising de-installing a service from a server, comprising: displaying the icon representing the service;
[**27B**] displaying the icon representing the server on which the service is installed; and
[**27C**] utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected service from the server on which it was installed.
Claim 28
A method according to claim 27 further comprising separating icon representing the service from the icon representing the server.
Claim 29
A method according to claim 27 further comprising testing whether the selected server is a valid computing platform for de-installation of the service.
Claim 30
A method according to claim 27 further comprising copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation.

Claim 31
[**31PRE**] A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:
[**31A**] a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers,
[**31B**] each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers,
[**31C**] each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a MAC_address;
[**31D**] wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes,
[**31E**] and associated system files for use in executing the one or more processes wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system,
[**31F**] each container having its own execution file associated therewith for starting one or more applications,
[**31F**] each container having its own execution file associated therewith for starting one or more applications,
[**31H**] wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and,
[**31I**] a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.
Claim 32
A computing system as defined in claim 31, further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.

Claim 33
A computing system as defined in claim 32, wherein the run time module includes an intercepting module associated with the plurality of containers for intercepting system calls from any of the plurality of containers and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the IP address, the host name and the MAC_Address.
Claim 34
[**34A**] A computing system as defined in claim 31, wherein the run time module performs: monitoring resource usage of applications executing;
[**34B**] intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;
[**34C**] comparing the monitored resource usage of the at least one respective application with the resource limits; and, forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

CERTIFICATE OF SERVICE UNDER 37 C.F.R. § 42.6 (E)(4)

I certify that on January 31, 2025, I will cause a copy of the foregoing document, including any exhibits or appendices filed therewith, to be served via USPS Priority Mail Express at the following correspondence address of record for the patent:

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

Date: January 31, 2025

/Dara Del Rosario/
Dara Del Rosario
Paralegal
WOLF, GREENFIELD & SACKS, P.C.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 13,991 words excluding a table of contents, a table of authorities, Mandatory Notices under § 42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: January 31, 2025

/Dara Del Rosario/
Dara Del Rosario
Paralegal
WOLF, GREENFIELD & SACKS, P.C.