

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. IPR2025-00488
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.....	xv
A. Real Party-In-Interest	xv
B. Related Matters.....	xv
1. United States Patent & Trademark Office	xv
1. USPTO Patent Trial and Appeal Board	xvi
2. U.S. District Court for the Eastern District of Texas	xvi
3. U.S. District Court for the Western District of Texas.....	xvi
4. U.S. District Court for the Northern District of California.....	xvii
C. Counsel and Service Information - § 42.8(b)(3) and (4).....	xvii
I. STANDING.....	1
II. GROUNDS	1
III. THE '814 PATENT	2
A. Specification	2
B. Person of Ordinary Skill in the Art (“POSA”).....	4
C. Prosecution History	4
D. Challenged Claims.....	4
IV. CLAIM INTERPRETATION	5
V. Grounds 1-2: Claims 1-34 ARE UNPATENTABLE OVER Schmidt-Tormasov (GROUND 1) OR OVER SCHMIDT-TORMASOV-Calder (GROUND 2).....	5
A. Schmidt-479 (EX1008)	5
B. Tormasov (EX1010).....	7
C. Calder (EX1006).....	10
D. Grounds 1-2	12
1. Schmidt-Tormasov Combination (Ground 1)	12
2. Schmidt-Tormasov-Calder Combination (Ground 2).....	17
E. Mapping to Challenged Claims	19
1. Claim 1	19

a.	[1PREA].....	19
i.	“1. In a system having a plurality of servers with operating systems that differ”	19
ii.	“operating in disparate computing environments”	23
iii.	“wherein each server includes a processor and an operating system”	24
iv.	“[OS] including a kernel [and] a set of associated local system files compatible with the processor”	27
b.	[1PREB].....	32
i.	“a method of providing at least some of the servers in the system with secure, executable, applications”	32
ii.	“[applications] related to a service”	35
iii.	“wherein the applications are executed in a secure environment”	37
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”	38
c.	[1A]	40
i.	“storing in memory accessible to at least some of the servers a plurality of secure containers of application software”	40
	(1) “storing in memory accessible to...the servers”	40
	(2) “containers”	41
	(3) “secure containers of application software”	45
ii.	“each container comprising one or more of the executable applications”	48
iii.	“and a set of associated system files required to execute the one or more applications”	50
iv.	“for use with a local kernel residing permanently on one of the servers”	51
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,”	53

e. [1C] “the containers of application software excluding a kernel,”	54
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,”	56
g. [1E] “wherein said associated system files...are copies or modified copies of the associated local system files...,”	60
h. [1F] “and wherein the application software cannot be shared between the plurality of secure containers of application software,”	62
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.”	63
2. Claim 2: “[C]laim 1...each container has an execution file associated therewith for starting the...applications.”	67
3. Claim 3	68
4. Claim 4: “[C]laim 1...pre-identifying applications and system files required for association with the one or more containers” before [1A]’s “storing step.”	69
5. Claim 5	70
6. 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.”	70
7. Claim 7	72
8. Claim 8	73
9. 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.”	74
10. 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.”	75
11. Claim 11	77
12. Claim 12	77
13. Claim 13: “[C]laim 1...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.”	78
14. Claim 14	79
a. [14PRE] “[C]laim 1...creating containers prior to said step of storing containers in memory”	79
b. [14A] “wherein containers are created by: a) running an instance of a service on a server;”	80
c. [14B] “b) determining which files are being used; and,”	81
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.”	82
15. Claim 15	82
a. [15A]	82
b. [15B]	82
c. [15C]	83
16. Claim 16	83
a. [16A]	83
b. [16B]	83
17. Claim 17	83
a. [17A]	83
b. [17B]	84
18. Claim 18	84
19. Claim 19	85
20. Claim 20	85

21. Claim 21	85
22. Claim 22	86
23. Claim 23	86
24. Claim 24	86
25. Claim 25	87
26. Claim 26	87
27. Claim 27	87
a. [27A]	87
b. [27B]	87
c. [27C]	88
28. Claim 28	88
29. Claim 29	88
30. Claim 30	88
31. Claim 31	89
a. Previously Addressed Limitations	89
b. [31I] “a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the...applications.”	90
32. Claim 32	91
33. Claim 33	91
34. Claim 34	92
a. [34A]	92
b. [34B]	92
c. [34C]	92
VI. Grounds 3-4: Claims 1-34 Are Unpatentable Over the Combinations OF GROUNDS 1-2 In Further View of Schmidt-629.....	93
1. Schmidt-629 (EX1116) and the Grounds 3-4 Combinations.....	93
2. Claims 1-4, 7-30, 32-34.....	94
3. Claims 5-6, 31	94
VII. DISCRETIONARY DENIAL IS UNWARRANTED	94

A. §314(a).....	94
1. Stay Potential.....	95
2. Trial Timing	95
3. Litigation Investment	95
4. Issue Overlap.....	96
5. Litigation Defendants.....	96
6. Other Circumstances	96
B. §325(d).....	96
VIII. CONCLUSION.....	96
IX. APPENDIX: CLAIM LISTING.....	97

TABLE OF AUTHORITIES

CASES

<i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, Paper 11 (Mar. 20, 2020)	94, 95
<i>BMW of North America, LLC v. Michigan Motor Techs., LLC</i> , IPR2023-01224, Paper 15, 11 (Feb. 15, 2024)	95
<i>Google LLC v. Security First Innovations, LLC</i> , IPR2024-00215, Paper 15, 6 (May 23, 2024)	5
<i>Markforged Inc. v. Continuous Composites Inc.</i> , IPR2022-00679, Paper 7 (Oct. 25, 2022)	96
<i>PLR Worldwide Sales Ltd. v. Flip Phone Games, Inc.</i> , IPR2024-00209, Paper 9 (May 10, 2024)	23, 65
<i>Protect Animals With Satellites LLC v. OnPoint Sys., LLC</i> , IPR2021-01483, Paper 11 (Mar. 4, 2022)	95
<i>Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking</i> , IPR2019-01393, Paper 24 (June 16, 2020)	95

REGULATIONS

37 C.F.R. §42.100(b)	5
37 C.F.R. §42.104(a)	1

STATUTES

35 U.S.C. §102(b)	1, 2
35 U.S.C. §102(e)	2
35 U.S.C. §103	1
35 U.S.C. §282(b)	5
35 U.S.C. §314(a)	94
35 U.S.C. §325(d)	96

OTHER AUTHORITIES

Director’s Interim Procedure for Discretionary Denials (June 21, 2022) 95, 96

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 <i>VirtaMove, Corp. v. Google LLC</i> , 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 <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sept. 06, 2024)
1080	First Amended Complaint for Patent Infringement Against Google LLC in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (May 21, 2024)
1081	Scheduling Order in <i>VirtaMove, Corp. v. Google LLC</i> , 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 (June 30, 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 <i>VirtaMove, Corp. v. Google LLC</i> , 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 <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1101	Plaintiff’s Disclosure of Proposed Claim Constructions in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1102	Google LLC’s Opening Claim Construction Brief in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 24, 2024)
1103	Plaintiff’s Responsive Claim Construction Brief in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 12, 2024)
1104	Google LLC’s Reply Claim Construction Brief in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 26, 2024)
1105	Plaintiff’s Sur-Reply Claim Construction Brief in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 13, 2024)
1106	Joint Claim Construction Statement in <i>VirtaMove, Corp. v. Google LLC</i> , 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.

1. USPTO Patent Trial and Appeal Board

Concurrently with the present petition, Petitioner is filing IPR2025-00487, 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.

2. 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.

3. 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.

4. 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	<u>E-mail</u> : EHunt-PTAB@wolfgreenfield.com GNieberg-PTAB@wolfgreenfield.com ASaraswat-PTAB@wolfgreenfield.com <u>Post and hand delivery</u> : Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206 <u>Telephone</u> : 617-646-8000 <u>Facsimile</u> : 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 as to the challenged claims. 37 C.F.R. §42.104(a).

II. GROUNDS

Ground Number and Reference(s)		Claims	Basis
1	Schmidt-479+Tormasov	1-34	§103
2	Schmidt-479+Tormasov+Calder	1-34	§103
3	Schmidt-479+Tormasov+Schmidt-629	1-34	§103
4	Schmidt-479+Tormasov+Calder+Schmidt-629	1-34	§103

The above references were not of record during prosecution of the '814 patent.

Schmidt-479, Tormasov, and Calder are prior art at least under pre-AIA §102(b) to the '814 patent's earliest claimed priority date.

The '814 patent was filed 2004-09-13, and claims priority to two provisional applications: No. 60/502,619 ("'619 Provisional," EX1107), filed 2003-09-15; and No. 60/512,103, filed 2003-10-20. However, the '814 patent's claims are not entitled to the '619 Provisional's filing date because they lack written description support in the '619 Provisional. For example, limitation [1A]'s "container" (*see*

infra §VIII (claim listing)) is missing from EX1107. EX1003 (“Bhattacharjee”), [0029]-[0035], [0047]-[0048], [0001]-[0014]. Thus, Schmidt-629’s 2002-09-26 publication date makes it §102(b) prior art. Even if the ’814 patent were entitled to the 2003-09-15 filing date, Schmidt-629 would still be prior art under §102(e).

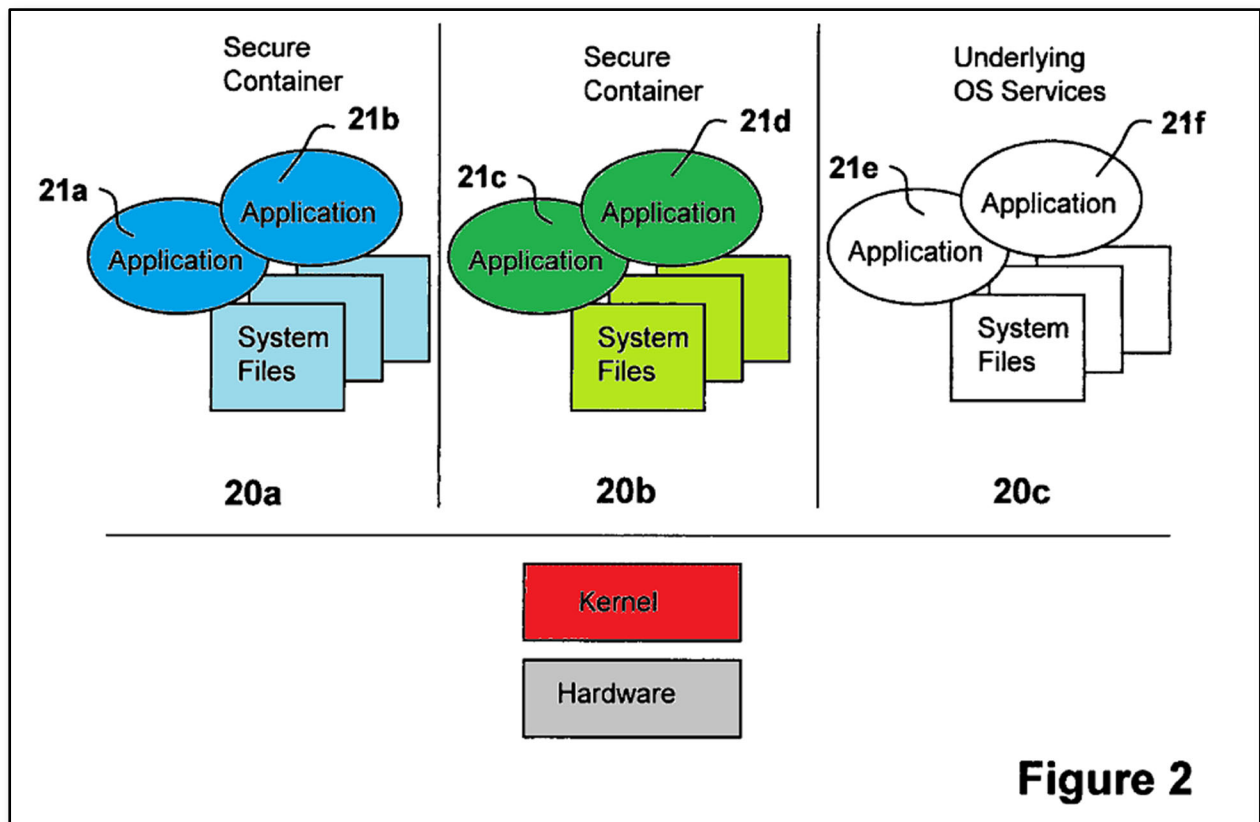
III. THE ’814 PATENT

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, [0036]-[0037].

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 runs 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 running 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, [0038].



Containers share the computer's hardware (grey) and OS kernel (red) (which controls access to the computer's resources) (FIG. 2, 1:56-61) can have their own unique identifier (e.g., IP address) (3:55-57). Bhattacharjee, [0039].

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, [0040]-[0042].

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* §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, [0043].

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. Bhattacharjee, [0044]-[0046].

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.

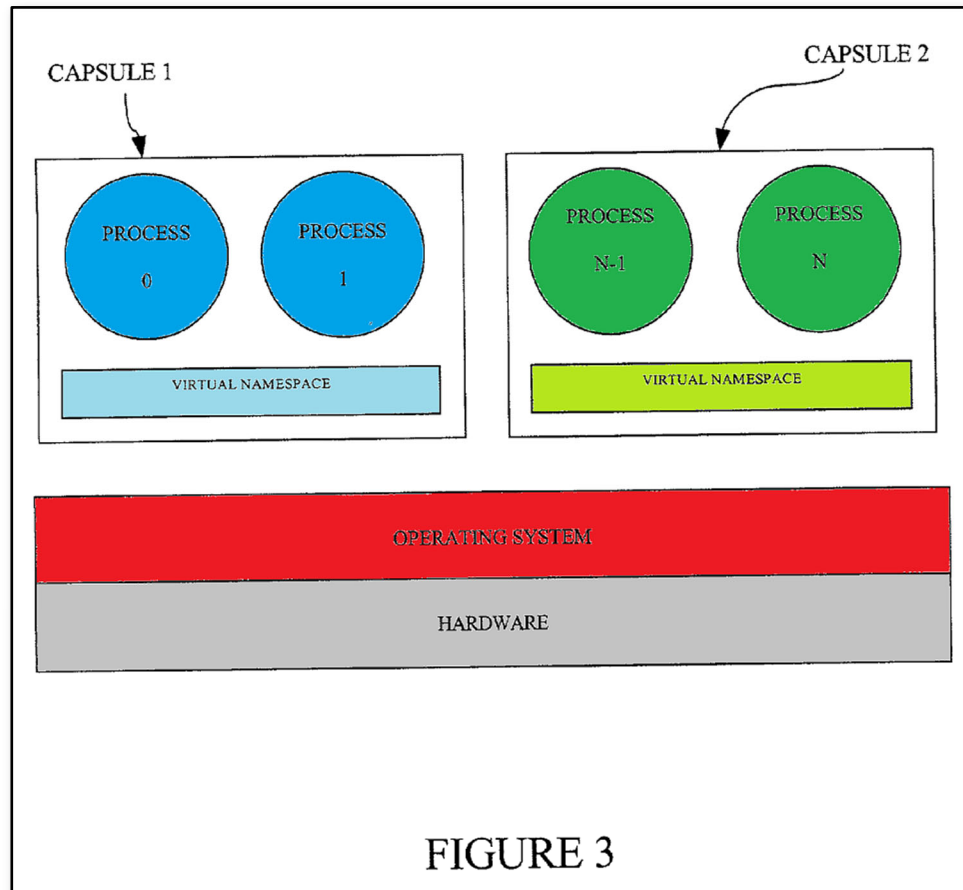
§42.100(b). In the concurrent district-court proceeding, Petitioner "Google" and Patent Owner "VirtaMove" have briefed competing constructions of various claim terms. EX1100-EX1101. As discussed further within the Grounds, this Petition presents alternative mappings of the prior art under both parties' 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, [0049]-[0065].

V. GROUNDS 1-2: CLAIMS 1-34 ARE UNPATENTABLE OVER SCHMIDT-TORMASOV (GROUND 1) OR OVER SCHMIDT-TORMASOV-CALDER (GROUND 2)

A. Schmidt-479 (EX1008)

Schmidt-479 discloses software-based "compute capsules" that each "encapsulate an active computing environment" on a server so the environment can be stored and subsequently started/restarted on a server. Schmidt-479, [0016]. Schmidt-479's Figure 3 (highlighted below) illustrates a "server computer"

([0065]) with two capsules (blue and green), “each hav[ing] two processes”² and respective “virtual namespace[s]” ([0044]). Bhattacharjee, [0066]-[0068].



Each virtual namespace has its own “file system view” ([0045]), which can be “automatically populated with system directories and files necessary for normal user operation” ([0048]). Those “files include...application binaries, libraries...and data files.” Schmidt-479, [0048]. Thus, each capsule “provides a

² Conventional “processes” are instances of executing computer applications.

EX1061, 1:13-16; Bhattacharjee, [0067].

private, customizable view of a shared file system, so that users can modify arbitrary files” within their own virtual namespace, “similar to the way a user might alter the file system on a dedicated computer,” but without changing “the underlying file[s] remain[ing]” in the underlying shared file system. Schmidt-479, [0017], [0019]. Bhattacharjee, [0069]-[0071].

Each “virtual namespace is host-independent,” which “enables the capsule to migrate between binary compatible machines and provides privacy and isolation between multiple capsules.” Schmidt-479, [0018].³ Capsules can be “migrated to a different instance of an operating system on a different machine.” Schmidt-479, [0044]. Bhattacharjee, [0072]-[0073].

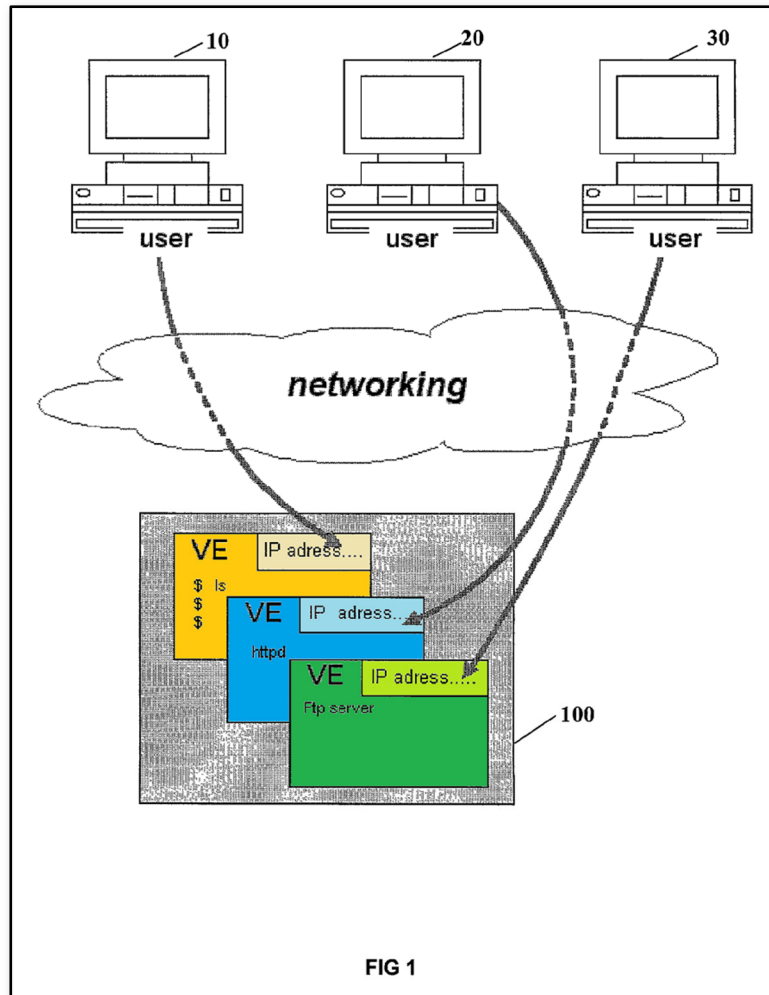
B. Tormasov (EX1010)

Tormasov provides a software-based technique to “efficient[ly] utilize[e]” “a single hardware system with a single operating system kernel” by providing each “end user” of “a server” with their own “virtual computing environment [VCE] that is functionally equivalent to a computer with a full-featured operating system.” Tormasov, [0017], [0024]. Like the ’814 patent (EX1001, 1:51-2:3), Tormasov

³ POSAs knew that “[b]inary compatible machines” include machines running different OSs from a binary-compatible family (e.g., different versions of Linux-based OSs). Bhattacharjee, [0072] (citing EX1127, 7:46-54; EX1128, 3:14-38).

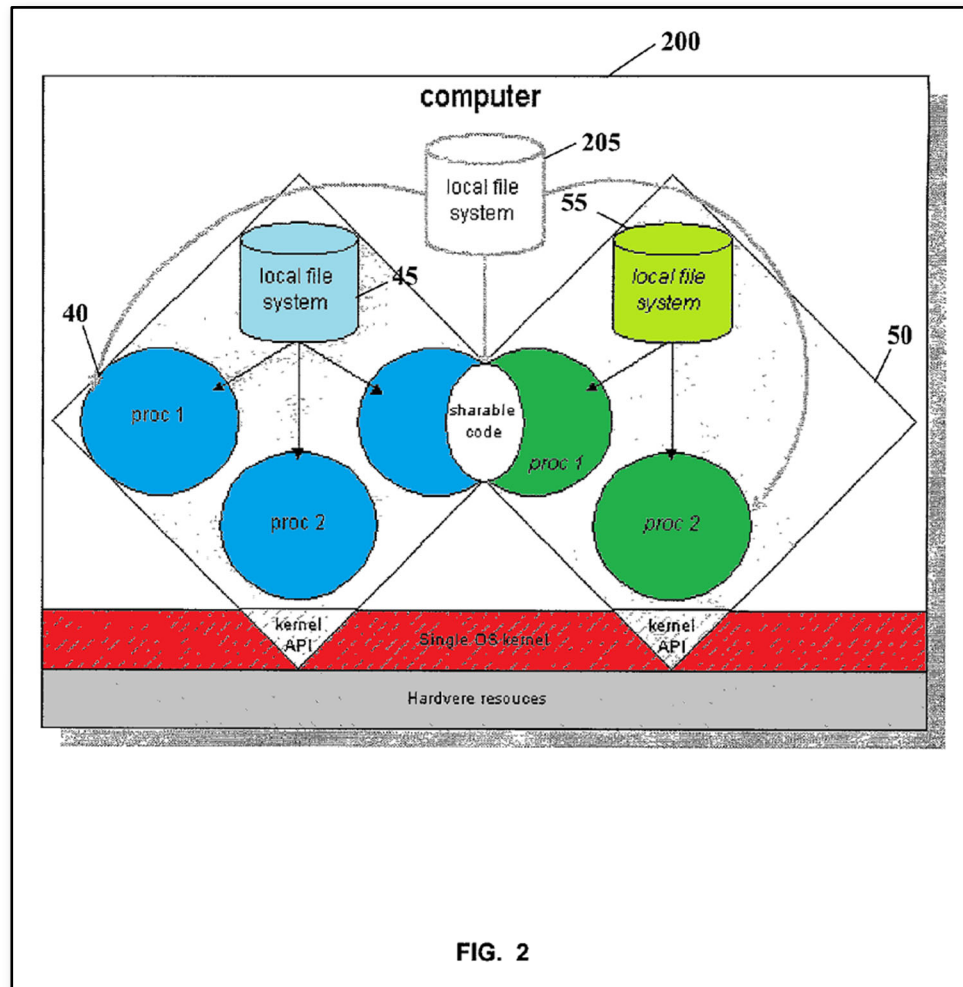
distinguishes VCEs from “a full hardware emulation-type solution” that executes multiple OS kernels (e.g., Virtual-Machine techniques). Tormasov, [0017], [0013]-[0016]. Bhattacharjee, [0074]-[0076].

Tormasov’s Figure 1 (color added below) shows “the same hardware node 100” (server) providing multiple VCEs (orange, blue, green) to potentially “different users.” Tormasov, [0024], [0017] (“server...provid[ing]” VCEs to “end user[s]”). For example, one VCE executes an httpd (website) application (blue) and another executes an Ftp (file-transfer-protocol) application (green). Bhattacharjee, [0077].



Tormasov’s Figure 2 (color added below) shows a server executing two VCEs sharing an OS kernel (red) and hardware (grey). Tormasov, [0026]; *compare with* Tormasov, FIG. 3 (alternative VM-based approach using multiple kernels). Each VCE has “its own unique file system” (light blue and light green) for executing processes (blue and green) – e.g., for the above-discussed

applications – where processes in different VCEs are “completely isolated from each other.” Tormasov, [0026]. Bhattacharjee, [0078].



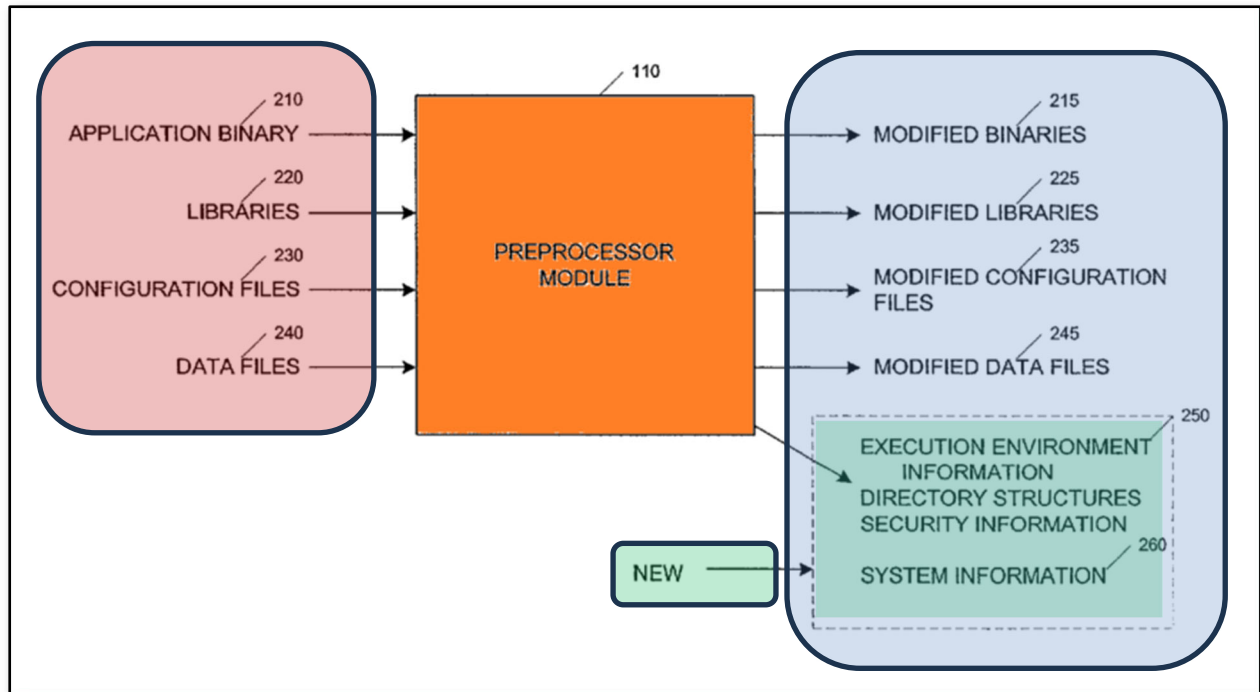
C. Calder (EX1006)

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) used by the application. Calder, [0076], [0082]; EX1014, [0026]-[0028] (“Typically, one software package includes all the files...required for a local setup procedure...including the Dynamic Link Libraries (DLL)” and “executable file[s]”); EX1005, 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/closing a file). EX1120, 19-20. Because different OSs may “provide different mechanisms for implementing system calls,” an application designed for one OS might not work with another OS. EX1097, [0082]. Bhattacharjee, [0079]-[0082].

Calder’s solution modifies an “application package” by adding “system libraries...that translate” “system calls” native to one OS (e.g., “Windows 2000”) to work with a non-native OS (e.g., “Linux”). Calder, [0101]. Bhattacharjee, [0083].

Calder’s Figure 2 (color added below) shows a “preprocessing module” (orange) that converts original application-package files (pink) into 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., not the application’s “native” (originally intended) OS). Calder, [0101]. Bhattacharjee, [0084]-[0085]; EX1054, 4:33-46 (“‘non-native’ server...run[s] a” “different” “operating system”).



D. Grounds 1-2

1. Schmidt-Tormasov Combination (Ground 1)

As discussed above (§V.A), Schmidt-479 teaches techniques to allow a “user’s computing environment [to] be represented in a form that can be understood by” first and second “computers and moved between the computers” so the user can seamlessly transfer their computing “interaction” from one computer to another. Schmidt-479, [0005]. Schmidt-479 does that by placing “active **computing environments**” in “capsules,” including one or more applications together with system files needed to allow the application(s) to be stored, moved, and started/restarted on any server. Schmidt-479, [0016]-[0018]. Multiple capsules can execute on the same server and share its hardware and OS (including kernel). Schmidt-479, FIG. 3, [0044], [0054]. Bhattacharjee, [0086].

Like Schmidt-479, Tormasov concerns providing multiple, independent virtual **computing environments** (“VCEs”) on a server, where each VCE is functionally equivalent to a “full-featured” OS and, thus, can have applications installed therein and can execute independent services (e.g., two VCEs can serve independent websites from one physical server). Tormasov, [0023]-[0024], FIG. 1; EX1122 (cited in Tormasov, [0035]) at 249 (“httpd” is a “service”); EX1110, [0033]-[0034], FIG. 3 (illustrating multiple servers providing multiple virtual environments). Unlike techniques that had multiple OS kernels on one server, Tormasov’s VCEs (like Schmidt-479’s capsules) share a server’s hardware and OS kernel. Tormasov, [0017], [0023]-[0025]. Bhattacharjee, [0087].

Based on these teachings, POSAs would have been motivated to implement in Schmidt-479’s capsules Tormasov’s VCEs (and their applications) to provide capsules that each provide an “equivalent to a...full-featured” OS (Tormasov, [0023]) and that could also beneficially be stored and executed on any server in a multi-server system (Schmidt-479, [0016]-[0018]). Bhattacharjee, [0088]-[0092] (citing EX1083, [0002], [0014], [0011], [0061]-[0063]).

POSAs also had motivation to implement Schmidt-Tormasov’s VCE-capsules in a system with servers running **different OSs**, because POSAs knew “[i]t is important...that an application be usable with several **different operating systems**.” EX1058, 1:14-21. It was background knowledge that “[s]oftware

developers often strive to tailor or ‘port’ their applications to a variety of computing platforms” (e.g., different “operation system[s]”). EX1057, 1:10-14. Bhattacharjee, [0093] (citing also EX1017, [0029]-[0030]).

For example, in “a large corporation having many end users, there are usually a number of **computers with different operating systems**” serving/executing applications. EX1049, 1:41-58. Typically, that application “software...[is] centrally hosted by a Server and used by a number of Clients.” EX1020, [0004]-[0005]; Bhattacharjee, [0094]-[0096] (citing also EX1058, 1:14-21; EX1057, 1:10-14; EX1017, [0029]-[0030]; EX1016, [0002]; EX1053, 1:36-41; EX1049, 1:12-14, 1:41-58; EX1051, 2:25-37). Furthermore, in those “larger networks, it is common...for the servers to be running **different network operating systems.**” EX1051, 2:35-37; Bhattacharjee, [0096]-[0107] (citing also EX1054, 1:14-35, 1:48-54, 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, 4:39-52). Schmidt-479 itself discloses an example system having multiple servers running different OSs “optimized for [an application’s] performance” (e.g., servers respectively running “Microsoft Windows,” “X11/Unix,” etc.). Schmidt-479, [0077]-[0078].

Thus, POSAs had motivation to implement Schmidt-Tomasov VCE-capsules in a system having multiple servers running different OSs, with a VCE-

capsule being executable on more than one different OS. Bhattacharjee, [0107]-[0108].

During prosecution (*supra* §III.C), the '814 patent's applicants said different versions of OSs in the same OS-family (e.g., OSs with the same tradename, like Unix-based OSs "Solaris 8" and "Solaris 10") are "different operating systems." EX1002, 300-301.

As discussed above (§V.A), Schmidt-479's "capsules" have "file system view[s]" containing application executable binaries and systems files that allow the capsule (including its applications) to be stored and started/restarted on "a different instance of an operating system on a different" server. Schmidt-479, [0044], [0048].

POSAs knew applications native to one OS (e.g., Windows 95) were typically compatible with other OSs in the same family (e.g., Windows 98). Bhattacharjee, [0108]. Thus, POSAs would have reasonably expected each Schmidt-Tormasov VCE-capsule to be executable on multiple, different OSs in the same OS family, like Solaris 8/10 or Windows 95/98. That motivation and reasonable expectation of success are further corroborated by references sharing inventor(s) with Schmidt-479 that disclose "arbitrarily bound[ing]" capsules "to different machines and **different operating systems.**" EX1009, [0018]; EX1116,

[0015]. Bhattacharjee, [0109]-[0115] (citing EX1017, [0019]-[0021]; EX1128, 1:51-57, 3:7-30; EX1050, 2:55-38).

Thus, each Schmidt-Tormasov VCE-capsule (and application(s) installed thereto) would have been storable and executable on servers running different OSs in the same OS family. Bhattacharjee, [0116].

POSAs would have also reasonably expected success given Schmidt-479's and Tormasov's overlapping teachings regarding compatible techniques. Schmidt-479's capsules include unique "file system" views and "**virtual namespaces**" to allow a "**computing environment**" to be stored as a unit executable on different servers (Schmidt-479, [0016]-[0018]) running, e.g., conventional OSs, like "Unix" (Schmidt-479, [0048]). Tormasov's "**virtual computing environments**" similarly use unique local file systems and namespaces to execute on conventional computing systems. Tormasov, FIGS. 1-2. Tormasov's VCEs are executable on conventional OSs, like those discussed in Schmidt-479. Indeed, Tormasov's "BIBLIOGRAPHY" cites textbooks regarding the "Unix Operating System" ([0033]), "Linux" OS ([0035]), and generic "Operating System[s]" ([0034]). Bhattacharjee, [0117]-[0118] (citing also EX1009, [0074]).

Thus, it would have required only ordinary programming skill to implement a Tormasov VCE in a Schmidt-479 capsule so the VCE-capsule (including its system files and applications) could be stored, installed, and executed on any of

multiple servers running different OSs in the same OS family, in a multi-server system (e.g., large-organization network). Bhattacharjee, [0119].

2. Schmidt-Tormasov-Calder Combination (Ground 2)

POSAs further knew that different servers in large organizations often executed OSs in **different** OS families. Bhattacharjee, [0120]-[0121] (citing EX1071, 2:4-14). Thus, POSAs had reason to make Schmidt-Tormasov's VCE-capsules executable on OSs in **different** families (e.g., Windows- and non-Windows-based OSs). As discussed above (§V.A), Schmidt-479 itself discloses a system of servers having OSs in different families. Schmidt-479, [0077]-[0078]. Bhattacharjee, [0122]-[0124] (citing EX1009, [0018]; EX1116, [0015]; EX1017, [0029]-[0030]).

POSAs had additional, independent motivation to make Schmidt-Tormasov's VCE-capsules executable on different OS families based on Calder.

As discussed above (§V.C), 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 making Windows-native applications executable on Unix-based OSs. Calder, [0101]. As discussed above (§V.D.1), Schmidt-Tormasov's VCE-capsule includes similar system files allowing the VCE-capsule (and application(s) therein) to execute on any binary-compatible machine.

Thus, POSAs would be motivated and reasonably expected success to apply Calder's teachings to adapt any suitable bundle of applications with system files (including Schmidt-Tomasov's VCE-capsules) to be executable on multiple, different OSs (including different OS-families). Bhattacharjee, [0125]-[0126].

Applying Calder's teachings, each Schmidt-Tomasov-Calder VCE-capsule comprises modified system and application files to allow the VCE-capsule (and its application(s)) to execute on servers running different OS-families in a multi-server system. For example, VCE-capsules with native Windows 2000 applications would have "system libraries added" to support "translat[ing] Windows 2000 system calls to Linux system calls." Calder, [0101]. Bhattacharjee, [0126].

POSAs would have reasonably expected success in the combination. Calder says its techniques work with the same conventional OSs discussed in Schmidt-479 and Tomasov (e.g., Windows-, Unix-, Linux-OSs). Calder, [0083]-[0084]. POSAs also knew those conventional OSs relied on system calls, which Calder's techniques translate for execution with non-native OSs. Calder, Abstract. Thus, it would have required only ordinary programming skill to implement in Schmidt-Tomasov's VCE-capsules software that, based on Calder, permits VCE-capsules to execute on different OS families in a multi-server, multi-OS system. Bhattacharjee, [0127]-[0128].

E. Mapping to Challenged Claims

1. Claim 1

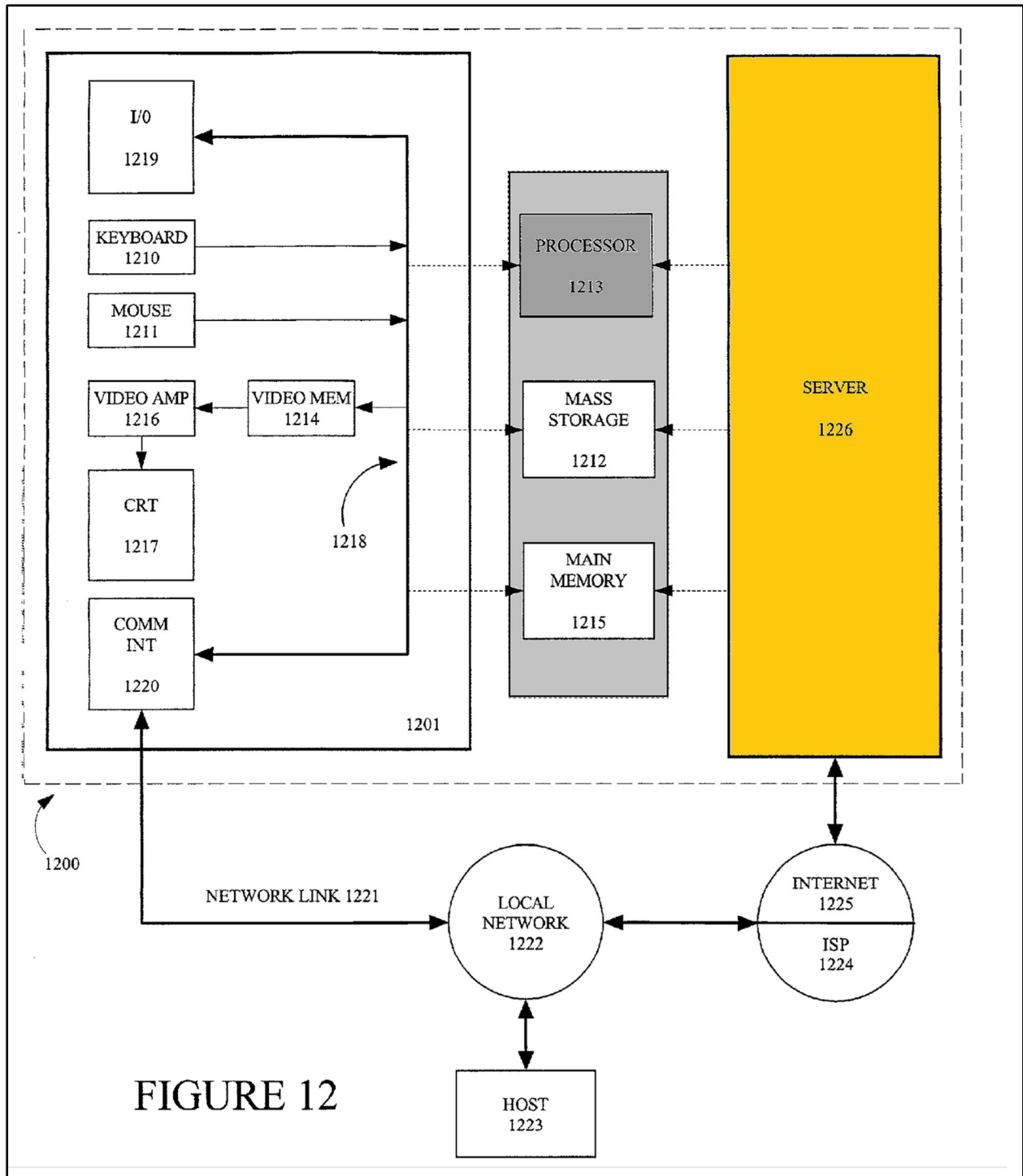
Claim 1's preamble, which the parties agree is limiting (EX1106, 6), is addressed below in two parts: [1PREA]'s "*system*" and [1PREB]'s "*method*."

a. [1PREA]

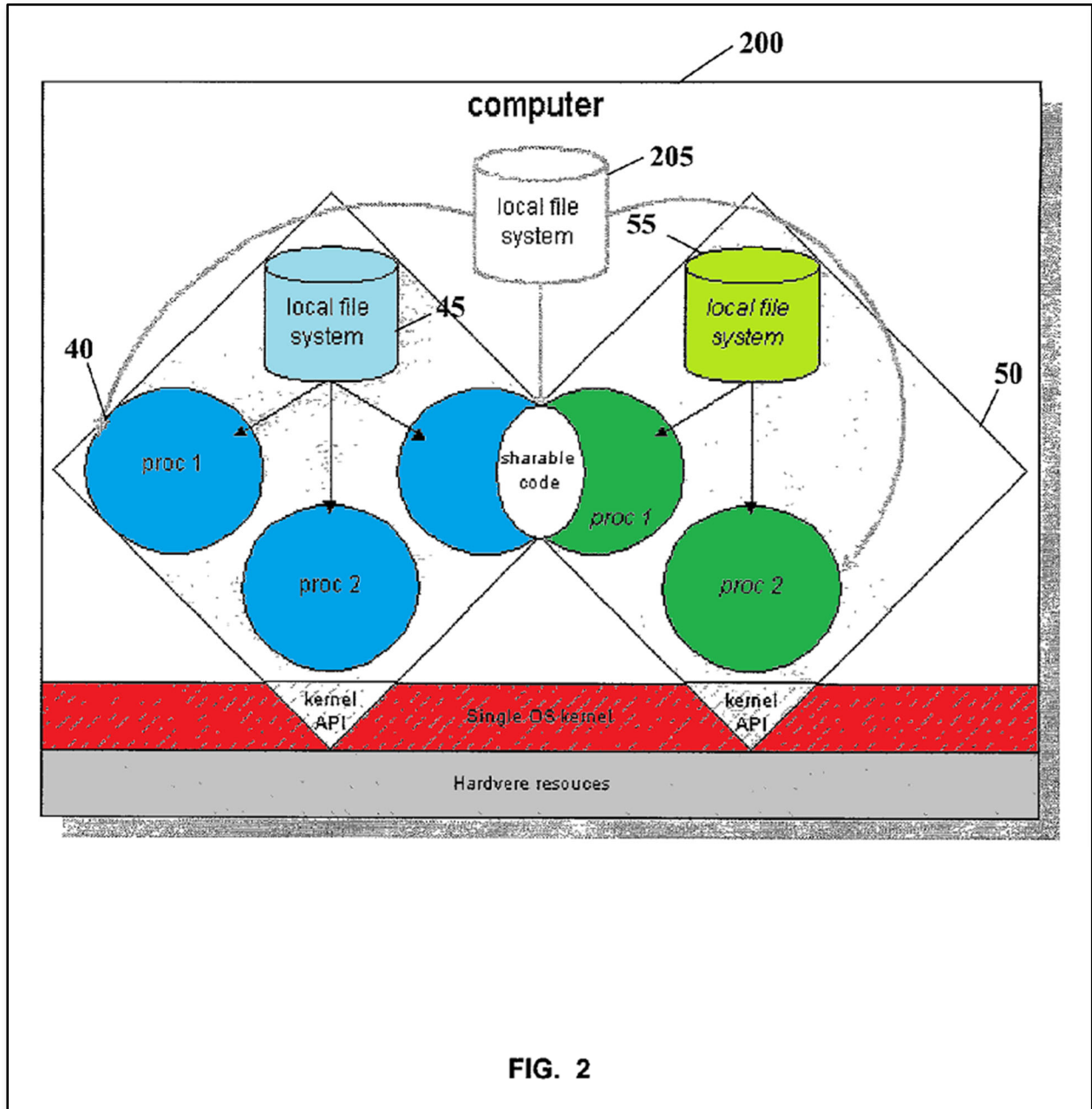
i. "1. In a system having a plurality of servers with operating systems that differ"

As discussed above (§§V.D.1-2), Schmidt-Tomasov's/Schmidt-Tomasov-Calder's VCE-capsules are implemented in *a system having a plurality of servers*, based on both Schmidt-479's and Tomasov's teachings.

Schmidt-479 (e.g., Figure 12, highlighted below) teaches to execute capsules on a "*server*" (yellow) having hardware (gray) (e.g., "processor 1213"). Schmidt-479, [0063]-[0067]; FIGs. 1-3, [0016]-[0019], [0042]-[0044] (discussing "computer **hardware** coupled to the *operating system*"), [0048].



Similarly, Tormasov (e.g., Figure 2, highlighted below) teaches to executes VCEs on a “**hardware computer**” (a “*server system*”) using its “[h]**ardware resources.**” Tormasov, [0017], [0026].



Thus, Schmidt-Tormasov's/Schmidt-Tormasov-Calder's *system* has physical *servers*, which meet the parties' joint construction of "*servers*" as "physical servers." EX1106, 6. Bhattacharjee, [0129]-[0135].

EX1088's definition of "*operating system*" – which is Google's construction (EX1100, 2) – 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,...and UNIX" as examples. EX1088, 378.

As discussed above (§§V.D.1-2), Schmidt-Tormasov's/Schmidt-Tormasov-Calder's *system* has *a plurality of servers* that execute conventional OSs (e.g., Unix- and Windows-based OSs). Schmidt-479, [0077] ("Unix," "Windows NT"); Tormasov, [0033] (citing EX1120: "Unix" OS), [0035] (citing EX1122: "Linux" OS); Calder, [0084] ("UNIX, LINUX," various "Windows[-based]" OSs). The '814 patent uses similar conventional OSs (e.g., "Windows"). See EX1001, 7:34-38.

Thus, Schmidt-Tormasov's/Schmidt-Tormasov-Calder's OSs meet Google's construction and VirtaMove's "ordinary meaning" construction of "*operating system*." EX1106, 4. Bhattacharjee, [0136]-[0139] (citing EX1121, 2-4 (cited in Tormasov, [0034]); Calder, [0083]).

As discussed above (§§V.D.1-2), Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s *servers* have *OSs that differ*. Bhattacharjee, [0140]-[0141].

ii. “operating in disparate computing environments”

Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s VCE-capsules “encapsulate” respective “***computing environment[s]***” (Schmidt-479, [0016]), specifically “virtual ***computing environments***” (VCEs) (Tomasov, [0023]). *See supra* §V.D.1-2.

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), Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s servers are (at least) “standalone” computers as discussed below, so the Board should find Schmidt-Tomasov/Schmidt-Tomasov-Calder meet [**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”).

Consistent with Schmidt-479’s Figure 12 and Tomasov’s Figure 2 (both highlighted above), Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s servers

(*supra* §V.E.1.a.i) are standalone computers (at least) because they are separate computer machines. Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s servers also operate independently of each other, which is what VirtaMove says makes accused computers “standalone.” EX1103, 9; Bhattacharjee, [0142]-[0145].

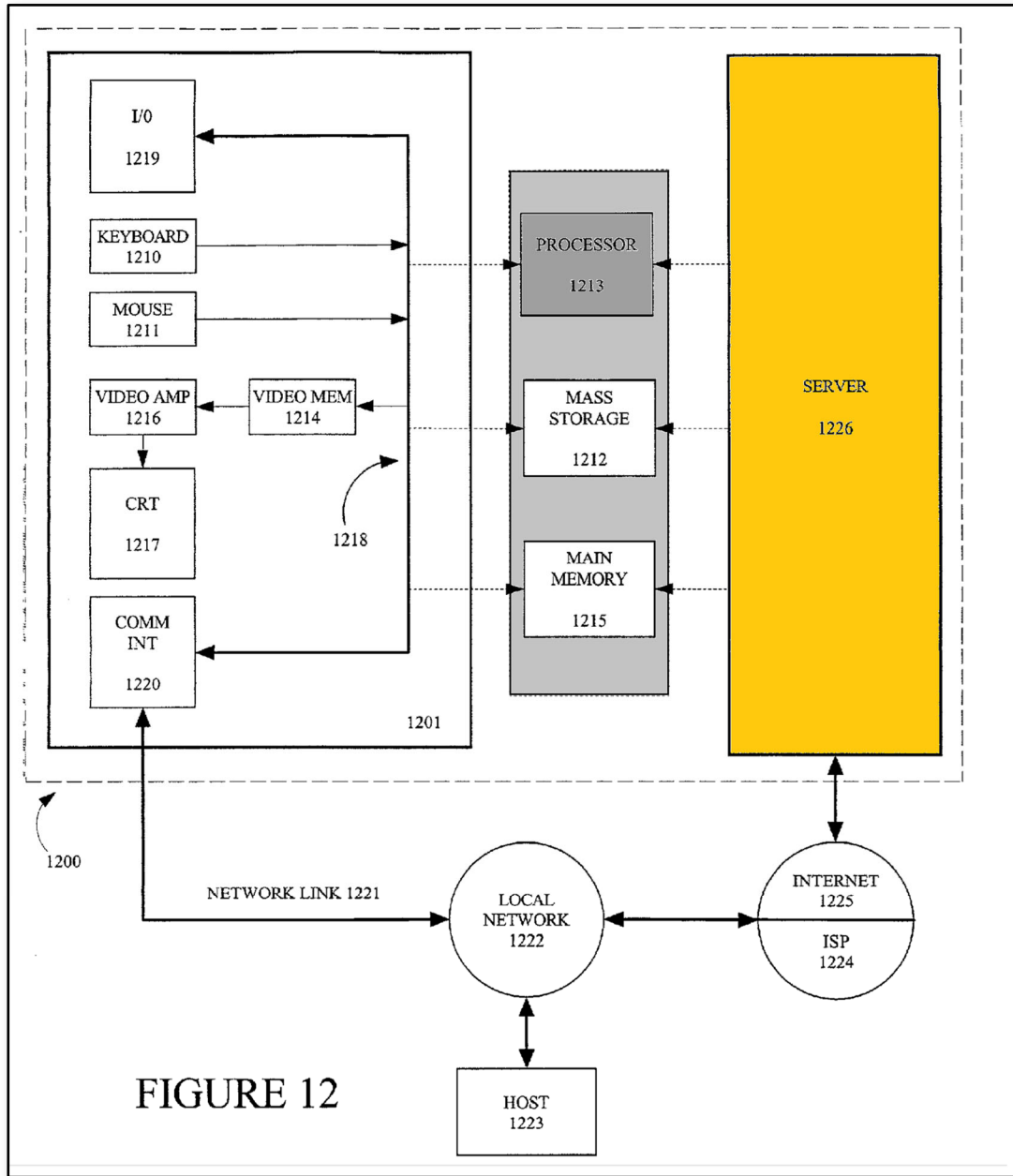
Additionally, during prosecution, the Examiner found that [1PREA] was met by a “network environment” (EX1002, 239-240) having a server and client (EX1089, FIG. 2, 6:28-35). VirtaMove did not disagree. EX1002, 286-333. Bhattacharjee, [0146]-[0151]. Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s system likewise has servers and client computers in a network environment. *See supra* §VII.D.1-2.

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

Each Schmidt-Tormasov/Schmidt-Tormasov-Calder *server includes a processor and an operating system* based on Schmidt-479’s and Tormasov’s similar teachings.

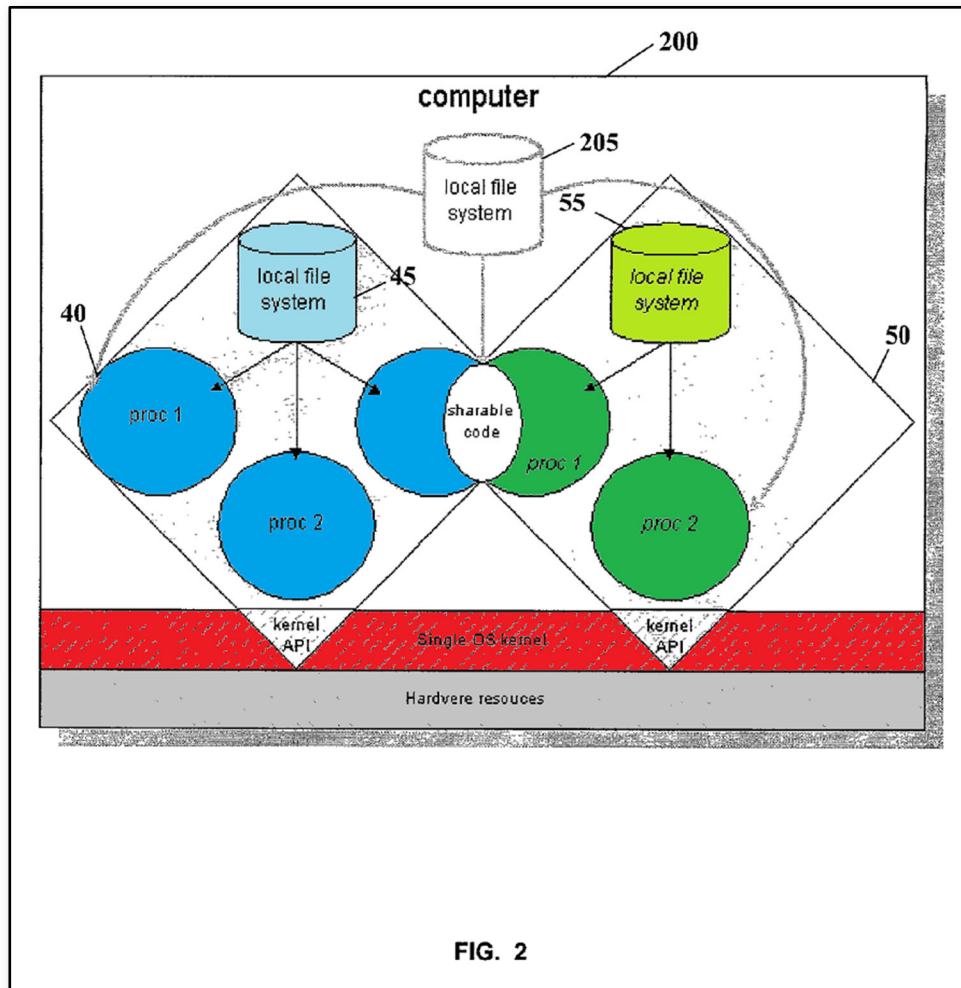
Schmidt-479’s Figure 12 (highlighted below) illustrates conventional server hardware: e.g., a “***processor***” (dark grey) that “reside[s]...wholly on [the] ***server***.” Schmidt-479, [0066], FIGS. 1-3, [0042]-[0044] (discussing “communicat[ion] with the ***operating system*** and then...***computer hardware***”). POSAs knew that typical and obvious conventional *processors* are physical hardware, meeting the parties’

joint construction of “*processor*” as “physical computer processor.” EX1106, 6; Bhattacharjee, [0152]-[0153] (citing EX1087, 1; EX1017, [0003]-[0017]).



Similarly, Tormasov’s Figure 2 (highlighted below) shows that *each* conventional *server* includes “**hardware** resource[s]” (grey) and “one **operating system**” (red). Tormasov, [0027]. Bhattacharjee, [0154]-[0162] (citing EX1121

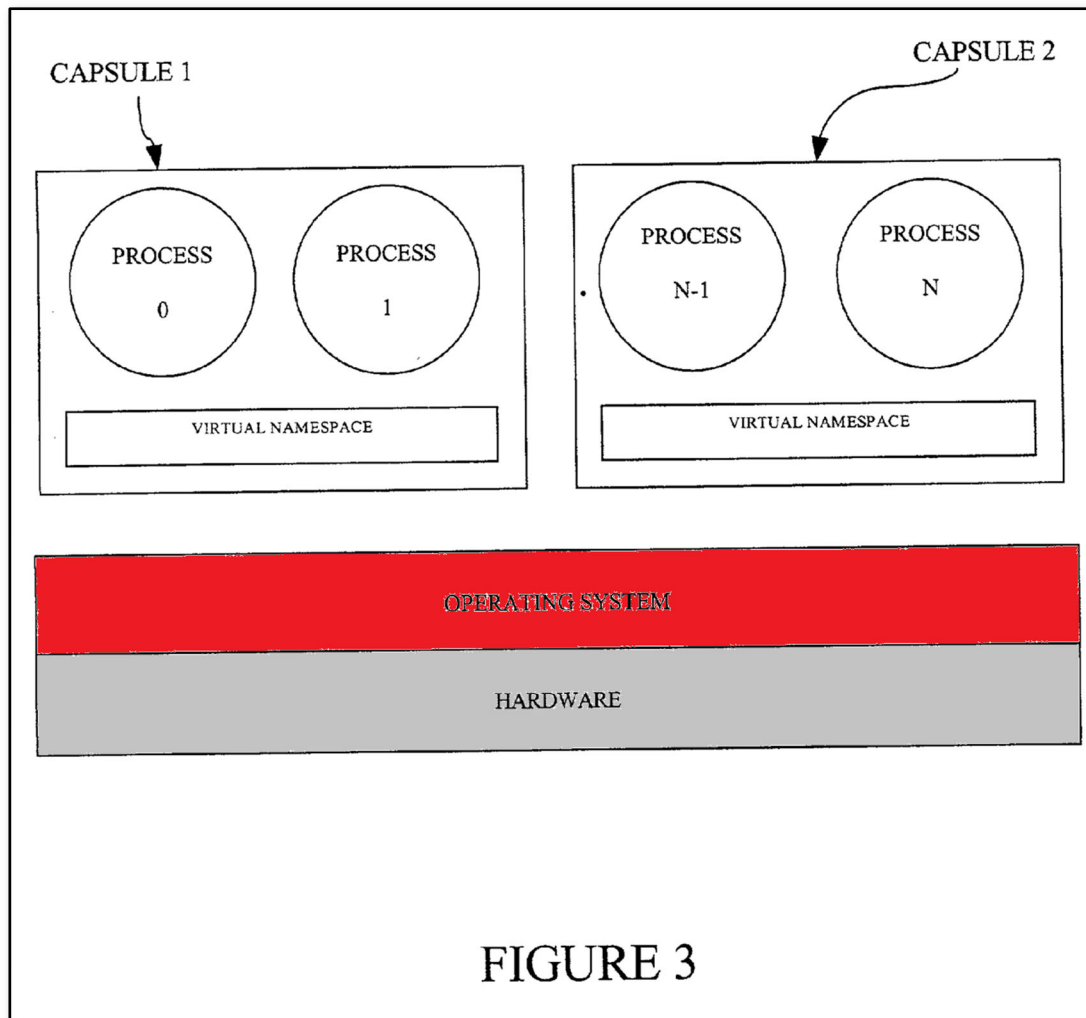
(cited in Tormasov, [0034]), 4; EX1113, claim 22; EX1017, [0003]-[0017]; EX1088, 378). *See supra* §V.E.1.a.i (discussing parties’ constructions of “operating system”).



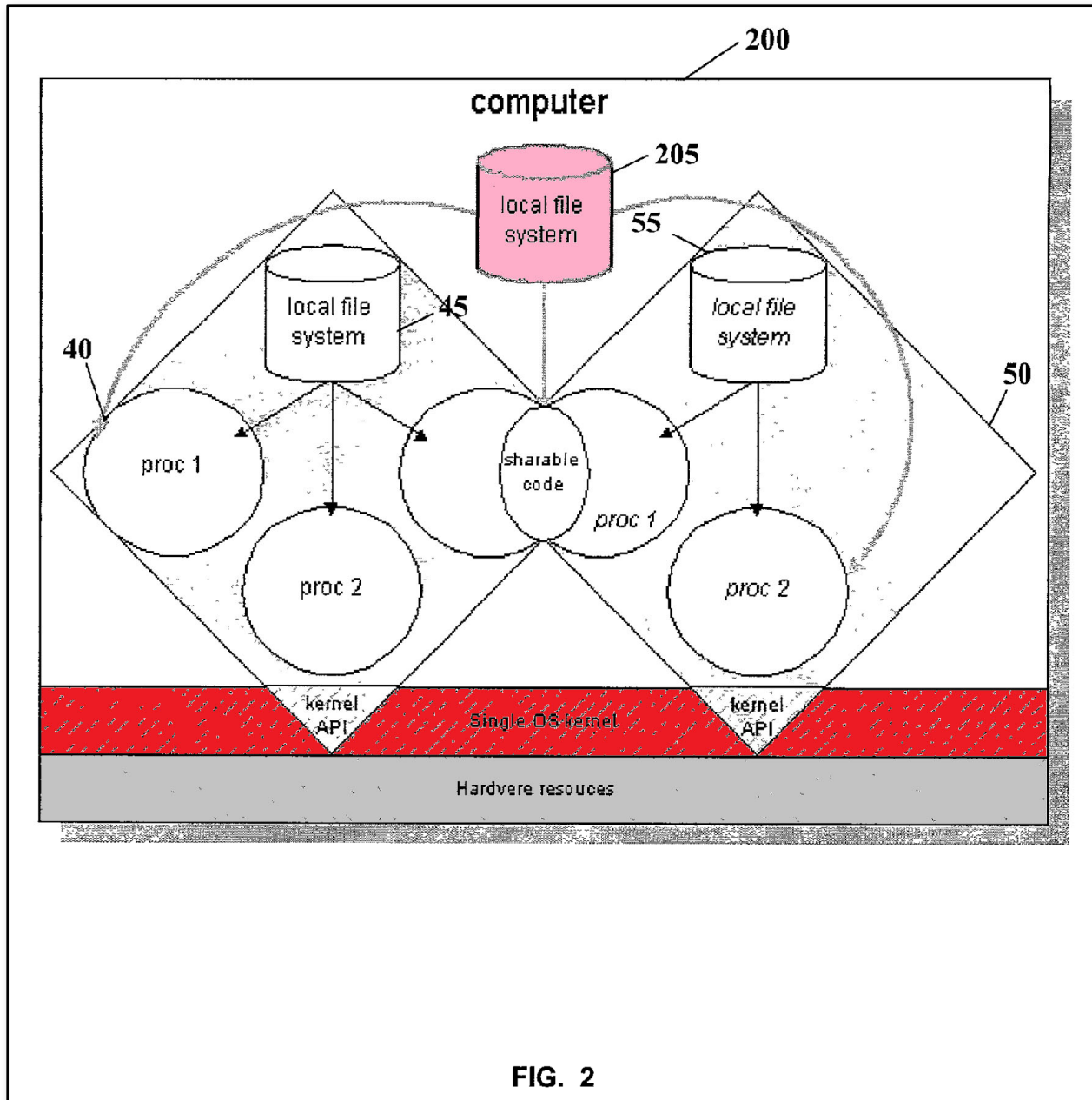
**iv. “[OS] including a kernel [and]⁴ a set of associated
local system files compatible with the processor”**

Schmidt-479’s Figure 3 (color added below) shows that each server includes an “operating system” (red) – with a *kernel* – and hardware (gray) (e.g., a *processor*). Bhattacharjee, [0163]-[0167] (citing EX1088, 300; EX1086, 2:53-55 (“the *operating system* is commonly called...the *kernel*”); EX1120 (cited in Tormasov, [0033] and EX1086, 2:58-60), 4-5; EX1017, [0009]-[0013])).

⁴ The parties agree “and” should be inserted. EX1106, 6.



Similarly, Tormasov's Figure 2 (color added below) shows that each server includes "an **ordinary** single *kernel* operating system" (red), hardware (gray), and a "common **file system** 205" (pink). Tormasov, [0025]-[0026].



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. POSAs would have understood an “ordinary...*kernel*” (Tomasov, [0025]) like the ones Schmidt-

Tomasov/Schmidt-Tomasov-Calder uses meets such a dictionary definition of “*kernel*.” Bhattacharjee, [0168]-[0170] (citing EX1121, (cited in Tomasov, [0034]), 2-4).

Thus, each *kernel* in each OS on a Schmidt-Tomasov/Schmidt-Tomasov-Calder server meets Google’s construction and VirtaMove’s “ordinary meaning” construction of “*kernel*.” EX1106, 4. Bhattacharjee, [0169].

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 uses conventional *system files* – like libraries and configuration files – that were often provided with conventional prior-art OSs. EX1001, 2:52-3:19. Bhattacharjee, [0171].

Schmidt-479’s servers have “an underlying file system” with “configuration files” and “system libraries.” Schmidt-479, [0050]-[0051], [0054]. Similarly, POSAs knew a typical, obvious implementation of an OS’s “*local* file system” (Tomasov, FIG. 2) had similar *local system files*. See EX1001, 2:52-3:19, 10:42-46. See also *infra* §V.E.1.c.iii (demonstrating Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s VCE-capsules have *system files*). Bhattacharjee, [0172].

Moreover, EX1122 (cited in Tomasov, [0035]) recognizes that when you “restrict users’ access to your system” and provide, e.g., an ftpd service in a restricted environment (an example that Tomasov discloses (FIG. 2)), “you need

to move all the files that ftpd needs into the restricted login directory,” including library files in the directory “lib.” EX1122, 122-123. Bhattacharjee, [0173] (citing EX1113, 2:7-11).

Similarly, another reference by Tormasov inventors that discusses aspects of VCEs (EX1113) notes the desirability in a shared system (like Schmidt-Tormasov/Schmidt-Tormasov-Calder) for “[e]ach of the computer users” to have their “own version...[of] the **system file area of the operating system.**” Tormasov-132, 2:7-11. Bhattacharjee, [0173]-[0183] (citing EX1120 (cited in Tormasov, [0033]), 19-20, 313-314, FIG 2.1; EX1029, [0027] (typically, “minimum...**OS components include[] a kernel...and system files**”); EX1090, [0053] (Linux “file system may include **configuration files**...[and] **shared libraries**”); EX1108, [0003]; EX1023, [0023] (Windows-based OSs have “**system files** which form the **kernel** part of the operating system”), [0027]; EX1011, [0032]; EX1067, 9:12-20; EX1064, 1:53-62; EX1062, 3:37-40; EX1034, [0041], [0047]).

Each Schmidt-Tormasov/Schmidt-Tormasov-Calder server’s *kernel* and *set of associated local system files* discussed above are *compatible with the server’s processor* because the processor uses those files, which would not occur if they were incompatible. Bhattacharjee, [0184]-[0187] (citing EX1067, 35:48-54 (“**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.

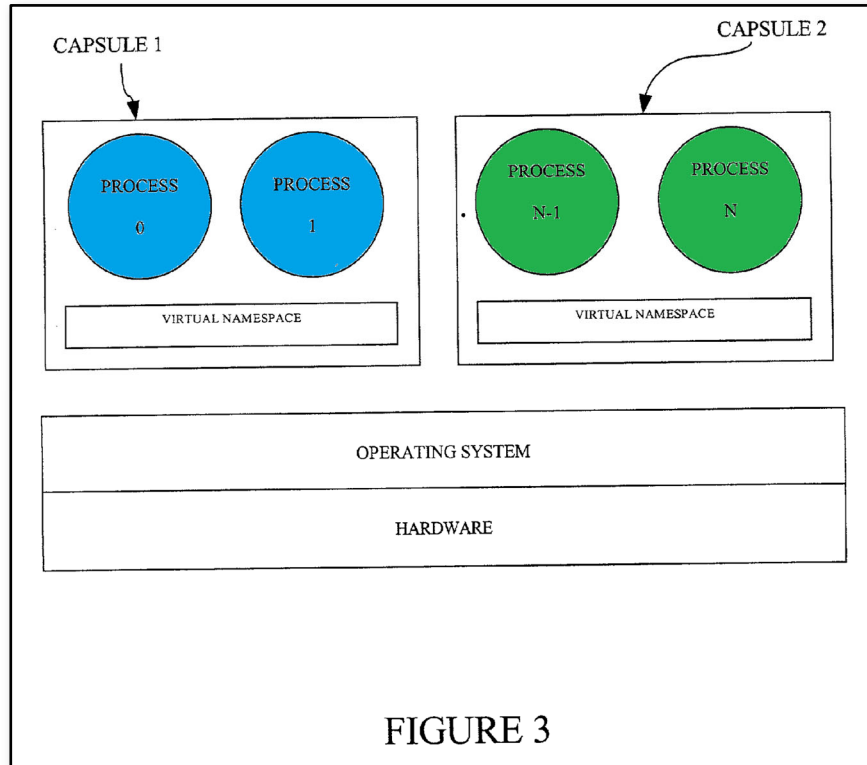
b. [1PREB]

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

The ’814 patent’s “example applications” include web-server applications (e.g., “Apache”). EX1001, 2:55-3:19. Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules *securely execute* web-server (and other similar) *applications*. Bhattacharjee, [0188]-[0189] (citing Schmidt-479, [0009]-[0010] (discussing “computer programs” for *applications* like a “word processor”); Tormasov, FIG. 1 (“httpd” web-site server *application*)).

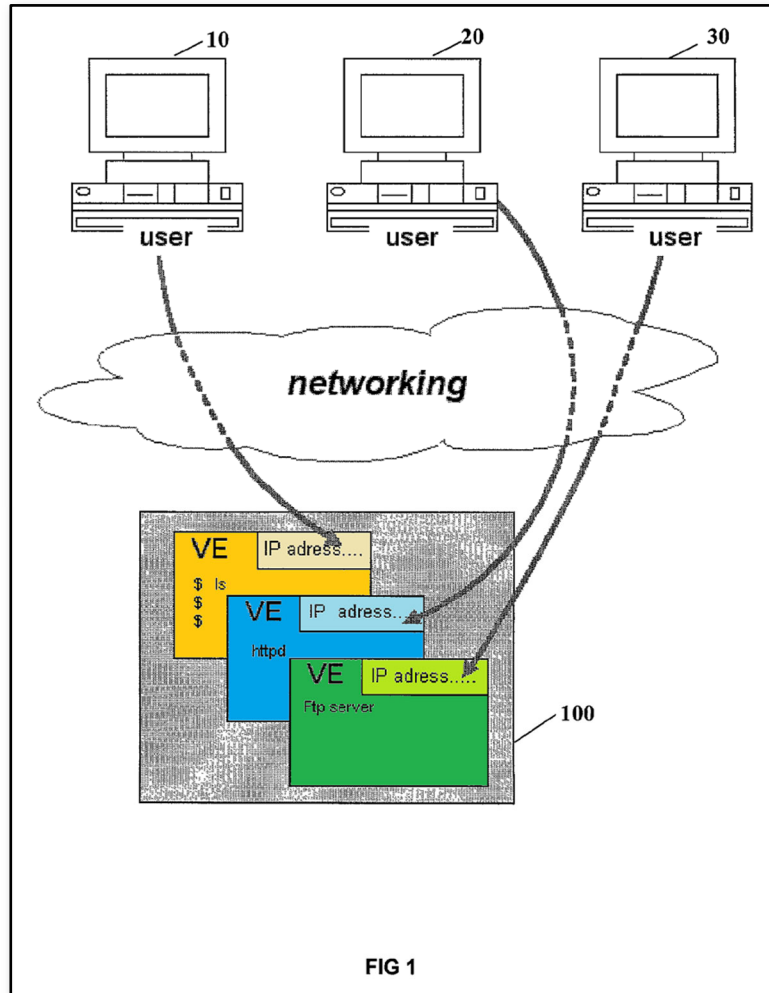
Schmidt-Tormasov/Schmidt-Tormasov-Calder uses Schmidt-479’s capsules, which “comprise[] one or more processes and their associated system environment.” Schmidt-479, [0038], FIG. 3 (color added below). These are “processes” of *executable applications provided to the servers*, like “a word processor, [or] a web browser” application. Schmidt-479, [0010], [0040], [0048] (file system views in each capsule “include, for instance: ***application binaries...***”), [0051] (“default set of [the] file system view mappings includes.../bin **for *applications***”), [0077] (servers with capsules of applications for, e.g., “audio or video services”). Bhattacharjee, [0190]-[0194]; Calder, [0076] (corroborating that

an “application package” typically includes “an *application* binary (also called *application program*)” that is *executable*).



Similarly, Tormasov’s Figure 1 (color added below) shows VCEs *providing* “*a method of* efficient utilization of a single hardware system” ([0017]) for *executing applications* for, e.g., an httpd (web) server (blue) and ftp (file-transfer protocol) server (green). Tormasov, [0012] (an identified prior-art problem being addressed was *executing* “complex *applications* requiring a database” or “CGI” “scripts” “on a remote server”), [0024] (VCEs allow “installation of programs...and other *applications*”). Bhattacharjee, [0195] (citing EX1111, 4:58-

64 (“traditional architecture” for a “work organization of computers... include[s]...*application* software”)).

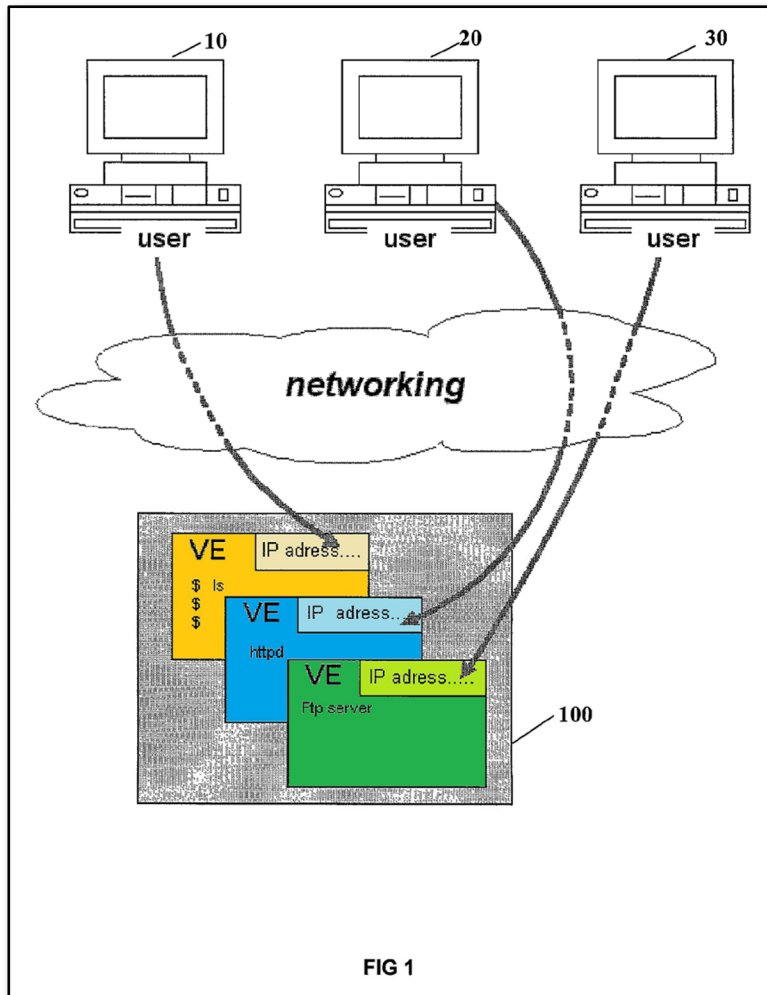


As discussed *supra* §V.D.1-2, Schmidt-Tormasov/Schmidt-Tormasov-Calder places one or more of the above applications in VCE-capsules, which are *provided to the system’s servers*. The application processes in a VCE-capsule are *secure* (at least) because, as discussed below (§V.E.1.c.i.(3)), VCE-capsules are *secure containers of application software*. Bhattacharjee, [0196]-[0197].

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, [0198]-[0199].

As discussed above (§V.A), Schmidt-479 teaches encapsulating various *applications related to* various “*services*” (e.g., “X11/Unix *services*...Windows NT *service*...and others”). Schmidt-479, [0077]. Similarly, as discussed above (§V.B) and illustrated in Tormasov’s Figure 1 (highlighted below), VCEs execute *applications related to* similar *services* as the '814’s examples, like applications for an “httpd” server (website service) or “Ftp server” (file-transfer-protocol service), by accessing the *server* over “networking.” Tormasov, [0006] (one express goal of Tormasov is to provide, e.g., a “hosting *service*” that allows users to provide services via “websites”), [0024] (VCEs allow “installation of programs, configuration of **network services**”); EX1122 (cited in Tormasov, [0035]), 118 (“FTP is...one of the most widely used *services* on the Internet....FTP *service*, **ftpd**,” which is shown in Tormasov’s FIG. 1 below). Bhattacharjee, [0200]-[0201].



POSAs understood the above-discussed applications served by Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s networked servers are specialized, software-based functionality provided by Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s network servers, meeting Google’s construction of “*service*” (EX1100, 3 (quoting EX1088, 475)), and VirtaMove’s “ordinary meaning” construction (EX1106, 4). Bhattacharjee, [0201]-[0203] (citing Schmidt-479, [0008]-[0010] (“computer **network** to the **servers**”)); Tomasov, FIG. 1 (accessing server over “**networking**”)).

iii. “wherein the applications are executed in a secure environment”

Schmidt-Tormasov/Schmidt-Tormasov-Calder *executes* the above-discussed *applications in a capsule* (“a **private**, portable, persistent *environment*”) (Schmidt-479, Abstract) for executing a “virtual computing *environment*” (VCE) (Tormasov, [0026]). *Applications executing in a VCE-capsule are executed in a secure environment* (at least) because “[c]apsules offer...*security within a shared environment.*” Schmidt-479, [0041]. The virtual computing *environment* executing in the VCE-capsule is further *secure* (at least) because “processes in different” VCEs (and their respective “unique file system[s]”) are “**completely isolated** from each other” (Tormasov, [0026]) and “are **never visible** to other” VCEs “running on the same computer” (Tormasov, [0018]). Bhattacharjee, [0204].

Based on Calder, POSAs would have been additionally motivated and reasonably expected success to “intercept[]” “system calls made by the application[(s)]” installed to Schmidt-Tormasov-Calder’s *environments* to restrict “access [to] approved files,” thereby protecting the servers and “the contents of the application package.” Calder, [0087], [0073] (“application package” is “**executed safely, securely**, and transparently”), [0100] (using “*security* information” to “**protect both the client computer [], as well as the contents of the**

application”); *see also infra* §V.E.c.i.(3) (demonstrating VCE-capsules meet *secure containers*). Bhattacharjee, [0205]-[0209].

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, a “GUI supports 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. Exemplary “tasks” using icon objects include “[i]nstall[ing] application specific files” and “[s]tarting a first application.” EX1001, 15:35-56. Those were known, conventional uses of icon objects (e.g., clicking an icon to activate an application’s executable object-file). Bhattacharjee, [0210]-[0218] (citing EX1047, 1:21-67; EX1059, 1:34-38; EX1047, 1:43-51).

POSAs knew that the conventional OSs that Schmidt-Tormasov/Schmidt-Tormasov-Calder servers use (e.g., Linux- and Windows-based OSs, *see supra* §V.D.1-2) provided and/or obviously and beneficially supported GUIs, where *applications each include an object* (e.g., icons, application files, and executables) that, in Schmidt-Tormasov/Schmidt-Tormasov-Calder, is *executable* by **two or more** of the different OSs in the system (e.g., Windows 95/98), which meets each

party's construction of "*at least some...*" EX1106, 4. Bhattacharjee, [0210]-[0216]-[0218] (citing EX1047, 1:39-45 (GUI's "[e]xecution of an application program is initiated by selecting its corresponding icon")). Thus, e.g., an icon for starting an executable file and the executable file itself are each *an object executable for performing a task related to the service* provided by the application being started by the icon. Bhattacharjee, [0219].

Furthermore, based on Schmidt-479, Schmidt-Tormasov's/Schmidt-Tormasov-Calder's active VCE-capsules (*supra* §V.A, §§V.D.1-2) "comprise[]" "processes and their associated system environment." Schmidt-479, [0038]. These are "processes" of *executable applications related to a service*. Schmidt-479, [0008] ("computing has become '*service*-oriented'" and when "the user accesses [] data or computing programs, the remote computer is said to be providing a *service*"), [0010] (exemplary applications include, e.g., "a word processor, a web browser"), [0040], [0048] ("files [in each capsule's file-system view] include, for instance: ***application binaries***"). Bhattacharjee, [0221]-[0223] (citing EX1121 (cited in Tormasov,[0034]), 43). Furthermore, "an ***object*** in one capsule" is isolated and protected from *objects* in another capsule. Schmidt-479, [0041]. Bhattacharjee, [0223]-[0224].

Additionally, based on Tormasov, each Schmidt-Tormasov/Schmidt-Tormasov-Calder VCE-capsule (*supra* §§V.D.1-2) separates "user processes on

the level of kernel *objects*/resources namespace,” which is a “collection of unique names” that each “applie[s] to such *objects* as files” that can be executed to support the services provided by applications. Tormasov, [0018]. Bhattacharjee, [0220].

Thus, Schmidt-Tormasov/Schmidt-Tormasov-Calder meets [1PREB].
Bhattacharjee, [0224].

Based on Calder, Schmidt-Tormasov-Calder’s VCE-capsule (*supra* §V.D.2) comprises additional system files (modified libraries and configuration files, Calder, [0082]) that allow the above-discussed *objects* “to execute and communicate on **any non-native [or native] platforms**” (i.e., different OS-families) (Calder, [0101]). That is an additional reason Schmidt-Tormasov-Calder meets [1PREB]. Bhattacharjee, [0225]-[0230].

c. [1A]

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

(1) “storing in memory accessible to...the servers”⁵

Based on Tormasov, Schmidt-Tormasov/Schmidt-Tormasov-Calder *stores* “[a]ll the processes of all” VCE-capsules running on a server “inside the same

⁵ See *infra* §V.E.1.c.i.(2) (demonstrating VCE-capsules are *containers*),

§V.E.1.c.i.(3) (demonstrating VCE-capsules are *secure containers of application software*).

physical *memory*” *accessible to that server*. Tormasov, [0026]. That meets VirtaMove’s construction, which only requires the claimed *memory* to be *accessible* to (usable by) one server. *See* EX1103, 12. Bhattacharjee, [0231]-[0235].

Calder teaches storing application packages *in the memory* of a “master control center” accessible to *servers*. Calder, [0078]-[0079]. That would have motivated POSAs (with reasonable expectation of success) to *store* any suitable application package (including Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules) in *memory* of a centralized-storage system that is *accessible to the* system’s plurality of *servers*, to beneficially provide at least two *servers* in the network a centralized location from which to read/write, manage, and use VCE-capsules, which meets both parties’ constructions of “*memory accessible to...the servers.*” EX1106, 4. Bhattacharjee, [0236]-[0237].

(2) “containers”

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.

As discussed above (§V.E.1.b.i), Schmidt-479, Tormasov, and Calder each teach executing similar *sets of software applications on a computing platform*, any

of which POSAs would have been motivated and reasonably expected success to *execute* in Schmidt-Tomasov's/Schmidt-Tomasov-Calder's VCE-capsules. Bhattacharjee, [0238]-[0239].

Based on Schmidt-479, each VCE-capsule comprises “the complete state necessary to allow the encapsulation to be suspended and revived on any binary compatible machine.” Schmidt-479, [0016]. That necessary data includes “[s]ystem environment information,” which includes “**configuration settings**, working directories and files...**installed software**, and internal program state.” Schmidt-479, [0038]. Each capsule further comprises “file system views,” which are “system directories and **files necessary [i.e., *required*] for normal user operation**...for instance: ***application* binaries, libraries**, certain **device drivers**...etc.” Schmidt-479, [0048]. Those are *files required to successfully execute software applications* on each server-computing platform in Schmidt-Tomasov/Schmidt-Tomasov-Calder. Bhattacharjee, [0240].

Based on Tomasov, each VCE-capsule “includes a complete ***set of*** processes [*executing software applications*] and ***files*** of an operating system that can be modified by the end user,” which is an additional teaching of processes and files *required to successfully execute software applications* (e.g., web-server applications) on each server's *computing platform*. Tomasov, FIG. 1, [0026]. Bhattacharjee, [0241]-[0243].

Calder’s teachings further confirm that Schmidt-Tomasov-Calder’s VCE-capsules include “**application** binar[ies],” “libraries,” and “configuration files” that *are required to successfully execute a set of software applications on any computing platform (server) running OSs non-native to the VCE-capsule.* Calder, [0082], [0102]. Bhattacharjee, [0244].

Thus, Schmidt-Tomasov/Schmidt-Tomasov-Calder meets part [1] of the joint construction of *container*. Bhattacharjee, [0245], [0246]-[0249].

[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.

Based on Schmidt-479, each VCE-capsule is *for use on a Schmidt-Tomasov server* and those “[c]apsules offer privacy, **isolation**, and security” because “capsule contents are **not externally visible or accessible** (unless they are explicitly exported). Thus, capsules **provide privacy and isolation** because they are completely disjoint, (i.e., it is impossible for an object in one capsule to name an object in another capsule).” Schmidt-479, [0041]. Bhattacharjee, [0250].

Based on Tomasov, each VCE-capsule further “includes a complete set of processes and files of an operating system that can be modified by the end user” (i.e., read/write files) ([0025]) and “its own **unique** [i.e., *mutually exclusive*] file system,” which includes its read/write files ([0026]). Tomasov further discloses

that “two processes in different” VCEs are “**completely isolated** from each other” ([0026]) and each VCE has a “root file system [that] is **independent** and...**never visible to other**” VCEs, including those “running on the same computer” (*server*) ([0030]). Bhattacharjee, [0251].

Thus, *each VCE-capsule is mutually exclusive of the other VCE-capsules* on that server, so the *read/write files* associated with⁶ each VCE-capsule – e.g., its “complete set of...files that can be modified” (Tormasov, [0025]) and the VCE-capsule’s read/write “objects” (Schmidt-479, [0041]) – *cannot be shared with other VCE-capsules*. Bhattacharjee, [0252]. *See also infra* §V.E.1.c.i.(3) (demonstrating VCE-capsules are *secure containers of application software*).

Each VCE-capsule’s read/write files are further prevented *from being shared with other VCE-capsules* because each VCE-capsule has its own root directory, as discussed below (§V.E.1.i). Bhattacharjee, [0253]. Conventional techniques for providing different root directories– like chroot (change-root) – existed to prevent users from corrupting other users’ files and/or the OS’s system files, and thus required only ordinary skill to implement. Bhattacharjee, [0253]-[0258] (citing EX1027, [0042]; EX1035, [0002]-[0003]; EX1072, [0014]-[0015], [0049]-[0053]).

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

POSAs also had motivation to prevent sharing of read/write files to protect them from being corrupted, based on Calder’s teaching to intercept system calls to “**protect[] the contents** of the application package.” Calder, [0087], [0100]. Bhattacharjee, [0254].

Thus, Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules meet part [2] of the parties’ construction of *container*. Bhattacharjee, [0259], [0260]-[0281] (citing EX1064, 1:53-62, 2:12-13; EX1048, 6:7-8; EX1097, [0051]).

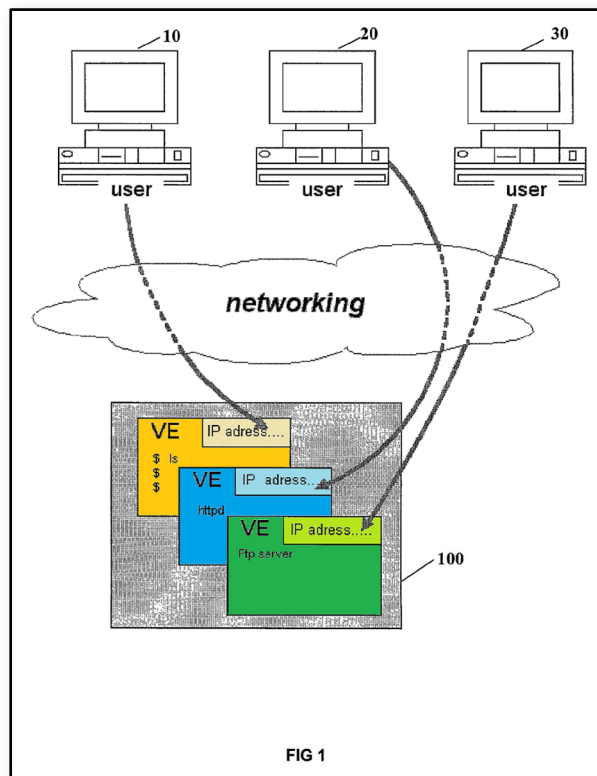
(3) “secure containers of application software”

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. Exemplary “resource[s]” “include CPU time, memory, file activity and network bandwidth.” EX1001, 10:15-17.

Schmidt-Tormasov/Schmidt-Tormasov-Calder uses Schmidt-479’s capsule to encapsulate a Tormasov VCE. *Supra* §§V.D.1-2.

Schmidt-479’s “[c]apsules offer privacy, **isolation** [*insulation*], and **security**” because they “are completely disjoint, (i.e., it is **impossible for an object in one capsule to name an object in another capsule**).” Schmidt-479, [0041], [0014] (“allow[ing] a user to **customize** their computing environment”), [0020] (giving an “untrusted guest...a limited set of **resources**”). Tormasov’s Figure 1 (color added below) illustrates VCEs executing applications that support a specific service (an application set), such as an httpd application set (blue) for a web-server service. Tormasov, [0024].



As discussed above (§V.D.1), each VCE-capsule (*container*) comprises a “virtual computing **environment**” that “is **perceived by the personal computer**

user **as if he has obtained full network root access** to a common computer” (Tormasov, [0023]) and allows “different users” running applications to “work with the same hardware node...**as if they worked on totally separate computers** with no associated hardware” (Tormasov, [0024]). VCE users can “perform all actions allowed for the ordinary computer,” like “installation of programs, configuration of network services,” and execution of “applications.” *Id.* Bhattacharjee, [0282]-[0290].

Thus, application sets in each VCE-capsule appear to have individual control of some critical system resource(s) (e.g., the processor and/or “network address,” Tormasov, [0024]; “naming their file system,” Schmidt, [0013]), which meets each party’s construction of “*secure containers of application software*” (EX1106, 5). Bhattacharjee, [0291].

Additionally, each VCE-capsule also “has its own unique file system” and application processes in different VCE-capsules are “completely isolated from each other.” Tormasov, [0026]. Thus, data (e.g., read/write files) within each application set in a VCE-capsule are insulated (“isolated”) from effects of other application sets in other VCE-capsules, which also meets each party’s construction of “*secure containers of application software*.” EX1106, 5. Bhattacharjee, [0294]-[0298].

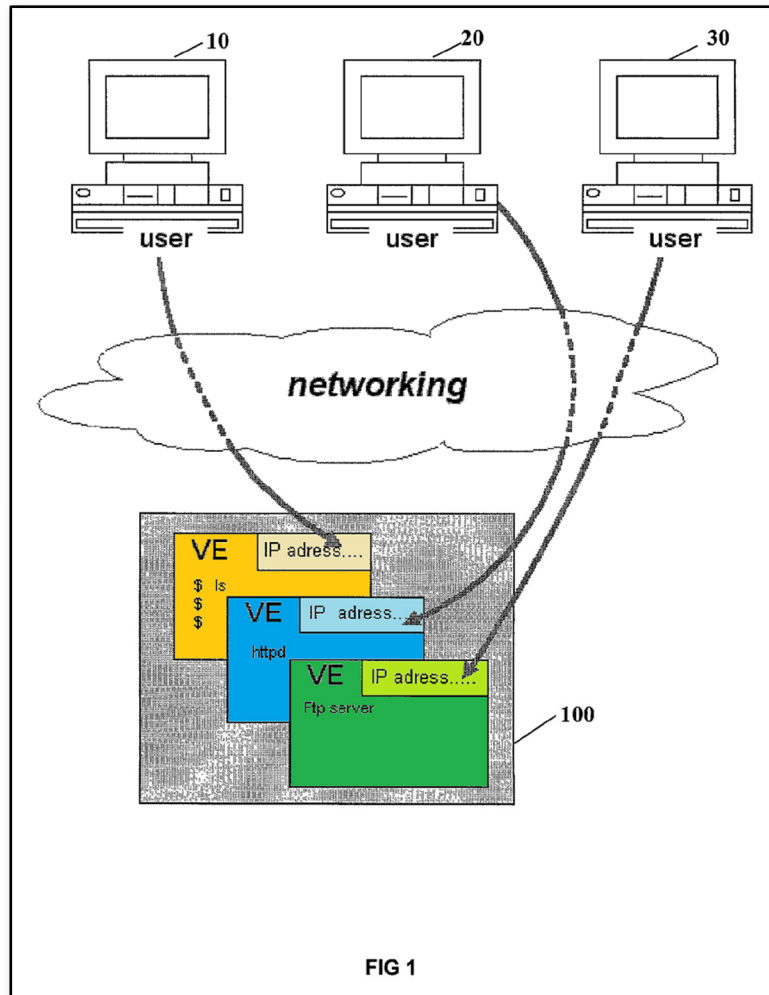
As discussed above (§V.E.1.b.iii), based on Calder, “applications” in Schmidt-Tormasov-Calder VCE-capsules are also “prevent[ed]” “from improperly modifying or accessing data on the” computers, which further insulates them from effects of other application sets. Calder, [0079]. Bhattacharjee, [0292]-[0293], [0299].

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

Based on Schmidt-479, each “[VCE-]capsule comprises one or more processes and their associated system environment.” Schmidt-479, [0038]. Each process is an *application* “program in **execution**.” EX1121, 43. Bhattacharjee, [0300]-[0301] (citing EX1121 (cited in Tormasov, [0034]), 42-43).

Each capsule further comprises “an **active** computing environment,” which includes “**active processes** and the complete state necessary to allow the encapsulation to be suspended and revived on any binary compatible machine.” Schmidt-479, [0016]. That “[s]ystem environment information may include, for instance,...**installed software, and internal program state**.” Schmidt-479, [0038]. Bhattacharjee, [0302].

Tormasov’s Figure 1 (color added below) further demonstrates that VCE-capsules *comprise executable applications*, like an http web-server application (blue) or Ftp (file-transfer-protocol) server application (green) that users can *execute*, as in a conventional OS. Tormasov, [0023]. Bhattacharjee, [0303].



Thus, Schmidt-Tormasov’s VCE-capsules (*containers*) *each comprise one or more of the executable applications*. Bhattacharjee, [0304].

Based on Calder, each Schmidt-Tormasov-Calder VCE-capsule also includes “modified ***application*** binaries” and modified *system files* so the *applications* in VCE-capsules (*containers*) can be *executed* on both native and non-native OSs. Calder, [0098]. Bhattacharjee, [0304]-[0306].

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 OSs typically provided *system files*. Bhattacharjee, [0307].

In Schmidt-Tormasov/Schmidt-Tormasov-Calder, each VCE-capsule “is functionally equivalent to a computer with a full-featured” OS (Tormasov, [0023]) and has its own “root file system” (Tormasov, [0030]). POSAs knew a typical, obvious implementation of a conventional prior-art OS local file system used in Schmidt-Tormasov/Schmidt-Tormasov-Calder would have its own *set of associated system files required to execute the VCE-capsule application* (and any additional applications therein). *See* EX1122 (cited in Tormasov, [0035]), 122-123 (when you “restrict users’ access to your system” and put an ftpd application in a restricted environment, as in Tormasov, “you need to move all the files that ftpd needs into the restricted login directory,” including *system files* like library files in the Unix-OS directory “lib”). Bhattacharjee, [0308]-[0316] (citing EX1029, [0027] (the “**minimum...OS components include[]** a kernel...and *system files*”)); EX1023, [0023], [0027]; EX1011, [0032]; EX1067, 9:12-20; EX1064, 1:53-62; EX1062, 3:37-40; EX1094, 3:34-63; EX1034, [0041], [0047]).

Schmidt-479 says it was “desirable” to allow “users” of computing environments (like VCE-capsules) to “arrange and organize the appearance of the data and computer programs [*applications*] in a way that is suitable for them.”

Schmidt-479, [0007]. Tormasov’s VCEs similarly “allow[] a root user” “to make file modifications and **configure** their own local parameters of the” OS.

Tormasov, [0030]. In the conventional OSs that Schmidt-Tormasov/Schmidt-Tormasov-Calder uses, those modifications/configurations would conventionally be made in configuration files, which are additional *system file required to execute the applications* that exist in VCE-capsules. Bhattacharjee, [0317]-[0318] (citing EX1061, 1:12-16, 2:28-33, 2:53-3:5 (“Typically, when” an OS “instantiate[s] an application,” it will “locate[]...and then process[]...the **configuration file[] for the application....**”))).

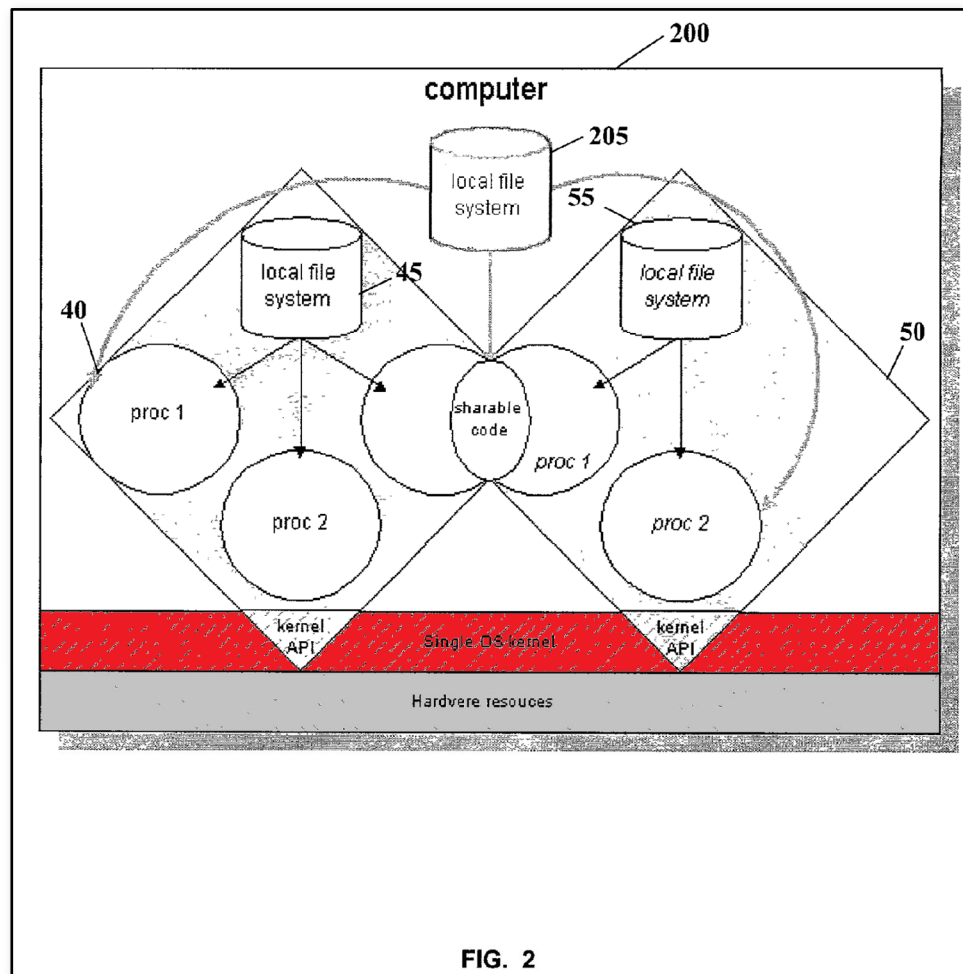
Moreover, based on Calder (*supra* §V.D.2), Schmidt-Tormasov-Calder’s VCE-capsules include additional modified *system files* (e.g., libraries and configuration files) *required to execute the applications* on non-native OSs.

Calder, [0077], [0082], [0094], [0101]. Bhattacharjee, [0319].

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

Tormasov’s Figure 2 (color added below) shows VCEs do not include, but rather share *use* of “a single hardware system with a single operating system [*local*] **kernel**” (red) *on one of the servers*. Tormasov, [0017], [0023]-[0025]

(VCEs “share [use] the same *kernel* of the [local] operating system” *residing on the server*), [0029]. Bhattacharjee, [0320]-[0321].



The conventional *local kernel* (red) used by VCE-capsules *resides permanently* in memory of the *server* (a hardware resource (gray)) because the kernel is not lost when power is removed from the server (e.g., it’s stored in non-volatile memory), and thus meets Google’s construction of “*local kernel...*” and VirtaMove’s “ordinary meaning” construction. EX1106, 4. Bhattacharjee, [0322]-[0326] (citing EX1121 (cited in Tormasov, [0034]), 2 (OS, and thus its *kernel*, “is

the first program run on a computer when the computer boots up,” which corroborates that the OS+*kernel* is stored in non-volatile memory); EX1091, 1:23-38).

Based on Calder, Schmidt-Tomasov-Calder’s VCE-capsules (*supra* §V.D.2) “execute [i.e., *use*] an interrupt call on the [*local*] operating system *kernel* of a client computer” (which are *servers* in Schmidt-Tomasov-Calder) to execute on non-native OSs. *See* Calder, [0150]. Bhattacharjee, [0327]-[0328].

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.E.1.c.iii, [1A]), each Schmidt-Tomasov/Schmidt-Tomasov-Calder VCE-capsule (*container*) comprises a *set of associated system files*, like libraries and configuration files. Schmidt-479, [0048]-[0052]; Tomasov, [0026]; Calder, [0077], [0082], [0094], [0101]. Moreover (*see supra* §V.E.1.c.iv), each server’s VCE-capsules use the server’s OS’s *local kernel*, and thus *system files* within each VCE-capsule *are compatible with the local kernel of at least that server’s “single operating system.”* Tomasov, [0017]. Bhattacharjee, [0329]-[0331].

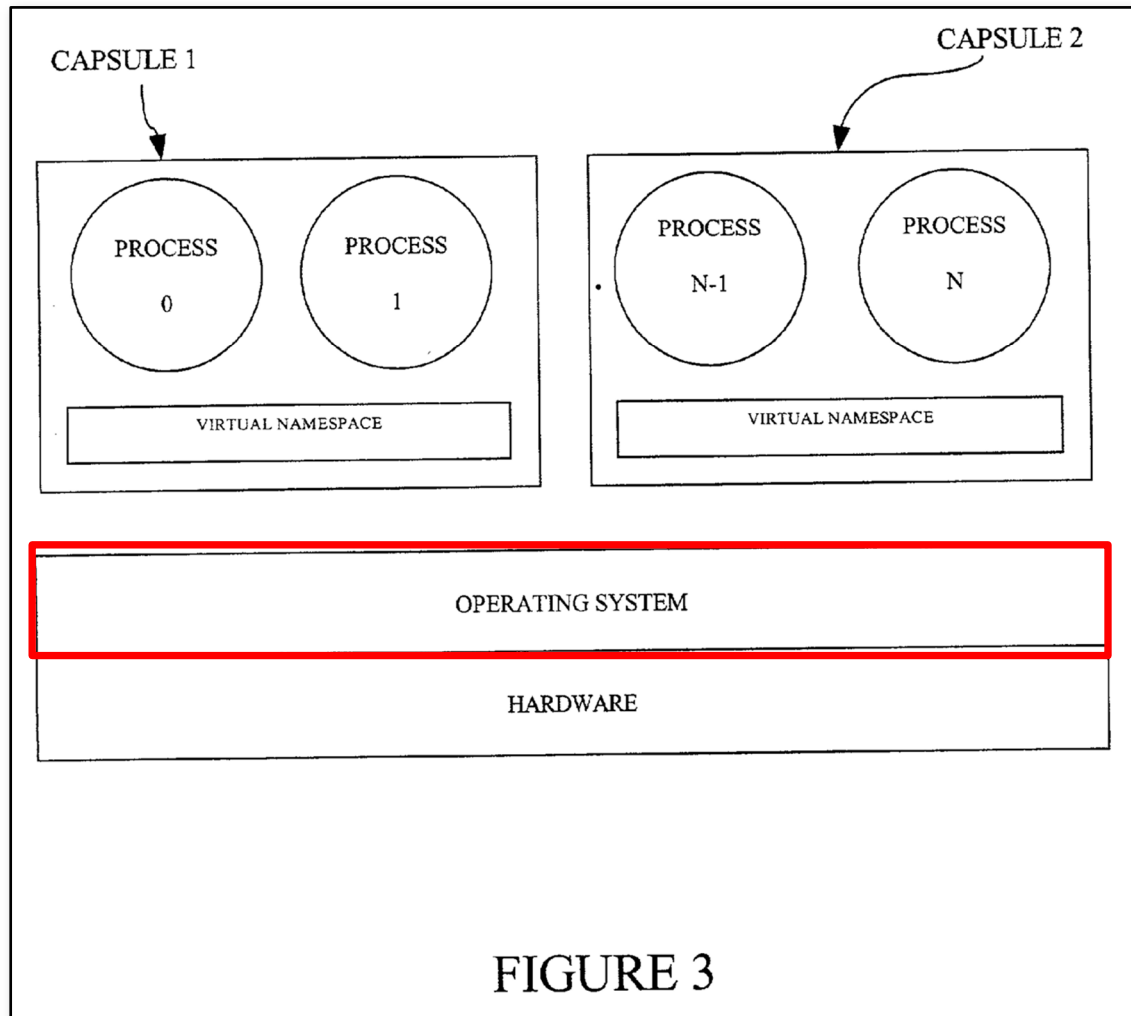
Furthermore, Schmidt-Tomasov/Schmidt-Tomasov-Calder VCE-capsules are compatible with at least two different OSs (*see* §V.E.1.a.i, [1PREA]), and thus their *system files* are also compatible with *the plurality of different* OSs under each

party's construction of "*at least some of the different operating systems*" (EX1106, 4; EX1100, 4; EX1101, 2). Schmidt-479, [0044] (capsules can be "migrated to a **different instance of an *operating system*** on a different machine"); EX1116, [0015] (capsules can be moved to "a **different *operating system***"); EX1009, [0018] (the "capsule is entirely self-contained and can be suspended in secondary storage, [and] arbitrarily bound to different machines and **different *operating systems***...."); Calder, [0101] (modifying copies of system files to execute an application on "any non-native [OS] platform" and thereby utilize non-native kernels). Bhattacharjee, [0332]-[0336].

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

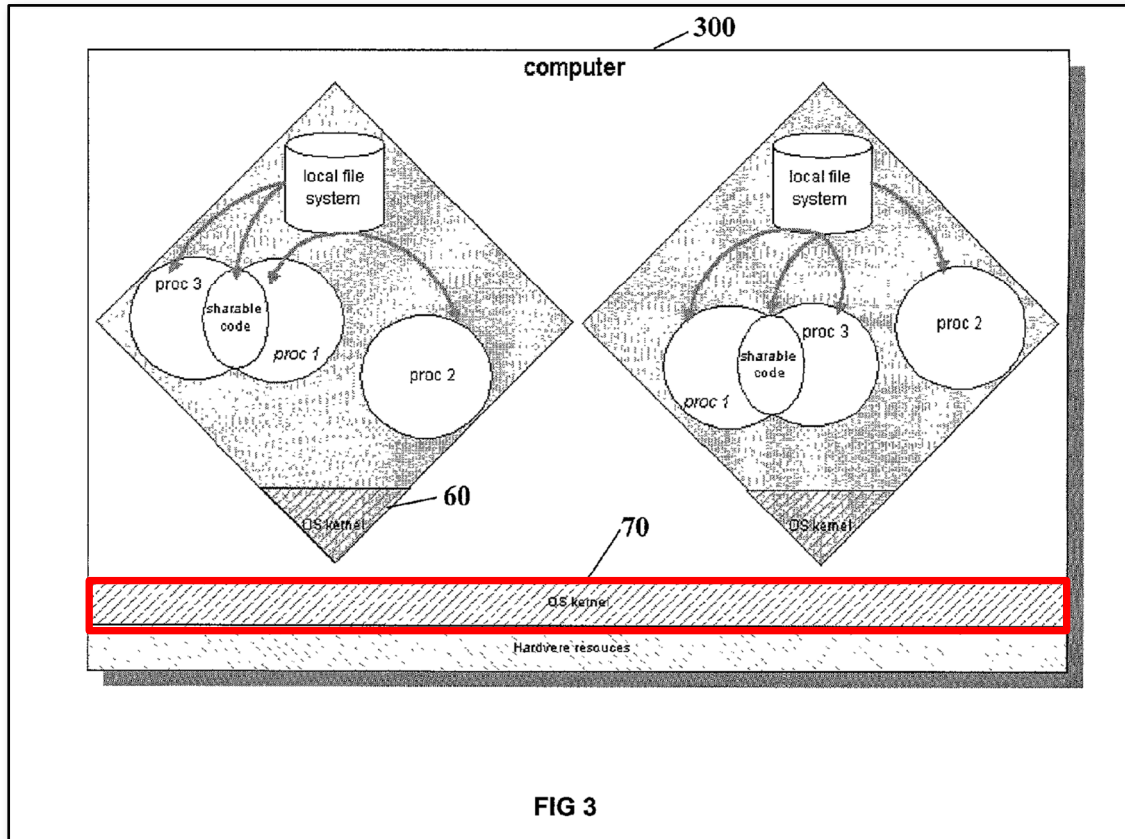
Schmidt-479's Figure 3 (color added below) shows the OS (red) and thus the OS kernel is *excluded* from capsules. Bhattacharjee, [0337]-[0338] (citing

EX1086, 2:53-55 (“the *operating system* is commonly called the *system kernel* [], or just the *kernel*”).



Tormasov’s Figure 2 (color added below) similarly shows the kernel (red) outside (*excluded* from) VCEs. Tormasov, [0014]-[0016] (distinguishing prior-art

techniques requiring “several...*kernels*”), [0025] (“processes...are shared...inside an ordinary *single kernel*” OS).

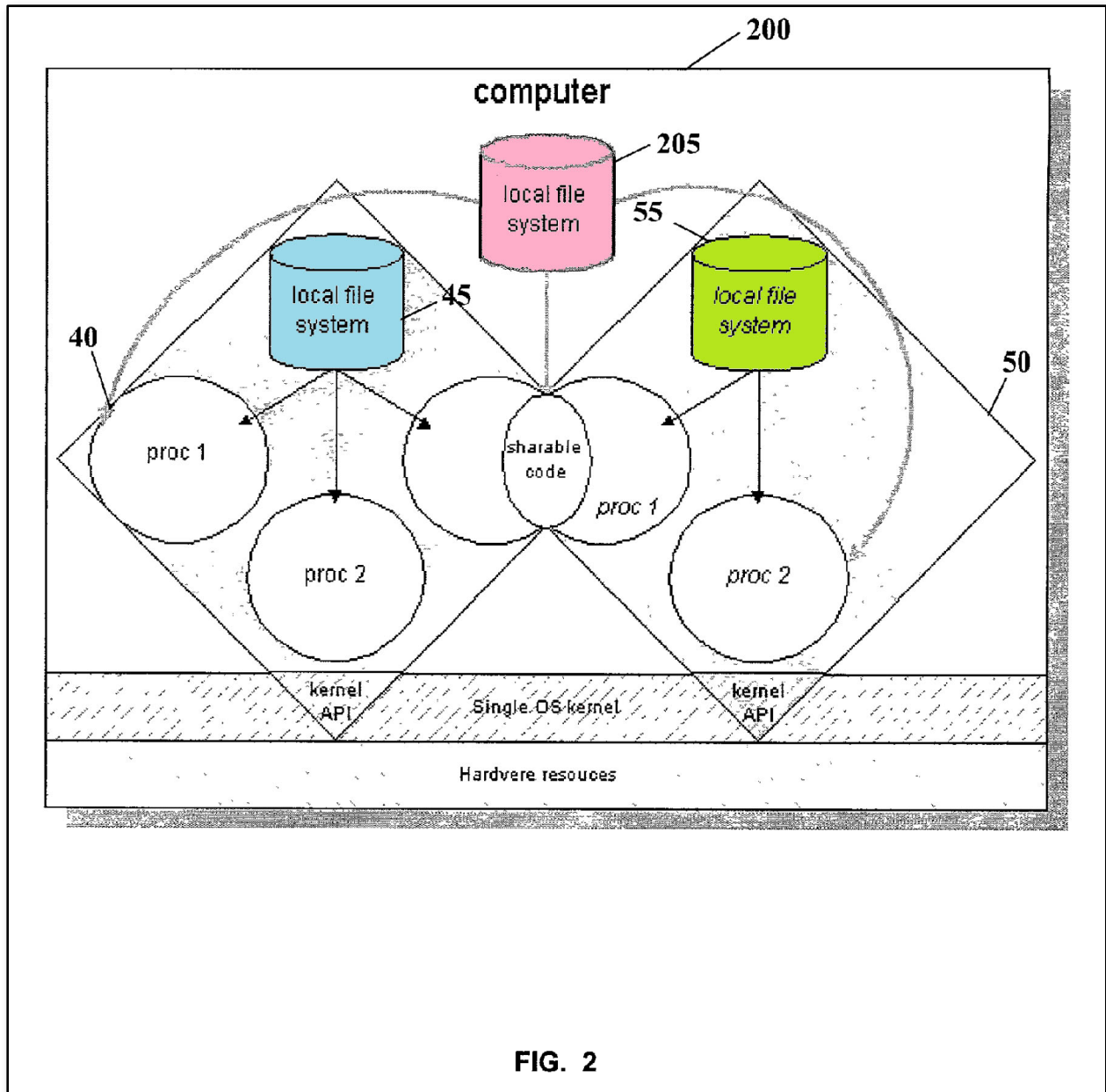


Thus, Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules *exclude a kernel*. Bhattacharjee, [0339]-[0340], [0260]-[0277]. *See also supra* §§V.D.1-2; §V.E.1.c.i.(2).

- 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,”

Tormasov’s Figure 2 (highlighted below) shows a “common *file system*” (pink) *that remains resident on the server*. Tormasov, FIG. 2, [0026]. POSAs

understood that a typical, obvious implementation of a conventional file system for a conventional, prior-art OS comprises *associated local system files* needed to execute the OS and to execute applications that are not in VCEs (and thus execute without accessing a VCE's local file system). Bhattacharjee, [0341]-[0343] (citing EX1029, [0027] (the “minimum...**OS components include[] a kernel...and *system files*”))). POSAs also understood that a “kernel API” like in Tormasov's Figure 2 is used, *e.g.*, to make system calls to the OS, which are activated and processed by libraries (*system files*) *that remain resident on the server*. Bhattacharjee, [0344]-[0345] (citing EX1086, 2:37-3:11).**



EX1122 (cited in Tormasov, [0035]) corroborates that an obvious implementation of a local file system in each VCE includes “cop[ies]” of “the files that” the application(s) (e.g., “ftpd”) “needs,” including *system files*, like various library files (“lib”) and configuration files. EX1122, 122-123; *see* EX1001, 2:55-3:13 (identifying the same Unix-OS files as “*system files*”). Bhattacharjee, [0346].

As discussed above (§V.E.1.c.iii), each VCE-capsule (*container*) installs *associated system files* to a “local file system” in the VCE-capsule (light blue and green above), which are *stored in memory* and *utilized* by each VCE-capsule’s application(s) (as illustrated by arrows running from the local file systems to application processes). Tormasov, [0026], FIG. 2. Bhattacharjee, [0346].

As discussed above (§V.B), Tormasov allows for “installation of programs....and other *applications*” inside a VCE, where the applications are only accessible to that VCE. Tormasov, [0024] (“users...can work with the same hardware node [] without noticing each other”). Users may also “stop and start the virtual computing environment in the same manner as with a common operating system.” Tormasov, [0025]. Thus, POSAs understood that when an application is installed to a VCE, the application uses the *associated system files* inside the VCE (e.g., blue above) *in place of the associated local system files that remain resident on the server* (pink above). Bhattacharjee, [0347].

Based on Schmidt-479, each VCE-capsule further includes its own “libraries” (*system files*) that allow the capsule (and its application(s)) to be stored and started/restarted on different servers. Schmidt-479, [0048], [0051]. Bhattacharjee, [0348]-[0355] (citing EX1077, 122).

Based on Calder, Schmidt-Tormasov-Calder’s VCE-capsules include additional, modified copies of *system files* – “libraries” and “configuration files”

(Calder, [0082]) – that *are utilized in place of local system files* to allow VCE-capsules (and their application(s)) to execute on non-native OSs. Calder, [0082]. Bhattacharjee, [0356]-[0357].

Thus, Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s *system files* meet [1D].

g. [1E] “wherein said associated system files...are copies or modified copies of the associated local system files...,”

As discussed above (§V.E.1.f, [1D]), Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s servers comprise respective “local file system[s]” (Tormasov, FIG. 2, [0026]) with *system files* (Tormasov, [0026]-[0031]; Schmidt-479, [0048], [0051]; Calder, [0082]).

When the VCE-capsule’s application(s) are native to the server’s local OS, at least some of a VCE’s *system files* are *copies of the server’s local system files* (e.g., identical libraries used to make that OS’s system calls), meeting [1E] under the apparent claim scope VirtaMove applies in district court. *See* EX1079, 30 (VirtaMove arguing [1E] is met because allegedly “the host OS and container will use one or more identical system files, for example when both...incorporate the same Linux distribution version, or...use the same version of libc,” a conventional Unix library). Bhattacharjee, [0358]-[0360] (citing EX1122 (cited in Tormasov, [0035]), 122-23 (when “using chroot() to restrict users’ access,” as in Schmidt-

Tomasov/Schmidt-Tomasov-Calder (*see infra* §V.E.1.i, [1G]), “you need” “*copies*” of “the files that” the application(s) “needs,” including various library *system files*)).

Based on Schmidt-479, Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s VCE-capsules (*containers*) can be installed to/restarted on any server by having “portions of” a first server’s “underlying file system” “map[ped, i.e., *copied*]...into their namespace.” Schmidt-479, [0045]-[0046], [0052] (“mounts of directories such as /lib” are “*copied*” into each user’s view of the file system); *see also* Tomasov, [0018] (noting that VCEs are also “separat[ed]” based on “namespace[s]”). Bhattacharjee, [0361].

“[T]he default set of file system view mappings includes standard system directories, such as.../lib for *system* libraries” (*system files*). Schmidt-479, [0051]. Schmidt-479 also provides a “*copy*-on-write mode of file access to support the modification of read-only or *system files*,” where editing those *system files* creates a *copy* in the user’s environment (Schmidt-479, [0046]), which Schmidt-Tomasov/Schmidt-Tomasov-Calder stores in VCE-capsules. Thus, Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s VCE-capsule comprises *associated system files* that are *copies of the associated local system files that remain resident on the server*. Bhattacharjee, [0362]-[0366] (citing EX1077, 122).

Moreover, based on Calder, Schmidt-Tomasov-Calder’s VCE-capsules include “*modified*” system files (e.g., “*modified* libraries [and], *modified* configuration files,” Calder, [0082]) that are based on “*cop[ying]* specific information from the **existing** [*local*] **system registry**” (Calder, [0121]). Bhattacharjee, [0367]-[0368].

Thus, Schmidt-Tomasov/Schmidt-Tomasov-Calder meets [1E].

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

Users accessing VCEs “can work with the same [server] hardware node [] without noticing each other, just as if they worked on totally separate computers with no associated hardware.” Tomasov, [0024]. Even when “two processes in different” VCEs are “started for execution from one file (for example from the shared file system) they would be **completely isolated from each other.**” Tomasov, [0026]. A VCE’s “processes are never visible to other” VCEs “running on the same computer.” Tomasov, [0030]. Similarly, each VCE’s “root file system...allows a root user of every” VCE “to make file modifications and configure their own local parameters of the” OS, where “changes done in the file system in one” VCE “do not influence the file systems in the other” VCEs. Tomasov, [0030]-[0031]. Bhattacharjee, [0369].

Because (i) users can run and configure applications in their VCEs as if they were using separate computers, (ii) each VCE's processes (e.g., executing applications) are **completely isolated**, and (iii) VCEs can be stopped and started independently, POSAs understood that *the application software* within a VCE-capsule (*container*) *cannot be shared between the plurality of secure containers of application software*. Bhattacharjee, [0370]-[0371].

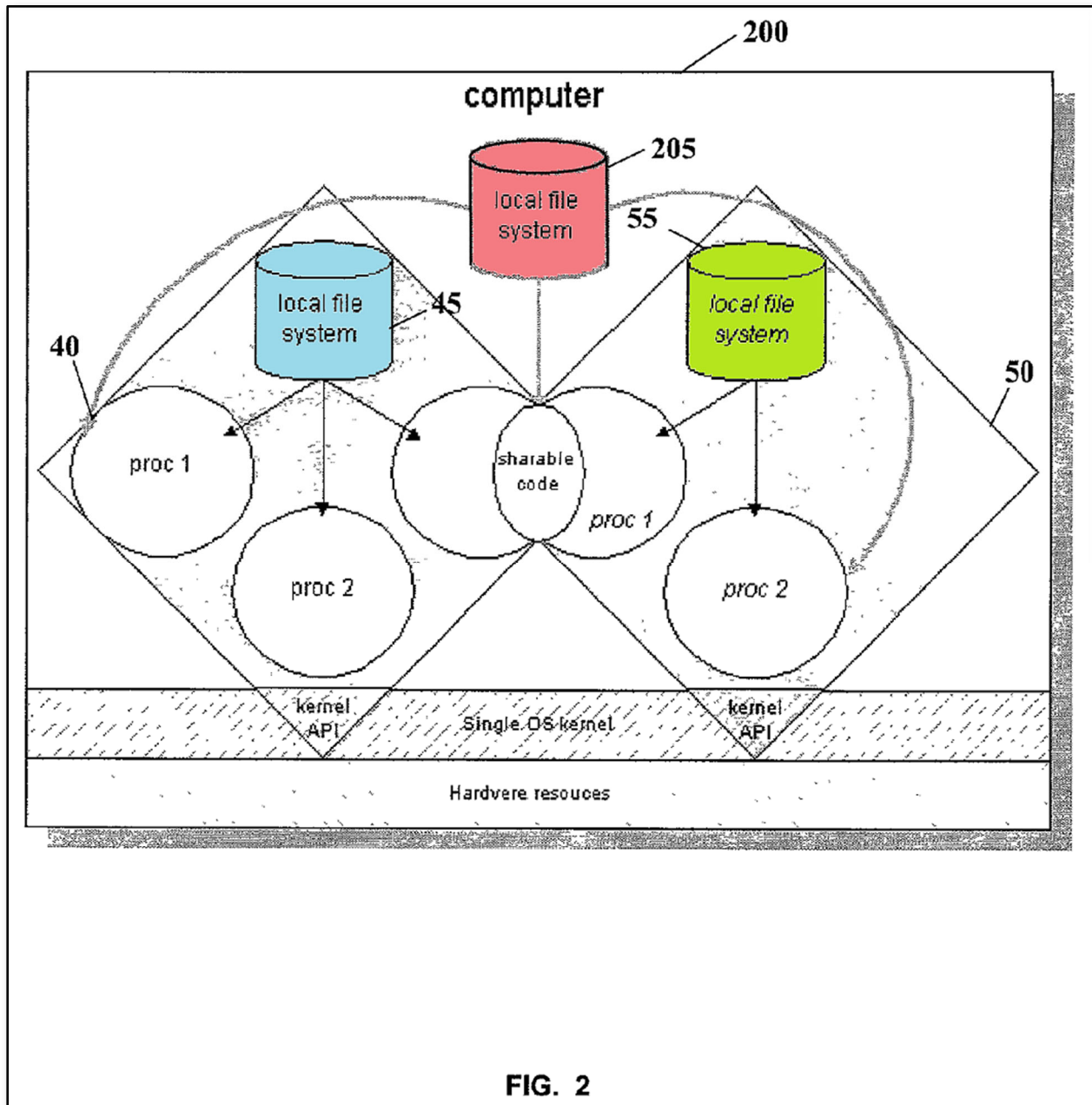
Moreover, as discussed above and further below (§V.E.1.i. [1G]), each VCE-capsule has a respective, unique root directory where applications are installed, which teaches (or at minimum renders obvious) that applications cannot be shared across VCE-capsules (because they are confined to different root directories). Tormasov, [0030]-[0031]; Bhattacharjee, [0372]-[0374] (citing EX1075, 2:9-15 (“sandbox[ed]” applications have “access only to the constrained environment and **cannot corrupt software applications outside it**, i.e. beyond the sandbox boundary”)).

Thus, Schmidt-Tormasov/Schmidt-Tormasov-Calder meets [1F].
Bhattacharjee, [0375].

- 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.**”

Based on Tormasov, each VCE-capsule “has a **completely independent root file system.**” Tormasov, Abstract. As Tormasov’s FIG. 2 (highlighted below)

illustrates, “[e]ach of...two [VCEs] 40, 50 has its own *unique file system* 45, 55 and each [VCE] can also see the common file system 205” (the computer’s *operating system’s root file system*). Tormasov, [0026].



Thus, each VCE’s local *root file system* (e.g., 45, 55) is different from the server OS’s *root file system* 205, 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, [0376]-[0379].

Moreover, a VCE’s “**root file system**...allows a root user” of the VCE “to make file **modifications** and **configure** their own local parameters,” but “**changes** done in the file system in one [VCE] **do not influence** the file systems in the other [VCE].” Tormasov, [0030]-[0031], [0018]. Because VCEs on the same computer (e.g., VCEs 40, 50) access the computer’s *operating system’s root file system* 205 but cannot see changes made in the other VCEs’ *unique root file systems* (e.g., VCE 50 cannot see VCE 40’s file system 45), POSAs would have understood this to teach that the changes made to one VCE’s *unique root file system* (e.g., file system 45) are not made to (and thus make it *different from*) the computer’s *operating system’s root file system* 205. Tormasov, [0029]-[0031]; Bhattacharjee, [0380]. This also meets VirtaMove’s interpretation of [1G]. See EX1103, 9.

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

VCE-capsule’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.

Bhattacharjee, [0381]-[0382] (citing, Tormasov, [0018] [0023]).

Additionally, Schmidt-479’s capsule technique (used in Schmidt-Tormasov/Schmidt-Tormasov-Calder, *supra* §§V.D.1-.2) uses “a **chroot** system call [that] re-assigns the *root* of the file system to be the *root* of the file system view, which gives capsules” “the illusion of a private file system.” Schmidt-479, [0052] (“it is not desirable” for users “to modify the underlying local-level files (for which they may not have permission)”), [0013], [0020]. Similarly, Tormasov teaches that a VCE’s “*unique file system*” provides a changed root environment (that could be implemented using a chroot system call, as in Schmidt-479) because, as discussed above in this section, the *root file systems* of VCEs on the same computer are isolated and invisible to each other. Tormasov, [0026], FIG. 2. Bhattacharjee, [0383]-[0386] (cites also EX1079, 40-51).

Thus, each VCE-capsule meets VirtaMove’s apparent claim scope in district court, which alleges that a *container* assigned to a “**chroot**” file system meets [1G]. EX1079, 40-51 (emphasis original); EX1120 (cited in Tormasov, [0033]), 109-110 (“*chroot* system call” means “[p]rocesses can change their notion of the file system root”) (emphasis original). Bhattacharjee, [0387]-[0392].

2. Claim 2: “[C]laim 1...each container has an execution file associated therewith for starting the...applications.”

POSAs knew as part of their background knowledge that conventional prior-art OSs had at least one *execution file associated therewith for starting applications*, e.g., upon start-up of the OS. For example, EX1041 evidences POSAs’ knowledge that Windows-based OSs have their “own initialization files” that contain “a list of **application programs to start automatically.**” EX1041, 2:8-17. Similarly, EX1013 evidences POSAs’ knowledge that a typical prior-art OS has “start-up data maintained by the operating system” that identifies “what **applications to load** [*start*] and the order in which the *applications* are **loaded** [*started*] by the operating system.” EX1013, [0003]. POSAs knew that those conventional OS initialization files and start-up data are, or typically were processed by, *execution files for starting the application(s)* identified in those files/data. Bhattacharjee, [0393]-[0397] (citing also EX1027, [0070]-[0073]).

Each Tormasov VCE is an “environment that is functionally equivalent to a computer with a full-featured operating system” that allows for “command shells, compilation and installation of programs, configuration of network services,” etc. Tormasov, [0023]-[0024]. Tormasov further discloses that a user “may stop and **start** the virtual computing environment **in the same manner as with a common operating system.**” Tormasov, [0025]. POSAs would thus have found it obvious to use a conventional *execution file to start* a Schmidt-Tormasov/Schmidt-

Tormasov-Calder VCE-capsule and, once the VCE-capsule was started, a conventional *execution file* to start the VCE-capsule's *application(s)* therein, in the same manner as was done when starting common prior-art OSs, as discussed above. Bhattacharjee, [0398]-[0399].

Additionally, VirtaMove alleges claim 2 is met by an “image configuration” that “includes information such as application arguments, environments, etc.” EX1079, 51-55. Based on Schmidt-479's teachings, each VCE-capsule includes “[s]ystem **environment information**,” such as “privileges, configuration settings, working directories and files, assigned resources, open devices, installed software, and internal program state.” Schmidt-479, [0038]. POSA would have found it obvious for that environment information to be read by an *execution file associated therewith for starting applications* – e.g., an *execution file for starting application(s)* and configuring the various settings, privileges, etc. that the application(s) need to execute properly. Bhattacharjee, [0400]-[0402] (citing EX1084, [0145]-[0153]).

Schmidt-Tormasov/Schmidt-Tormasov-Calder meets claim 2 in either of the above ways. Bhattacharjee, [0403].

3. Claim 3

Schmidt-Tormasov/Schmidt-Tormasov-Calder meets claim 3 for the same reasons discussed above (§V.E.2) that POSAs had motivation and expected

success to include *execution files* that, as discussed above, *included instructions related to an order in which executable applications will be executed* within VCE-capsules (e.g., to first start applications that other applications need to execute properly). Bhattacharjee, [0404]-[0409] (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” before [1A]’s “storing step.”

As discussed above (§V.E.1.c.i.(2)), Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules are transportable *containers* of software applications stored in memory accessible to servers before being installed to a server (*see supra* §§V.E.1.c.i.(1)); POSAs would have understood this involves *pre-identifying the application(s) and system files required for association* with the VCE-capsules (*containers*) *prior to said storing step* in [1A]. Bhattacharjee, [0410]-[0415] (citing EX1084, [0145]-[0150]).

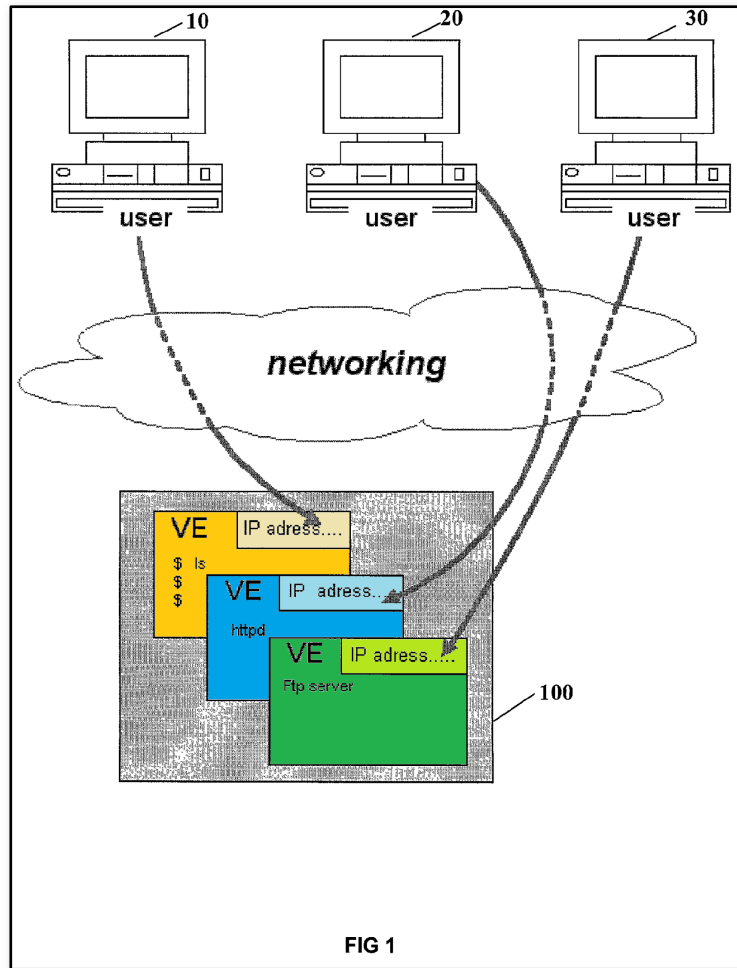
Based on Calder, Schmidt-Tormasov-Calder VCE-capsules stored in memory accessible to servers also have *pre-identified* applications and associated system files (Calder, [0082]), for use on servers running non-native OSs (Calder, [0073]), which also meets claim 4. Bhattacharjee, [0416]-[0417].

5. Claim 5

Tormasov “separate[es]” VCEs by *assigning each* a “namespace” ([0029]), which is a *container specific identity*. POSAs would *modify at least some of the associated system files in VCE-capsules to provide an association* with their namespace so applications could identify/communicate with applications outside their VCE. Bhattacharjee, [0418]-[0419].

6. 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.”

Tormasov’s Figure 1 (highlighted below) shows that each VCE is *assigned* its own “**IP address**” meeting claim 6: e.g., the VE executing an httpd application is accessed at one *assigned IP address* (light blue) and the VE executing the Ftp server is accessed at a different *assigned IP address* (light green). Bhattacharjee, [0420].



Tormasov’s VCEs allow different users of the same server hardware to deploy, e.g., independent, different web servers. Tormasov, [0011], FIG. 1. The users “perceive” themselves as having “obtained **full network root access** to a common computer” (Tormasov, [0023]) and “can perform all actions allowed for the ordinary computer,” such as “**configuration of network services**” and working “with the same hardware node” “as if they worked on **totally separate computers with no associated hardware**” (Tormasov, [0024]). Hosting independent web-server applications and configuring independent “network services” as if they each

had “no associated hardware” further teaches (or at least renders obvious) that each VCE is *assigned a unique identity that includes at least an IP address*, because POSAs knew that different web servers were conventionally (and beneficially) assigned different IP addresses. Bhattacharjee, [0421]-[0423] (citing EX1130, 2:6-14; EX1131, 1:27-56).

Based on Schmidt-479 (*supra* §§V.D.1-2), the server’s “physical resources” used by a VCE-capsule (e.g., “Internet addresses”) are assigned “unique” “tokens.” Schmidt-479, [0054]-[0054]. That also motivates *assigning a unique Internet (IP) address* to facilitate moving capsules to different machines, as Schmidt-479 teaches (Schmidt-479, [0058]). Bhattacharjee, [0424]-[0425].

POSAs would have assigned VCE-capsules additional unique identifiers – e.g., *host name* and *MAC address* – to support communication protocols that used those unique identities (e.g., MAC addresses for Ethernet networks; hostnames for remote access). Bhattacharjee, [0426]-[0431] (citing EX1039, [0002]; EX1019, [0012], [0023]; EX1130, 2:16-14; EX1131, 1:27-56).

7. Claim 7

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, [0008]. Schmidt-479 “create[s]” “*mount points* [that] are established for each of

the required directories” mapped to a capsule. Schmidt-479, [0052]. “[O]nce *mounts* have been established, a chroot system call re-assigns the root of the file system to be the root of the file system view.” Schmidt-479, [0052]. POSAs would have *modified* similar configuration files (*system files*) to *define container specific mount points associated with the VCE-capsules* as an obvious, conventional way to mount each VCE-capsule’s local file-system view. Bhattacharjee, [0432]-[0435] (citing EX1018, [0044]).

8. Claim 8

Calder teaches a “**server that transmits *application* packages to the member computers**” (Calder, [0075]), which motivates (and provides reasonable expectation of success in) *retrieving* Schmidt-Tormasov-Calder’s VCE-capsules (*secure containers of applications and associated system files, supra §V.E.1.c*) *from a computing system (e.g., a server) storing (having) those VCE-capsules.* Bhattacharjee, [0436]-[0444] (citing, e.g., EX1084, [0162]-[0166], FIGs. 11A).

Moreover, when VCE-capsules are executing on a server having multiple VCE-capsules (*e.g., Tormasov, [0017], [0027], FIG. 1*), each VCE-capsule is *retrieved from that computer system*, as claimed. Bhattacharjee, [0445]-[0446].

9. 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 any suitable application package to manage its use of *associated computer hardware resources*, like “**memory**,” “**network bandwidth**,” and “**disk usage**.” Calder, [0125]-[0126]. The “grant[ed] allocation” of resources is “predicted” using “heuristics” (e.g., “amount of virtual memory currently being consumed”). Calder, [0126]. Bhattacharjee, [0447]-[0448].

Schmidt-Tomasov’s/Schmidt-Tomasov-Calder’s VCE-capsule share a server’s “**resources**” (e.g., hardware, network bandwidth, shared memory), but are isolated and thus unaware of competition for those *resources*. Tomasov, [0025]-[0026]. Bhattacharjee, [0447]-[0448].

Thus, based on Calder, POSAs had motivation for Schmidt-Tomasov/Schmidt-Tomasov-Calder to allocate to (*associate with*) VCE-capsules *hardware resource usage* – using *information related to at least one of CPU memory, network bandwidth, and disk allocation* – before *execution* of the *applications within* VCE-capsules, to ensure those applications had sufficient *resources* to execute properly. Bhattacharjee, [0449]-[0459] (citing EX1036, 1:16-26; EX1063, 1:52-2:62, 4:41-46; EX1027, [0176]; EX1037, 6:29-38; EX1076, 13; EX1083, [0035]-[0040]; EX1094, 2:5-21). POSAs would have reasonably

expected success because using such *information* to pre-allocate resources to applications/processes was conventional, as the cited corroboration demonstrates. Bhattacharjee, [0460]-[0461].

10. 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.”

Schmidt-479 teaches “it is **not desirable**” to permit users sharing one computing device “to modify the underlying local system-level files” (e.g., modifying “/lib for system libraries”). Schmidt-479, [0051]-[0052]. To prevent unauthorized modifications, Schmidt-479 executes a “**chroot** system call [to] re-assign[] the root of the file system.” Schmidt-479, [0052]. Thus, *applications residing within a capsule have no access to system files or applications in other capsules or to system files within the server’s OS*. Bhattacharjee, [0462].

Similarly, Tormasov teaches another way of preventing applications from having *access to system files or applications in other* environments by ensuring “processes [*executing applications*] in different” VCEs are “**completely isolated** from each other” ([0026]) and “are **never visible to other**” VCEs “running on the same computer” ([0018]). Similar to Schmidt-479’s chroot, Tormasov’s VCEs have an “independent” “root file system” comprising respective *system files*, as discussed above (§V.E.1.i, [1G]). Thus, a VCE’s *application has no access to*

system files or applications in other VCEs. Bhattacharjee, [0463]. *See also supra* §V.E.1.c.i.(3) (demonstrating VCE-capsules are *secure containers*).

Thus, Schmidt-479 and Tormasov each teach, or at minimum render obvious, techniques for preventing *applications residing within* VCE-capsules from accessing *system files within the operating system during execution thereof* to protect the server OS's system files from corruption by users and/or applications executing in VCE-capsules. Permitting a VCE-capsule's "root user" to access the OS's system files would contradict Tormasov's desire for VCEs to "not influence" each other (Tormasov, [0031]). Bhattacharjee, [0464].

Additionally, Schmidt-479 and Tormasov separate applications using namespaces. Schmidt-479, [0041]; Tormasov, [0018], [0029]. POSAs knew that a "namespace" can "isolate software packages" so that "any files and components installed in that namespace are visible to the application while files and components installed **in other namespaces are not.**" EX1015, [0080], [0086]; Bhattacharjee, [0465]-[0470] (citing also EX1076, 5; EX1077, 120). *See* EX1079, 59 (VirtaMove arguing "namespaces" "isolat[e] processes in an environment"). Thus, POSAs reasonably expected success in beneficially using namespaces in Schmidt-Tormasov/Schmidt-Tormasov-Calder to prevent applications from *accessing system files or applications in other VCE-capsules (containers)*.

11. Claim 11

In Schmidt-479, capsule data (e.g., files) are *stored in network file storage*. Schmidt-479, [0063]-[0070]. Calder similarly discloses application “files...***stored remotely***.” Calder, [0139]. Bhattacharjee, [0471]-[0472]. Those disclosures motivate implementing Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules to *include files stored remotely in network file storage* to provide the well-known benefits of remote access. Bhattacharjee, [0473] (citing EX1092, 1:21-30).

Another reference by a Schmidt-479 inventor discloses a remote “[c]apsule directory service” that manages and “keep[s] track of” capsule “unique locators” (e.g., “IP address[es]”). EX1116, [0057]. POSAs had motivation to store similar *parameters* for VCE-capsules in a similar remote directory, to facilitate management thereof. Bhattacharjee, [0474]-[0476].

12. Claim 12

As discussed above (§V.E.8), POSAs would retrieve VCE-capsules (*containers*) from network-server storage, and thus had motivation to merge retrieved files with claim 11’s parameters (§V.E.11) *to affect* claim 1’s *step of storing* (§V.E.1.c.i, [1A]), by using *parameters forming descriptors of containers* (§V.E.11), such as information about where to store VCE-capsules. Bhattacharjee, [0477]-[0479].

13. Claim 13: “[C]laim 1...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.”

EX1114, filed by Tormasov’s inventors, corroborates POSAs’ background knowledge that distributed storage systems used “data storage algorithms” based on tracking a history of “*resource* utilization” to “provide optimate data storage” for distributed systems. EX1114, [0018]. Additionally, Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s servers run conventional prior-art OSs like Windows- and Unix-based OS (*see supra* §V.E.1.a.i); those OSs typically included tools for measuring/*tracking* processes related to applications, including tracking statistics for *resource allocation and monitoring the status of application(s)*. Bhattacharjee, [0480]-[0488] (citing, e.g., for Windows-based OSs, EX1033, 2:62-3:18; e.g., for Unix-based OSs, EX1060, 5:57-61; for generic OSs, EX1121, G-14).

POSAs had motivation and would have expected success to use those (or other conventional tracking) tools in Schmidt-Tormasov/Schmidt-Tormasov-Calder to track the history of processes in VCE-capsules for the same reasons POSAs used those tools to track process history (e.g., to project server-resource needs). Bhattacharjee, [0489]-[0492] (citing EX1093, 1:66-3:6; EX1121, G-5).

14. Claim 14

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

As discussed above (§V.E.4), Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules (*containers*) contain pre-identified applications and system files stored therein.

Based on Schmidt-479, VCE-capsules are “stored off-line, for instance on a disk drive[’s]” *memory* accessible to servers on which the VCE-capsule will be started/restarted. Schmidt-479, [0038]. Thus, each VCE-capsule is *created prior to storing* the capsule *in memory* of one storage device from which it is subsequently *stored in memory* accessible to a destination server that executes the VCE-capsule. Bhattacharjee, [0493]-[0494].

Similarly, users can log-in to a server providing multiple VCEs. Tormasov, FIG. 1, [0017]. From this teaching, POSAs would have understood that the VCE-capsules (and applications therein) are *created prior to storing in memory* so they are available to users when accessing the server. Bhattacharjee, [0495]. Tormasov also says users “may stop and start the [VCE] in the same manner as with a common operating system” ([0025]), from which POSAs would have understood that VCEs are stored in memory when stopped, and retrieved when started. Another reference by Tormasov inventors (EX1111) similarly discloses “computers with a set of **installed** virtual environments” (5:38-41), which further

corroborates POSAs' understanding that VCEs (and applications and system files contained therein) are created before *storing in memory accessible* to the server.

Bhattacharjee, [0495]-[0497].

Calder's "transferr[ing]" of pre-packaged applications "from a server to the client" (Calder, [0078]), provides additional teaching to have Schmidt-Tomasov's/Schmidt-Tomasov-Calder's VCE-capsules *created prior to the step of storing containers in memory* accessible to the servers retrieving VCE-capsules.

Bhattacharjee, [0498]-[0499].

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

To create capsules, Schmidt-479 *runs* applications processes and related system files and stores them in capsules that can be started/restarted on the same or a different server. Schmidt-479, [0044]-[0048]. Bhattacharjee, [0500]-[0502] (citing EX1040, 1:14-25; EX1099, [0104]).

As discussed above (§§V.E.1.b.ii-iv), applications in Schmidt-Tomasov's/Schmidt-Tomasov-Calder's VCE-capsules relate to *a service* (e.g., an httpd (web-server) *service*). *See* Tomasov, FIG. 2 (executing various *services* in VCEs). Based on Schmidt-479 and/or POSAs' background knowledge, POSAs had motivation and reasonable expectation of success to create Schmidt-Tomasov/Schmidt-Tomasov-Calder VCE-capsules by *running an instance of a* VCE-capsule and its application(s) *on a server* before encapsulating them so as, as

explained below (§V.E.14.c, [14B]), to identify the system files required to provide that service, and to test the VCE-capsule (and the applications therein) for compatibility with the various server hardware-OS combinations in the system. Bhattacharjee, [0503]-[0509] (citing EX1129, 1:11-20; EX1031, [0064]-[0068]; EX1020, [0007]).

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

As discussed above (§V.E.14.b, [14A]), based on Schmidt-479, POSAs had motivation to run applications on a server to *determine which files* (e.g., binaries, libraries, configuration files, etc.) *are being used* so the VCE-capsule could be stored and properly executed on any server in the system. Bhattacharjee, [0510]-[0511].

Alternatively, POSAs would have had additional motivation to identify *which files are being used* by an application as a conventional way to *determine which files* were required to execute applications in isolated computing environments like VCE-capsules to ensure applications were beneficially sandboxed (i.e., free to operate without harming the underlying server). Bhattacharjee, [0512]-[0514] (citing EX1077, 120-122 (“Sandboxing Applications” into secure directories to protect computers from applications – like Schmidt-Tormasov/Schmidt-Tormasov-Calder protects servers from applications

in VCE-capsules (*supra* §V.E.c.iv) – first by “[r]unning the application” to identify required system files)).

- 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.”**

As discussed above (§§V.E.1.c.i-iv), Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules include *copies of applications and system files* required to execute the applications, so the VCE-capsule can be copied to and executed on multiple servers in the system, thereby providing independent instances of the applications (e.g., multiple web-server applications). Each encapsulated VCE (and applications and system files therein) is a *second instance* of that data beyond the computer’s local first instance or another VCE’s instance, as claimed. Bhattacharjee, [0515]-[0517]; *see* EX1079, 67 (VirtaMove arguing claim is met by storing “different instance[s]” of an alleged *container*).

15. Claim 15

- a. **[15A]**

See supra §V.E.14, §V.E.6. Bhattacharjee, [0518]-[0519].

- b. **[15B]**

As discussed above (§V.E.1.a.i), Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules (*containers*) are *installed on “server system[s].”* Tormasov, [0017]; Schmidt-479, [0064]-[0067]. Bhattacharjee, [0520]-[0521].

c. [15C]

POSAs would *test the applications and files within* each VCE-capsule “to ensure proper operation...on several different various operating systems” in the system. EX1129, 1:11-20. Bhattacharjee, [0522]-[0525].

16. Claim 16

a. [16A]

See supra §V.E.14.a. Bhattacharjee, [0526].

b. [16B]

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

Schmidt-479 can create capsule files using a ““copy-on-write mode,”” which “modif[ies] the file within their virtual namespace only, while **the underlying file remains unchanged.**” Schmidt-479, [0019], [0045]-[0048]. Those “unchanged” “underlying file[s]” are *a skeleton set of system files that serve as a starting point* for creating VCE-capsules. Bhattacharjee, [0527]-[0531] (citing EX1077, 119-122).

17. Claim 17

a. [17A]

As discussed above (§V.E.1.b.ii), Schmidt-Tormasov/Schmidt-Tormasov-Calder makes a VCE-capsule’s *service* available by *installing* it (e.g., an httpd (web-server)) *on a target server selected from one of the system’s plurality of*

servers. Tormasov, FIG. 1. Bhattacharjee, [0532]-[0537] (citing EX1014, [0015]; EX1020, [0069]-[0078]).

b. [17B]

Tormasov wants to reduce “time and effort setting up and administering” “complex software packages,” including “administration of a remote node on a data storage network.” Tormasov, [0011]-[0012]. Bhattacharjee, [0538].

Thus, POSAs had motivation and reasonably expected success to *install* and configure VCE-capsules using a GUI that *associates 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* because of the well-known convenience of using conventional GUIs (e.g., supporting drag-and-drop installation operations using icons). Bhattacharjee, [0539]-[0546] (citing, e.g., EX1014, [0030]-[0031]).

18. Claim 18

Tormasov teaches to reduce “time and effort setting up and administering” software packages on a “*remote* node” ([0011]-[0012]), which suggests/motivates system administrators configuring VCEs on *target servers* from remote locations (e.g., headquarters) using a conventional *remote*-administration GUI.

Bhattacharjee, [0547]-[0552] (citing EX1014, [0030]; EX1020, [0024], [0046], [0108]; EX1109, [0002]-[0005]).

19. Claim 19

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 Schmidt-Tormasov/Schmidt-Tormasov-Calder meet claim 19 the same way they meet claim 18 (§V.E.18). Bhattacharjee, [0553]-[0554].

If claim 19 requires computers with different OSs and/or hardware, POSAs knew that was typical for servers in large organizations. Bhattacharjee, [0555]-[0557] (citing corroboration); *supra* §§V.D.1-2.

20. Claim 20

POSAs would implement [17B]’s associating step (*supra* §V.E.17.b) by *relatively moving the unique icon representing the service to the unique icon representing a server* as a conventional, convenient, drag-and-drop function for GUIs. Bhattacharjee, [0558]-[0562] (citing corroboration).

21. Claim 21

Conventional *distributed software applications* are “spread over several computers that are connected via a network.” EX1025, [0038]. Calder’s techniques “relate[] to *distributed computing*” ([0002]), with files “stored

remotely **on separate machines**” ([0139]). Thus, POSAs had motivation for some VCE-capsules to have *distributed software applications*. Bhattacharjee, [0563]-[0572] (citing corroboration).

22. Claim 22

“**Computing devices typically include a *console*...to control the computing device** manually.” EX1022, [0004]. Thus, POSAs had motivation for Schmidt-Tormasov/Schmidt-Tormasov-Calder to *update a console on the selected target server with information indicating that the service is resident* thereon so the server’s *console* recognized and properly executed its *services*. Bhattacharjee, [0573]-[0576] (citing corroboration).

23. Claim 23

It would have been obvious to POSAs to *test 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* to avoid wasting time and/or and potentially damaging Schmidt-Tormasov/Schmidt-Tormasov-Calder servers with incompatible VCE-capsules. Bhattacharjee, [0577]-[0581] (citing corroboration).

24. Claim 24

Tormasov permits “different ***users***” ([0024]) to “connect[] to a server system” to execute a VCE related to *the service* (e.g., httpd service), as discussed above (§V.B, §§V.E.1.b.ii-iii). POSAs would thus *create a user account for*

accessing *the service*, a conventional system-administrator function.

Bhattacharjee, [0582]-[0585] (citing corroboration).

25. Claim 25

Tormasov permits “***installation*** of...***applications***” into a *server’s* VCEs (Tormasov, [0024]), which *installs the files specific to the selected application on the selected server*. Bhattacharjee, [0586]-[0587].

26. Claim 26

Schmidt-Tormasov’s/Schmidt-Tormasov-Calder’s VCE-capsules implement “***access*** restrictions enforce[d] inside the” server’s OS kernel. Tormasov, [0018]. That typically included *setting file access permissions to allow an individual user to access the one of the applications to be distributed* and installed in the VCE-capsule. Bhattacharjee, [0588]-[0590] (citing corroboration).

27. Claim 27

a. [27A]

Tormasov’s users “may **stop**” VCEs “in the same manner as with a common operating system” ([0025]), which teaches/motivates *de-installing* the VCE (and the *service* it provides) *from the server*. Bhattacharjee, [0591]-[0592].

b. [27B]

As discussed above (§V.E.17.b, [17B]), POSAs would install a service using drag-and-drop to associate the *service’s* and *target server’s* icons, and thus would implement those icons to be *displayed*, as a user-friendly way to identify the

services installed on each server. Bhattacharjee, [0593]-[0597] (citing corroboration).

c. [27C]

POSAs knew a conventional, user-friendly administrator feature was using icons to de-install *services* from *target servers*. Bhattacharjee, [0598]-[0602] (citing corroboration).

28. Claim 28

As discussed above (§§V.E.27.b-c, [27B]-[27C]), POSAs would want distinct icons for de-installing services from servers and thus would also *separate* those icons as a conventional, user-friendly, GUI-based way to de-install services. Bhattacharjee, [0603]-[0605] (citing corroboration).

29. Claim 29

As discussed above (§V.E.23), POSAs would *test whether the selected server is a valid platform* for installing services and thus would also *test whether it is a valid computing platform for de-installing the service* to prevent improper de-installation. Bhattacharjee, [0606]-[0607].

30. Claim 30

Schmidt-479 supports a “virtual desktop system architecture” that presents capsule environments executing on a remote server. Schdmit-479, [0073]-[0074]. In that implementation, POSAs would implement any *data file changes specific to the service* being executed by the server to be *copied back to a storage medium*

from which the data file changes originated to allow data-file changes to be retrievable for later use. Bhattacharjee, [0608]-[0610].

31. Claim 31

a. Previously Addressed Limitations

Schmidt-Tormasov/Schmidt-Tormasov-Calder meet the following limitations of claim 31 for the reasons discussed in the corresponding section(s) above. Bhattacharjee, [0611]-[0631].

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

Schdmit-479 uses “a chroot *system call* [that] re-assigns the root of file system” seen by applications *running* in a capsule ([0052]) and “hash tables” that are “segmented based on capsule identity,” which includes “system entry points (e.g., *system calls*)” made by those running applications ([0059]). That chroot functionality teaches/suggests a *run time module for monitoring system calls from applications associated with one or more* Schmidt-Tomasov/Schmidt-Tomasov-Calder VCE-capsules (*containers*) *and for providing control of the one or more applications* in the VCE-capsules (e.g., by re-directing application requests for file access). *See also* EX1118 (filed by Schmidt-479 inventors), [0061], [0066], [0072] (describing capsule creation/re-starting via *system calls*). Thus, Schmidt-Tomasov/Schmidt-Tomasov-Calder meets [31I]. Bhattacharjee, [0632]-[0633].

Calder translates *system calls* for use on non-native OSs. Calder, [0101]. Calder taught a run-time software module(s) (e.g., an “interception *module*”) that *monitors system calls from the applications* to translate them, and thereby *provide control of the one or more applications*. Calder, [0086]-[0088]. That is another way Schmidt-Tomasov-Calder meets [31I]. Bhattacharjee, [0634]-[0637].

32. Claim 32

As discussed above (§V.E.9, claim 9), POSAs would *pre-allocate*, *predetermined* system resources to VCE-capsules and thus it was also obvious to include a scheduler comprising values related to an allotted time in which processes within a VCE-capsule (container) may utilize predetermined resources, a conventional way to assign server-resources to applications. Bhattacharjee, [0638]-[0642] (citing corroboration).

33. Claim 33

As discussed above (§V.E.32, claim 32), Schmidt-Tormasov/Schmidt-Tormasov-Calder meets a *run time module* that *intercepts system calls* before reaching the host's kernel and redirects them into the VCE-capsule. Schmidt-479, [0052], [0059]; Calder, [0139]. As discussed above (§V.E.6, claim 6), each VCE-capsule is assigned, *e.g.*, an IP address. Schmidt-479 discloses a *run-time module* that monitors "where interface objections are named," including in "*system calls*." Schmidt-479, [0059]. Exemplary named interface objects are communication interfaces (*e.g.*, for IP-based networks). Schdmidt-479, [0070].

An obvious, conventional way to ensure *unique identifiers* were conveyed to applications was for the *run time module* that *intercepts system calls* from any of the plurality of VCE-capsules (containers) to replace the host's identifier with the VCE-capsule's identifier (or vice-versa) to ensure the VCE-capsules *can run*

independently of one another without contention, as Schmidt-479 and Tormasov intended. Schmidt-479, [0054]; Tormasov, [0030]. For example, in response to system calls requesting the server's IP address, the run-time module would provide the VCE-capsule's assigned, *unique* IP address. Bhattacharjee, [0643]-[0647].

34. Claim 34

a. [34A]

See supra §V.E.9, §V.E.13. Bhattacharjee, [0648]-[0649].

b. [34B]

POSA knew conventional applications run in *user mode* and make *system calls to kernel mode* (e.g., to access hardware resources). Bhattacharjee, [0650]-[0651]. As discussed above (§V.E.1.c.i.(2), §V.E.1.c.iv), applications in VCE-capsules share a *kernel*, and system calls are *intercepted* and *redirected* into the VCE-capsule (e.g., if a requested system file is within the VCE-capsule) or passed to the kernel (e.g., if server hardware is required). Schmidt-479, [0052], [0059]; Tormasov, FIG. 1; Calder, [0074]-[0076]. Bhattacharjee, [0650]-[0656] (citing EX1120, 15).

c. [34C]

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* §V.E.32 and only *forward the system calls based on the comparison* to ensure allocated resource-limits are not violated. Bhattacharjee, [0657]-[0659].

VI. GROUNDS 3-4: CLAIMS 1-34 ARE UNPATENTABLE OVER THE COMBINATIONS OF GROUNDS 1-2 IN FURTHER VIEW OF SCHMIDT-629

1. Schmidt-629 (EX1116) and the Grounds 3-4 Combinations

Schmidt-629 and Schmidt-479 share an inventor, were filed within months of each other, and describe complementary (and overlapping) aspects of the same capsule technique. *Compare* Schmidt-629, [0033]-[0035] *with* Schmidt-479, [0038]-[0040]. Schmidt-629 teaches to implement each “capsule [to] ha[ve] a unique locator, such as IP address, assigned to it,” to beneficially support “mov[ing]” the capsule “to a different machine having a potentially different operating system or” to a “different network.” Schmidt-629, [0015]-[0017], [0056]-[0057]. For example, the IP addresses allow capsules having moved to different locations to communicate with each other, by “us[ing] the unique locator (i.e., IP address) of the target capsule as the final destination” of a communication. Schmidt-629, [0016]. Bhattacharjee, [0660]-[0664].

Thus, based on Schmidt-629, POSAs would have been motivated and reasonably expected success to assign a unique locator identifier, such as an IP address, to each of Schmidt-Tormasov’s VCE-capsules (Ground 3) and each of Schmidt-Tormasov-Calder’s VCE-capsules (Ground 4). Schmidt-629, [0015]-[0017]. Bhattacharjee, [0665]-[0667].

2. Claims 1-4, 7-30, 32-34

Claims 1-4, claims 7-30, independent claim 31's limitations other than [31C], and the limitations added by (dependent) claims 32-34 are rendered obvious by Schmidt-479's and Tormasov's teachings (with or without Calder's teachings) in the Grounds 3-4 combinations for the same reasons as in the Grounds 1-2 combinations. *Supra* §§V.E.1-4, §§V.E.7-31.a, §§V.E.32-34. Bhattacharjee, [0668], [0673], [0677]-[0679].

3. Claims 5-6, 31

Claim 5 recites, *inter alia*, that each container has *a container specific identity assigned to it*; claim 6 and limitation [31C] recite, *inter alia*, that each container is assigned a *unique [associated] identity* which includes an *IP address*. *See infra* §IX.

As discussed above (§VI.1), in view of Schmidt-629, each VCE-capsule in the Grounds 3-4 combinations is “*assign[ed]*” “a *unique* locator,” e.g., “an *IP address*” (Schmidt-629, [0015]), which is an additional reason the Grounds 3-4 combinations render obvious claims 5-6 and [31C]. Bhattacharjee, [0669]-[0672], [0674]-[0676].

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)

Schmidt-479, Tormasov, Calder, and Schmidt-629 were not before the Office during prosecution, so denial is unwarranted.

VIII. CONCLUSION

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

Dated: 01-31-2025

Respectfully submitted,

/Gregory Nieberg/
Gregory Nieberg, Reg. #57,063

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 13,999 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.