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

_____

DECLARATION OF SAMRAT BHATTACHARJEE, PH.D.

Google Exhibit 1003
Google v. VirtaMove

TABLE OF CONTENTS (CONDENSED)

I. PERSONAL AND PROFESSIONAL BACKGROUND 1

II. MATERIALS REVIEWED AND CONSIDERED 6

A. Public Accessibility of Non-Patent Prior Art Exhibits 6

III. MY UNDERSTANDING OF PATENT LAW 6

A. Anticipation 8

B. Obviousness 9

IV. UNPATENTABILITY GROUNDS 12

V. THE '814 PATENT 14

A. Background and Specification 14

B. Person of Ordinary Skill in the Art ("POSA") 17

C. Prosecution History 19

D. Challenged Claims 20

E. The Independent Claims Are Not Supported By Provisional Application No. 60/502,619 20

VI. CLAIM INTERPRETATION 21

A. Terms with Agreed Upon Constructions 22

1. Preamble of Claim 1 22

2. "*container*" (claims 1, 2, 4, 6, 9, 10, 13, 14) 22

3. "*a kernel a set of associated local system files [sic]*" (claim 1) 22

4. "*processor*" (claim 1) 23

5. "*servers*" (claim 1) 23

6. "*within a container*" (claim 1) 23

7. Terms whose antecedent refer to the same entity 23

B. Terms with Disputed Constructions 24

1. "*operating system*" (claims 1, 10) 24

2. "*kernel*" / "*operating system kernel*" (claim 1) 24

3. "*disparate computing environments*" (claim 1) 25

4. "*service*" (claims 1, 14) 26

5. "*at least some of the different operating systems*" (claim 1) 26

"*at least some of the plurality of different operating systems*" (claim 1)26

6. "memory accessible to at least some of the servers" (claim 1).....26

7. "*local kernel residing permanently on one of the servers*" (claim 1) 27

8. "*secure containers of application software*" (claim 1) 27

9. "*an operating system's root file system*" (claim 1) 27

VII. Application of Prior Art to Challenged Claims: Claims 1-34 would have been obvious to POSAs over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations. 29

A. Schmidt-479 (EX1008) 29

B. Tormasov (EX1010) 35

C. Calder (EX1006).....39

D. The Schmidt-Tormasov and Schmidt-Tormasov-Calder Combinations (Grounds 1-2).....46

1. The Schmidt-Tormasov Combination (Ground 1).....46

2. The Schmidt-Tormasov-Calder Combination (Ground 2).....69

E. Grounds 1-2: Mapping of Schmidt-Tormasov (Ground 1) and Schmidt-Tormasov-Calder (Ground 2) to Claims 1-34 74

1. Claim 1 74

a. [1PREA].....75

b. [1PREB] 105

c. [1A] 124

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," 169

e. [1C] "the containers of application software excluding a kernel," 171

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," 176

g. [1E] "wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server,"....185

h. [1F] "and wherein the application software cannot be shared between the plurality of secure containers of application software," 189

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."....192

2. Claim 2: "[C]laim 1, wherein each container has an execution file associated therewith for starting the one or more applications."....200

3. Claim 3: "[C]laim 2, wherein the execution file includes instructions related to an order in which executable applications within will be executed."....205

4. Claim 4: "[C]laim 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 [[1A]]. 208

5. Claim 5: "[C]laim 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." 211

6. Claim 6: "[C]laim 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."....211

7. Claim 7: "[C]laim 2 further comprising the step of modifying at least some of the system files to define container specific mount points associated with the container."....219

8. Claim 8: "[C]laim 1, wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers."....221

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." 225
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." 238
11. Claim 11: "[C]laim 2, wherein containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location." 242
12. Claim 12: "[C]laim 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." 244
13. Claim 13: "[C]laim 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." 245
14. Claim 14 254
    a. [14PRE] "[C]laim 1 comprising the step of creating containers prior to said step of storing containers in memory" 254
    b. [14A] "wherein containers are created by: a) running an instance of a service on a server;" 257
    c. [14B] "b) determining which files are being used; and," 261
    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." 266
15. Claim 15 268
    a. [15A] "[C]laim 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;" 268
b. [15B] "installing the container on a server; and," 268
c. [15C] "testing the applications and files within the container." 268
16. Claim 16 270
a. [16A] "[C]laim 1 comprising the step of creating containers prior to said step of storing containers in memor," 270
b. [16B] "wherein a step of creating containers includes: using a skeleton set of system files as a container starting point and installing applications into that set of files." 270
17. Claim 17 272
a. [17A] "[C]laim 1 further comprising installing a service on a target server selected from one of the plurality of servers," 272
b. [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." 273
18. Claim 18: "[C]laim 17 wherein the target server and the graphical user interface are at remote locations." 280
19. Claim 19: "[C]laim 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." 284
20. Claim 20: "[C]laim 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." 286
21. Claim 21: "[C]laim 20 further comprising starting a distributed software application." 289

22. Claim 22: "[C]laims 20 further comprising updating a console on the selected target server with information indicating that the service is resident on the selected target server."....293
23. Claim 23: "[C]laim 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."....296
24. Claim 24: "[C]laim 17 further comprising creating a user account for the service."....299
25. Claim 25: "[C]laim 17, further comprising the step of installing files specific to the selected application on the selected server." 300
26. Claim 26: "[C]laim 17 further comprising the steps of setting file access permissions to allow a user to access the one of the applications to be distributed."....301
27. Claim 27 302
    a. [27A] "[C]laim 1, further comprising de-installing a service from a server, comprising:"....302
    b. [27B] "displaying the icon representing the service; [and] displaying the icon representing the server on which the service is installed; and" 303
    c. [27C] "utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected service from the server on which it was installed." 305
28. Claim 28: "[C]laim 27 further comprising separating icon representing the service from the icon representing the server."....307
29. Claim 29: "[C]laim 27 further comprising testing whether the selected server is a valid computing platform for de-installation of the service."....309
30. Claim 30: "[C]laim 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." 309
31. Claim 31 310

a. [31PRE] “A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:” 310
b. [31A] “a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers, ” 311
c. [31B] “each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers,” 311
d. [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;” 313
e. [31D] “wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes,” 313
f. [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,” 314
g. [31F] “each container having its own execution file associated therewith for starting one or more applications,” 314
h. [31G] “in operation, each container utilizing a kernel resident on the server and” 315
i. [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,” 316
j. [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.” 317

32. Claim 32: “[C]laim 31, further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.” 318
33. Claim 33: “[C]laim 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 [sic].” 321

34. Claim 34 322

a. [34A] “[C]laim 31, wherein the run time module performs: monitoring resource usage of applications executing;” 322

b. [34B] “intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;” 323

c. [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.” 325

F. Grounds 3-4: Mapping of the Combinations of Grounds 1-2 in Further View of Schmidt-629 to Claims 1-34 326

1. Schmidt-629 (EX1116) and the Grounds 3-4 Combinations 326

2. Description of the Grounds 3-4 Combinations 329

3. Claims 1-4, 7-30, 32-34 330

4. Mapping of the Grounds 3-4 Combinations (the Ground 1-2 Combinations in Further View of Schmidt-629) to Claims 1-34 331

a. 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.” 331

b. 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.” 332

c. Claim 31: 333

d. Claims 32-34 334
VIII. APPENDIX: CLAIM LISTING 1

TABLE OF CONTENTS (FULL)

I. PERSONAL AND PROFESSIONAL BACKGROUND 1
II. MATERIALS REVIEWED AND CONSIDERED 6
A. Public Accessibility of Non-Patent Prior Art Exhibits 6
III. MY UNDERSTANDING OF PATENT LAW 6
A. Anticipation 8
B. Obviousness 9
IV. UNPATENTABILITY GROUNDS 12
V. THE '814 PATENT 14
A. Background and Specification 14
B. Person of Ordinary Skill in the Art ("POSA") 17
C. Prosecution History 19
D. Challenged Claims 20
E. The Independent Claims Are Not Supported By Provisional Application No. 60/502,619 20
VI. CLAIM INTERPRETATION 21
A. Terms with Agreed Upon Constructions 22
1. Preamble of Claim 1 22
2. "*container*" (claims 1, 2, 4, 6, 9, 10, 13, 14) 22
3. "*a kernel a set of associated local system files [sic]*" (claim 1) 22
4. "*processor*" (claim 1) 23
5. "*servers*" (claim 1) 23
6. "*within a container*" (claim 1) 23
7. Terms whose antecedent refer to the same entity 23
B. Terms with Disputed Constructions 24
1. "*operating system*" (claims 1, 10) 24
2. "*kernel*" / "*operating system kernel*" (claim 1) 24
3. "*disparate computing environments*" (claim 1) 25
4. "*service*" (claims 1, 14) 26

5. "*at least some of the different operating systems*" (claim 1) 26

"*at least some of the plurality of different operating systems*" (claim 1)26

6. "memory accessible to at least some of the servers" (claim 1).....26

7. "*local kernel residing permanently on one of the servers*" (claim 1) 27

8. "*secure containers of application software*" (claim 1) 27

9. "*an operating system's root file system*" (claim 1) 27

VII. Application of Prior Art to Challenged Claims: Claims 1-34 would have been obvious to POSAs over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations. 29

A. Schmidt-479 (EX1008) 29

B. Tormasov (EX1010) 35

C. Calder (EX1006).....39

D. The Schmidt-Tormasov and Schmidt-Tormasov-Calder Combinations (Grounds 1-2).....46

1. The Schmidt-Tormasov Combination (Ground 1).....46

2. The Schmidt-Tormasov-Calder Combination (Ground 2).....69

E. Grounds 1-2: Mapping of Schmidt-Tormasov (Ground 1) and Schmidt-Tormasov-Calder (Ground 2) to Claims 1-34 74

1. Claim 1 74

a. [1PREA].....75

i. "1. In a system having a plurality of servers with operating systems that differ".....75

ii. "operating in disparate computing environments".....81

iii. "wherein each server includes a processor and an operating system" 85

iv. "[OS] including a kernel [and] a set of associated local system files compatible with the processor" 92

b. [1PREB].....105

i. "a method of providing at least some of the servers in the system with secure, executable, applications".....105

ii. "[applications] related to a service".....111

iii. "wherein the applications are executed in a secure environment" 114

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".....117

c. [1A] 124

i. "storing in memory accessible to at least some of the servers a plurality of secure containers of application software".....124

(1) Schmidt-Tormasov/Schmidt-Tormasov-Calder VCEs-capsules (*containers*) are "stor[ed] in memory accessible to…the servers" 125

(2) Schmidt-Tormasov/Schmidt-Tormasov-Calder VCEs-capsules are "containers of application software".....127

(3) Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules are "secure containers of application software".....150

ii. "each container comprising one or more of the executable applications" 156

iii. "and a set of associated system files required to execute the one or more applications".....158

iv. "for use with a local kernel residing permanently on one of the servers" 164

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," 169

e. [1C] "the containers of application software excluding a kernel,".....171

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," 176

g. [1E] “wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server,”.....185

h. [1F] “and wherein the application software cannot be shared between the plurality of secure containers of application software,” 189

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.” 192

2. Claim 2: “[C]laim 1, wherein each container has an execution file associated therewith for starting the one or more applications.”.....200

3. Claim 3: “[C]laim 2, wherein the execution file includes instructions related to an order in which executable applications within will be executed.”.....205

4. Claim 4: “[C]laim 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 [[1A]]. 208

5. Claim 5: “[C]laim 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.” 211

6. Claim 6: “[C]laim 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.”.....211

7. Claim 7: “[C]laim 2 further comprising the step of modifying at least some of the system files to define container specific mount points associated with the container.”.....219

8. Claim 8: “[C]laim 1, wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers.”.....221

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." 225

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." 238
11. Claim 11: "[C]laim 2, wherein containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location." 242
12. Claim 12: "[C]laim 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." 244
13. Claim 13: "[C]laim 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." 245
14. Claim 14 254
    a. [14PRE] "[C]laim 1 comprising the step of creating containers prior to said step of storing containers in memory" 254
    b. [14A] "wherein containers are created by: a) running an instance of a service on a server;" 257
    c. [14B] "b) determining which files are being used; and," 261
    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." 266
15. Claim 15 268
    a. [15A] "[C]laim 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;" 268

b. [15B] “installing the container on a server; and,” 268
c. [15C] “testing the applications and files within the container.” 268
16. Claim 16 270
a. [16A] “[C]laim 1 comprising the step of creating containers prior to said step of storing containers in memor,”.....270
b. [16B] “wherein a step of creating containers includes: using a skeleton set of system files as a container starting point and installing applications into that set of files.” 270
17. Claim 17 272
a. [17A] “[C]laim 1 further comprising installing a service on a target server selected from one of the plurality of servers,”.....272
b. [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.”.....273
18. Claim 18: “[C]laim 17 wherein the target server and the graphical user interface are at remote locations.” 280
19. Claim 19: “[C]laim 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.”.....284
20. Claim 20: “[C]laim 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.” 286
21. Claim 21: “[C]laim 20 further comprising starting a distributed software application.”.....289
22. Claim 22: “[C]laims 20 further comprising updating a console on the selected target server with information indicating that the service is resident on the selected target server.”.....293

23. Claim 23: "[C]laim 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." 296
24. Claim 24: "[C]laim 17 further comprising creating a user account for the service." 299
25. Claim 25: "[C]laim 17, further comprising the step of installing files specific to the selected application on the selected server." 300
26. Claim 26: "[C]laim 17 further comprising the steps of setting file access permissions to allow a user to access the one of the applications to be distributed." 301
27. Claim 27 302
    a. [27A] "[C]laim 1, further comprising de-installing a service from a server, comprising:" 302
    b. [27B] "displaying the icon representing the service; [and] displaying the icon representing the server on which the service is installed; and" 303
    c. [27C] "utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected service from the server on which it was installed." 305
28. Claim 28: "[C]laim 27 further comprising separating icon representing the service from the icon representing the server." 307
29. Claim 29: "[C]laim 27 further comprising testing whether the selected server is a valid computing platform for de-installation of the service." 309
30. Claim 30: "[C]laim 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." 309
31. Claim 31 310
    a. [31PRE] "A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:" 310

b. [31A] “a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers, ” 311
c. [31B] “each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers,”.....311
d. [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;”.....313
e. [31D] “wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes,”.....313
f. [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,”.....314
g. [31F] “each container having its own execution file associated therewith for starting one or more applications,”.....314
h. [31G] “in operation, each container utilizing a kernel resident on the server and”.....315
i. [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,” 316
j. [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.” 317

32. Claim 32: “[C]laim 31, further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.”.....318
33. Claim 33: “[C]laim 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 [sic]." 321

34. Claim 34 322

a. [34A] "[C]laim 31, wherein the run time module performs: monitoring resource usage of applications executing;".....322

b. [34B] "intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;" 323

c. [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." 325

F. Grounds 3-4: Mapping of the Combinations of Grounds 1-2 in Further View of Schmidt-629 to Claims 1-34.....326

1. Schmidt-629 (EX1116) and the Grounds 3-4 Combinations.....326

2. Description of the Grounds 3-4 Combinations 329

3. Claims 1-4, 7-30, 32-34.....330

4. Mapping of the Grounds 3-4 Combinations (the Ground 1-2 Combinations in Further View of Schmidt-629) to Claims 1-34 331

a. 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."....331

b. 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." 332

c. Claim 31:.....333

i. Limitations [31PRE]-[31B], [31D]-[31H].....333

ii. Limitation [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”.....333
d. Claims 32-34.....334
VIII. APPENDIX: CLAIM LISTING.....1

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
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
1037	U.S. Patent No. 7,103,745
1038	U.S. Patent No. 6,263,440

Exhibit	Description
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
1075	PCT Application Publication No. WO2002056156

Exhibit	Description
1076	Kamp et al., "Jails: Confining the omnipotent root," Proceedings of the 2nd International SANE Conference (Vol. 43, p. 116) (2000)
1077	Prevelakis et al., "Sandboxing Applications," USENIX Annual Technical Conference, FREENIX Track (pp. 119-126) (2001)
1078	Plaintiff VirtaMove Corp.'s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sept. 06, 2024)
1079	Chart re: '814 Patent accompanying Plaintiff VirtaMove Corp.'s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sept. 6, 2024)
1080	First Amended Complaint for Patent Infringement Against Google LLC in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (May 21, 2024)
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
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
1100	Google LLC's Proposed Claim Terms for Construction in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1101	Plaintiff's Disclosure of Proposed Claim Constructions in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1102	Google LLC's Opening Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 24, 2024)

Exhibit	Description
1103	Plaintiff's Responsive Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 12, 2024)
1104	Google LLC's Reply Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 26, 2024)
1105	Plaintiff's Sur-Reply Claim Construction Brief in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 13, 2024)
1106	Joint Claim Construction Statement in *VirtaMove, Corp. v. Google LLC*, 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 18, 2024)
1107	U.S. Provisional Application No. 60/502,619
1108	U.S. Patent Application Publication No. 20050188268
1109	U.S. Patent Application Publication No. 20020140743
1110	U.S. Patent Application Publication No. 20020143906 ("Tormasov-906")
1111	U.S. Patent No. 7,076,633 ("Tormasov-633")
1112	U.S. Patent Application Publication No. 20060089950 ("Tormasov-950")
1113	U.S. Patent No. 7,222,132 ("Tormasov-132")
1114	U.S. Patent Application Publication No. 20020147815 ("Tormasov-815")
1115	U.S. Patent No. 7,209,973 ("Tormasov-973")
1116	U.S. Patent Application Publication No. 20020138629 ("Schmidt-629")
1117	U.S. Patent No. 6,944,860 ("Schmidt-860")
1118	U.S. Patent Application Publication No. 20020174265 ("Schmidt-265")
1119	U.S. Patent Application Publication No. 20020116659 ("Tormasov-659")
1120	Bach, Maurice. 1987. Design of the Unix Operating System by Marice J Bach, 1 edition (Feb. 27, 1987) Prentice Hall; ISBN: 0132017997.
1121	Crowley, Charles. 1997. Operating Systems: a design-oriented approach. Irwin. 1997. ISBN 0-256-15151-2.
1122	Eckel, George and Chris Hare. 1995. Building a Linux Internet Server, Chris Hare", G. Eckel, New Riders Publishing; ISBN: 1562055259.
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

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

I, Samrat Bhattacharjee, Ph.D., declare:

I have been retained by Wolf, Greenfield & Sacks, P.C., counsel for Petitioner Google LLC, to assess claims 1-34 (the "challenged claims") of U.S. Patent No. 7,519,814 ("the '814 patent") (EX1001). I am being compensated for my time at my standard rate of $700.00 per hour, plus actual expenses. My compensation is not dependent in any way upon the outcome of the inter partes review of the '814 patent.

I. PERSONAL AND PROFESSIONAL BACKGROUND

[0001] My qualifications for forming the opinions set forth in this declaration are summarized below and explained in more detail in my current curriculum vitae, provided as EX1004. EX1004 also includes a list of my publications.

[0002] I am a tenured professor in the Computer Science Department at the University of Maryland. I am also an Affiliate Professor with the Department of Electrical and Computer Engineering at the University of Maryland. I have been an appointed faculty member at the University of Maryland since 1999.

[0003] I received a Ph.D. in computer science from the Georgia Institute of Technology in 1999. Before that I received a B.S. degree in Mathematics and Computer Science with Highest Distinction (Summa Cum Laude) and was recognized as the Outstanding Department Major at Georgia College and State University, Milledgeville, Georgia in 1994.

[0004] I have been involved in the research and development of computing systems for over two decades. I perform research and I teach in the areas of computer software and the management and deployment of server applications, such as computer networks, distributed systems, computer security, cryptography, and operating systems, among others. A major theme of my research has been the development of technology to improve the performance and security of data transfers and data storage on networks that deploy and manage servers and server applications, like the Internet. My research has examined problems ranging from network and operating system support audio and video streaming, large scale Internet video distribution, the design of novel network architectures for better and secure content distribution and storage, efficiency of content distribution over wireless networks, and so on.

[0005] For example, starting in the late 1990s, the focus of my research was the development of network and operating system technology to design a new network architecture that allowed for more efficient data transfer. Part of my Ph.D. research was to develop new and more efficient architectures for video delivery on the Internet, and I have published papers on this architecture during my graduate studies. I continued to work on the management and deployment of server applications for video delivery and security and storage of digital content for server applications as a faculty member, and have published various papers in

these areas. During 2007, I was a visiting researcher at AT&T Labs, and one of the projects I focused on was a video content delivery platform using server applications. This work resulted in both publications and a granted US patent (US 8,752,100 B2). Among other research, I have authored papers and developed research software for public-key infrastructure KeyChains and a cryptographic library for "chit"-based security, which all involve the management and deployment of server applications.

[0006] Along with working on research problems related computer software and the management and deployment of server applications, I routinely taught this material in both graduate and undergraduate courses. A number of my papers in these domains are with students at the University of Maryland, many of whom started in these areas after taking my classes.

[0007] Before the earliest alleged priority date of the '814 patent I had had extensive experience working with various Operating Systems, including at the kernel level. My experience included kernel modifications to both Linux and various flavors of SunOS (including Solaris). For example, I developed an OS for networking applications, and also performed research on various performance improvements for networking applications within the kernel. I also taught the undergraduate OS course at Maryland, and started a reading group to introduce students, including undergraduates to OS research and code for the management

and deployment of server applications. Inspired partly by this experience, a graduate student developed a teaching OS from scratch, which I and others than modified to teach undergraduates about OSs, including the management and deployment of server applications.

[0008] As a result of that experience, I also co-authored a paper on how to teach low-level systems/OS to undergraduates. Evolutions of this teaching OS is still used to teach undergraduate OS at Maryland.

[0009] For over 25 years, I have developed professional and academic experience in the field of computer software and the management and deployment of server applications. For example, I have authored or co-authored over 100 articles in peer-reviewed journals, conference proceedings, and workshops relating computer software and the management and deployment of server applications. Over the years I have received several fellowships, prizes, and awards for teaching excellence and technical papers, including an Alfred P. Sloan Jr. Fellowship in 2004 and the National Science Foundation CAREER award in 2001. I have also received an award from the SIGCOMM foundation for a "Test of Time" paper (that is given to an influential paper published ten years ago). My research work has been supported by multiple grants from the US National Science Foundation, and the Department of Defense. I have also started a Joint Ph.D. program with the University of Maryland and the Max Planck Society in Germany, and co-founded

the annual Cornell, Maryland, Max Planck Research School that provides research exposure to about 80 students from across the world during a week-long school.

[0010] I have served on numerous paper and proposal review panels and participated on program committees for ACM/IEEE groups, technical journals, and conferences in the field of computer software and the management and deployment of server applications. I have created research computer software for all of my projects, much of which is available online for others to build on and involves the management and deployment of server applications.

[0011] I am a named inventor on four U.S. Patents. These patents are generally related to computer software and the management and deployment of server applications.

[0012] I have served as an expert witness and technical consultant in litigation and Inter Partes Review matters concerning computer software and the management and deployment of server applications, including videoconferencing, data conferencing and screen sharing, voice over IP (VoIP) telephony, multimedia networking, distributed systems, operating systems, computer networks, and datacenter networking, among others. I have testified in several trials and depositions as an expert witness.

II. MATERIALS REVIEWED AND CONSIDERED

[0013] My findings, as explained below, are based on my years of education, research, experience, and background in the field of computer software and the management and deployment of server applications, as well as my investigation and study of relevant materials for this declaration. When developing the opinions set forth in this declaration, I assumed the perspective of a person having ordinary skill in the art, as set forth in Section V.B below. In forming my opinions, I have studied and considered the materials identified in the table in the Appendix Listing of Exhibits above.

A. Public Accessibility of Non-Patent Prior Art Exhibits

[0014] My opinions regarding public accessibility of the following exhibits are discussed in my Appendix Non-Patent Literature attached hereto in the sections of that Appendix summarized in the table below:

Exhibit	Appendix Section
EX1076	§II.A
EX1077	§II.B
EX1088	§II.C
EX1120	§II.D
EX1121	§II.E
EX1122	§II.F

III. MY UNDERSTANDING OF PATENT LAW

[0015] In developing my opinions, I discussed various relevant legal principles with Petitioner's attorneys. I understood these principles when they

were explained to me and have relied upon such legal principles, as explained to me, in the course of forming the opinions set forth in this declaration. My understanding in this respect is as follows:

[0016] I understand that "inter partes review" (IPR) is a proceeding before the United States Patent & Trademark Office for evaluating the patentability of an issued patent's claims based on prior-art patents and printed publications.

[0017] I understand that, in this proceeding, Petitioner has the burden of proving that the challenged claims of the '814 patent are unpatentable by a preponderance of the evidence. I understand that "preponderance of the evidence" means that a fact or conclusion is more likely true than not true.

[0018] I understand that, in IPR proceedings, claim terms in a patent are given their ordinary and customary meaning as understood by a person of ordinary skill in the art ("POSA") in the context of the entire patent and the prosecution history pertaining to the patent. If the specification or prosecution history provides a special definition for a claim term that differs from the meaning the term would otherwise possess, that special definition controls. If a claim element is expressed as a "means" for performing a specified function, I understand that it covers the corresponding structure described in the specification and equivalents of the described structure. I have applied these standards in preparing the opinions in this declaration.

[0019] I understand that determining whether a particular patent or printed publication constitutes prior art to a challenged patent claim can require determining the effective filing date (also known as the priority date) to which the challenged claim is entitled. I understand that for a patent claim to be entitled to the benefit of the filing date of an earlier application to which the patent claims priority, the earlier application must have described the claimed invention in sufficient detail to convey with reasonable clarity to the POSA that the inventor had possession of the claimed invention as of the earlier application's filing date. I understand that a disclosure that merely renders the claimed invention obvious is not sufficient written description for the claim to be entitled to the benefit of the filing date of the application containing that disclosure.

[0020] I understand that for an invention claimed in a patent to be patentable, it must be, among other things, new (novel—i.e., not anticipated) and not obvious from the prior art. My understanding of these two legal standards is set forth below.

A. Anticipation

[0021] I understand that, for a patent claim to be "anticipated" by the prior art (and therefore not novel), each and every limitation of the claim must be found, expressly or inherently, in a single prior-art reference. I understand that a claim limitation is disclosed for the purpose of anticipation if a POSA would have

understood the reference to disclose the limitation based on inferences that a POSA would reasonably be expected to draw from the explicit teachings in the reference when read in light of the POSA's knowledge and experience.

[0022] I understand that a claim limitation is inherent in a prior art reference if that limitation is necessarily present when practicing the teachings of the reference, regardless of whether a person of ordinary skill recognized the presence of that limitation in the prior art.

B. Obviousness

[0023] I understand that a patent claim may be unpatentable if it would have been obvious in view of a single prior-art reference or a combination of prior-art references.

[0024] I understand that a patent claim is obvious if the differences between the subject matter of the claim and the prior art are such that the subject matter as a whole would have been obvious to a person of ordinary skill in the relevant field at the time the invention was made. Specifically, I understand that the obviousness question involves a consideration of:

- the scope and content of the prior art;
- the differences between the prior art and the claims at issue;
- the knowledge of a person of ordinary skill in the pertinent art; and

- if present, objective factors indicative of non-obviousness, sometimes referred to as "secondary considerations." To my knowledge, the Patent Owner has not asserted any such secondary considerations with respect to the '814 patent.

[0025] I understand that in order for a claimed invention to be considered obvious, a POSA must have had a reason for combining teachings from multiple prior-art references (or for altering a single prior-art reference, in the case of obviousness in view of a single reference) in the fashion proposed.

[0026] I further understand that in determining whether a prior-art reference would have been combined with other prior art or with other information within the knowledge of a POSA, the following are examples of approaches and rationales that may be considered:

- combining prior-art elements according to known methods to yield predictable results;
- simple substitution of one known element for another to obtain predictable results;
- use of a known technique to improve similar devices in the same way;
- applying a known technique to a known device ready for improvement to yield predictable results;

- applying a technique or approach that would have been "obvious to try," *i.e.*, choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success.
- known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art;
- some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior-art reference or to combine prior-art reference teachings to arrive at the claimed invention. I understand that this teaching, suggestion or motivation may come from a prior-art reference or from the knowledge or common sense of one of ordinary skill in the art.

[0027] I understand that a universal motivation known in a particular field to improve technology can provide a motivation to combine prior art references even absent any hint or suggestion in the references themselves. I also understand that obviousness is determined in light of all the facts and that a given course of action often has simultaneous advantages and disadvantages, and that this does not necessarily obvious motivation to combine teachings from multiple references.

[0028] I understand that for a single reference or a combination of references to render the claimed invention obvious, a POSA must have been able to arrive at the claimed invention by modifying, implementing, or combining the teachings of the applied references.

IV. UNPATENTABILITY GROUNDS

[0029] The table below summarizes the Grounds 1-4 that I discuss in this declaration:

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

[0030] As I discuss further below (§VII.E), in my opinion claims 1-34 of the ’814 patent would have been obvious to a person of ordinary skill in the art over the combination of Schmidt-479 (EX1008) and Tormasov (EX1010) (**Ground 1**) and the combination of Schmidt-479, Tormasov and Calder (EX1006) (**Ground 2**).

[0031] As I also discuss below (§VII.F.4), it is also my opinion that claims 1-34 would have also been obvious to a person of ordinary skill in the art over the Ground 1 and Ground 2 combinations in further view of Schmidt-629

(EX1116). **Ground 3** is the Ground 1 combination in further view of Schmidt-629; **Ground 4** is the Ground 2 combination in further view of Schmidt-629.

[0032] The '814 patent's earliest-alleged priority date on the patent itself is September 15, 2003, based on the filing of U.S. Provisional Patent Application No. 60/502,619 (the "'619 provisional"; EX1107). *See* EX1001, code (60). The '814 patent also alleges priority to a second U.S. Provisional Patent Application No. 60/512,103 (the "'103 provisional"), filed October 20, 2003. For purposes of this declaration, I have been asked to assume that the priority date for the '814 patent is in the September-October 2003 timeframe.

[0033] I understand that Schmidt-479 (published 2002-07-18), Tormasov (published 2002-09-05), and Calder (published 2002-05-30) qualify as prior art to the '814 patent even if the '814 patent were entitled to the September 15, 2003 filing date of the '619 provisional. However, as I discuss below (§V.E), it is my opinion that the '619 provisional application does **not** provide written description support for (at least) either independent claim of the '814 patent.

[0034] In any event, I understand that Schmidt-629 is prior art to the '814 patent whether or not the '814 patent is not entitled to the September 15, 2003 filing date of the '619 provisional based on (at least) Schmidt-629's filing date (March 22, 2001).

[0035] None of Schmidt-479, Tormasov, Schmidt-629, or Calder were of record during prosecution of the ’814 patent. *See* EX1002; EX1001, code (56).

V. THE ’814 PATENT

A. Background and Specification

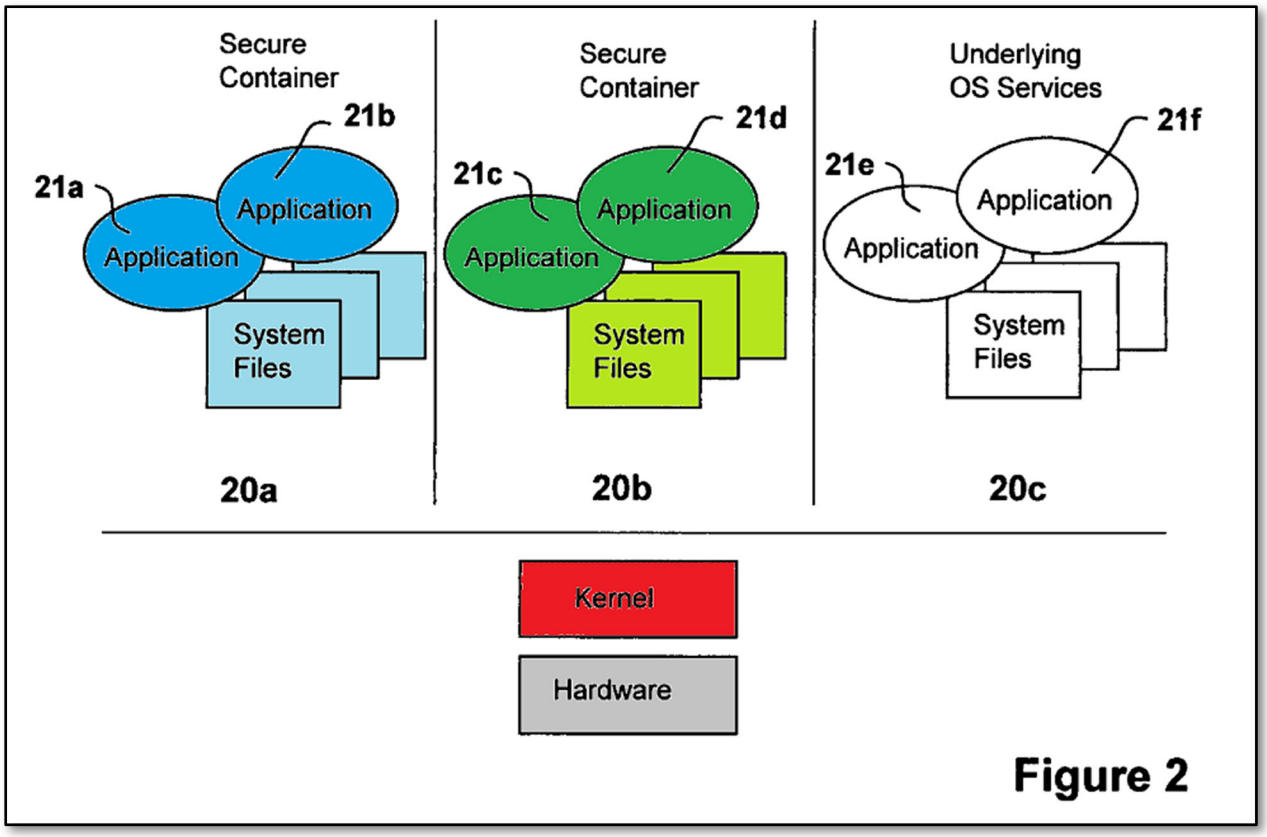
[0036] Computer Operating Systems (“OSs”) typically support various system files (like configuration files and libraries) used by the applications installed to a computer. Configuration files provide information about, for example, the computer’s IP address or user-defined settings (e.g., the available and/or selected font(s), font size(s)). Libraries provide pre-existing code for common functions (e.g., opening files) that any application can use. *See, e.g.*, EX1086, 2:62-3:11 (“System calls look like ordinary function calls in C programs, and libraries map these function calls to the primitives needed to enter the operating system in a well-known manner”; referring to conventional Unix-based “system calls, such as open (to open a file for reading or writing), close, read, write”).

[0037] Typically, multiple applications are installed on each computer. However, POSAs knew that problems can sometimes arise when multiple applications are installed on the same computer. For example, the ’814 patent says an application may replace a shared library with a new version of the library but another application may have been designed in a way such that the application

requires (or works better with) the previous, replaced version of the shared library. Another known problem with having multiple applications installed and executing on one computer is that one application may over-consume shared computing "resources," like the processor or "network port numbers," and prevent other applications from executing properly. EX1001, 1:31-36.

[0038] The patent says one known solution to these problems is to "separate[]" applications by installing sets of applications on "individual computer system[s]" (EX1001, 1:27-30), but that can quickly become costly by having to purchase, install, and manage individual computers for each set of applications. An alternative prior-art solution is "Virtual Machine [VM] technology," where one computer can run multiple VMs, each of which simulates a different machine with its own hardware, OS, etc. EX1001, 1:51-61. The patent alleges, however, that VMs "imposes significant performance overhead," like storage for multiples copies of entire OSs on one machine. EX1001, 1:62-65. The patent says another known technique separated "application[s] from the underlying" OS but allegedly did "not isolate applications into distinct environments." EX1001, 2:4-12.

[0039] As illustrated in the '814 patent's Figure 2 (which I have annotated below), the '814 patent's alleged invention is directed to a software "container" having copies of system files for applications, so those applications can be used in different operating environments.



As shown above, "a single server" can execute sets of "[a]pplications" (blue and green) that "are segregated" into respective "secure containers" 20a-20b. EX1001, 7:4-15. Each container has respective system files (light blue and light green) that cannot be overwritten by another container. EX1001, 7:10-12, 9:53-57. Unlike VMs, containers do not comprise an entire OS within each container or simulate having different machines for each container. Rather, containers share a computer's hardware (grey) ***and*** OS kernel (red). EX1001, FIG. 2, 1:56-61. (The kernel is core of an OS that controls access to the underlying computer's resources (e.g., hardware).) Each container is allegedly unlike prior-art Application

Virtualization techniques because each container can have a unique identity (e.g., unique IP address). EX1001, 2:10-12, 3:55-57.

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

[0040] I have been informed and understand that for purposes of assessing whether prior-art references disclose every element of a patent claim (thus "anticipating" the claim) and/or would have rendered the claim obvious, the patent and the prior-art references must be assessed from the perspective of a person having ordinary skill in the art ("POSA") to which the patent is related, based on the understanding of that person at the time of the patent claim's priority date. I have been informed and understand that a POSA is presumed to be aware of all pertinent prior art and the conventional wisdom in the art, and is a person of ordinary creativity. I have applied this standard throughout my declaration.

[0041] A POSA at the time of the '814 patent's earliest claimed priority date (i.e., in the September-October 2003 timeframe) 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.

[0042] The declaration in this matter demonstrates that the challenged claims of the '814 patent would have been obvious based on the references named

in the combinations below (§§VII.D.1-2, §VII.F.1) that I understand are prior art even if VirtaMove were to demonstrate a priority date that is before the earliest-alleged priority date on the face of the '814 patent (i.e., September 15, 2003).[1] I am also submitting a declaration in IPR2025-00487 that demonstrates that the challenged claims of the '814 patent would have been obvious based on the '814 patent if the '814 patent were entitled to claim priority to a date that is earlier than the alleged priority date on the face of the '814 patent (i.e., September 15, 2003, *see* EX1001, code (60)), but where that date is not earlier than the prior art date for the references in the grounds of IPR2025-00487.

[1] As I discussed above (§IV), I understand that Schmidt-629 is prior art to the '814 patent regardless of an alleged priority date because the '814 patent cannot claim priority to the filing date of its earliest alleged provisional application (i.e., the provisional application filed September 15, 2003). *See* §V.E (demonstrating that the independent claims are not supported by that September 15, 2003 provisional). I further understand that whether or not the '814 patent can claim priority to the September 15, 2003 provisional, Schmidt-629 would still be prior art to the '814 patent as to any priority date for the '814 patent that occurs after Schmidt-629's March 22, 2001 filing date.

C. Prosecution History

[0043] The original application's claims were rejected as anticipated by U.S. Patent 6,381,742 ("Forbes"). EX1002, 238-271. The applicants responded by adding Limitations [1E]-[1G] and the word "remain" to Limitation [1D]. EX1002, 287-288, 291; *see infra* §VIII (Claim Listing). The applicants also argued, among other arguments, that an example of different OSs occurs when one server has "one container…running an application…compatible with Solaris 8" (a Unix-based OS) and "another container" "running an application…compatible with Solaris 10." EX1002, 300-301. Solaris 8 and 10 are different versions of the Solaris OS (i.e., they're in the same OS-family, similar to Microsoft Windows 95, 98, XP, etc.). Thus, the inventors' discussion during prosecution establishes that in the context of the '814 patent, different OS, includes OSs that differ by being different versions of the same OS-family, as opposed to different OS only being form different OS-families (e.g., a Windows-based vs. a Unix-based OSs). The Examiner subsequently allowed the claims because the prior art allegedly failed to disclose containers of application software that had unique file systems and that allowed different versions of an OS to execute on the "same system/server environment." EX1002, 341-344.

D. Challenged Claims

[0044] Claims 1-34 are reproduced in the Claim Listing (*infra* §VIII), with labeled limitations (e.g., [**1PREA**]-[**1G**]).

[0045] The ’814 patent’s two independent claims (1 and 31) each recite a system having secure containers of application programs comprising associated system files for executing a container’s applications. The containers do not include an OS kernel, but instead all containers on one server use that server’s underlying OS kernel.

[0046] Each dependent claims recites further functionality that was routine and conventional before the ’814 patent’s earliest claimed priority date—like using execution files to start applications and pre-allocating computer resources to applications.

E. The Independent Claims Are Not Supported By Provisional Application No. 60/502,619

[0047] I have been asked to consider whether Provisional Application No. 60/502,619 (EX1107; the “’619 application”) provides written description support for the independent claims of the ’814 patent. (I have not been asked to consider whether ’619 application provides written description support for any other claim or to consider whether Provisional Application No. 60/512,103 provides written description support for any claim of the ’814 patent.)

[0048] In my opinion, the '619 application does **not** provide written description support for the independent claims of the '814 patent. The '619 application is directed to "Drag & Drop Application Management." EX1107, 1. The '619 application discusses an alleged invention that "relates to software that manages the operation of a data center." EX1107, 1. As the '619 application itself says, the alleged invention discussed in that application is about "installing a software application on a computing platform" using a "software application icon." EX1107, 1. That is, the '619 application is about methods and systems for installing and de-installing applications. However, the '619 application is not about containers of application software. Notably, the '619 application does not use the term "*container*" at all and does not otherwise describe or illustrate anything that meets the parties' agreed upon construction of a *container* (*infra* §VI.A.2). For at least this reason, it is my opinion that the '619 application does not provide written description support for any independent claim of the '814 patent.

VI. CLAIM INTERPRETATION

[0049] I understand that determining whether a claimed invention is novel and/or non-obvious requires comparing the prior art to the claim. I understand that in some scenarios, an express construction defining the outer bounds of one or more claim terms may be appropriate during an IPR

proceeding—for example, where the prior art would meet a challenged claim under one proposed construction but not another construction.

[0050] I have been informed that the parties have agreed to constructions of certain terms and dispute the construction of other terms. Below is my understand of the parties' current positions on claim constructions:

A. Terms with Agreed Upon Constructions

1. Preamble of Claim 1

[0051] The parties agree that the preamble of claim 1 is limiting. EX1106, 6.

2. "*container*" (claims 1, 2, 4, 6, 9, 10, 13, 14)

[0052] The parties agree that the construction of "*container*" should be:

> An aggregate of files required to successfully execute a set of software applications on a computing platform. 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.

3. "*a kernel a set of associated local system files [sic]*" (claim 1)

[0053] The parties agree that an "*and*" should be inserted between *"a kernel*" and "*a set of associated local system files.*" EX1106, 6.

4. "*processor*" (claim 1)

[0054] The parties agree that the construction of "*processor*" should be: "physical computer processor." EX1106, 6.

5. "*servers*" (claim 1)

[0055] The parties agree that the construction of "*servers*" should be: "physical servers." EX1106, 6.

[0056] I note that VirtaMove further argues that the scope of "*servers*" should include "a physical server running virtual machine software." *See* EX1105, 4.

6. "*within a container*" (claim 1)

[0057] The parties agree that the construction of "*within a container*" should be: "associated with a container." EX1106, 6.

7. Terms whose antecedent refer to the same entity

[0058] The parties agree that the following terms and their respective antecedent should refer to the same entity. *See* EX 1106, 6-7.

- "*a processor*" / "*the processor*" (claim 1)
- "*a plurality of servers*" / "*the servers*" (claim 1)
- "*one of the servers*" / "*the server*" (claim 1)
- "*operating systems*" / "*the operating systems*" (claims 1, 10)

- "*a plurality of different operating systems*" / "*the plurality of different operating systems*" (claim 1)

- "*one or more of the executable applications*" / "*the one or more of the executable applications*" (claim 1)

- "*a plurality of secure containers of application software*" / "*the containers of application software"* / "*the plurality of secure containers of application software*" / "*the containers*" (claims 1, 6, 9)

B. Terms with Disputed Constructions

1. "*operating system*" (claims 1, 10)

Google's Construction	VirtaMove's Construction
"The software that controls the allocation and usage of hardware resources such as memory, central processing unit (CPU) time, disk space, and peripheral devices." (EX1102, 4)	"No construction necessary; plain and ordinary meaning." (EX1103, 7) VirtaMove further argues that the parties do not "dispute [] the scope of" this term. EX1105, 4.

2. "*kernel*" / "*operating system kernel*" (claim 1)

Google's Construction	VirtaMove's Construction
"The 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." (EX1102, 4)	"No construction necessary; plain and ordinary meaning." (EX1103, 7) VirtaMove further argues that the parties do not "dispute [] the scope of" this term. EX1105, 4.

3. "*disparate computing environments*" (claim 1)

Google's Construction	VirtaMove's Construction
Indefinite	"Environments run by standalone or unrelated computers" (EX1103, 8)

[0059] The '814 patent provides a definition of the term "[d]isparate computing environments" at EX1001, 2:17-19: "[e]nvironments where computers are stand-alone or where there are plural computers and where they are unrelated." My understanding of Google's basis for arguing the claimed "*disparate computing environments*" is indefinite is that "the specification provides no objective boundaries of 'relatedness'" for the computers running the environments. EX1102, 6-7. However, my understanding is that Google has not argued that it would be unclear to POSAs what it means to have "stand-alone" computers.

[0060] As I discuss further below (§VII.E.1.a.ii), it would have been obvious to implement the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations (*infra* §§VII.D.1-2) to provide environments on multiple, standalone computers and thus meet Limitation [**1PREA**]. In other words, even though the full scope of *disparate computing environments* is unclear because the claim language read in light of the specification does not provide a sufficiently clear boundary, the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations would have been well-within that boundary.

[0061] Thus, I understand that the Board can find that the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations (alone and further in view of Schmidt-629) render claim 1 unpatentable without having to resolve whether the term "*disparate computing environments*" renders the claim indefinite.

4. "*service*" (claims 1, 14)

Google's Construction	**VirtaMove's Construction**
"specialized, software-based functionality provided by network servers and comprised of one or more applications" (EX1106, 4; EX1102, 7)	"No construction necessary; plain and ordinary meaning." (EX1106, 4; EX1103, 9)

5. "*at least some of the different operating systems*" (claim 1)

"*at least some of the plurality of different operating systems*" (claim 1)

Google's Construction	**VirtaMove's Construction**
"at least two or more of the different operating systems" / "at least two or more of the plurality of different operating systems" (EX1106, 4; EX1102, 11)	"No construction necessary; plain and ordinary meaning." (EX1106, 4; EX1103, 12)

6. "memory accessible to at least some of the servers" (claim 1)

Google's Construction	**VirtaMove's Construction**
"memory that at least two or more of the servers can read from or write to" (EX1106, 4; EX1102, 11)	"memory that at least some of the servers can read from or write to" (EX1106, 4; EX1103, 12)

7. "*local kernel residing permanently on one of the servers*" (claim 1)

Google's Construction	VirtaMove's Construction
"local kernel in one of the server's memory that is not lost when power is removed from it" (EX1106, 4)	"No construction necessary; plain and ordinary meaning." (EX1106, 4)

8. "*secure containers of application software*" (claim 1)

Google's Construction	VirtaMove's Construction
"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; EX1102, 13)	"[c]ontainers 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; EX1103, 14)

9. "*an operating system's root file system*" (claim 1)

Google's Construction	VirtaMove's Construction
Indefinite (EX1106, 5; EX1102, 14)	"No construction necessary; plain and ordinary meaning." (EX1106, 5; EX1103, 15) Further arguing that "[t]he point of the claim is that each claimed container will have its own root file system rather than sharing a root file system with *any* operating system's root file system. If a container shared its root file system with even one operating system, then it would not be unique." EX1105, 8 (emphasis original)

[0062] This term is part of limitation [**1G**], which states "*wherein each of the containers has a unique root file system that is different from* ***an operating system's root file system***." In the related district court litigation, Google argues this term is indefinite because "the claim language does not indicate *which* 'operating system' th[e] comparison should be made" to determine whether "each of the containers [] unique root file system" is "*different from an operating system's root file system*." EX1102, 14.

[0063] As I discuss further below (§VII.E.1.a.i), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) each have a unique root file system that is different from each other and different from the root file system of every server in the world. Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder would have met limitation [**1G**] as to **any** OS's root file system in the world. In other words, even though the full scope of limitation [**1G**] is unclear because the claim language read in light of the specification does not provide a sufficiently clear boundary, the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations would have been well-within that boundary.

[0064] Thus, I understand that the Board can find that the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations (alone and further in view

of Schmidt-629) render claim 1 unpatentable without having to resolve whether the term *an operating system's root file system* renders the claim indefinite.

* * *

[0065] As I discuss further below (§§VII.E-F), the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations (alone and further in view of Schmidt-629) render obvious all challenged claims under both parties' claim constructions. For all other claim terms, I have applied the term's plain and ordinary meaning as it would have been understood by POSAs in the September-October 2003 timeframe (which would have been the same in the September-October 2002 timeframe).

VII. APPLICATION OF PRIOR ART TO CHALLENGED CLAIMS: CLAIMS 1-34 WOULD HAVE BEEN OBVIOUS TO POSAS OVER THE SCHMIDT-TORMASOV AND SCHMIDT-TORMASOV-CALDER COMBINATIONS.

A. Schmidt-479 (EX1008)

[0066] Schmidt-479 is a U.S. Patent Application Publication that was filed on January 18, 2001, and published on July 18, 2002. Schmidt-479 "relates to virtual namespaces for active computing environments." Schmidt-479, [0002]. As I discuss further below, Schmidt-479 discloses a software-based capsule technique to allow for various applications and associated services to be conveniently installed on one or more servers in unique computing environments on one server, sharing the server's one OS kernel. Schmidt-479's capsule

technique, which I discuss further below, "may be embodied in any form of computer program product. A computer program product comprises a medium configured to store or transport computer readable code, or in which computer readable code may be embedded." Schmidt-479, [0071]. Thus, Schmidt-479 is in the same field of endeavor as the '814 patent. *See* EX1001, 1:12-16 ("FIELD OF THE INVENTION: The invention relates to computer software. In particular, the invention relates to management and deployment of server applications.").

[0067] Schmidt-479 says as background information that it is "desirable for a user to be interacting with a computer, to stop the interaction with the computer, to move to a new computer, and to begin interacting with the new computer at precisely the point where the user stopped interacting with the first computer." Schmidt-479, [0005]. However, applications on a Unix OS, for example, are assigned a "Unix process identifier (PID), which is a token by which one process may refer to another process." Schmidt-479, [0012]. Schmidt-479 says, as POSAs knew, that "[a] PID is only valid on a particular machine and is not globally unique," and thus "[a] process cannot move to another machine and maintain the same PID." Schmidt-479, [0012].[2] For example, when a user

[2] A POSA understood that a conventional "processes" is an instance of executing computer applications. *See, e.g.*, EX1061, 1:13-16.

migrates to another machine IP-based communications would be disrupted because the user / applications being used by would no longer be associated with the previous IP address on the initial server.

[0068] Schmidt-479's solution to the above-identified issues of how to migrate applications to different machines, including those with different IP addresses, are software capsules having "tokens [that] are unique to each capsule and valid across all machines in the system." Schmidt-479, [0054]. Schmidt-479's Figure 3 (which I have colored below) illustrates two capsules (blue and green) on a computing device (e.g., a server, with hardware (grey) and an OS (red)). Each capsule is also assigned "a virtual namespace within them" (light blue and green, respectively) so that capsules (and the applications installed therein) "may communicate with hardware in a host-independent manner through the virtual namespace and could do so as well if migrated to a different instance of an operating system on a different machine." Schmidt-479, [0044].

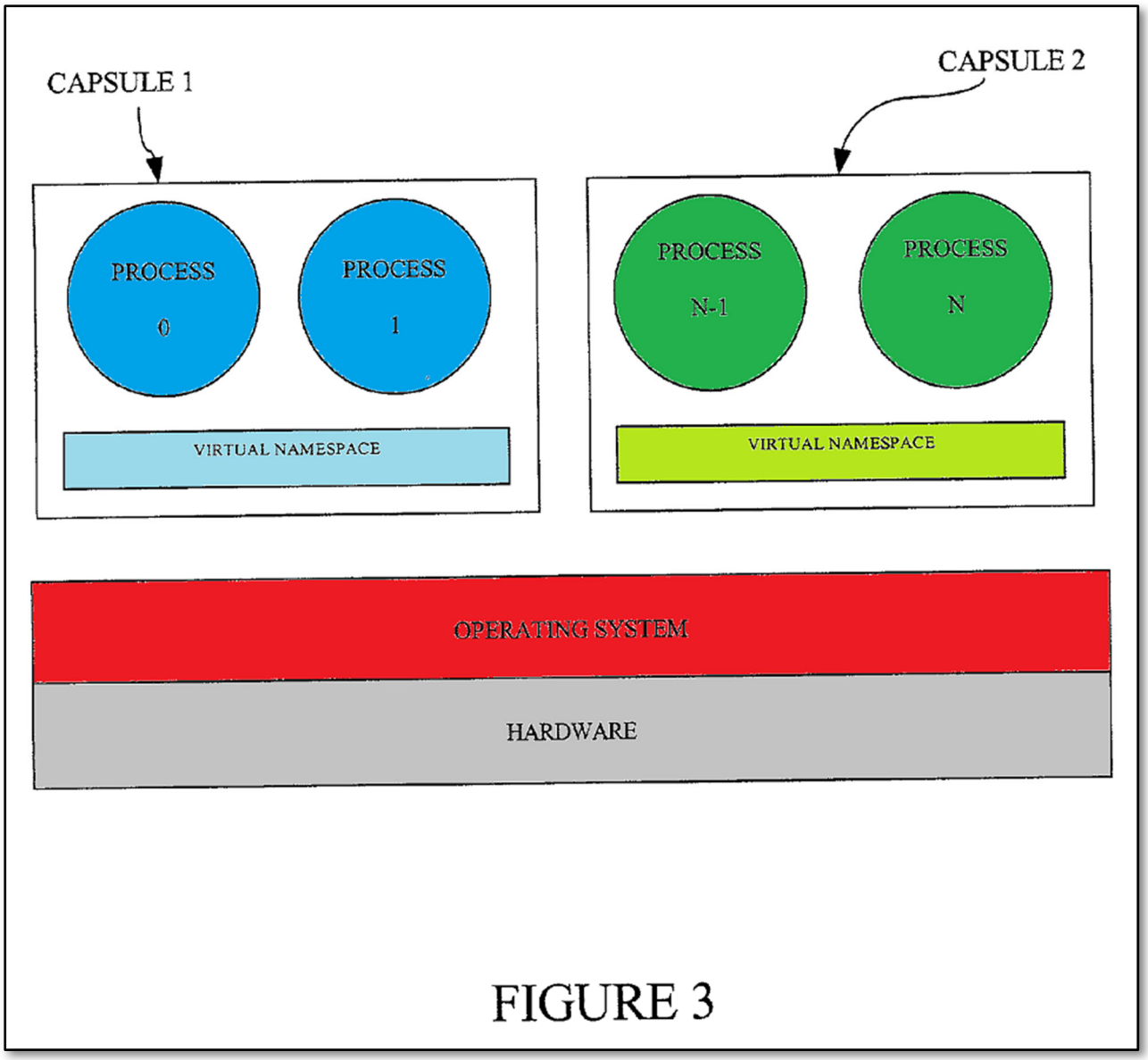


FIGURE 3

[0069] Each virtual namespace has its own "file system view" (Schmidt-479, [0045]), which can be "automatically populated with system directories and files necessary for normal user operation" (Schmidt-479, [0048]). Those "files include…application binaries, libraries…and data files." Schmidt-479, [0048]. Thus, Schmidt-479 says that each capsule "provides a 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 shared file system outside the virtual namespace. Schmidt-479, [0017], [0019].

[0070] Each capsule "comprises one or more processes and their associated system environment," including "state information relating to exactly what the processes are doing," where that state information includes system and other conventional files like "configuration settings, working directories and files, assigned resources, open devices, [and] **installed software**." Schmidt-479, [0038].

[0071] Schmidt-479 further discloses that each capsule's "default set of file system view mappings includes standard system directories, such as /bin **for applications**…." Schmidt-479, [0051]. As Schmidt-479 states, /bin is a directory that "contains application programs on Unix systems" and is an example of the "many standard, machine-local directories that are the same across systems." Schmidt-479, [0048]. That is, conventional deployments of a Unix-based OS will have a directory in the file system called "/bin" where application executables would be installed.

[0072] Additionally, Schmidt-479 says that each capsule's "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] For example, capsules can be "migrated to a different instance of an operating system on a different machine." Schmidt-479, [0044].

[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. My opinion in this regard is corroborated by EX1127, which discuses a server running the "server OS, which may be Redhat Linux, or another **binary compatible version of Linux**," which corroborates my opinion that binary compatible machines include machines that execute different but binary compatible OSs in the same OS-family (e.g., Linux). EX1127, 7:46-54. My opinion is further corroborated by EX1128, which says that "**binary compatibility** refers to maintaining compatibility between a binary (i.e., compiled) version of a first software module, e.g., an application module, and a second software module, e.g., **an operating system**." EX1128, 3:14-20. Thus, "[a]n application that is binary compatible with the ABI [Application Binary Interface] of a particular version of an operating system **should be compatible with <u>any other version of the operation system</u> that maintains the same ABI**." EX1138, 3:24-38 (noting that, for example, the "Solaris™ ABI changes infrequently").

[0073] Thus, Schmidt-479 allows one physical server with one OS (and OS kernel) to have thereon multiple capsules, where each capsule has its own view of a file system so that users of the capsule can have their own copies of applications (and associated files required to execute those applications) stored to, installed in, and executed within a capsule without affecting the file system within other capsule or the file system of the underlying OS. Furthermore, as discussed above, Schmidt-479 discloses a technique for storing a capsule and subsequently starting or restarting that capsule (and the applications installed thereon) any binary compatible server in a system, including a different server than the server the capsule was previously installed to and executed on.

B. Tormasov (EX1010)

[0074] Tormasov is a U.S. Patent Application Publication that was filed on July 30, 2001 and published on September 5, 2002. Tormasov generally relates "to the provision of full independent computer system services across a network of remote computer connections." Tormasov, [0003]. As I discuss further below, the computing devices on which Tormasov stores and installs its virtual computing environments (VCEs) include physical, hardware servers. Tormasov, [0017]. Tormasov's VCEs are software environments that permit users to install computer software applications for various services (e.g., website applications for website services). Tormasov, FIG. 1, [0017], [0024]. Thus, Tormasov is in the

same field of endeavor as the '814 patent. *See* EX1001, 1:12-16 ("FIELD OF THE INVENTION: The invention relates to computer software. In particular, the invention relates to management and deployment of server applications.")

[0075] Like Schmidt-479 (discussed above §VII.B), Tormasov provides a software-based technique to "efficient[ly] utilize[e]" "a single hardware system ['a server system'] with a single operating system kernel" by providing "a virtual computing environment [VCE] that is functionally equivalent to a computer with a full-featured" OS. Tormasov, [0017]. Like the '814 patent (EX1001, 1:51-2:3), Tormasov explicitly distinguishes VCEs from "a full hardware emulation-type solution" that require multiple OSs kernels (e.g., Virtual-Machine techniques). Tormasov, [0017], [0013]-[0016]. Tormasov says that "software emulators of the VMware type" (i.e., VM techniques), typically only permit "from about 2-3 to about 10-15" simultaneous emulators on one server because "the size of the emulated memory is close to the size of the memory used by the process or in which the computer emulator works." Tormasov, [0014]-[0015]. Tormasov says as background information that "[f]or an effect use of [server] equipment, the number of computers in a virtual environment installed in one host computer should be at least two to three times larger than the numbers mentioned above." Tormasov, [0016].

[0076] Thus, to overcome the limits of prior art techniques like VMs, Tormasov discloses having one server in a system of servers that can each execute multiple VCEs, each providing an equivalent of a full-featured OS, but without needing to run multiple OSs running kernels on the server. Tormasov, [0017]. That is, each server can execute multiple equivalents of a full-featured OS at the same time, using one server's hardware and OS kernel. (Tormasov also illustrates an alternative "full hardware emulation solution" [0022] in Figure 3, which includes multiple kernels on one machine.)

[0077] Tormasov's Figure 1 (which I have added color to below) shows "the same hardware node 100" (server) providing multiple VCEs (orange, blue, green) to "different users." Tormasov, [0024], [0017] (the "server…provide[s]" VCEs to "end user[s]"). In this example, one VCE executes an httpd (website) application (blue), another executes an Ftp (file-transfer-protocol) application (green), and a third VCE is executing a command-line shell (e.g., a Unix shell; orange).[4]

[4] POSAs knew that an Apache web-server application is an example of an httpd web-server. *See, e.g.*, EX1097, [0005] ("Many computer **network** server **applications** are developed using freely available tools like Linux™, **Apache™**,

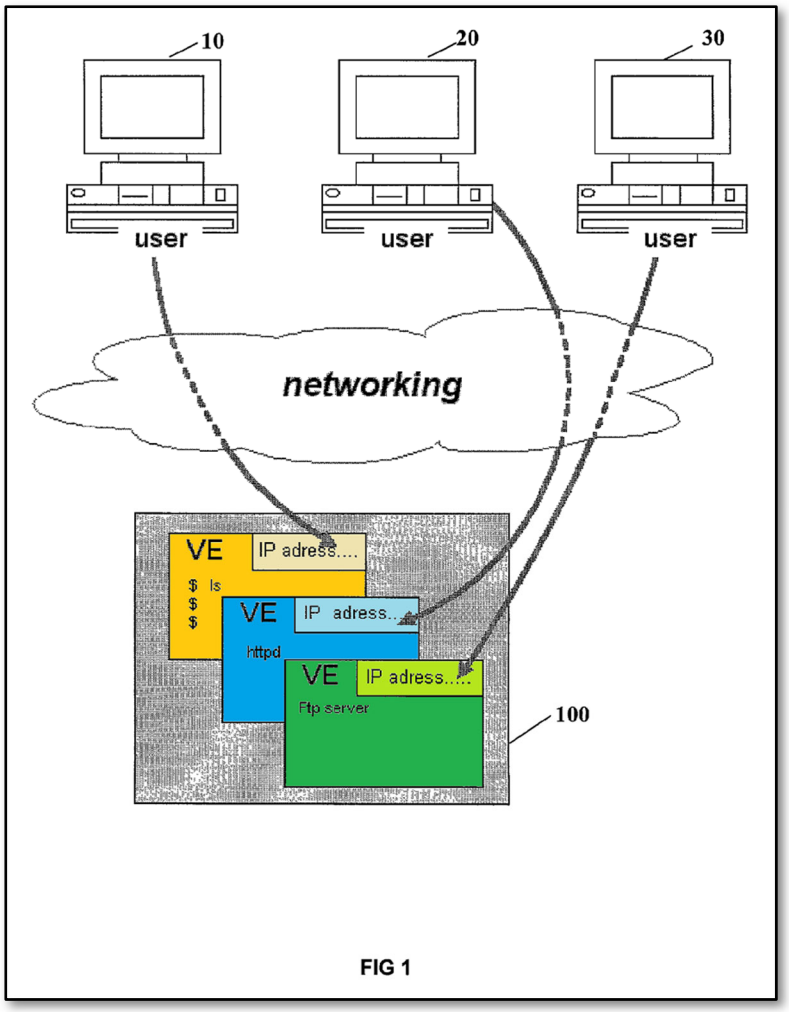


FIG 1

[0078] Tormasov's Figure 2 (which I have annotated below) shows additional details of a server executing two VCEs sharing the server's OS kernel (red) and hardware (grey). Tormasov, [0026]; *compare with* Tormasov, FIG. 3 (illustrating an alternative VM-based approach using multiple kernels). As shown below, each VCE has "its own unique file system" (light blue and light green) for

---

PHP™ and MySQL™."); EX1076, pdf page 6 (referring to "the popular **Apache web server**").

executing processes (blue and green) – e.g., processes of the above-discussed applications –that "are completely isolated from each other." Tormasov, [0026].

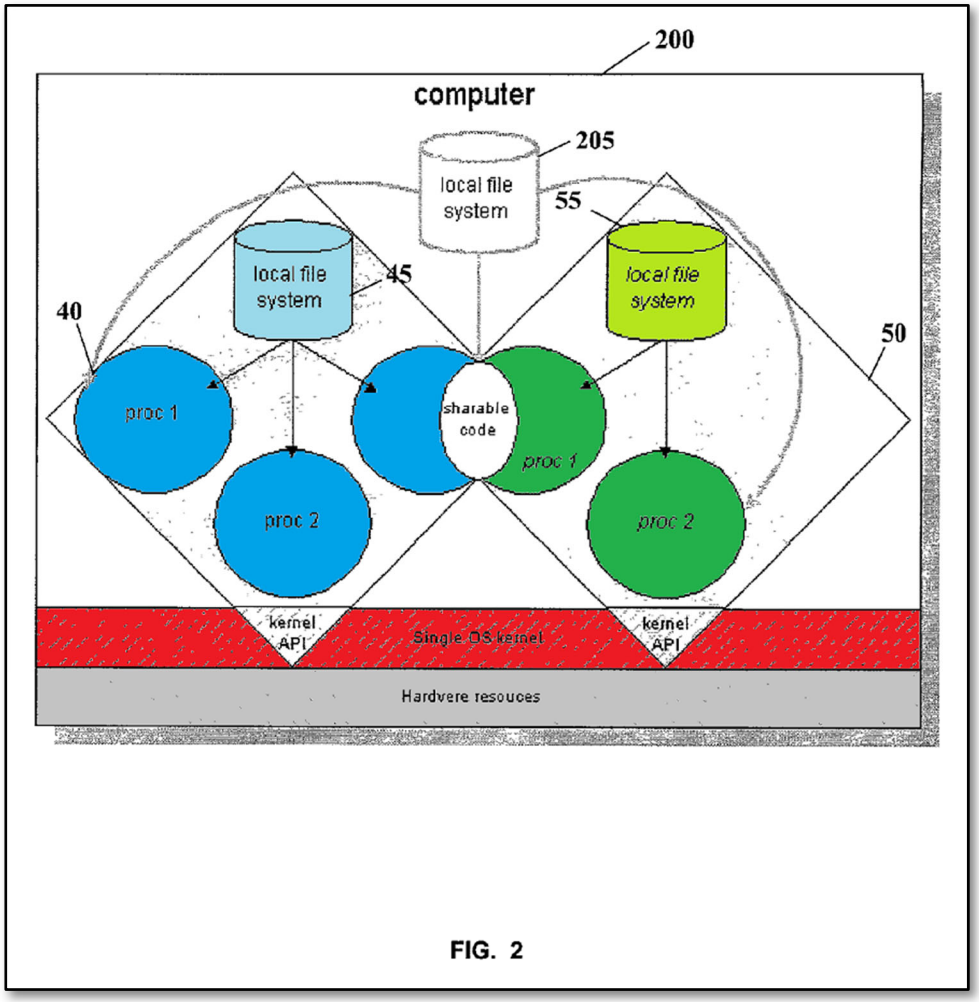


FIG. 2

C. Calder (EX1006)

[0079] Calder is a U.S. Patent Application Publication that was filed on November 29, 2000 and published on May 30, 2002. Calder generally "relates to distributed computing, and more particularly, relates to secure peer-to-peer Internet or enterprise distributed computing. The invention also relates to the secure execution of an application on a client computer." Calder, [0002]. As I

discuss further below, the computing devices on which Calder stores and installs application packages include server computers. Calder, [0007], [0073]-[0078]. Thus, Calder is in the same field of endeavor as the '814 patent. *See* EX1001, 1:12-16 ("FIELD OF THE INVENTION: The invention relates to computer software. In particular, the invention relates to management and deployment of server applications.").

[0080] Calder teaches techniques for modifying "an application package to be executed safely, securely, and transparently on a remote machine." Calder, [0073]. Calder teaches that conventional software "[p]ackages" that are used to install applications on computing systems typically include the application's binaries ("also called application program," which typically include, e.g., executable files to start and run the application) and system files, like libraries ("DLL[s]") and configuration files. Calder, [0076], [0082]. As I discuss further below, Calder's invention takes conventional packages for installing applications and outputs modified versions of the files in those conventional packages (e.g., "modified binaries [], modified libraries [], modified configuration files [],"Calder, [0082]).

[0081] My opinion that a typical, conventional software package in the prior art for installing an application typically includes, among other files, executables and libraries is corroborated by EX1014, which discloses:

Typically, **one software package includes all the files which are normally required for a local setup procedure and which correspond to the application being considered**. Those installation files clearly depends upon the type and the complexity of the particular application for which an installation is required. **Such installation files, including the Dynamic Link Libraries (DLL)** and all the subsequent files which are to be copied on the hard disk drive of PC client 3, for instance, are well known from the skilled man and will not, for that reason, be developed with more details. Typically, it is sufficient to observe that those files include all the files which are normally involved **in a local setup procedure and that the particular executable file—the setup.exe—which causes the launching of the installation procedure**, has to support an unattended mode, which is that which is being executed when the user types the "-s" switch on the command line (unattended or silent setup).

In addition to the installation files required for a standard local setup procedure, the software package located on server 2 further includes an **additional description file, hereinafter referred to as package.ini file**. Package.ini file may take the form of a text file and contains the description of the installation files which are involved in the setup procedure. **It particularly includes the precise list of the installation files required for a local setup procedure**, plus an additional line carrying the command which is required for starting the local setup procedure.

Considering the example of the Microsoft Office™ software package which is marketed by Microsoft™ Corp., server 2 is arranged to store

> **the standard Microsoft installation files**. In addition server 2 includes a package.ini description file which defines the list of those files and further comprises an additional line to run the silent setup procedure, i.e. the following line: "setup.exe -s".

EX1014, [0026]-[0028]. *See also* EX1005 ("Blaser") (noting that "[w]hen an application is installed to an OS, a number of globally accessible files are often placed to the computing system, including for example shared libraries and system configuration").

[0082] As Calder notes (and POSAs knew), one common function provided by libraries is to make system calls, which are requests to use a service or functionality provide by the underlying OS's kernel.[5] Some examples of conventional functions that are implemented via conventional system calls, are functions to open a file in the file system and use the computer's network

[5] My opinion that system calls are often made using one of an OS's libraries is corroborated by EX1120 (cited Tormasov, [0033]), which discloses that in a Unix-based OS, for example, "[s]ystem calls look like ordinary function calls in C programs, and libraries map these function calls to the primitives needed to enter the operating system….Programs frequently use other libraries such as the standard I/O library to provide a more sophisticated use of the system calls." EX1120, 19-20.

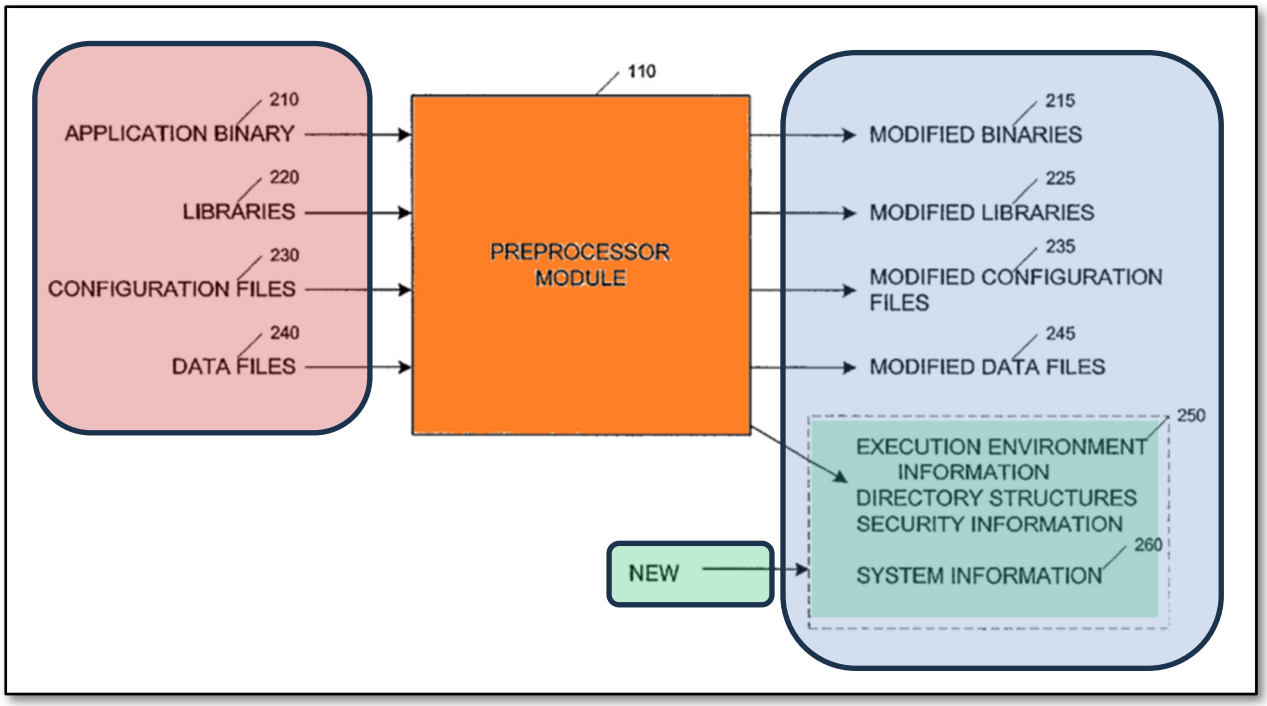
connection. However, POSAs knew that different OSs (then and now) often have different system calls and/or syntaxes for executing system calls, in particular OSs in different OS families.[6] Thus, an application package intended for one OS (a Windows-based OS) may not work with another OS (a Mac-based OS).

[0083] Calder's solution to permit an application made for one OS to execute on a different OSs having a potentially different set of system calls (e.g., OSs in different OS families) is to modify an "application package" by adding "system libraries…that translate" "system calls" intended for one OS (e.g., "Windows 2000") to work with a computer having a different OS, including an OS in a different OS-family (e.g., "Linux"). Calder, [0101]. For example, as I discuss further below, Calder teaches that "the application package is modified using a preprocessing module which, among other things, modifies the binaries of applications in the application package such that an interception module is loaded when the binary is executed." Calder, [0073].

[6] My opinion that OS in different families often used different system calls is corroborated by EX1097, which states (as POSAs knew) that "[d]ifferent operating systems provide different mechanisms for implementing system calls." EX1097, [0082].

[0084] As shown in Calder's Figure 2 (which I have annotated below), Calder's software-implemented "preprocessing module [orange]" processes an application package's original files (pink) to create modified files (blue) (e.g., modified libraries and configuration files) that "allow for the application 405 to execute and communicate on any non-native platforms" (i.e., to execute on an OS that is different from the application's "native" (originally intended) OS). Calder, [0101].[7]

[7] My opinion that POSAs understood Calder's non-native platform to be a computer executing an OS that is different from the application's native OS is corroborated by EX1054, which states (as POSAs knew) that a "'non-native' server is [] a server…**running an operating system that is <u>different</u> than**" "the client['s]" OS. EX1054, 4:37-41.



[0085] Calder discloses that "[i]n one embodiment, system resources are controlled by using the virtual layer 415 to intercept part or all of the application programming interface (API) routines that utilize these resources." Calder, [0086]. The virtual layer includes "an interception module" that "intercept[s]" "[p]art or all of the system call made by the application." Calder, [0087]. "By intercepting these interfaces, the interception modules can prevent a user of the client computer 140 from accessing the contents of the application package 115 and/or the application from improperly modifying or accessing data from the client computer." Calder, [0088].

D. The Schmidt-Tormasov and Schmidt-Tormasov-Calder Combinations (Grounds 1-2)

1. The Schmidt-Tormasov Combination (Ground 1)

[0086] As I discussed above (§VII.A), Schmidt-479 taught as background information that people often use "computer programs [applications]" that communicate with other "computers over a computer network." Schmidt-479, [0005], [0007]. Schmidt-479 further taught that it was often "**desirable**…to stop the interaction with the [remote] computer, to move to a new computer, and to begin interacting with the new computer at precisely the point where the user stopped interacting with the first computer." Schmidt-479, [0005]. Schmidt-479's application "capsules" address that need by using "virtual namespaces for active **computing environments**" that have stored therein the application and system files needed to allow those applications to be stored, moved, and started/restarted on a different server. Schmidt-479, [0002]. Schmidt-479's capsules share one server's hardware and OS (and kernel). Schmidt-479, FIG. 3, [0044], [0054].

[0087] As I discussed above (§VII.B), Tormasov, like Schmidt-479, is also about computing environments that can share a computing device's / server's hardware and operating system. Tormasov discloses servers that provide multiple, independent virtual **computing environments** ("VCEs") that are each "functionally equivalent to a computer with [a] full-featured operating system." Tormasov, [0023]. Because each VCE is functionally equivalent to a full-featured

OS, each VCE installed to and executing on a server can execute independent application-based services (e.g., independent websites and/or independent FTP-server applications). Tormasov, [0024], FIG. 1 (showing an httpd application and an ftpd application in different VCEs on one server)).

[0088] My opinion that POSAs understood based on their background knowledge the httpd application shown in Tormasov's Figure 1, for example, to be a service is corroborated by EX1122 – which is cited in Tormasov's "BIBLIOGRAPHIC" at Tormasov, [0035] – which notes that "httpd" is a "**service**." EX1122, 249.

[0089] My opinion that POSAs would have had understood Tormasov to be teaching (or at least suggesting) executing Tormasov's VCE on the servers of a system having a plurality of servers is further corroborated by EX1110 ("Tormasov-906"), which is another reference by the same inventors as Tormasov. Tormasov-906's virtual environments (VEs) that are each "a fully functional virtual machine" but, unlike prior art Virtual Machine techniques, "emulate[] no hardware" and share a server's OS's kernel, as in Tormasov. Tormasov-906, [0033]. Tormasov-906 discloses that VEs permits users "to install any software supported by the underlying operating system, such as their own web-servers with CGI-scripts and dynamic modules, email servers, ftp servers, RealAudio/Video

servers, X-servers with remote access and sshd servers." Tormasov-906, [0034]. Tormasov-906's Figure 3 shows multiple "Server[s] with [multiple] VE's [sic]."

[0090] As I discussed above (§VII.B), unlike prior-art techniques for executing independent environments on one machine by using multiple virtual machines on one physical machine (i.e., multiple environments executing independent OS kernels on one physical machine), Tormasov's VCEs (like Schmidt-479's capsules) can execute at the same time on the same server while sharing that one server's hardware and OS kernel and without providing a distinct kernel for each VCE (or capsule). Tormasov, [0017], [0023]-[0025]; Schmdit-479, FIG. 3.

[0091] Thus, in my opinion, POSAs would have had reason to place inside each Schmidt-479 capsule a Tormasov VCEs (and applications installed therein) so an installed capsule would have provided its user a "functional[] equivalent to a computer with full-featured operating system" (Tormasov, [0023]) and that could also beneficially be stored and executed on any server in a multi-server system (Schmidt-479, [0004]-[0005]), which Schmidt-479 says (as POSAs knew) would have been a "desirable" feature (Schmidt-479, [0005]).

[0092] My opinion that POSAs would have had motivation to have computing environments of applications that could be moved to and/or accessed from different servers (and a reasonable expectation of success in doing so) is

further corroborated by EX1083 ("Schmidt-500"), which is a published patent application by Brian Schmidt that was filed around the same time as Schmidt-479 and also addresses aspects of the inventor's compute capsule for "computing environments." Schmidt-500, [0002], [0014]. Schmidt-500 discloses as background information that users often use different machines to "access" the same program from "the **servers** where the programs where being executed." Schmidt-500, [0011]. Schmidt-500 further discloses an example of a system (illustrated in Schmidt-500's FIG. 10) where various servers 1010, 1011, and 1012 are providing "services" – e.g., "X11/Unix services, archived or live audio or video services, Windows NT service," etc. – being executed in Schmdit-500's capsules. *See* Schmidt-500, [0061]-[0063], FIG. 10 (executing capsules on servers 1010-1012).

[0093] In my opinion, POSAs would have also had motivation to implement Schmidt-Tormasov's VCE-capsules in a system having a plurality of servers running **different OSs**. That is because POSAs knew "[i]t is **important…that an application be usable with several different operating systems**." EX1058, 1:14-21. Similarly, "[s]oftware developers often strive to tailor or 'port' their applications to a variety of computing platforms," but "system-dependent variables, such as…operating system, make porting a difficult task."

EX1057, 1:10-14. For example, POSAs knew "it is desirable to have the 32-bit applications operate and function correctly on a 64-bit platform":

> The capability of running 32-bit applications on the 64-bit platform **is highly desirable for easing the transition from the popular 32-bit platform to the new 64-bit platform**. Not only may this ease the transition, it may **reduce the burden on software developers to immediately port their software to the 64-bit platform**. It also **gives the new 64-bit platform an existing library of software** (e.g., the old 32-bit applications). Furthermore**, it is desirable to have 32-bit and 64-bit applications interoperate**.

EX1017, [0029]-[0030].

[0094] POSAs also knew that networks for large corporate systems often had "computers spread across a wide geographical area that are networked together," with groups of users "interconnected" to "one or more server computers" that "**run[] different network operating systems**." EX1049, 1:12-14, 1:41-52. That is another reason why POSAs would have had reason to implement each of Schmidt-Tormasov's VCE-capsules (and the applications installed therein) to each be compatible with OSs that differ.

[0095] In particular, POSAs would have had motivation to make each Schmidt-Tormasov VCE-capsule (and the application(s) therein) compatible with multiple, different OSs (e.g., one VCE-capsule that would have been compatible with both a server running a Windows-based OS and another server running a

Linux-based OS) and deploy those VCE-capsules in a system having a plurality of servers running different OSs. That is because POSAs knew "[i]t is **important…that an application be usable with several <u>different</u> operating systems**." EX1058, 1:14-21. Similarly, "[s]oftware developers often strive to tailor or 'port' their applications to a variety of computing platforms," but "system-dependent variables, **such as…operating system** [i.e., operating system differences], make porting a difficult task." EX1057, 1:10-14. For example, POSAs knew "it is desirable to have the 32-bit applications operate and function correctly on a 64-bit platform":

> The capability of running 32-bit applications on the 64-bit platform **is highly <u>desirable</u> for easing the transition from the popular 32-bit platform to the new 64-bit platform**. Not only may this ease the transition, it may **reduce the burden on software developers to immediately port their software to the 64-bit platform**. It also **gives the new 64-bit platform an existing library of software** (e.g., the old 32-bit applications). Furthermore**, it is <u>desirable</u> to have 32-bit and 64-bit applications interoperate**.

EX1017, [0029]-[0030].

[0096] Moreover, as EX1049 corroborates, POSAs knew as part of their background knowledge in the art that "in a distributed system of a large corporation having many end users, there are usually a number of computers **with <u>different</u> operating systems**." EX1049, 1:41-58. As EX1051 similarly

corroborates, POSAs also knew that "[i]n larger networks, it is common…for the servers to be running **different network operating systems**." EX1051, 2:35-37. And, as EX1020 corroborates, in systems for large networks, it would have been common and beneficial to have "software…centrally hosted by a Server and used by a number of Clients." EX1020, [0004]-[0005].

[0097] My opinion that POSAs would have had motivation to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1054. EX1054 states that "[i]n the prior art, a user of a Windows NT client [] is typically authenticated by a Windows NT server." EX1054, 1:48-54; 4:48-50. EX1054 further describes a "server system" that "run[s] any of a number of computer operating systems," including various "[w]ell-known server operating systems like "Novell Netware, IBM OS/2 Warp Server, IBM AS/400®, Microsoft Windows NT, and many variations of OSF UNIX." EX1054, 1:14-35. "The server system is accessed by the client system for specific functions," including "execution of applications, and storage of and access to user information." EX1054, 1:27-35. As I discussed above, in a typical large network in the prior art, there would have been client computers running a variety of different OS. Thus, in light of that background knowledge, POSAs would have known that it would have been

conventional for the client computers in a conventional large network to access servers running a corresponding OS. For example, in a typical large network that included Windows-based OSs, POSAs knew as part of their background information the Windows-based clients in the network would have often accessed Windows-NT-based servers to execute Windows-based applications; OS/2-based clients would have often accessed OS/2-based server to execute OS/2-based applications; etc.

[0098] My opinion that POSAs would have had motivation to make each Schmidt-Tormasov VCE-capsule (and the applications therein) compatible with multiple, different OSs and deploy the VCE-capsules in a system with a plurality of servers running different OSs is further corroborated by EX1066. EX1066 states that it was known that "[t]here exist many **legacy server applications that have provided and continue to provide valuable information processing servers** to organizations worldwide. Many of these legacy server applications, which are the result of many years of development, contain, store or access many years (and sometimes decades) worth of data." EX1066, 1:17-37. Thus, in light of that background knowledge, POSAs would have had reason to make legacy applications in VCE-capsules executable on different OS, such as more-recent OSs that had additional features, such as improved security, speed, and the ability to support more-recent applications. Thus, for example, POSAs

would have had motivation to deploy several servers, each running a different state-of-the-art OS, while maintaining the ability for the network to execute and serve legacy applications designed for older OSs.

[0099] My opinion that POSAs would have had motivation to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1068. EX1068 states that "[a] current trend in computing is **to interconnect a variety of computer architectures and computer operating systems in a single network** 100." EX1068, 1:19-27. For example, a network that "includes a variety of servers, i.e., a Unix server 150, a Netware server 160, an OS/2 server 170, and a Windows NT server 180," which is "referred to as heterogeneous networks" and can "include local area networks." EX1068, 1:19-27. Thus, EX1068 further corroborates my opinion that it would have been typical and obvious in the prior art to have networks with a plurality of servers, where the servers executed different OSs.

[0100] My opinion that POSAs would have had motivation to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1071.

EX1071 notes as background information that "companies having large networked facilities **can internally have thousands of networked computers accessible by numerous content servers** on a single network. **Each of the content servers can be running any of various operating systems** as can the computers with which the servers are communicating." EX1071, 2:4-14. EX1071 further notes as background information that in those large systems "in the case of networked software, **each server has an individual copy of networked software** for a subset of the users." EX1071, 2:4-14. Thus, EX1071 further corroborates my opinion that it would have been typical and obvious in the prior art to have large networks that deployed a plurality of servers running different OSs, where the servers had copies of networked software for subsets of users. POSAs understood that the subsets of users were typically grouped based on location and/or role in the organization (e.g., engineering department, finance department, word-processing department, etc.).

[0101] My opinion that POSAs would have had motivation to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1016. EX1016 discloses as background information:

> Many businesses and other enterprises **use a variety of computer systems that are specially adapted for certain purposes**. For example, **a manufacturing division of an enterprise may use one kind of computer system specifically designed** to handle manufacturing data **while the sales division of the same enterprise may use another kind of system** for sales information. The **engineering division of the enterprise may use an entirely different computer system** as well. Using different computer systems for different divisions of an enterprise makes sense because **each kind of computer system will provide certain strengths that suit that division**.

EX1016, [0002]. Thus, EX1016 further corroborates my opinion that POSAs knew it was conventional to have large networks (e.g., large corporate, academic, and/or government networks) with a variety of computer systems, specialized for different divisions. POSAs further knew that the systems for different divisions often had need for different OSs. For example, a Unix- / Linux-based OS for the engineering division and a Windows-based OS for the secretarial division. At the same time, as I discussed above/below with corroboration, POSAs would have understood that certain applications would have beneficially been available across all divisions, regardless of the OS being run on the division's server(s) (e.g., e-mail applications, applications unique to the organization, etc.).

[0102] My opinion that POSAs would have had reason to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1053. EX1053 discloses as background information that UNIX was "[o]riginally designed for software development, computer-aided design (CAD), and other **technical applications run by engineers**." EX1053, 1:36-41. Thus, EX1053 further corroborates my opinion discussed above that in large organizations different divisions would have had need for applications (and corresponding application servers) designed for particular OSs (such as Unix-based OSs for an engineering division).

[0103] My opinion that POSAs would have had reason to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1049. EX1049 discloses as background information:

> The client computers within a group of computers are typically used by end users. **End users often have preferences regarding what operating system runs on their computer**. Therefore, in a distributed system of a large corporation having many end users, **there are usually a number of computers with different operating systems**.

EX1049, 1:53-58. Thus, EX1049 corroborates my opinion that it would have been conventional in a large network to have client computers running a variety of different OSs because different users had different preferred OSs (e.g., Windows-based versus Mac-based). As I also discussed above with corroboration, it would have also been conventional to have application servers running OSs matched to the OSs on the client devices (e.g., a Windows-based server to serve Windows-based clients).

[0104] My opinion that POSAs would have had reason to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1012. EX1012 discloses as background information:

> **Modern data processing systems are often implemented in "client-server" configurations in which a number of "client" computers**, typically personal computers or workstations, **communicate with one or more "server" computers and request such server computers to perform processing tasks**.
>
> In conventional client-server configurations, **each client computer is required to be compatible in some manner with each server computer so that the client and server computers can understand one another**. In some environments, this requirement is easily met, as **each client and server computer is of the same platform type, e.g.,**

UNIX, Windows, Macintosh OS. In other environments, **special translation protocols are implemented to permit a particular client platform type to communicate with a particular server platform type**.

As client-server systems become larger and serve entire enterprises, such as large corporations or institutions, conventional techniques for facilitating communication between and among client and server computers become more difficult to use, while **the need for processing that is seamlessly distributed among the systems' constituent client and server computers generally increases**.

For example, if a user of a particular client machine **creates a particular report application program** that provides a printed report of general value throughout the organization, **it would be extremely valuable for other users, perhaps working with clients and servers of different platform types, to be able to use the report application program**.

EX1012, [0002]-[0005]. Thus, EX1012 further corroborates my opinions that it would have been conventional in conventional large organizations in the prior art (1) to have a client-server model where the clients and servers ran a variety of different OSs; (2) to have servers running OSs matched to the client OSs and to support serving applications between clients and servers running different OSs; (3) to have applications where it would be beneficial for the same application

program to be available across the organization, regardless of what platform (e.g., OSs) the client devices were running.

[0105] My opinion that POSAs would have had reason to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1093. EX1093 discloses as background information:

> The present invention relates to the monitoring of server performance in a network like the Internet. System administrators, as well as information technology (IT) executives, generally like to know the productivity of their systems, and want to accurately forecast future needs for upgrading or expanding existing systems. Capital expenditures must be budgeted according to changes in server resource requirements, based on key server operating parameters such as CPU utilization, memory availability, I/O usage, and permanent storage (disk) capacity. This objective has become very difficult to achieve, due to extremely **complex environments of hundreds or thousands of networked servers running on many different platforms, including UNIX and its variations (such as AIX, HP-UX, System V, and Mach), Windows NT, OS/2, NetWare, etc. (UNIX is a trademark of UNIX System Laboratories; AIX is a trademark of International Business Machines Corp (IBM); HP-UX is a trademark of Hewlett-Packard Co.; System V is a trademark of American Telephone & Telegraph Co.; Mach is a**

trademark of NeXT Computers; Windows NT is a trademark of Microsoft Corp.; OS/2 is a trademark of IBM; and NetWare is a trademark of Novell Corp.). In addition to these platforms, some servers are managed by specialized systems, such as SAP (a suite of bundled business software including an accounts receivable program, a materials management program, a scheduling program, etc.).

EX1093, 1:66-2:25.

[0106] My opinion that POSAs would have had reason to make each of Schmidt-Tormasov's VCE-capsules (and the applications therein) compatible with multiple, different OSs and deploy them in a system with a plurality of servers running different OSs is further corroborated by EX1127. EX1127 discloses a computer accessing application programs from a remote "server farm" having a plurality of servers running different OSs (including OSs in different OS families):

> Referring again to FIG. 1A, appliance 132 may include a mother board and sound card, a video card, ports to plug in a keyboard and a monitor as well as some other digital equipment, but need not have a hard drive installed in it. Upon startup, appliance 132 automatically calls up the server farm 100 through either a telephone wire 112 or a cable modem, a digital subscriber line (DSL) or a network cable 111 on either a wide area network (WAN) or a local area network (LAN) to one of the remote servers in the server farm 100. The remote server acts as an application host and allows the user of the appliance 132 to choose from a number of operating systems offered by the remote server. **For**

> **example, the user can choose to operate in Windows NT, Windows 95, Windows 98, or Linux**. Once the automatic download of the OS is completed, the user is then able to choose from a number of applications. The applications may be the same applications that are commonly locally installed. The applications can be downloaded when they are to be used, without incurring large delays. The downloaded software is stored in random access memory (RAM), at least until the user logs out of the system. User data are stored on one of the severs of the server farm 100. No results have to be stored locally. Because all software is downloaded from the network, and all work is remotely saved in the server farm 100, the appliance 132 need not have a hard drive. The appliance can thus be less expensive than a conventional computer.

EX1127, 4:39-52.

[0107] Moreover, 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].

[0108] Thus, as I have demonstrated above with multiple pieces of corroborating evidence, POSAs would have had motivation to implement Schmidt-Tormasov's VCE-capsules (and the applications therein) in a system having multiple servers running disparate OSs (e.g., a conventional, large corporate

network), where each VCE-capsule (and the applications therein) could be executed on those different OSs.

[0109] As I discussed above (§V.C), during prosecution, the '814 patent's applicants told the Patent Office that different OSs in the same OS-family (e.g., different versions of OSs with the same tradename, like Unix-based OSs "Solaris 8" and "Solaris 10") are OSs that differ. EX1002, 300-301. By that same reasoning, Windows 95 and Windows 98 would be different OSs, different Linux distributions would be different OSs, etc.

[0110] As I discussed above (§VII.A), Schmidt-479's "capsules" have "file system views" containing applications and systems files (e.g., libraries) that allow the capsule and the applications / application executables therein to be stored and started/restarted on "a different instance of an operating system on a different" server. Schmidt-479, [0044], [0048].

[0111] POSAs knew that applications native to one OS (e.g., Windows 95) could often execute on another OS in the same family (e.g., Windows 98). In other words, POSAs knew that it would have been conventional to have applications that were executable on different OSs, at least under the inventors' argument that different versions of OSs in the same OS-family are different OS. My opinion that POSAs knew that applications native to one OS could often execute on another OS in the same family is corroborated by EX1017.

[0112] EX1017 discloses that it was known that "[p]eople want to run desired applications in their chosen platform in the ideal world" but it can sometimes be "very difficult to run an application in a different host platform that it wasn't written for." EX1017, [0019]. However, EX1017 also notes as background information that "[e]ach platform has its corresponding body of native applications that are designed to run under it." EX1017, [0020]. Thus, "[w]hen a new generation of platform is released, software developers generally upgrade their products to run under the new generation platform." EX1017, [0020]. EX1017 provides as additional background information that "OS developers **wish to make their products backwards compatible**" so that "**older generations of applications may run on the latest generation of the OS** (and thus the latest generation of platform)." EX1017, [0021]. In other words, product developers sought to develop applications and OS so that "non-native applications can run under a native platform (including the new OS)." EX1017, [0021]. POSAs knew that many applications for conventional OSs in the prior art were designed to be executable on multiple platforms (OSs), in particular OSs from the same (or kernel-compatible) OS family.

[0113] My opinion that POSAs knew that applications native to one OS (e.g., Solaris 8) could often execute on another OS in the same family (e.g., Solaris 10) is further corroborated by EX1128. EX1128 discloses that "it would be

desirable to provide a tool for application software developers to assess the compatibility of their software programs with existing versions of operating systems on which the software may execute." EX1128, 1:51-57. EX1128 further discloses "[a]spects of" the reference's invention, in context of "certifying binary compatibility between application software and the Sun Solaris ™ operating system":

> Aspects of the present invention will be described with reference to certifying binary compatibility between application software and the Sun Solaris ™ operating system, or more particularly between application software and the Sun Solaris™ Application Binary Interface (ABI). It will be appreciated, however, that the present invention is not so limited.
>
> **The term binary compatibility refers to maintaining compatibility between a binary (i.e., compiled) version of a first software module, e.g., an application module, and a second software module, e.g., an operating system.** The Solaris™ operating system complies with an ABI, which is a set of supported run-time interfaces available for an application to use. Among other things, the Solaris™ ABI defines the interfaces provided by the system libraries, the interfaces provided by the kernel system calls, the locations and formats of various system files and directories, and the input and output syntax and semantics of system utilities. **An application that is binary compatible with the ABI of a particular version of an operating system should be**

compatible with any other version of the operating system that maintains the same ABI.

The Solaris™ ABI changes infrequently, and only with significant advance notice.

EX1128, 3:7-30. Thus, EX1128 corroborates my opinion that it was conventional in the prior art for applications native to one OS (e.g., one particular version of the Solaris OS) to be compatible with other versions in the same OS family (e.g., another version of the Solaris OS that had a binary compatible ABI, which, as EX1128 recognized, "changes infrequently" between versions).

[0114] My opinion that POSAs knew that applications native to one OS were typically executable on another OS in the same family is further corroborated by EX1050. EX1050 discloses that:

> The manner in which UNIX implements System Calls, particularly to storage, is in many respects quite advantageous to system performance. In UNIX, t**he System Call is the interface between UNIX and an application program**. **A System Call by the application program requests the "kernel" portion of the UNIX operating system to perform one particular task or service on behalf of the operating system**. T**he "kernel" portion of UNIX includes approximately 60 System Calls which are not changed between different hardware systems, and are the standard interface to UNIX**. Other programs in UNIX adopt the kernel to the particular hardware environment.

EX1050, 2:25-38. As EX1050 states, all Unix distributions would have "approximately 60 System Calls which are **not changed** between different hardware systems, and are the **standard interface** to UNIX." Thus, POSAs would reasonably expect that software applications that relied on only standard UNIX system calls would for (at least) that reason be compatible with multiple versions of a Unix-based OS. POSAs would have reasonably expected the same to have been true regarding different versions of OSs within other OS families.

[0115] Thus, EX1017, EX1128, and EX1050 corroborate my opinion that POSAs would have reasonably expected that applications in Tormasov-Schmidt's VCE-capsules would have been compatible with multiple, different OSs (at least) under VirtaMove's argument during prosecution of the '814 patent that OSs that differ include OSs in the same OS-family (e.g., an application originally intended for Windows 95 then, and now, can also run on a computer running Windows 98).

[0116] Thus, Schmidt-Tormasov's VCE-capsules (and applications installed thereto) could have been (and POSAs would have had motivation for them to be) stored and executed on servers running different OSs in the same family. POSAs would have also known that some OSs in different OSs families had compatible kernels and used compatible system calls. Thus, POSAs would have also wanted Schmidt-Tormasov's VCE-capsules (and applications installed

thereto) to be executable on OSs in **different** OS-families but with compatible kernels, and would have reasonably expected success because, as noted above, it was conventional for applications to be compatible on (at least) some different OSs (e.g., OSs in the same OS family).

[0117] In my opinion, POSAs reasonably expected success in the Schmidt-Tormasov combination discussed above. 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]). Schmidt-479 further states that "[a]n embodiment of the invention maybe implemented in **any type of computer system or programming or processing environment**." Schmidt-479, [0072]. *See also* EX1009 ("Schmidt-529"), [0074] (disclosing embodiments of capsules "the present invention [of Schmidt's capsule technique] **can be implemented in any operating system environment**").

[0118] Tormasov's "virtual **computing environments**" similarly use unique local file systems and namespaces for execution on conventional computing systems having conventional hardware. Tormasov, FIGS. 1-2. Tormasov's VCEs are executable on any conventional OS, like those discussed in Schmidt-479. Indeed, Tormasov's "BIBLIOGRAPHY" cites textbooks describing features of the

"Unix Operating System" ([0033]), the "Linux" OS ([0035]), and designing a generic "Operating System" ([0034]).

[0119] Thus, in my opinion, it would have required only POSAs' ordinary skill to place a Tormasov's VCE (and applications installed thereto) inside a Schmidt-479 capsule so the VCE-capsule (and the system files and applications therein) could be stored, installed, and executed on any one of multiple servers running different kernel-compatible OSs in a multi-server system (e.g., large-organization networks of large corporations, academic institutions, and/or large government entities).

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

[0120] As I discussed above (§VII.D.1), POSAs would have had motivation to make Schmidt-Tormasov's VCE-capsules (and the applications therein) executable on servers in a system having a plurality of servers running different OS, and would have wanted and reasonably expected in having the VCE-capsules to be executable on (at least) different OSs in the **same** OS family (where the OS's are "different" under VirtaMove's prosecution statements that Solaris 8 and 10 are different OSs.)

[0121] However, POSAs also knew that the networks for large organizations in the prior art would have often had server computers and client computers executing OSs in **different** OS families and/or OSs with incompatible

kernels (e.g., one or more Windows-based servers and clients and one or more Unix-based OSs servers and clients). For example, EX1071 notes as background information that "companies having large networked facilities can internally have thousands of networked computers accessible by numerous content servers on a single network. Each of the content servers can be **running any of various operating system**s as can the computers with which the servers are communicating." EX1071, 2:4-14. EX1071 further notes as background information that POSAs knew in those large systems "in the case of networked software, each server has an individual copy of networked software for a subset of the users." EX1071, 2:4-14. Thus, EX1071 further corroborates my opinion that it would have been typical and obvious in the prior art to have large networks that deployed a plurality of servers running different OSs, where some servers had copies of networked software intended primarily for different subsets of users. POSAs would have understood that the subsets of users were typically grouped based on location (e.g., by country, time zone, etc.) and/or role in the organization (e.g., engineering department, finance department, word-processing department, etc.).

[0122] Thus, in my opinion, POSAs would have had motivation to make Schmidt-Tormasov VCE-capsules (and the applications therein) that were

executable on OSs in **different** families (e.g., Windows- and non-Windows-based OSs).

[0123] As I discussed above (§VII.A), Schmidt-479 discloses an example of a system having servers with OSs in different families. Schmidt-479, [0077]-[0078]. Moreover, other contemporaneous references published by one of more of Schmidt's inventors disclose that a "compute capsule is entirely self-contained and can be suspended in secondary storage, arbitrarily bound to different machines and **different operating systems**, and transparently resumed." EX1009, [0018].[8] *See also* EX1116, [0015] ("The capsule has a unique locator, such as an IP address, assigned to it. Using the unique locator, the capsule may be moved to a different machine **having potentially a different operating system** or on different network and maintain the open network connections it had prior to the migration."); EX1017, [0029]-[0030] (discussed *supra* §VII.D.1).

[0124] If VirtaMove were to argue Schmidt-479 (alone and/or in view of EX1009 and EX1116) does not teach/suggest making Schmidt-479's capsules (and the applications therein) executable on different OS families with a reasonable

[8] EX1117 claims priority to an application filed 2001-01-19, and thus I understand would be prior art to the '814 patent's earliest alleged priority date identified on the face of the '814 patent.

expectation of success, POSAs would have had motivations to do so (for the reasons I discussed above and below), and a reasonable expectation of success in doing so, by combining Schmidt-Tormasov with teachings from Calder. As I explain below, POSAs would have been motivated to apply Calder's teachings to adapt any suitable bundle of applications with system files (including Schmidt-Tormasov's VCE-capsules) to be executable on multiple, different OSs (including different OS-families).

[0125] As I discussed above (§VII.C), Calder's teaches modifying an "application package" (e.g., its libraries and configuration files) so "the application" can "execute and communicate on any non-native platforms," like allowing Windows-native applications to work on a server running a Unix-based OS. Calder, [0101]. As I discussed above (§VII.D.1), Schmidt-Tormasov's VCE-capsules would have included similar system files as discussed in Calder (before modified by Calder's technique) that would have allowed the capsule (and application(s) therein) to execute on any binary-compatible machine. Thus, even if a Schmidt-Tormasov's VCE-capsules would have been limited to execution on one OS (or one OS family), POSAs would have recognized that Calder's teachings could make any suitable application package with system files (like Schmidt-Tormasov's VCE-capsules) executable on multiple, different OSs (including in

different OS-families), which POSAs would have had reason to do, for the reasons I discussed above.

[0126] In the Schmidt-Tormasov-Calder combination (which I may also refer to as "Schmidt-Tormasov-Calder"), each VCE-capsule would have further comprised modified system files (e.g., modified libraries and modified configuration files, Calder, FIG. 2, [0101]) that would have allowed the VCE-capsule (and the application(s) within the VCE-capsule) to be executable on different OSs running in a system having servers running OSs that differ (by being different versions of an OS within the same OS-family and by being OSs from different OS families), as taught by Calder. For example, a native Windows 2000 VCE-capsule would have had "system libraries added to" the VCE-capsule (and applications therein) that would have executed software to "translate Windows 2000 system calls to Linux system calls" (Calder, [0101]), and vice versa

[0127] In my opinion, POSAs would have reasonably expected success in arriving at the Schmidt-Tormasov-Calder combination. Schmidt's, Tormasov's, and Calder's techniques are software-based methods, which would have required only ordinary skill to adapt (as necessary) to execute in the Schmidt-Tormasov-Calder combination discussed above. Moreover, Calder says its technique works with the same conventional OSs discussed in Schmidt-479 and Tormasov (e.g., Windows-, Unix-, and/or Linux-OSs). Calder, [0083]-[0084]. POSAs would have

also known that conventional prior art OSs relied on system-calls, which Calder's technique translates to allow applications to execute on non-native OSs (as well as their native OSs). Calder, Abstract.

[0128] Thus, based on Calder's disclosure, it would have required only POSAs' ordinary skill to add to Schmidt-Tormasov's VCE-capsules software that would have permitted the VCE-capsules and their applications to execute on different OS families in a multi-server, multi-OS system.

E. Grounds 1-2: Mapping of Schmidt-Tormasov (Ground 1) and Schmidt-Tormasov-Calder (Ground 2) to Claims 1-34

[0129] As I explain in the sections below (§§VII.E.1-34), in my opinion POSAs would have found the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations meet all limitations of claims 1-34 of the '814 patent, and therefore would have rendered those claims obvious to POSAs.

1. Claim 1

[0130] The parties agree that claim 1's preamble is limiting. EX1106, 6. As I discuss below (§§VII.E.1.a.-b), the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations each meet the preamble. I have divided claim 1's preamble into two parts addressed in separate sections below: the preamble's "system" portion, which I have designated limitation [**1PREA**] (§VII.E.1.a), and the preamble's "method" portion, which I have designated limitation [**1PREB**] (§VII.E.1.b).

a. [1PREA]

[0131] For convenience, I have further divided limitation [**1PREA**] into four sub-elements. For the reasons discussed below (§§VII.E.1.a.i-iv), it is my opinion that Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations meet [**1PREA**] under both parties' constructions.

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

[0132] As I discussed above (§VI.A.5), the parties have jointly construed "*servers*" as "physical servers" (EX1106, 6) and VirtaMove further argues that claim scope includes "a physical server running virtual machine software" (EX1105, 4).

[0133] Schmidt-479's Figure 12 (which I have highlighted below) teaches an example of executing capsules on a "***server*** 1126" (yellow), having various hardware (light gray), such as a "processor 1213" (dark gray). Schmidt-479, [0063]-[0067]. Schmidt-479 further discloses that, as "FIG. 3 shows" each capsule's "virtualized namespace" will "interact[] with the system…through a resource interface coupled to the ***operating system***, which is in turn coupled to the **machine hardware**." Schmidt-479, [0042]. *See also* Schmidt-479, FIGs. 1-3, [0042]-[0044]. Similarly, EX1083 ("Schmidt-500"), which is another reference by Schmidt et al, discloses executing Schmidt's capsules on a "host system" (a hardware computer) in a system having multiple "computers 1010, 1011, and

1012" offering various "services" for different conventional *OSs* (e.g., "X11 / Unix services…Windows NT service[s]"). EX1083 ("Schmidt-500"), [0061]-[0063], FIG. 10 (plural ***servers*** 1010-1012).

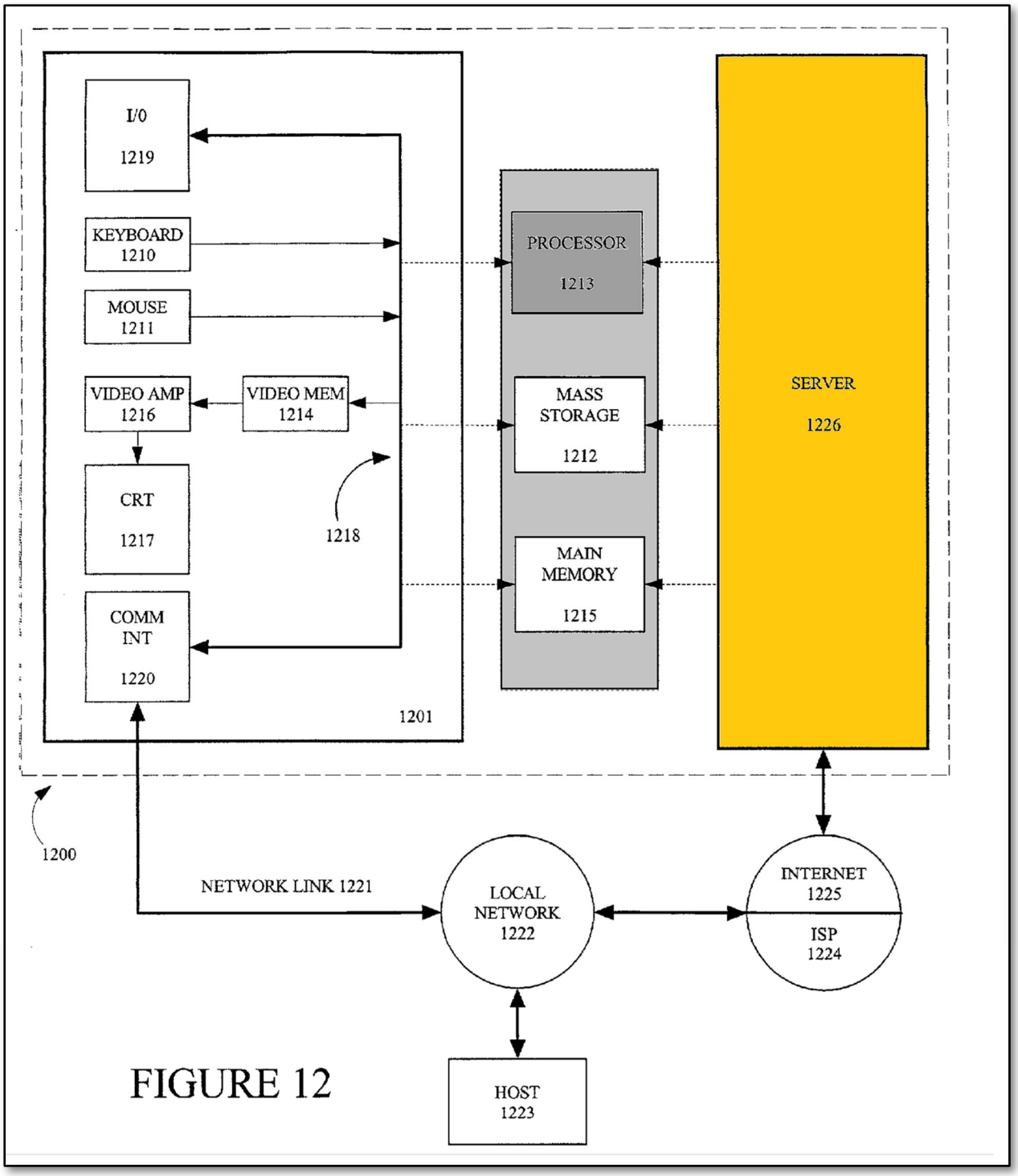

I/0
1219
KEYBOARD
1210
MOUSE
1211
VIDEO AMP
1216
VIDEO MEM
1214
CRT
1217
1218
COMM
INT
1220
1201
PROCESSOR
1213
MASS
STORAGE
1212
MAIN
MEMORY
1215
SERVER
1226
1200
NETWORK LINK 1221
LOCAL
NETWORK
1222
INTERNET
1225
ISP
1224
HOST
1223

FIGURE 12

[0134] Tormasov's Figure 2 (which I have annotated below) similarly teaches to execute VCEs on a "**hardware** computer" (a "***server*** system") using the computer's "**[h]ardware** resources" (Tormasov, [0017], [0026]).

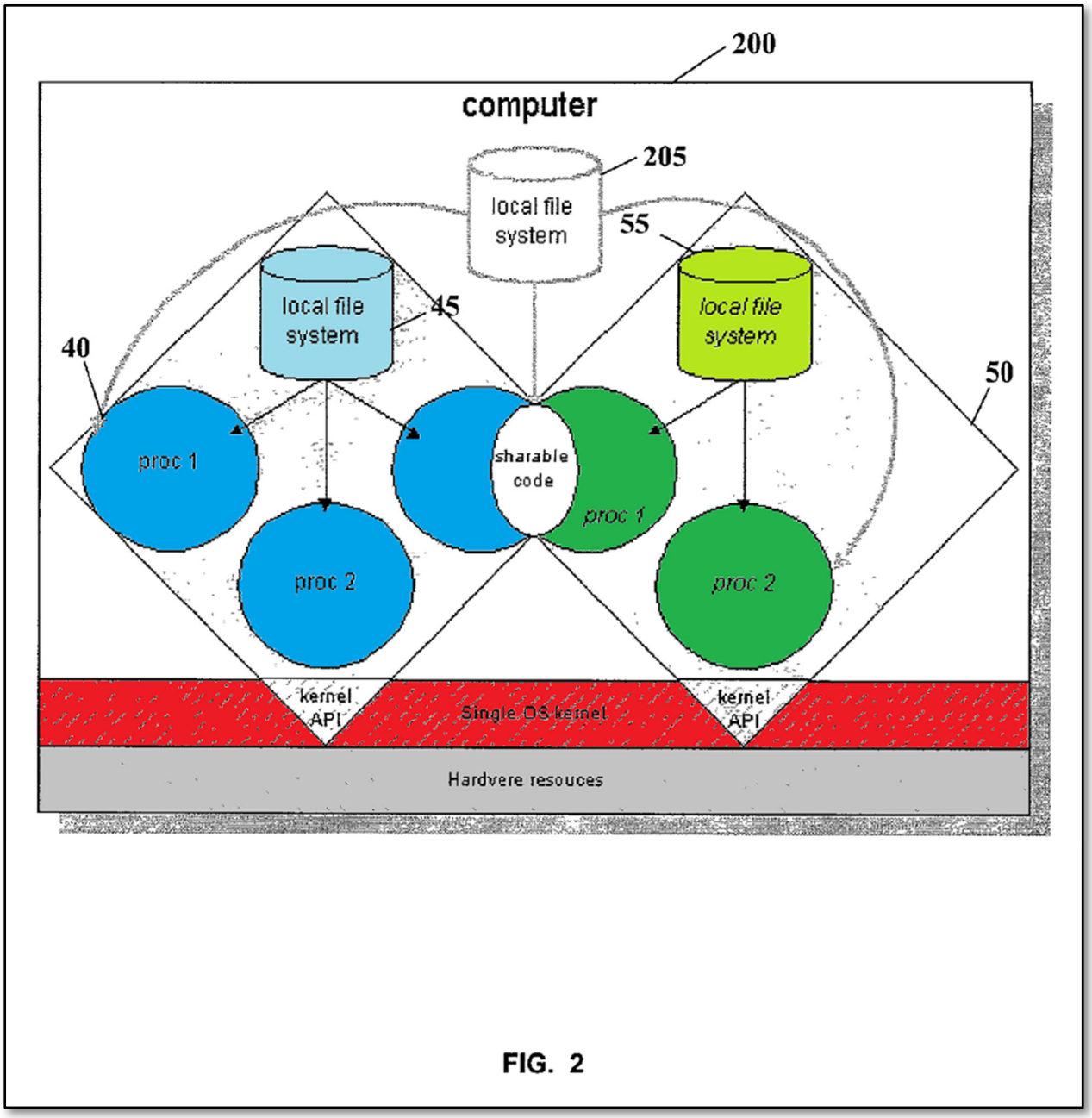


FIG. 2

[0135] Thus, based on Schmidt-479's and Tormasov's teachings, the *servers* in the Schmidt-Tormasov and Schmidt-Tormasov-Calder *systems* would have been physical *servers* (not merely software-based servers), which meet the parties' joint construction of "*servers*" as "physical servers." EX1106, 6; *supra* §VI.A.5. *See also* Schmidt-479, [0063] (below the heading "Embodiment of Computer Execution Environment (**Hardware**)").

[0136] EX1088's definition of "*operating system*" 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. That definition – which is Google's construction of "*operating system*" (EX1100, 2; EX1106, 4) – is consistent with a POSA's understanding of a typical, conventional operating system in the prior art. The '814 patent discloses that exemplary OSs include "Linux, Windows and Unix systems" and that "the invention is not limited to these operating systems and programming languages and other operating systems and programming language may be used." EX1001, 7:34-38.

[0137] As I discussed above (§§VII.D.1-2), the *plurality of servers* in the Schmidt-Tormasov and Schmidt-Tormasov-Calder *systems* would have executed conventional, prior art *operating systems*, including the same example OSs discussed in the '814 patent and identified in EX1088's definition discussed

above (e.g., Windows- and Unix-based OS). POSAs would have understood each of those OSs to be software that would have controlled the allocation and usage of hardware resources on each server such as its memory, CPU time, disk space, and peripheral devices. Thus, in the Schmidt-Tormasov and Schmidt-Tormasov-Calder *systems*, the OSs on the *servers* meet Google's construction of "operating system" (EX1106, 4) and VirtaMove's "plain and ordinary meaning" construction (EX1106, 4). I further note that VirtaMove has argued that it is not disputing that Google's construction is, at least, a valid exemplary definition of an *operating system* that is within the scope of the claims. EX1105, 4 (alleging that the parties do not "dispute [] the scope of" this term)).

[0138] My opinion regarding a POSA's understanding of the conventional, prior art *OSs* that would have been used in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's *servers* is further corroborated by Calder. Calder notes that "system resources" are typically accessed via "a system interface" to the computing device's OS. Calder, [0083]. Thus, Calder demonstrates that in a conventional computer, the computer's "resources" are controlled via the OS (which is why requests for resources are made via an interface to the OS).

[0139] My opinion that the conventional, prior art *OSs* that would have been used in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's *servers* would have met each party's constructions of "*operating system*" is further corroborated

by EX1121, which is cited as part of Tormasov's "BILBIOGRAPHY" (*see* Tormasov, [0034]). EX1121 discloses that an "***operating system*** is the layer of software that nearly every application software package expects to be present" (EX1121, 2), to provide "services" that "are invoked with a special 'trap' or 'system call' instruction" (EX1121, 3). EX1121 further states that "[t]he functions of an operating system" include being "a resource managed" that "manages the hardware resources of the computer system," such as the "*Processor*," "*Memory*," "*Input/output (I/O) controllers*," "*Disk devices*," and "*Other devices*." EX1121, 4.

[0140] As I discussed above (§VII.D), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's *systems* would have *had a plurality of servers with OSs that differ*, including within and across OS-families (e.g., various Windows-based and non-Windows-based OSs).

[0141] Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder meet *a system having a plurality of servers with operating systems that differ*, as claimed.

ii. "operating in disparate computing environments"

[0142] As I discussed above, the VCE-capsules in the Schmidt-Tormasov combination (§VII.D.1) and the Schmidt-Tormasov-Calder combination (§VII.D.2) "encapsulate" respective "***computing environment[s]***" (Schmidt-479,

[0016]), specifically "virtual ***computing environment[s]***" (VCEs) (Tormasov, [0023]).

[0143] 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. The '814 patent defines "disparate computing environments" as "[e]nvironments where computers are stand-alone or where there are plural computers and where they are unrelated" (EX1001, 2:17-19), but does not explain what the patent means to have "stand-alone" or "unrelated" computers.

[0144] As I discussed above (§VI.B.3), Google argues in the District Court litigation that this claim term is indefinite because it is unclear to POSAs what it means to have "unrelated" computers in context of the entire claim language. As noted above, Patent Owner, however, construes the term as "[e]nvironments run by standalone ***or*** unrelated computers." EX1103, 8. While Google explains that it is unclear what "unrelated" means (EX1102, 5-7; §VI.B.3), in my opinion, as I discuss below, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's plurality of servers (and client computers) would have been "standalone" computing devices, which I understand is sufficient for the Board to find that Schmidt-Tormasov and Schmidt-Tormasov-Calder meet *operating in disparate*

computing environments (even though it is unclear to POSAs what it means to have "unrelated" servers in context of the claim).

[0145] Consistent with Schmidt-479's Figure 12 and Tormasov's Figure 2 (both highlighted above), the servers in the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations (*supra* §V.E.1.a.i) are standalone computers (at least) because they are separate computer machines. *See also* Schmidt-479, FIGs. 1-3. For example, Schmidt-479 disclsoes that capsules can be "transparently and dynamically bound to **different physical resources** on **any host** in the system." Schmidt-479, [0054]. Each "host in the system" is a separate computer machine. Similarly, Tormasov's FIG. 1 illustrates different user "personal computers" 10-30 connecting to "the same hardware node 100." Tormasov, [0024]. Each hardware node 100 is a separate physical server.

[0146] Additionally, 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;

[0147] As I discussed above (§§VI.D.1-2), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's plural servers are separate computer machines and thus are stand-alone computers, which meets *operating in disparate computing environments*, even though it is unclear what makes computers "unrelated" in the context of claim 1.

[0148] Moreover, Patent Owner has argued that "standalone" "indicate[s] the ability of computers to operate independently of each other." EX1103, 9. The servers in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have (or in an obvious implementation thereof would have) operated independently of each other (because one server does not require the support of another server in order to execute applications on the server). Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's servers also meet *operating in disparate computing environments* under VirtaMove's apparent claim scope.

[0149] Furthermore, during prosecution, the examiner found that [**1PREA**]'s "*plurality of servers…operating in disparate computing environments*" was met by EX1010's disclosure of a "network environment" (EX1002, 239-240) that had a server and client (EX1089, FIG. 2, 6:28-35). The applicants did not disagree that a system like that taught *disparate computing environments*. As I discussed above (§VII.A-D), Schmidt-479, Tormasov, Calder, and the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations include systems with servers and client computers operating in a network environment. *See, e.g.*, Schmidt-479, FIG. 3, FIG. 12; Tormasov, FIGs. 1-2. For example, based on Schmidt-479's and Tormasov's similar teachings, in the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations, end-user client computers can access servers that provide VCE-capsules (and the servers providing layers are or had

reason to be clients of other servers, e.g., servers that manage and provide VCE-capsules to the server computers that interact with end-user client computers).

[0150] For example, Schmidt-479 notes as part of the background information of systems for which Schmidt-479's capsules could be used, "computer systems have been implemented that allow a user to access any computer in the system and still use the computer in the same manner." Schmidt-479, [0009]. The user would access those remote "computer systems" (server computers) over a network via a local, physical computer (client computer). As I discussed above, Tormasov's Figure 1 illustrates a similar system: different "personal computers" 10-30 (client computers) are connecting over a network to "the same hardware node 100" (a server computer). Tormasov, [0024].

[0151] Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder also meet *operating in disparate computing environments* under a claim scope found by the Patent Examiner that VirtaMove did not disagree with (as I explained above).

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

[0152] As I discuss below, e*ach server* in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's systems would have *included a processor and an operating system* based on Schmidt-479's and Tormasov's similar teachings.

[0153] For example, Schmidt-479's Figure 12 (annotated below) shows conventional computer hardware, which includes a "***processor*** [dark grey]"

that "reside[s]…wholly on [*each*] ***server***." Schmidt-479, [0066], FIGS. 1-3, [0008]-[0009], [0042]-[0044].

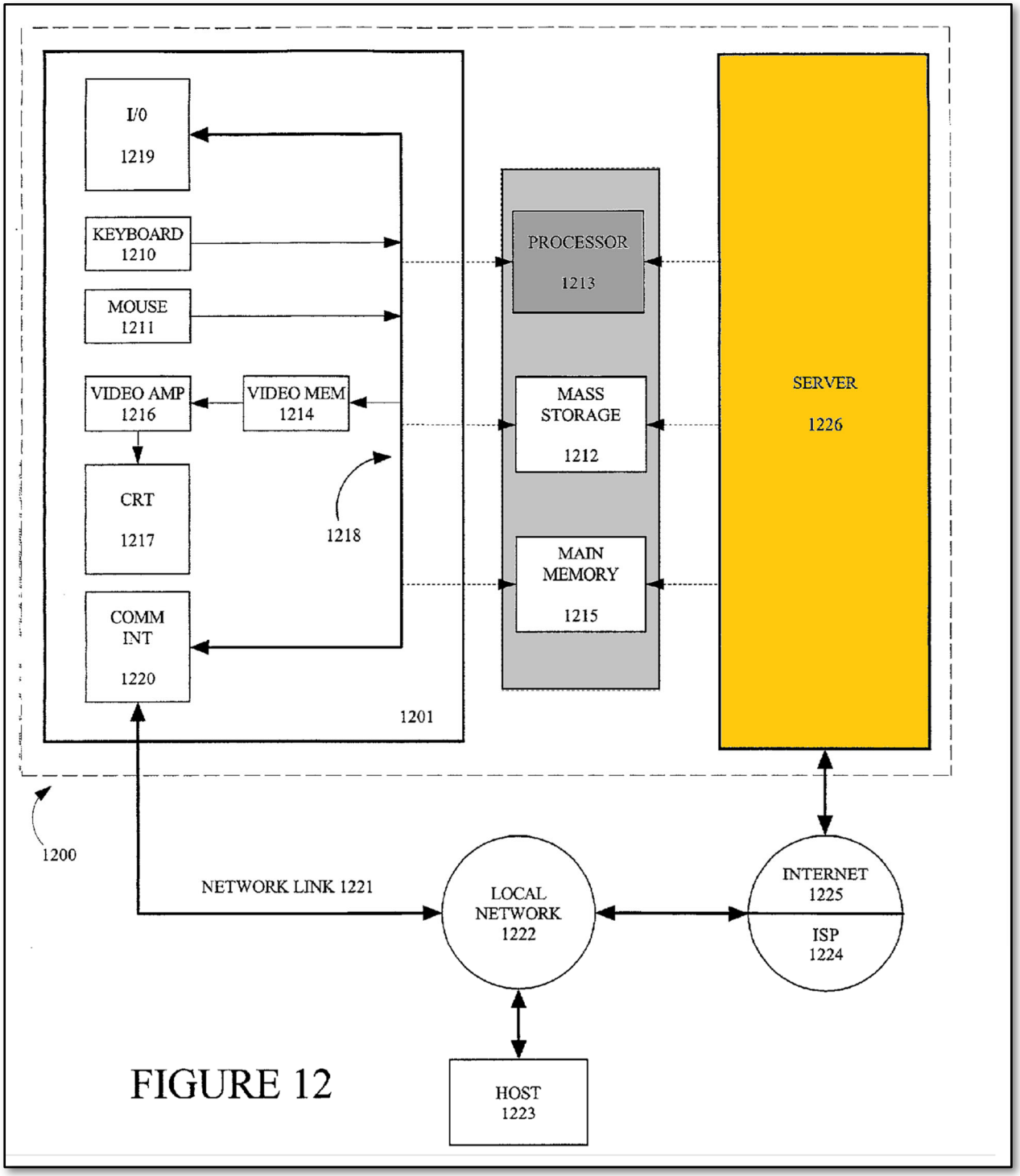


FIGURE 12

[0154] Similarly, Tormasov's Figure 2 (which I have highlight below) shows that *each server includes* "**hardware** resource[s] [grey]" and "one ***operating system*** [red]." Tormasov, [0027].

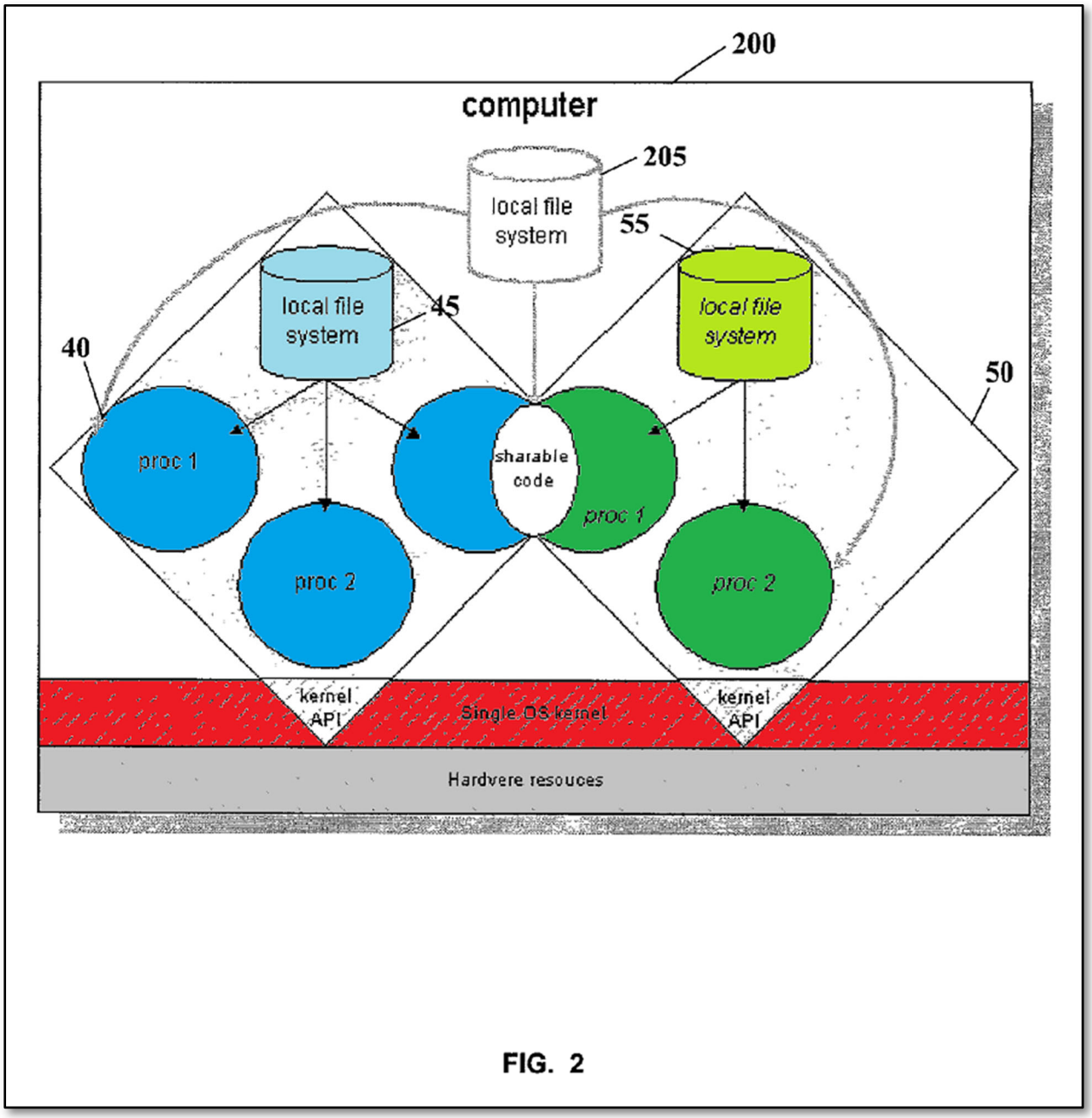


FIG. 2

[0155] Consistent with Schmidt-479's and Tormasov's similar disclosures discussed above, POSAs also knew that typical and obvious conventional *processors* are physical hardware (not merely software-based *processors*), and thus meet the parties' joint construction of "*processor*" as "physical computer processor." EX1106, 6.

[0156] My opinion that conventional *processors* in the prior art would have been physical computer *processors* is further corroborated by EX1121 (which is cited in Tormasov at [0034]). EX1121 is a textbook about the design of operating systems and discloses that the "main ***hardware*** *resources* in a computer system" include "the processor[, which] is the part of the computer system capable of executing instructions." EX1121, 4 (italicized emphasis original).

[0157] My opinion that conventional *processors* in the prior art would have been physical computer *processors* is further corroborated by EX1113 ("Tormasov-132"), which is an issued US patent that was filed around the same time as Tormasov by one or more of the same inventors that also discusses aspects of implementations of Tormasov's virtual computing environments. EX1113's claim 22, for example, claims "[a] computer useable medium having computer program logic stored thereon for executing on a ***processor***," where the "computer program code means [are] for establishing a namespace."

[0158] My opinion that conventional *processors* in the prior art would have been physical (hardware) computer *processors* is further corroborated by EX1017. EX1017 discloses as background information:

> Applications designed to run on a specific computing platform do not operate on a different computing platform. Generally, software is inextricably linked to the **computing platform on which it is designed to operate**. Software written and compiled to operate within minicomputer running a specific implementation of the Unix operating system will not function within a hand-held computer using a proprietary operating system.
>
> **A computing platform typically includes an *operating system* (OS) and computing hardware architecture**. Examples of OSs include these Microsoft® operating systems: MS-DOS®, Windows® 2000, Windows NT® 4.0, Windows® ME, Windows® 98, and Windows® 95. Examples of computing **hardware architecture** include those associated with these Intel® **microprocessors**: 80286, Pentium®, Pentium® II, Pentium® III, and Itanium™.
>
> Examples of c**omputing platforms** includes 16-bit platforms (such as Microsoft® MS-DOS® and Intel® 80286), 32-bit platforms (such as Microsoft® Windows® NT® and Intel® Pentium® II), and 64-bit platforms (such as Intel® Itanium™ and an appropriate 64-bit OS). A computing platform may also be called a platform, computing environment, or environment.
>
> …

Typical **Computer Architecture**

Typical computer architecture is multi-layered. From the bottom up, it includes the **hardware layer**, the ***operating system (OS)*** layer, and the application layer. Alternatively, these layers may be described as the hardware layer, the kernel mode layer, and the user mode layer.

FIG. 1 illustrates the layers of **typical computer architecture 100**. The top of the architecture is the user mode 110. It includes applications, such as applications 112a-e. These applications communicate with a set of APIs 120. Typically, this API set is considered part of the ***OS***, and thus, part of the computing platform.

…

Although FIG. 1 does not show the components 132-144 of the kernel 130 with connections between them, these components are connected as is necessary. The coupling lines are omitted from the drawing for the sake of simplicity and clarity.

Below the kernel mode 130, there is the hardware layer 150. This layer includes all of the hardware of the actual computer. This includes the ***processor(s)***, memory, disk I/O, other I/O, etc. The platform also includes the hardware layer.

Therefore, a computing platform includes the hardware layer, the kernel layer, and typically the user- mode APIs 120.

EX1017, [0003]-[0017].

[0159] Thus, the references discussed above further corroborate my opinion that it would have been typical and obvious for *each server* in the Schmidt-Tormasov and Schmidt-Tormasov-Calder *systems* to have *included a* physical (hardware-based) *processor* that meets the parties' joint construction of "*processor*" as "physical computer processor." EX1106, 6.

[0160] As I discussed above (§§VII.D.1-2, §VII.E.1.a.i), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's *servers* would have run conventional prior art *OSs* (e.g., Unix- and Windows-based *OSs*). Tormasov, [0033] (citing EX1120: which is about the "Unix" *OS*), [0035] (citing EX1122: which is about the "Linux" *OS*); Schmidt-479, [0077] (which references using the "Unix," Windows NT" *OSs*); Calder, [0084] (which references using "UNIX, LINUX," and various "Windows[-based]" *OSs*).

[0161] As I discussed above (§VII.E.1.a.i), POSAs knew that the conventional prior art OSs like those that would have been used in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's *servers* would have been software that controls the allocation of hardware resources such as memory, CPU time, disk space, and peripheral devices, which meets Google's construction of "*operating system*" and VirtaMove's "plain and ordinary meaning" construction. EX1106, 4; *see also* EX1088, 378 (providing a definition of an "*operating system*" that

matches Google's construction and is consistent with a POSA's understanding of a conventional, typical prior art OS).

[0162] Thus, the conventional prior art OSs that would have been used in Schmidt-Tormasov and Schmidt-Tormasov-Calder meet each party's construction of *operating system* (EX1106, 4), as I discussed above (§VII.E.1.a.i).

iv. "[OS] including a kernel [and][9] a set of associated local system files compatible with the processor"

[0163] 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. In other words, the patent recognizes that *system files* like shared libraries and configuration files were not only known in the prior art but were often included as *local system files* with conventional prior art OSs.[10] Indeed, the patent provides in its "**BACKGROUND**" section an "example" of "shared libraries, supplied by the [Linux Apache] OS distribution, which are 'system' files" and examples of "configuration files, also provided with the OS distribution." EX1001, 2:55-3:13,

[9] My understanding is that the parties agree "and" should be inserted. EX1106, 6.

[10] Whether any OS available today includes *system files* is not something I have been asked to analyze and it would require me at least to study the documentation and operation of any particular OS.

3:14-19 (admitting that "these shared library files and configuration files form system files provided by the operating system" and "[t]here may be any number of other files included as system files," including "for example, to support maintenance activities or to start other network services to be associated with a container"), 10:39-50 (discussing making containers using "s copy of the system files provided with a given operating system (OS) distribution"; i.e., using "the OS provided system files before an application is installed").

[0164] 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. VirtaMove argues that "*kernel*" should be given its "plain and ordinary meaning." EX1106, 4. *See also supra* §VI.A.3 (where I discuss my understanding of each party's construction of "*kernel*").

[0165] Schmidt-479's Figure 3 (which I have highlighted below) shows that each server includes an "operating system" (red) – with a *kernel* – and hardware (gray) (e.g., a *processor*).

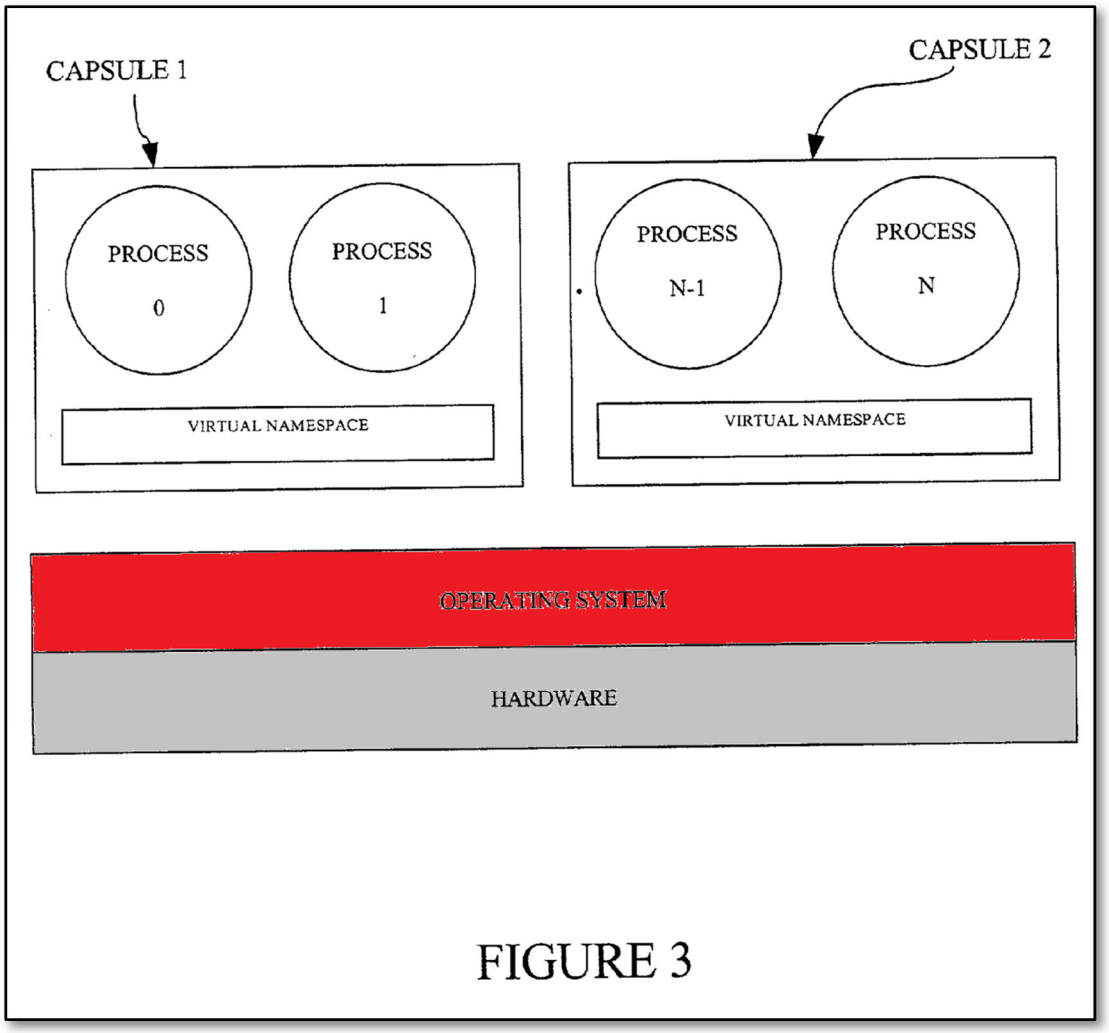


FIGURE 3

[0166] My opinion that POSAs would have understood that the conventional prior art OSs that are used in Schmidt-479's servers (*e.g.*, Windows-, Unix-, Linux-based OSs) would have had a conventional, prior art *kernel* is corroborated by EX1086. EX1086 discloses that "the operating system is commonly called the system ***kernel*** 130, or just the ***kernel***, emphasizing its isolation from user programs." EX1086, 2:53-55; *see also* EX1120 (cited in

Tormasov at [0033] and EX1086 at 2:58-60) at 4-5 ("[T]he operating system is commonly called the *system kernel*, or just the kernel….") (emphasis original).

[0167] My opinion that POSAs would have understood that the conventional prior art OSs that are used in Schmidt-479's servers (*e.g.*, Windows-, Unix-, Linux-based OSs) would have had a conventional, prior art *kernel* is corroborated by EX1017. EX1017 discloses:

> Typical **Computer Architecture**
>
> Typical computer architecture is multi-layered. From the bottom up, it includes the hardware layer, **the operating system (OS) layer**, and the application layer. Alternatively, these layers may be described as the hardware layer, **the *kernel* mode layer**, and the user mode layer.
>
> …
>
> **The next layer of the architecture is the *kernel* mode 130. This may be generally called the "*kernel*" of the OS. Since it is part of the OS, it is part of the computing platform**.

EX1017, [0009]-[0013].

[0168] Similar to Schmidt-479's disclosure and a POSA's background knowledge of OSs discussed above, Tormasov's Figure 2 (which I have colored below) shows that each server includes "an ordinary single ***kernel* operating system** [red]" and a "common **file system** 205 [pink]." Tormasov, [0025]-[0026].

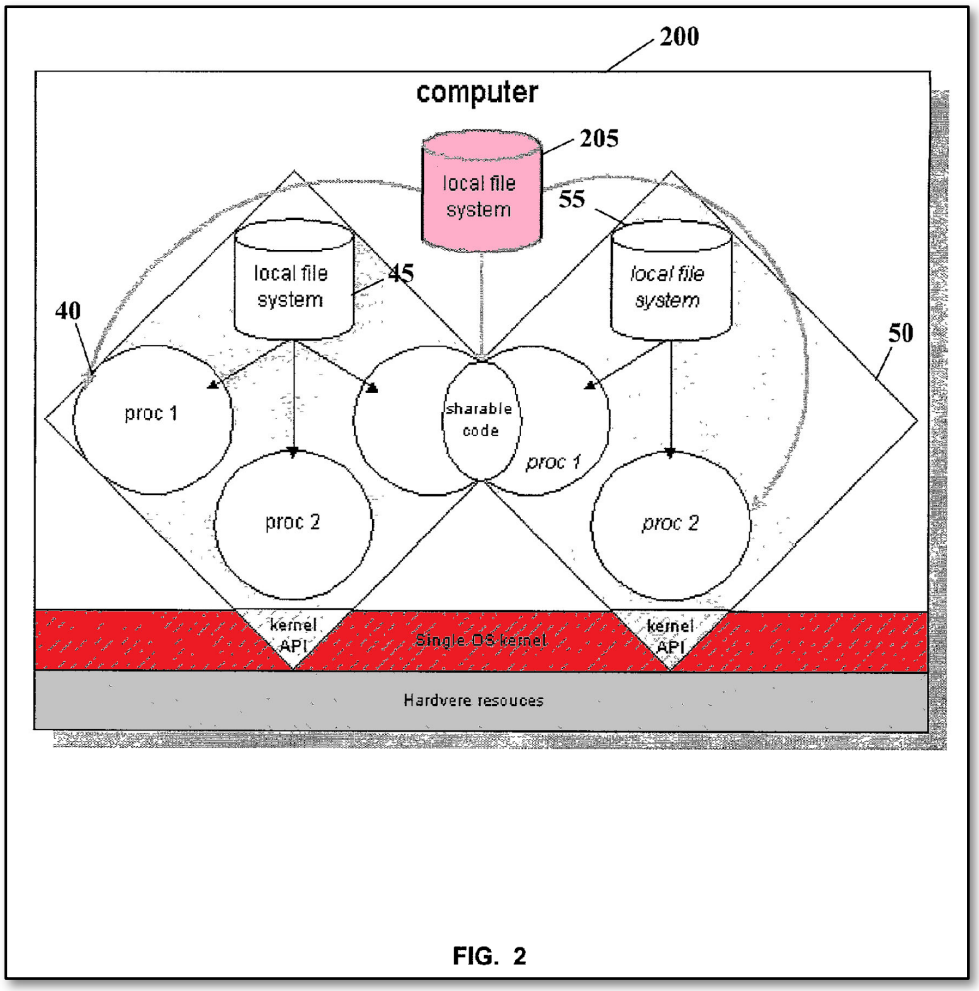


FIG. 2

[0169] POSAs understood that conventional OS *kernels* like the ones Schmidt-479 and Tormasov teach to use in their servers are the core of the OS and the portion of the system that manages memory, files, and peripheral devices, maintains the time and date, launches applications, and allocates system resources. *See* EX1088, 300 (providing a definition of "*kernel*" that is consistent with a POSA's understanding of a conventional OS *kernel* in the prior art). Thus, the *kernel* that would have been used in each Schmidt-Tormasov and Schmidt-Tormasov-Calder server (or, at least, an obvious implementation of those *kernels*

for use with conventional prior art OSs) meets Google's construction of ***kernel*** (EX1106, 4) and VirtaMove's "plain and ordinary meaning" construction (EX1106, 4).

[0170] My opinion in that regard is further corroborated by EX1121. EX1121 discloses that disclosing that an operating system "provides an interface between the hardware and the user programs" (EX1121, 2) and that "[t]he functions of an operating system" include being "a resource manager" that "manages the hardware resources of the computer system," like "the processor," "the memory," the "[d]isk devices." EX1121, 4. As I discussed above with corroboration, the part of the OS that manages access to the above resources in a typical, conventional prior art OS was the OS's *kernel*. Thus, EX1121's description of the resources managed by an operating system is consistent with EX1088's definition of "*kernel*" (which is also Google's construction of "*kernel*," EX1106, 4) and VirtaMove's "plain and ordinary meaning" construction (EX1106, 4), that I discussed above.

[0171] 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. However, the '814 patent does not allege to have invented OS *system files* and instead relies on conventional

system files that were typically provided with prior art OSs, like shared libraries and configuration files. EX1001, 2:52-3:19.

[0172] As I discussed above (§VII.A), Schmidt-479's servers have "an underlying file system" with "configuration files" and "system libraries." Schmidt-479, [0050]-[0051], [0054] . POSAs would have also understood that a typical, conventional prior art OS's "***local*** file system" as used in Tormasov (Tormasov, FIG. 2) comprises (or in a typical, obvious implementation thereof would have comprised) *local system files*, like shared libraries **and** configuration files. *See also infra* §VII.E.1.c.iii (where I demonstrate that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have comprised *system files*).

[0173] Moreover, EX1122 (which is cited Tormasov, [0035]) discloses that when you "restrict users' access to your system" and provide, for example, an ftpd service in a restricted environment, as in Tormasov (*see* §VII.B), "you need to move all the files that ftpd needs into the restricted login directory," including the library files in the directory "lib." EX1122, 122-123. Similarly, another reference (EX1113; "Toramsov-132") that was filed around the same time as Tormasov by one or more of Tormasov's inventors and that also discusses aspects of implementations of Tormasov's virtual environments. EX1113 notes as background information that in an "[o]rdinary computer…all of the *system* data

storage *files*, including the **configuration of files**, are usually seen by all the users of the operating system, and they are typically seen only in read-only mode" but "it is often necessary to provide for the possibility of multiple users making corrections to the system area." EX1113, 1:42-48. EX1113 further discloses that in light of the above, it would have been desirable in a shared system (as in Tormasov) for "[e]ach of the computer users [to] ha[ve] his/her own version…[of] the ***system file*** **area** of the operating system." EX1113, 2:7-11. That "own version" of the *system file* area would have been understood by POSAs to be a copy of a conventional OS's *system file* area (e.g., libraries and configuration files) that the virtual environment would have had access to in order to freely install applications, configure the environment, etc., without disturbing the copies of that *system file* area that is provided by the underlying OS and copied to other virtual environments on the same physical machine.

[0174] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would comprise (or in a conventional implementation would have comprised a *kernel and local system files*) is further corroborated by EX1120, which Tormasov cites as part of its "BIBLIOGRAPHY" (*see* Tormasov, [0033]). EX|1120 is about the Unix OS and discloses that the Unix OS has "three levels: user, ***kernel***, and hardware," where "[t]he system call and **library** interface represent[ing] the border

between user programs and the kernel." EX1120, 19-20, FIG. 2.1. EX1120 further discloses that the Unix OS includes "[s]ystem configuration" functionality so that "administrators can hard-code configuration data into files that are compiled and linked when building the kernel code." EX1120, 313-314.

[0175] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is corroborated by EX1029. EX1029 discloses that the "minimum…OS components **include[] a *kernel***, hardware drivers, **and *system files***." EX1029, [0027]

[0176] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is corroborated by EX1090. EX1090 discloses that the Linux "file system may include **configuration files**, application and OS binaries, **shared libraries**." EX1090, [0053].

[0177] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is corroborated by EX1108. Similarly, EX1108 discloses as background information that "[u]sers can tailor the operation of [various applications] applications to suit their needs by specifying various **configuration parameters**." EX1108, [0003]. EX1108 further discloses

as background information that the "values of the **configuration parameters can be stored in application-specific configuration *files*** such as **UNIX resource files** or in a central registry such as the **Windows® registry file**." EX1108, [0003].

[0178] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is further corroborated by EX1011. EX1011 discloses that "[e]very windows application is registered in a **windows *kernel*** with an application name." EX1011, [0032]

[0179] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is further corroborated by EX1067. EX1067 discloses that conventional "**operating *system files***" include "[t]he operating system ***kernel***, which forms the operating system's core program" EX1067, 9:12-20.

[0180] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is further corroborated by EX1064. EX1064 discloses:

> **A traditional operating system is logically layered and divided into two main portions: the *kernel*** and user programs.[11] The kernel interfaces with and controls the hardware, and also provides the user programs with a set of abstract system services called system calls. The kernel runs at a kernel level, where it can execute privileged operations and allows the kernel to have full control over the hardware as well as user level programs. This centralization provides an environment where all programs share the underlying hardware in a coordinated fashion.

EX1064, 1:53-62.

[0181] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is further corroborated by EX1062. EX1062 discloses that "[m]any computing platforms **support standard shared libraries**. For example, the **Windows platform supports Dynamic Link Libraries (DLLs)**, the **Unix platform supports Shared Libraries**, and the

[11] Though EX1064 calls the "two main portions" "the kernel [or 'kernel level'] and user **programs**," POSAs conventionally refer to the two main portions of a conventional OS as "kernel **mode**" and "user **mode**." E.g., 1034, [0041] ("Kernel mode is a privileged processor mode that allows system code to have direct access to hardware resources."), [0047] ("A user mode is a mode of the host microprocessor that does not allow code to directly access hardware resources.").

Macintosh platform supports Code Fragments." EX1062, 3:37-40. As I discussed above, libraries are examples of *system files* identified in the '814 patent. Thus, EX1062 corroborates my opinion that conventional OSs like those discussed in Schmidt-479, Tormasov, and Calder had *local system files* (e.g., shared libraries).

[0182] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers would have comprised a *kernel and local system files* is further corroborated by EX1023. EX1023 discloses that Windows-based OSs have "***system files*** which form the ***kernel*** part of the operating system." EX1023, [0023]. EX1023 further notes that these OSs have "customized **configuration files** [examples of *system files* in the '814 patent]" and "***system files*** of the operating system." EX1023, [0027].

[0183] Thus, each conventional prior art OS *kernel* that would have been used in each server in Schmidt-Tormasov and Schmidt-Tormasov-Calder meets Google's construction and VirtaMove's "plain and ordinary meaning" construction of "*kernel*." EX1106, 4.

[0184] Each Schmidt-Tormasov's and Schmidt-Tormasov-Calder server's ***kernel*** and ***set of associated local system files*** on each server are ***compatible with the*** server's ***processor*** because the processor in each server uses those *local system files* and the *kernel* on the server. Otherwise, if the *kernel* and

local system files were not compatible with the server's *processor*, the *processor* would not be able to use the *kernel* and *local system files.*

[0185] My opinion that each server in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have had a conventional *kernel* and *set of associated local system files* that would have been compatible with the server's processor is further corroborated by EX1067. EX1067 discloses that "for Windows 2000 to boot correctly on a computer…**the active kernel must be compatible with the computer's processor** configuration." EX1067, 35:48-54

[0186] My opinion that each server in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have had a conventional *kernel* and *set of associated local system files* that would have been compatible with the server's processor is further corroborated by EX1045. EX1045 discloses that "kernel programs tend not to be compatible from processor to processor." EX1045, 16:60-62. Similarly, EX1043 discloses that "computer programs written for" a processor "are frequently unsuitable for use with computers based on a different processor." EX1043, 1:17-20. In other words, EX1043 and EX1045 corroborate that because the conventional processors that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used for its servers **are** able to run the corresponding OS that would have been installed to the server (and its corresponding kernel and local

system files) POSAs understood that each processor would have been compatible with that kernel and set of associated local system files.

* * *

[0187] Thus, for the reasons I discussed above (§§VII.E.1.a.i-iv), Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1PREA**].

b. [1PREB]

[0188] For convenience, I have further divided limitation [**1PREB**] into four sub-elements. For the reasons discussed below (§§VII.E.1.b.i-iv), it is my opinion that Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1PREB**] under both parties' constructions.

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

[0189] The '814 patent's "example applications" include an "Apache" web-server application. EX1001, 2:55-3:19. As I discuss below, in Schmidt-Tormasov and Schmidt-Tormasov-Calder, each VCE-capsule *securely executes* web-server (and other similar) *applications*. For example, as I discussed above (§VII.B), Tormasov's FIG. 1 illustrates one virtual environment executing an "httpd" web-site server application and another virtual environment executing an "Ftp" file-transfer-protocol server application. Similarly, discusses as background information of example of *applications* to *execute* in Schmidt-479's capsules

include “computer programs” for applications like a “word processor” or “a web browser.” Schmidt-479, [0009]-[0010].

[0190] Schmidt-Tormasov and Schmidt-Tormasov-Calder each use Schmidt-479’s capsules (*see supra* §VII.D.1-2), which “comprise[] one or more processes and their associated system environment.” Schmidt-479, [0038]. Schmidt-479’s Figure 3 (which I have added color to below), illustrates two capsules, with one or more processes: for example, capsule 1 has processes 0 and 1 (blue) and capsule 2 has processes N-1 and N (green).

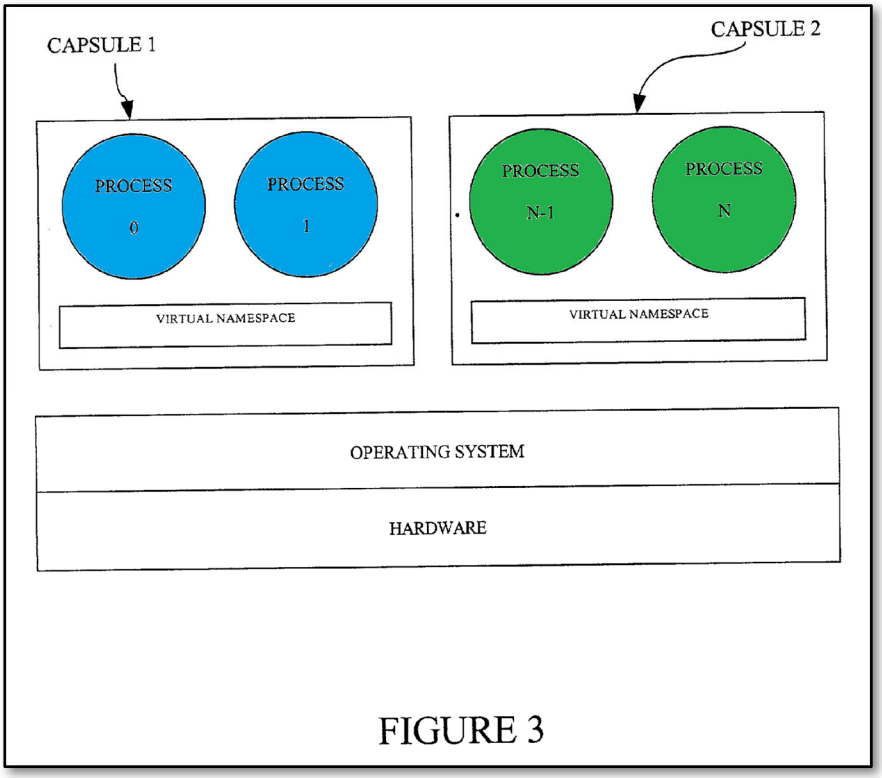


FIGURE 3

[0191] These “processes” that *execute* in the capsules are for *executable, applications*, like “a word processor, a web browser, and an interactive multimedia simulation” (i.e., various applications). Schmidt-479, [0010].

Schmidt-479 further discloses that the "use of compute capsules is completely transparent," such that when applications are executing inside capsules the "***applications*** need not take any special measures, such as source code modification, re-compilation, or linking with special libraries." Schmidt-479, [0040]. Schmidt-479 further discloses that that the file system views in each capsule "include, for instance: ***application* binaries**…," which are *executables* files for *executing* those *applications* inside a capsule. Schmidt-479, [0048]. As corroborated by Calder, a typical "***application*** package" includes "an ***application* binary** (also called ***application* program**)" (which is an *executable* application). Calder, [0076]. Thus, when Schmidt-479 disclsoes that each capsule includes "***application*** binaries" (Schmidt-479, [0048]), POSAs would understand that Schmidt-479 is teaching that each capsule includes *executable* files for starting an *applications* program (like the word-processor application discussed above).

[0192] Thus, Schmidt-479 repeatedly confirms that the capsules on servers comprise *executable applications*.

[0193] Schmidt-479 further discloses that each capsule's "default set of file system view mappings includes standard system directories, such as /bin **for *applications***…." Schmidt-479, [0051]. As Schmidt-479 states, /bin is a directory that "contains ***application* programs** on Unix systems" and is an example of the "many standard, machine-local directories that are the same across systems."

Schmidt-479, [0048]. That is, Schmidt-479 is disclosing (as POSAs knew) that conventional deployments of a prior art Unix-based OS had a directory in the file system called "/bin" where application executables are installed. Though conventional, prior art Unix-based systems will have a /bin directory in their file systems, the applications in any given /bin directory will typically vary (though there are some standard applications that are included with any given Unix distribution). Because Schmidt-479 permits capsules to be "move[d]" "to a different binary compatible machine, or" "suspend[ed] [] on disk for later revival on the same or a different machine" (Schmidt-479, [0006]), POSAs would have understood Schmidt-479 to be disclosing that each capsule includes copies of a /bin directory (and the applications installed therein) to ensure that the applications within the capsule could be moved to or started/restarted on another server machine in the system, which may not already have all of the applications in the capsule in the server's local /bin directory.

[0194] But even if Schmidt-479 were not disclosing that, POSAs would have had reason to include copies of each capsule's entire view of the /bin directory (and also the /lib and /etc directories, for example) when preparing a capsule for storage for moving / starting / restarting for the reason I discussed above: to ensure the capsule (and the applications therein) would execute properly on the servers in the system, even if the server did not already have the same

application(s) (and required libraries and/or configuration files) in its local file system. POSAs would have reasonably expected success because it would have required only POSAs' ordinary skill to include copies of the application executable and system files in each capsule (including any of the capsule user's modification thereof). To the extent not already being done by Schmdit-479, the cost of doing so would have been to increase the size of the capsule, which POSAs would have found to be a beneficial tradeoff as it would have better ensured compatibility and proper execution of each capsule on any server in a multi-server system, as Schmidt-479 desired. Indeed, as I discussed above, Tormasov teaches a similar computing environment that allows for installation of any application implemented in the environment.

[0195] Similar to Schmidt-479, Tormasov's Figure 1 (which I have added color to below) shows VCEs *providing* "***a method of*** efficient utilization of a single hardware system" ([0017]) for *executing applications*, e.g., httpd (web) server (blue) and ftp (file-transfer protocol) server (green). Tormasov similarly states that it is directed to solving problems associated executing "complex ***applications*** requiring a database" or "CGI" "scripts" "on a remote ***server***," which are *executable applications* commonly run on *servers*. Tormasov, [0012]. Tormasov reinforces that VCEs allow for the "installation of programs…and other ***applications***" implemented in each VCE. Tormasov, [0024]. A related reference

by the Tormasov inventors (EX1111, "Tormasov-633") similarly discloses executing VCEs within a "traditional architecture" for a "work organization of computers" that "include[s]…***application*** software." EX1111, 4:58-64.

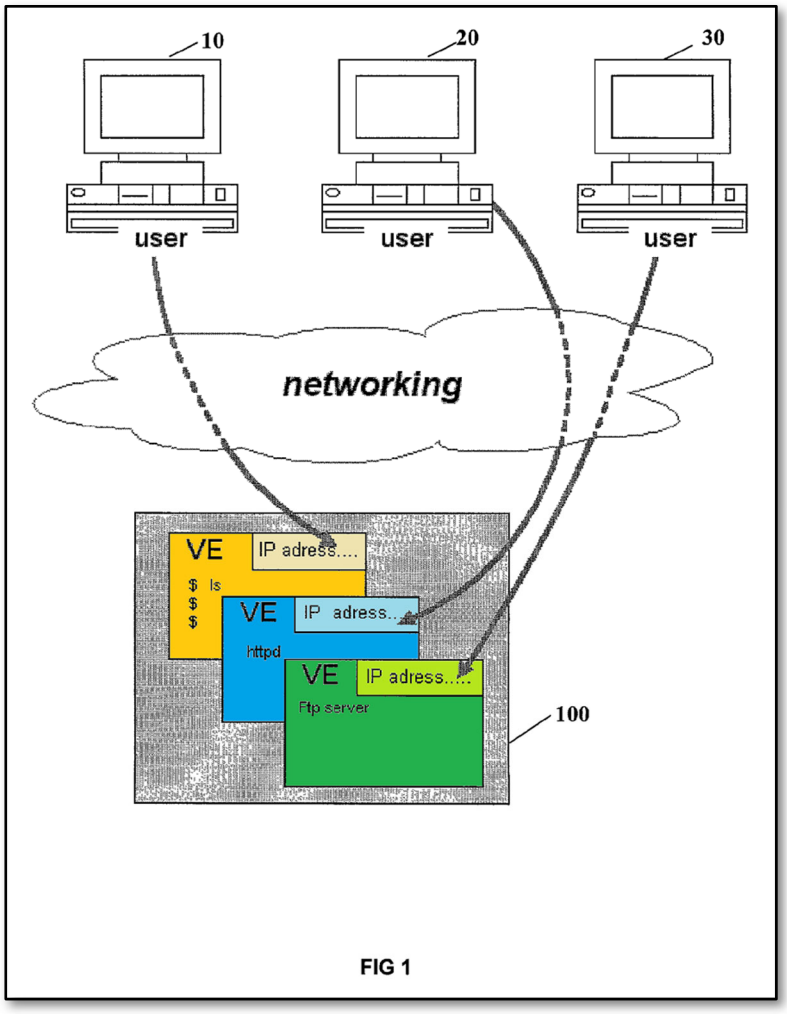


FIG 1

[0196] Thus, for the reasons I discussed above in this section and in §VII.E.1.a.i above, Schmidt-Tormasov and Schmidt-Tormasov-Calder would have placed *application executables* in VCE-capsules for installation of the VCE-capsules (and the application(s) therein) on servers. *See* Schmidt-479, [0051] (each capsule's "default set of file system view mappings includes standard system

directories, such as /bin **for *applications***….”); Tormasov, [0024] (VCEs permit for “installation of programs…and other ***applications***”). The application “processes in different” VCEs are *secure* (at least) because, as discussed below (§VII.E.1.b.iii), the VCE-capsules would have been *secure containers of application software*.

[0197] Thus, Schmidt-Tormasov’s and Schmidt-Tormasov-Calder’s methods would have *provided at least some of the servers in the system with secure, executable, applications*, as claimed.

ii. “[applications] related to a service”

[0198] As I discussed above (§VI.B.4), Google construes ***service*** to be “specialized, software-based functionality provided by network servers and comprised of one or more applications” (EX1106, 4); VirtaMove contends that “[n]o construction [is] necessary” and that the “plain and ordinary meaning” applies (EX1106, 4).

[0199] The ’814 patent discusses conventional *applications related to services*. For example, the ’814 patent says “applications such as Apache, MySql and PHP” can be placed “in a single container for supporting 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 application-based “***services***” like “CRM (Customer Relation Management) tools, Accounting, and Inventory.” EX1001, 7:18-21.

[0200] As discussed above (§VII.A), Schmidt-479 teaches encapsulating various *applications related to* various "***services***" (e.g., "X11/Unix ***services***…Windows NT ***service***…and others"). Schmidt-479, [0077]. As I discussed above (§VII.B) and as illustrated in Tormasov's Figure 1 (which I have added color to below), Tormasov's VCEs execute *application related to* similar *services* as the '814's examples, such as applications for an "httpd" server (an http *service*) or "Ftp server" (file-transfer *service*). Indeed, one expressly stated goal of Tormasov is to provide, e.g., a "hosting ***service***" that allows users to provide services via "websites." Tormasov, [0006]; *see also* Tormasov, [0024] (VCEs allow "installation of programs, configuration of network services, work with offices and other applications"); EX1122 (cited in Tormasov at [0035]), 119-120 ("FTP is…one of the most widely used ***services*** on the Internet….FTP service, ftpd").

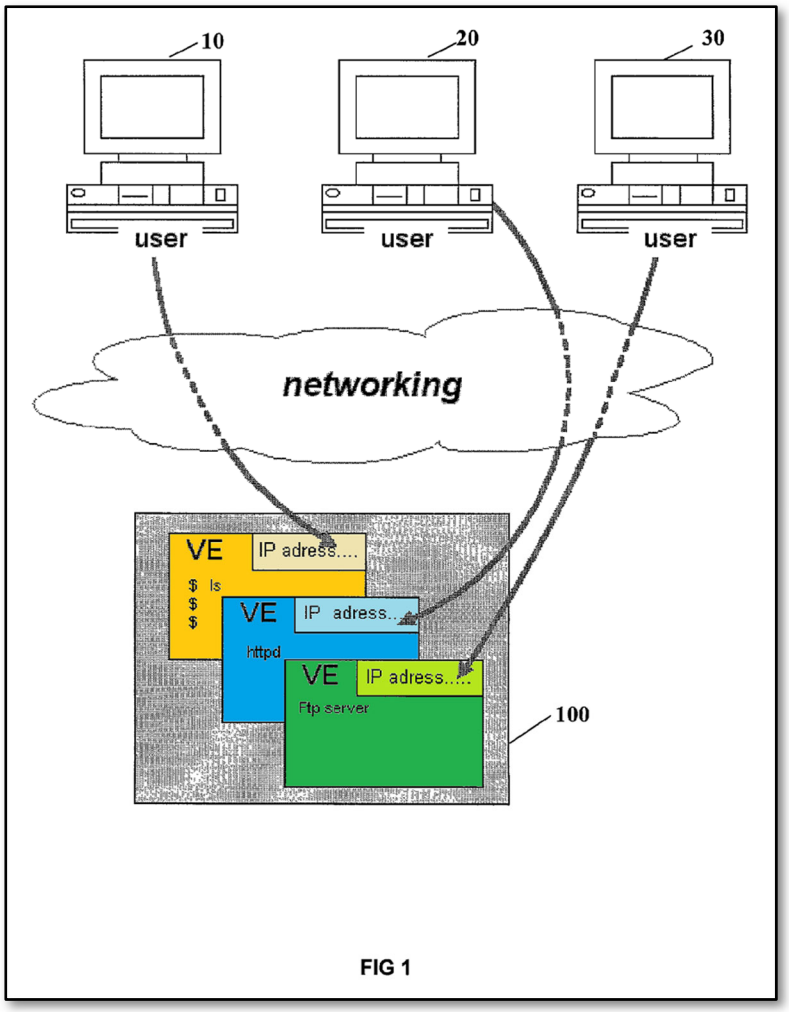


FIG 1

[0201] As illustrated in Tormasov's Figure 1 (highlighted above), Tormasov's server computer 100 is accessed via "networking" (i.e., it is a networked server). Schmidt-479 similarly teaches that the servers that store and execute capsules are networked servers. For example, Schmidt-479 discloses as background information (and as an example of a system in which capsules could be implemented) the prevalence of "computer **networks** [that] allow a user to access data and computer programs that exist elsewhere in the **network**"; e.g., client computers "access[ing]" computer applications provided by servers "via the

computer **network to the servers** where the programs were being executed." Schmidt-479, [0008]-[0010].

[0202] The above-discussed applications that would have been provided by Schmidt-Tormasov's Schmidt-Tormasov-Calder's networked servers are conventional *services* because they are specialized, software-based functionality that comprise of one or more applications (e.g., web-server application) (EX1088, 475), and thus meet Google's construction of "*service*" and VirtaMove's "plain and ordinary meaning" construction. EX1106, 4; *see also supra* §VI.B.4.

[0203] Thus, each VCE-capsule's applications in Schmidt-Tormasov and Schmidt-Tormasov-Calder *are related to a service* (e.g., website service).

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

[0204] The VCE-capsules in Schmidt-Tormasov and Schmidt-Tormasov-Calder each *execute* the above-discussed *applications in a* capsule, which is "a **private**, portable, persistent ***environment***." Schmidt-479, Abstract. Specifically, each VCE-capsule is 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]. Moreover, based on Tormasov, the virtual computing ***environment*** (VCE) executing in each

VCE-capsule is further *secure* (and is itself a *secure environment* for *executing* its *applications*) (at least) because "processes in different" VCEs (and their respective "unique file system") are "**completely isolated** from each other" (Tormasov, [0026]) and "are **never visible** to other" VCEs "running on the same computer" (Tormasov, [0018]).

[0205] Based on Calder, POSAs would have had additional motivation to secure Schmidt-Tormasov-Calder's ***environments*** based on Calder's teaching to "**protect** the contents of the [application's] project from improper tampering" and "**protect the non-secure machine** from improper tampering by the project." Calder, [0006]. For example, teaches to modify an "***application*** package" so it is "***executed*** safely, ***securely***, and transparently on a remote machine." Calder, [0073]. Calder discloses providing that protection by "intercepting" "system calls made by the application" to restrict "access [to] approved files" in the file system, thereby protecting the servers (and their file system) and "the contents of the application package" from corruption by each other. Calder, [0087]. Calder also teaches to use "***security*** information" in "application ***package***" that "protect[s] both the client computer [], as well as the contents of the application" from corruption from each other. Calder, [0100].

[0206] Thus, based on Calder, POSAs would have also had motivation to implement at least some VCE-capsules to have their system calls intercepted in

the manner Calder describes so as to further *secure* the *environments* that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules provided for *executing applications*, to protect the applications from improper tampering and protect the server on which the VCE-capsules were installed from corruption by the applications.

[0207] POASs would have reasonably expected success in doing so because, as I discussed above (§VII.C), Calder's techniques are based on software executable in the same conventional OSs that would have been used in Schmidt-Tormasov and Schmidt-Tormasov-Calder, and thus it would have required a POSA's ordinary programming skill to incorporate in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules software to intercept system calls, as taught by Calder, to protect the server and application package.

[0208] For either of the above-discussed reasons, VCE-capsules would have provided a *secure environment* for executing applications.

[0209] Thus, applications executing Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have been applications that *are executed in a secure environment*. *See also infra* §VII.E.1.c.i.(3) (where I demonstrate that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have been *secure containers of application software*).

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"

[0210] The '814 patent discloses an example where "**[a]n icon is an *object*** supported by a GUI [graphical user interface]" that "**[n]ormally an icon associates an image with an *object*** [e.g., a small graphic corresponding to an application] that can be operated on through a GUI." EX1001, 14:21-23. For example, the patent says that "the GUI supports the ability to **select an *object*, move it and initiate an operation**" – for example, "'copy[ing] and transfer[ring] of files" – "when the ***object*** is placed on a specific destination." EX1001, 14:4-8. Thus, examples of *objects* in the '814 patent are conventional icons associated with applications or with files that can be moved around a conventional GUI and *executed* on clicking the icon or by dragging-and-dropping the icon (e.g., moving the icon of an application or file to a GUI image of a directory can copy or move the corresponding file / application file(s) to that directory).

[0211] Additional examples of "***tasks***" in the '814 patent that use icon objects include "[i]nstall[ing] application specific files" and "[s]tarting a first application." EX1001, 15:35-56.

[0212] Thus, the '814 patent discloses conventional uses of application icons in prior art GUI-based OS (e.g., clicking an icon object to activate an application's executable object).

[0213] As I discussed above (§VII.D), Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used two or more conventional OSs from the prior art on respective servers in the Schmidt-Tormasov and Schmidt-Tormasov-Calder systems. Those conventional prior art OSs (e.g., Linux- and Windows-based OSs) used and/or beneficially supported installation of GUIs, where applications would each include an object (icons, application executables, and other application files) that would have been executable by the different OSs in the system.

[0214] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used included GUIs where applications were associated with one or more icon objects that were executable by the OS for performing a task related to a service (e.g., an icon for an accounting program related to performing a task related to an accounting service) regard is further corroborated by EX1047 discloses as background information:

> In graphical user interfaces employing **an *object*-oriented programming paradigm, *application* programs are typically represented to a user by an icon** …. ***Execution* of an *application* program is initiated by selecting its corresponding icon**, ….

EX1047, 1:21-67.

[0215] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used included GUIs where

applications were associated with one or more icon objects that were executable by the OS for performing a task related to a service (e.g., an icon for an accounting program related to performing a task related to an accounting service) regard is further corroborated by EX1059. EX1059 discloses as background information:

> Hardware changes force the development of separate software executables (e.g., **software *objects***) for the different hardware components. **A software *object* is a single software *executable*.** Conventionally, each hardware component type requires its own object or executable.

EX1059, 1:34-38.

[0216] My opinion that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used included GUIs where applications were associated with one or more icon objects that were executable by the OS for performing a task related to a service (e.g., an icon for an accounting program related to performing a task related to an accounting service) regard is further corroborated by EX1047. EX1047 discloses as background information:

> Execution of an application program is initiated by selecting its corresponding icon, most often using a pointing device such as a mouse. **When an application program is selected, a message is sent to the corresponding application program *object*, indicating that the application program object is to invoke certain of its methods.** For example, if a word processing program is selected, the methods

contained within **the application program *object* may include starting [i.e., *executing*] the word processing program**.

EX1047, 1:43-51.

[0217] The conventional prior art OSs which would have been used in Schmidt-Tormasov and Schmidt-Tormasov-Calder (e.g., Windows-based OSs), are GUI-based OS and associate executable applications with icons ("objects"), where the icons are conventionally included with the application. The icon objects were conventionally operated upon (e.g., clicked) by a user to cause the OS to execute the corresponding application associated with the icon.

[0218] My opinion that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's servers would have included and had reason to include *applications* that *each include an object executable by at least of the different OS for performing a task related to the service* is further corroborated by EX1047. EX1047 discloses as background information that an OS (GUI's "**[e]xecution of an application program is initiated by selecting its corresponding icon**." EX1047, 1:39-45. EX1059 discloses as background information that another common "**software *object* is a single software *executable***." EX1059, 1:34-39.

[0219] Thus, an icon would have conventionally been included with and/or associated with Tormasov's web-server application (e.g., an icon mapped to an executable file for starting the web-server application). That exemplary icon

and the executable file the icon is associated with for starting the web-server application are each *an object executable* for *performing a task related to the* web-server *service*. Tormasov, FIG. 1, [0006]-[0007].

[0220] Additionally, based on Tormasov, each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule 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].

[0221] Furthermore, based on Schmidt-479, Schmidt-Tormasov and Schmidt-Tormasov-Calder's active VCE-capsules "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**").

[0222] Schmidt-479 further discloses that "an ***object*** in one capsule" is isolated and protected from *objects* in another capsule. Schmidt-479, [0041]. As I discussed above, conventional *objects* would have been *an object executable by at*

least some of the different operating systems for performing a task related to a service provided by an *application* associated with the *object* (e.g., an application's icon object for installing, configuring, and/or starting execution of the application).

[0223] Furthermore, POSAs knew that when using a conventional OS "[a] *program* is a static ***object*** that can exist in a file" and "[a] *process* is a dynamic ***object*** that is a program in ***execution***." EX1121, 43 (italics of "program" and "process" original); *see also* Tormasov, [0034] (citing EX1121 as part of Tormasov's "BIBLIOGRAPHY"). As I discussed above, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) each include one or more application programs for executing one or more services (e.g., web-server programs for executing web server-related services). Thus, the applications in each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule are each associated with program, which is *each an object executable by at least some of the different operating systems for performing a task related to the service* associated with that application program.

[0224] Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1PREB**].

[0225] Based on Calder, Schmidt-Tormasov-Calder's VCE-capsule comprise additional system files (libraries and configuration files, Calder, [0082])

that allow the above-discussed *object(s)* "to execute and communicate on **any non-native [or native] platforms**" (i.e., different OS-families) (Calder, [0101]).

[0226] That is an additional reason Schmidt-Tormasov-Calder meets [**1PREB**].

[0227] As I discussed above (§VI.B.5), Google construes *at least some of the different operating systems* as "at least two or more of the different operating systems" (EX1106, 4; EX1102, 11); VirtaMove contends that "[n]o construction is necessary" and that the "plain and ordinary meaning" applies (EX1106, 4; EX1103, 12).

[0228] As I discussed above (§VII.E.1.a.i), Schmidt-Tormasov and Schmidt-Tormasov-Calder would have had a plurality of servers having OSs that differ (i.e., at least two or more different OS), thus meeting Google's construction of *at least some of the different operating systems*. As I discussed above, Schmidt-Tormasov and Schmidt-Tormasov-Calder would have also been consistent with the patent's example disclosure of *applications each including an object executable by at least some of the different operating systems for performing a task related to the service*, and thus meets VirtaMove's construction to the extent that construction includes (at least) the patent's example embodiments.

[0229] Thus, in Schmidt-Tormasov and Schmidt-Tormasov-Calder *the applications each include an object executable by at least some of the different operating systems for performing a task related to the service*, as claimed.

* * *

[0230] Thus, for the reasons I discussed above (§§VII.E.1.b.i-iv), Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1PREB**], and thus meets claim 1's preamble.

c. [1A]

[0231] For convenience, I have divided limitation [**1A**] into four sub-elements. For the reasons discussed below (§§VII.E.c.i-iv), it is my opinion that Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1A**] under both parties' constructions.

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

[0232] For convenience, I have further divided limitation [**1A**] into three sub-elements. For the reasons discussed below (§§VII.E.1.c.i.(1)-(3)), it is my opinion that Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1A**] under both parties' constructions.

(1) Schmidt-Tormasov/Schmidt-Tormasov-Calder VCEs-capsules (*containers*) are "stor[ed] in memory accessible to…the servers"

[0233] Below (§VII.E.1.c.i.(2)), I demonstrate that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCEs-capsules meet the claimed *containers*. In this section, I demonstrate that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCEs-capsules would have been *stored in memory accessible to the* Schmidt-Tormasov and Schmidt-Tormasov-Calder servers.

[0234] Tormasov *stores* "[a]ll the processes of all" VCEs running on a server "inside the same physical ***memory***." Tormasov, [0026]. Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules thus would have also been *stored in memory accessible to* each *server* executing the VCE-capsule, which meets VirtaMove's claim construction, which only requires the claimed *memory* to be *accessible* to one server. *See* EX1103, 12; *infra* §VII.E.1.c.i.(2) (where I demonstrate that VCE-capsules are *containers of application software*).

[0235] In my opinion, POSAs would have also had reason to *store* Schmidt-Tormasov's VCE-capsules in the *memory accessible to the* system's plurality of *servers* (e.g., accessible to at least two of the server), to beneficially provide a centralized storage location from which to store, manage, and distribute VCE-capsules to *at least two* servers in the network. That obvious implementation meets Google's construction of *memory accessible to at least some of the servers*.

EX1106, 4; EX1100, 4; EX1101, 2. *See infra* §§VIII.E.1.c.i.(3) (where I demonstrate that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have been *secure containers of application software*).

[0236] For example, Calder teaches *storing* application packages *in* the *memory* of a server (a server system) that "act[s] as the master control center" that would have been *accessible to* the network's "***server[s]***.'' Calder, [0078]-[0079]. POSAs would have understood that any suitable application package (like a Schmidt-Tormasov VCE-capsule, whether or not modified by Calder's techniques) could have been stored in a similar master control center, for the benefits Calder identified: "for all of the data processing, data transmissions, security information, and results processing" as application packages are transferred to server devices in the system. Calder, [0078].

[0237] Thus, POSAs would have had motivation and a reasonable expectation of success to *store* Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (and the applications therein) on the *memory* of a centralized-storage system (something like Calder's master control center), where that memory would have been *accessible to the servers* in the network, as a centralized location from which to read/write, manage, and use VCE-capsules. Those obvious implementations of Schmidt-Tormasov and Schmidt-Tormasov-Calder meet Google's construction of "*memory accessible to at least some of the*

servers" (EX1106, 4; *supra* §VI.B.6) and is another way it would have been obvious for Schmidt-Tormasov and Schmidt-Tormasov-Calder to meet VirtaMove's "plain and ordinary meaning" construction (EX1106, 4, *supra* §VI.B.6). *See also infra* §VII.E.1.c.i.(3) (where I demonstrate that VCE-capsules would have been "*secure containers of application software*").

(2) Schmidt-Tormasov/Schmidt-Tormasov-Calder VCEs-capsules are "containers of application software"

[0238] The '814 patent (at 2:23-42) provides a definition of "container" that is discussed below with labels [**1**]-[**6**] added. However, as I discussed above (§VI.A.2), the parties have agreed to a joint construction of the claimed *container* that is the combination of part [**1**] and the first sentence of part [**3**] that I discuss below. *See* EX1106, 6.[12] For completeness, I demonstrate how Schmidt-Tormasov and Schmidt-Tormasov-Calder meet the patent's full definition of "*container*" (i.e., parts [**1**]-[**6**]) in light of both parties' constructions, and thus also meet the parties' joint construction of *container* (i.e., part [**1**] and the first sentence of part [**3**] below).

[12] I note that the parties have each offered a competing construction of the related term "*secure containers of application software*," which I address below in §VII.E.1.c.i.(3)).

***[1]**: "**An aggregate of files required to successfully execute a set of software applications on a computing platform is referred to as a container**." EX1001, 2:23-25; see also EX1106, 6.*

[0239] As I discussed above (§VII.E.1.b.i), Schmidt-479 teaches *sets of software applications* in a capsule for *executing on a computing platform* and Tormasov teaches similar *sets of software applications* in a VCE for *executing on a computing platform*. Calder similarly teaches *sets of software applications* in a application packages for *executing on a computing platform*. POSAs would have had motivation to include any of the *software applications* discussed in Schmidt-479, Tormasov, or Calder in Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules and would have had a reasonable expectation of success in doing so because those applications were typical, conventional, and beneficial applications for executing on server *computing platforms* like the networked-servers that would have been used in the Schmidt-Tormasov and Schmidt-Tormasov-Calder systems that I discussed above (§§VII.D.1-2).

[0240] 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 data needed to store and start capsules (and applications therein) includes "[s]ystem environment information," which includes "privileges, **configuration settings**, working

directories and files, assigned resources, open devices, **installed software**, and internal program state." Schmidt-479, [0038]. Each capsule further comprises "file system views," which are "system directories and **files <u>necessary</u> [i.e., *required*] for normal user operation**…. Such files include, 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 both the Schmidt-Tormasov system and Schmidt-Tormasov-Calder system. (For example, *files required to successfully execute* the conventional *software applications* that Schmidt-479 identifies, like word-processor applications and applications for network services, hat I discussed above §VII.A, §VII.E.1.b.ii)

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

[0242] POSAs knew that the processes in each VCE-capsule are processes of an *executing software applications* because POSAs knew that "[w]hen

a user requests a program [e.g., a *software application*] to be ***executed***, a new ***process*** is created to encompass its execution." EX1064, 2:12-13.

[0243] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet part [**1**] of the patent's "container" definition, which is included in the parties' joint construction of *container*. *See* EX1104, 3; EX1103, 10; EX1105, 6.

[0244] Calder's teachings further confirm that Schmidt-Tormasov-Calder's VCE-capsules would have included "***application*** binar[ies]" (*executable* files for that *application*), "libraries," and "configuration files" that would have been *required to successfully execute a set of software applications on any computing platform* (e.g., server) in the system, including those servers running OSs that would have been non-native for the VCE-capsule (and the applications therein). Calder, [0082], [0102]. (It would have been a matter of design choice for a POSA whether **every** or some subset of the VCE-capsule in Schmidt-Tormasov-Calder would have been *executable* on **every** OSs in the system.)

[0245] That is an additional reason why Schmidt-Tormasov-Calder's VCE-capsules meet part [**1**] of the '814 patent's "container" definition, which is included in the parties' joint construction of *container*. *See* EX1104, 3; EX1103, 10; EX1105, 6.

> ***[2]****: "A container is not a physical container but a grouping of associated files, which may be stored in a plurality of different locations that is to be accessible to, and for execution on, one or more servers." EX1001, 2:25-29.*

[0246] As I discussed above for "container"-definition part [**1**], each of Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would *not* have been *a physical container but* would have been *a grouping of associated files* (e.g., application binaries and system files, i.e., software) *that* would have been *for execution on, one or more servers* in the system.

[0247] As I also discussed above (§VII.E.1.c.i.(1)), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would have been *stored in a plurality of different locations* (e.g., network-storage devices) that would have been accessible to a control server *that* would have been *accessible to one or more servers*. *See also* Calder, [0075], [0078], [0081] (teaching that application packages – *grouping of associated files* – "may reside within the memory of a server" or "a dedicated preprocessing computer" from which packages can be retrieved for *execution on servers*).

[0248] In my opinion, POSAs would have also had reason to *store executing* VCE-capsule *in a plurality of different locations accessible to one or*

more servers, such as a conventional networked-storage device having multiple storage locations.

[0249] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet **part** [**2**] of the '814 patent's "container" definition

> ***[3]**: "**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**. The term 'within a container', used within this specification, is [sic] to mean 'associated with a container.'" EX1001, 2:29-34; see also EX1106, 6.*

[0250] Based on Schmidt-479, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-"[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].

[0251] Based on Tormasov, each VCE in a VCE-capsule would have been *for use on a* Schmidt-Tormasov and Schmidt-Tormasov-Calder *server* and would have "include[d] a complete set of processes and files of an operating system that can be modified by the end user" (i.e., read/write files) (Tormasov, [0025]) and "its own **unique** [i.e., *mutually exclusive*] file system," which includes

its read/write files (Tormasov, [0026]). Tormasov further discloses that "two processes in different" VCEs are "**completely isolated** from each other" (Tormasov, [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*) (Tormasov, [0030]).

[0252] Thus, *each* VCE-capsule would have been *mutually exclusive of the other* VCE-capsules on that server, so the *read/write files* associated with each VCE-capsule[13] – e.g., each VCE's "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.[14]

[0253] Schmidt-Tormasov and Schmidt-Tormasov-Calder would have further prevented each VCE-capsule's read/write file *from being shared with other*

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

[14] For the same reason, data within each VCE-capsule's application set would have been insulated from effects of other application sets, as required by the parties' constructions of a *secure containers of application software*. *See infra* §VII.E.1.c.i.(3).

VCE-capsules (*containers*) by providing each VCE-capsule its own, unique root directory, as I discuss below (§VII.E.1.c.i, [**1G**]).

[0254] POSAs would have also had motivation to make the read/write files in each VCE-capsuled *mutually exclusive* to allow for preventing the sharing of read/write files to protect those read/write files within a VCE-capsule from corruption/interference by other VCE-capsules. *See* Calder, [0087] ("**protecting the contents** of the application package 115 **from user access**"), [0100]; EX1072, [0049]-[0052].

[0255] A POSA's motivation (and reasonable expectation of success in doing so) would have been reinforced by their background knowledge that techniques for preventing read/write files in one logical area of a computer system storage from modifying read/write files in another logical area of computer system storage were well-known and conventional. An example of which is the well-known chroot (change-root) functionality in Unix-based OSs (and similar functionality for other OS families), which existed precisely to provide the benefit of protect one user of a computer system (or one application / set of applications) from potentially corrupting the system files for another user / application. Because these features were well-known in the prior art, POSAs would have had a reasonable expectation of success in incorporating into Schmidt-Tormasov and Schmidt-Tormasov-Calder using only ordinary skill because Schmidt-Tormasov

and Schmidt-Tormasov-Calder would have used conventional OSs that were already capable of supporting that functionality.

[0256] My opinion that POSAs would have wanted to provide different VCE-capsules different views of a file system and that functionality like chroot were well-known in the prior art and used to prevent read/write files in one directory / storage area from being access from application / users assigned to a different directory / storage area is corroborated by EX1027. EX1027 discloses:

> As used herein, "chroot" refers to a command for changing the active root directory. That is, chroot is a command that executes to change a user's active root directory. For example, in a typical Unix system, the root directory is indicated as "/". Thus, the command "cd/"[15] will normally execute to change to the root directory of the filesystem. The chroot command may be executed to change a user's active root directory. For instance, the command "chroot/home" **changes a user's active root directory to "/home"**. Thereafter, executing the command "cd/" results in changing to the "/home" directory rather than "/", as "/home" is now the user's active root directory. As a further example, the chroot command may be utilized to specify a particular compartment as the active root directory. For instance, the command

[15] I note that, conventionally, there should be a space between "cd" and "/" in the UNIX system, but POSAs would have no problem understanding what EX1027 is disclosing.

> "chroot/compt/FOO" changes the active root directory to compartment FOO. Accordingly, as the "cd/" executes to change to the active root directory (which may be changed by the chroot command)**, the chroot command may be utilized as a further security mechanism. For instance, the chroot command may be utilized to effectively prevent a user (or a process) from escaping the active root directory specified by the chroot command.** That is, for example, **a user (or process) may not escape the active root directory specified by the chroot command** (e.g., compartment FOO) to the actual root directory ("/") of the system. Once the active directory of a user (or process) has been changed via the chroot command, it is said to have been chroot-ed. Accordingly, via kernel controls, **the ability of a process to transition to root from within a compartment is removed** in this exemplary implementation so that **the chroot cannot be escaped**. **This exemplary implementation provides the ability to make at least selected files within chroot immutable.**

EX1027, [0042].

[0257] My opinion that POSAs would have wanted to provide different VCE-capsules different views of a file system and that functionality like chroot were well-known in the prior art and used to prevent read/write files in one directory / storage area from being access from application / users assigned to a different directory / storage area is corroborated by EX1035. EX1035 discloses:

> Many file systems present a global view of data accessible to all processes running on a machine with access to those file systems.

There are situations, however, where it is advantageous to provide different processes with different views of a file system. Machines with multiple network interfaces, machines with multiple disparate network views, and machines running multi-level trusted systems where different users get access to different parts of the system, are all such situations.

Conventional systems may implement differing file system views, for example, by utilizing file and folder permissions or access control lists. For example, in a UNIX operating system**, the chroot ( ) system call limits a file system view to a certain subset of files. Other systems enable a user to move parts of a globally valid directory (e.g., folder) tree for a private view. Yet other systems enable a user to create a new directory tree with a private view of processes by mounting file systems over a network.**

EX1035, [0002]-[0003].

[0258] My opinion that POSAs would have wanted to provide different VCE-capsules different views of a file system and that functionality like chroot were well-known in the prior art and used to prevent read/write files in one directory / storage area from being access from application / users assigned to a different directory / storage area is corroborated by EX1072. EX1072 discloses:

Chroot Environment

In one embodiment, the local machine is running the **Unix™ operating system. A user interacts with the system through an interface**

running from a chroot (or change root) directory. All actions initiated by the interface also run from the chroot directory. The chroot directory (or environment) is an alternate root directory that appears as the root directory for a given process or set of processes. Thus, a malicious user is unable to tamper with the local machine's system files in the main directory because the user is confined to the directory tree below the chroot directory (i.e., the user only has access to files stored in the directory tree below the chroot directory).

FIG. 6 illustrates the process of user interaction with a computer system through a local machine running a Unix operating system in accordance with one embodiment of the present invention. At block 600, a user requests access to the computer system at the local machine. At block 610, the chroot environment (or directory) is made available to the user. **The chroot environment is a restricted run-time environment; processes initiated in the chroot environment and all processes spawned thereby do not have access to anything outside the chroot environment**.

At block 620, a user context is dynamically created at the local machine for the user. At block 630, a user interface is initiated in the chroot environment. **Since the user interface is initiated in the chroot environment, the user will not be able to tamper with the local machine's system files, which are maintained outside the chroot environment**. However, the system files contained in the chroot environment are used by the user and remain at risk for user tampering.

> In one embodiment, all users of a local machine share a single chroot environment. **Protection within the chroot environment is provided (only) by the standard Unix user semantics. Use of the chroot environment provides greater protection for the system resources outside of the chroot environment**.

EX1072, [0049]-[0053].

[0259] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet part [**3**] of the '814 patent's "container" definition, and therefore meet part [**2**] of the parties' joint construction of the claimed *container*. *See also supra* (where I demonstrate that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet part [**1**] of the '814 patent's definition of "container" and the parties' construction of the claimed *container*).

> ***[4]****: "A container comprises one or more application programs including one or more processes, and associated system files for use in executing the one or more processes; but containers do not comprise a kernel[.]" EX1001, 2:34-37.*

[0260] Schmidt-479's capsules comprise, among other things, "one or more active ***processes***," which Schmidt-479 can suspend and store for moving to or starting / restarting on any server in the system. Schmidt-479, [0016]. Based on Schdmit-479, each of Schmidt-Tormasov and Schmidt-Tormasov-Calder's VCE-capsule would have also comprised "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" inside each capsule "may include, for instance, privileges, ***configuration*** **settings**, working directories and ***files***, assigned resources, open devices, **installed software** [i.e. ***applications***], and internal program state." Schmidt-479, [0038].

[0261] Schmidt-479 discloses that each capsule's "default set of file system view mappings **includes standard system directories**, such as **/bin for applications**, **/lib for *system* libraries**, /usr/man for manual pages, etc." Schmidt-479, [0051]. As I discussed above (§V.A, §VII.E.1.a.iv), **libraries** in a /lib directory are examples of *system files* identified by the '814 patent. EX1001, 2:52-3:19. As Schmidt-479 states, Unix, for example, includes "many standard, machine-local directories that are the same across systems." Schmidt-479, [0048]. That is, conventional deployments of a Unix-based OS (then and now) have a directory in the file system called "/lib" where software libraries are installed. (Another conventional Unix directory is /etc, for configuration files, which are also *system files*. *See* EX1001, 2:52-3:19.)

[0262] Though conventional Unix-based OSs will have a /lib directory in their file systems, the libraries and/or versions of a given library in any given /lib directory may vary (though there are some standard libraries that were included with each Unix distribution). Because Schmidt-479 permits capsules to

be "move[d]" "to a different binary compatible machine, or" "suspend[ed] [] on disk for later revival on the same or a different machine" (Schmidt-479, [0006]), POSAs would have understood Schmidt-479 to be disclosing that each stored capsule includes copies of the /lib directory mapped to the capsule to ensure that the libraries (and the applications that use those libraries) that are within the capsule were available when the capsule was moved to or started/restarted on another server machine in the system, whose local /lib directory may not already have the libraries (or versions therefore) that the applications in the capsule requires to execute properly.

[0263] But even if Schmidt-479 were not disclosing that, POSAs would have had reason to include copies of each capsule's entire view of the /lib directory (and also the /etc configuration-file directory, for example) when preparing a capsule for storage for moving / starting / restarting for the reason I discussed above: to ensure the capsule (and the applications therein) would execute properly on the servers in the system, even if the server did not already have in its local file system the libraries and/or configuration files needed to properly execute the capsule (and the applications therein). POSAs would have reasonably expected success because it would have required only POSAs' ordinary skill to include copies of the system files in each capsule (including any of the capsule user's modification thereof). To the extent not already being done by

Schmdit-479, the cost of doing so would have been to increase the size of the capsule, which POSAs would have found to be a beneficial tradeoff as it would have better ensured compatibility and proper execution of each capsule on any server in a multi-server system, as Schmidt-479 desired. Indeed, as I discussed above, Tormasov teaches a similar computing environment that allows for installation of any application inside the environment, along with copies of the necessary *system files*.

[0264] Thus, the files inside each stored capsule includes (or POSAs would have had reason for them to include) *system files*, like libraries in the Unix directory /lib and configuration files in the Unix directory /etc (and corresponding libraries and configuration files for other conventional OSs). *See also* Schmidt-479, [0049]-[0052]. Those system files are / would have been *for executing the one or more processes* in the capsule associated with the *one or more application programs comprised* within the capsule.

[0265] Thus, for the reasons I discussed above in this section and in §VII.E.1.b.ii above, Schmidt-Tormasov and Schmidt-Tormasov-Calder would have placed *application programs including one or more processes, and associated system files for use in executing the one or more processes* on servers in the system.

[0266] As shown in Schmidt-479's Figure 3 (which I have copied below), the capsules are for use with the server's OS, but are stored outside that OS.

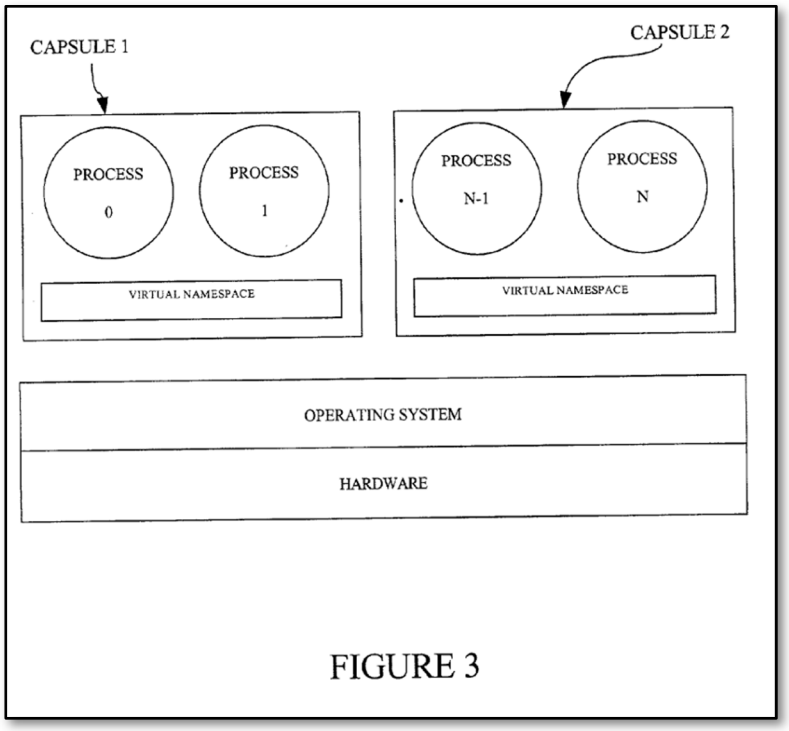


FIGURE 3

[0267] Schmidt-479 recognizes that its OSs comprises a kernel (as would have been true of a conventional OS). *See, e.g.*, Schmidt-479, [0011]-[0012], [0054], [0059]-[0060]. POSAs thus understood that each capsule *does not comprise an* OS *kernel* (or would *not* have needed to *comprise a kernel*) because the capsules share the server's Operating System (and thus also its kernel). *See also* Schmidt-529, [0074] (disclosing an embodiment of Schmidt capsules where "a thin layer of software is **interposed between the application and *kernel***").

[0268] Like Schmidt-479's capsules, Tormasov's Figure 1 (which I have annotated below) shows that each Tormasov VCE *comprises one or more application programs*, like an http web-server application (blue) or Ftp (file-transfer-protocol) server application (green).

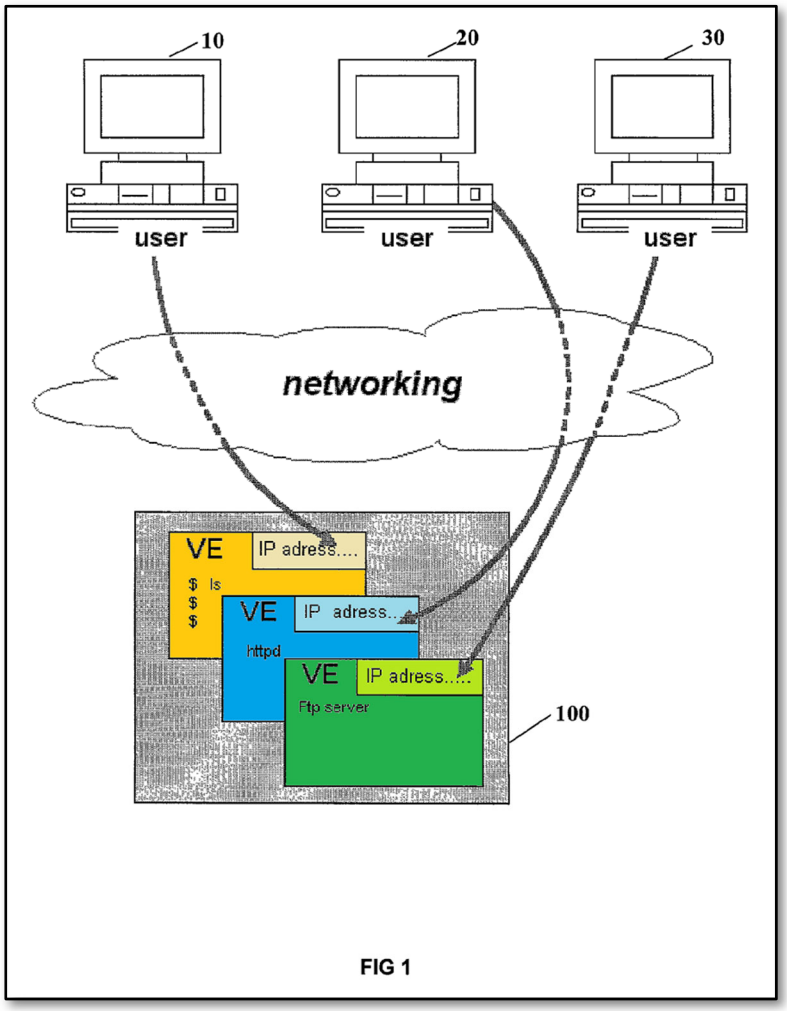


FIG 1

[0269] As illustrated in Tormasov's Figure 2 (which I have annotated below), VCEs include (1) *application programs including one or more processes* (blue and green) and (2) local file systems (light blue and green) *for use in*

executing the one or more processes, but *do not comprise a kernel* (red), which remains outside VCEs. Tormasov, [0025].

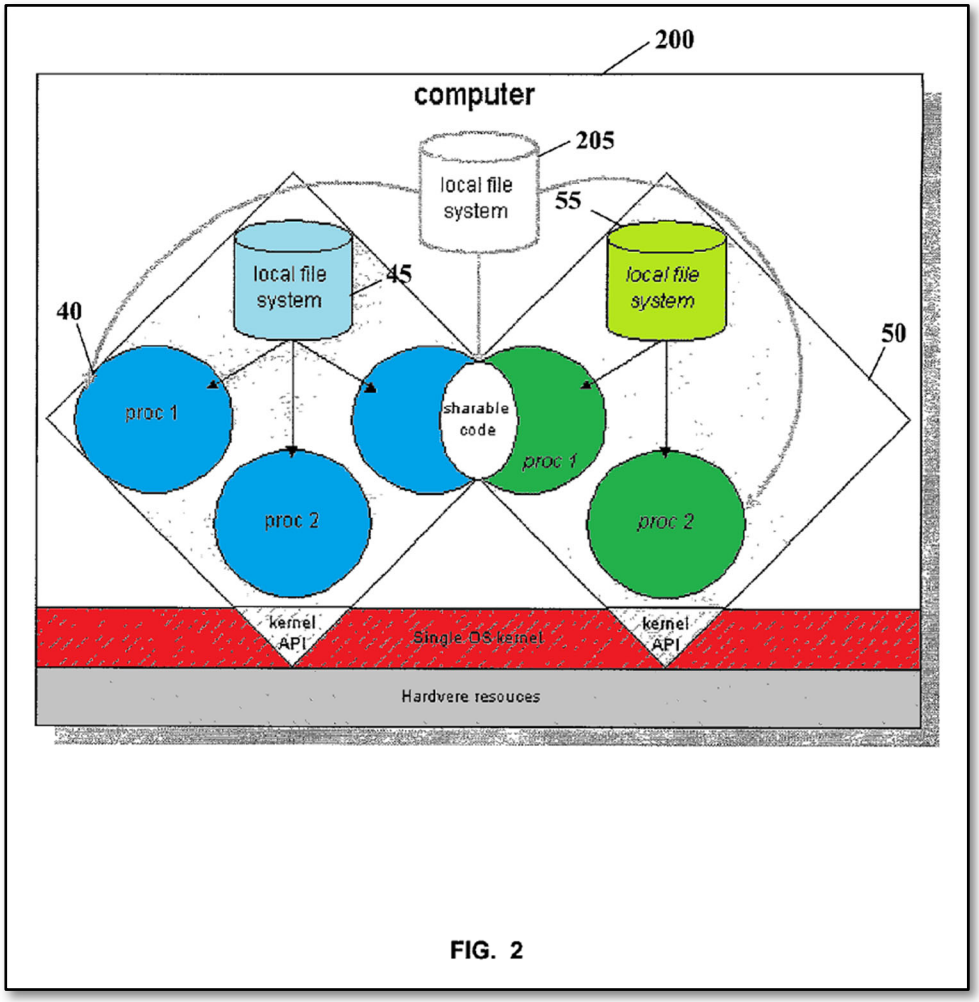


FIG. 2

[0270] As I discussed above (§§VII.D.1-2), in Schmidt-Tormasov and Schmidt-Tormasov-Calder, the local file system in each VCE comprises (or POSAs would have had reason for them to comprise) *associated system files for use in executing the one or more processes* (e.g., shared libraries and configuration files), of the *application programs* within VCE-capsules (*containers*) that were executable on an multiple servers in the system, including servers with different

versions of OSs in the same OS-family (e.g., Windows 95 and Windows 98). *See, e.g.*, EX1122 (cited Tormasov, [0035]), 122-23 (to execute properly, "fttpd," one of Tormasov's example applications inside VCEs, "needs" copies of certain library files that exist in the directory "lib" and configuration files that exist in the directory "etc").

[0271] As I discussed above (§VII.D.2), based on Calder, Schmidt-Tormasov-Calder would have further *included one or more processes, and associated system files for use in executing the one or more processes* on non-native OSs (e.g., OSs in **different** OS-families). Calder, [0082]-[0083], [0101]

[0272] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet part [**4**] of the '814 patent's "container"-definition.

[0273] Similarly, based on Calder's teachings, each VCE-capsule would have comprised *application programs* and *associated system files* [e.g., libraries and configuration files] *for use in executing* the applications on native **and** non-native OSs, as I discussed above (§VII.D.2). *See* Calder, [0082]-[0083], [0101].

[0274] Calder similarly discloses that when applications execute there are "**requests for *process* creation** and termination." Calder, [0122]-[0131]. *See also* EX1001, 16:11-15 ("programs/processes").

[0275] POSAs also knew as part of their background knowledge that "[t]ypically,…a computer application is instantiated…in the form of one or more application ***processes***…." EX1061, 2:28-33. Similarly, EX1064 corroborates that POSAs knew that "[w]hen a user requests a program to be executed, a new ***process*** is created to encompass its execution." EX1064, 2:12-13.

[0276] Furthermore, in Schmidt-Tormasov-Calder, the VCE-capsule's system files are modified so the application(s) in the VCE-capsule could have been "execute[d] and communicate[d] on any non-native platforms." Calder, [0101]. For example, "system libraries [would have been] added…that translate Windows 2000 system calls to Linux system calls." Calder, [0101]. "[W]hen a program runs under the Windows operating system," for example, "it accesses **the** operating system via the Windows API," with "[o]nly the WIN32 API calls [] allowed to access **the** operating system." Calder, [0096]. POSAs knew that the WIN32 API is how a 32-bit Windows's kernel is accessed. *See* EX1048, 6:7-8 ("**Win32 kernel**")). Thus, rather than *comprise a kernel*, Schmidt-Tormasov-Calder VCE-capsules would not have (and would not require) a kernel because, based on Calder's teachings, the applications that would been in each VCE-capsule would have been executable on different OSs by translating (modifying) system calls to make them understandable by the server's base OS's kernel (if the OS+kernel were not native for the VCE-capsule's applications).

[0277] Thus, an additional reason Schmidt-Tormasov-Calder's VCE-capsules would have met part [**4**] of the 814 patent's "container" definition would have been based on Calder's disclosure that I discussed above.

> *[5]: "[E]ach container has its own execution file associated therewith for starting one or more applications." EX1001, 2:38-39.*

[0278] Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet part [**5**] of the '814 patent's "container" definition for the same reason they meet claim 2, as I discuss below (§VII.E.2).

> *[6]: "In operation, each container utilizes a kernel resident on the server that is part of the operating system (OS) the container is running under to execute its applications." EX1001, 2:39-42.*

[0279] As I discussed above (§§VII.D.1-2, §VII.E.1.a.i), the conventional prior art OSs that would have been resident on Schmidt-Tormasov's and Schmidt-Tormasov-Calder's servers (e.g., Windows-based OSs, Unix-based OSs, etc.) would have had a *kernel* that would have been *utilized* by the VCE-capsule's (*containers*) and the applications installed therein. The *kernel* of the OS on each server would have been *utilized* to, for example, respond to system calls from applications in VCE-capsule to request use of the server's resources and services (e.g., requests to open files, use the server's processor hardware, etc.). *See*

Calder, [0150] (applications can "execute an interrupt call on the operating system kernel of a client computer").

[0280] My opinion that conventional system calls that would have been used by Schmidt-Tormasov's and Schmidt-Tormasov-Calder's applications to perform functionality that would have been part of the OS kernel is corroborated by EX1097, which discloses:

> System Call—**Application programs interact with operating systems by System Calls**. A System Call occurs when **an application program invokes a function that is implemented by its operating system, for example, open or read a file or get the current time**.

EX1097, [0051]. The above-discussed functionality are examples of functionality provided by the OS's kernel. *See* EX1064, 1:53-62 (noting that in a "traditional operating system" the "kernel" "provides the user programs with a set of abstract system services called system calls"). Thus, EX1097 corroborates my opinion that applications within VCE-capsules would have made conventional system calls from their respective VCE-capsules, which would have included system calls to *utilize a kernel resident on the server that* would have been *part of the operating system (OS) the* VCE-capsule (*container*) would have been *running under to execute its applications* on the servers in the system.

[0281] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet part [**6**] of the '814 patent's "container" definition.

(3) Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules are "secure containers of application software"

[0282] Google's construction of "*secure containers of application software*" (EX1106, 5) is consistent with the '814 patent's definition of "*secure* application *container*": "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." *See also* EX1001, 2:43-48; 7:38-40. VirtaMove's construction is similar to Google's construction but replaces "an environment" with "containers." EX1106, 5. The '814 patent further says that an "application set" is "a set of one or more software applications that support a specific service." EX1001, 7:39-41.

[0283] Thus, a *container* is a *secure container of application software* under the parties' construction if it either (or both) (i) is an environment/container where each application set appears to have individual control of some critical system resources or (ii) data within each application set is insulated from effects of other application sets.

[0284] As I discussed above (§§VII.D.1-2), Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used Schmidt-479's capsule technique to encapsulate a Tormasov VCE (and applications installed thereto). *Supra* §§V.D.1-2.

[0285] As I discuss below in this section, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet (i) and (ii) above. Below I start by demonstrating how Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet prong (i) of each party's construction of *secure containers of application software* before turning to prong (ii). *See* EX1106, 5.

[0286] **"each application set appears to have control of some critical system resource":** 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].

[0287] For example, Schmidt-479 says it "is desirable to retain some of the benefits of machine-centric computing." Schmidt-479, [0013]. "In particular, machine-centric computing allows a user to **customize their computing environment**, for instance by arranging and naming their file system." A POSA would understand that disclosure to be teaching (or at a minimum motivating) a capsule *appears to have control of* the file system (a *critical system resource*) because the user is free to arrange and name the file system as desired.

[0288] Relatedly, a potential use of capsules is where "a capsule might be created for an **untrusted guest** by populating the namespace with reference to a **limited set of *resources***." Schdmit-479, [0020]. A POSA would understand that

disclosure to be teaching (or at a minimum motivating) creating capsules where the user of the capsule is preventing from *having control of some critical system resource* (because they are "untrusted"), but the user's capsule would *appear to have control of some critical system resource(s)* in order to simulate the capsule *having control* of the physical server and its *resources*, but not actually *having* that *control* (because the user of the capsule is "untrusted").

[0289] Tormasov's Figure 1 (which I have annotated 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].

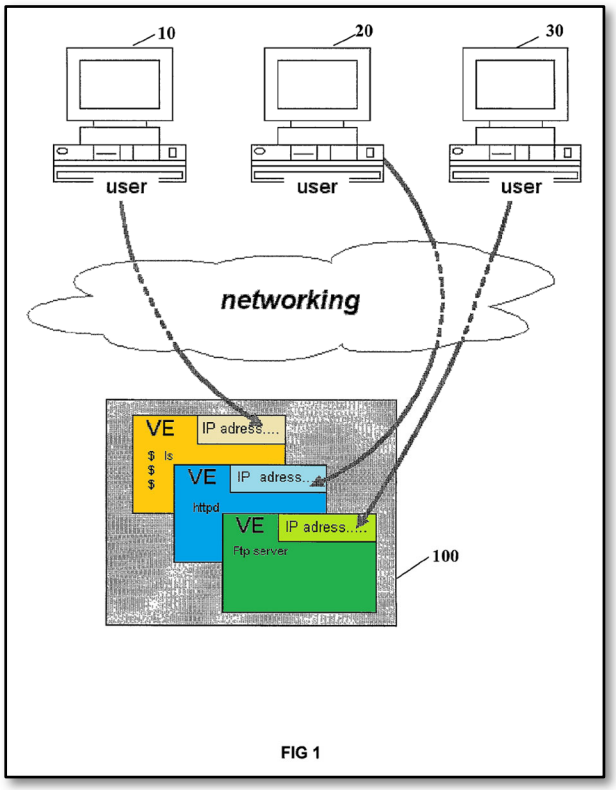


FIG 1

[0290] As I discussed above (§VII.B), each VCE-capsule provides 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]). Each VCE user(s) can "perform all actions allowed for the ordinary computer," like "installation of programs, configuration of network services," and execution of "applications." Tormasov, [0024].

[0291] Thus, application sets in each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule appear to have individual control of some critical system resource(s) (e.g., the server's processor, file system, and/or "network address") (Tormasov, [0024]), which meets both parties' constructions of *secure containers of application software* (EX1102, 13; EX1103, 14).

[0292] Moreover, based on Calder, an application's "resource requests [are] virtualized" to give the appearance of having the ability to "control[] resources," including "memory usage," "network bandwidth," "and disk usage. *See also* Calder, [0126]. The resources actually being used by the application are "transparen[t]" to and "hidden from the application." Calder, [0130].

[0293] That is another reason why Schmidt-Tormasov-Calder's VCE-capsules appear to have individual control of some critical system resource(s), which meets both parties' constructions of *secure containers of application software* (EX1102, 13; EX1103, 14).

[0294] "**data within each application set is insulated from effects of other application sets**": Each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule layer also meets prong (ii) of the parties' construction of a *secure application container* because "data within each application set [within each VCE-capsule] is insulated from effects of other application sets" in other VCE-capsules. EX1102, 13; EX1103, 14.

[0295] As I discussed above, each root user of a VCE perceives itself to have "**full network root access** to a common computer" and can modify the file system in the VCE as "**if they worked on totally <u>separate</u> computers** with no associated hardware," such as "installation of programs, configuration of network services," and execution of "applications." Tormasov, [0023]-[0024]. Each VCE also "has its own unique file system" and its application's processes are "completely **<u>isolated</u>** from each other." Tormasov, [0026].

[0296] Thus, data within each application set (read/write files) are insulated ("separated" and "isolated") from effects of other application sets in other VCEs, which is another way Schmidt-Tormasov's and Schmidt-Tormasov-

Calder's VCE-capsules would have met the parties' constructions of *secure containers of application software* (EX1006, 5). *See also* §VII.E.1.b.iii.

[0297] (POSAs would thus understand that Tormasov is, or is similar to, a chroot – changed root – environment – where each VCE has its own root file system – with the additional feature that each VCE's root file system is unique and insulated from effects by changes made to root file systems in other VCEs.)

[0298] Additional isolation is provided by Schmidt-479's because "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].

[0299] Moreover, based on Calder, the "applications" in VCE-capsules would have also been "prevent[ed]" "from improperly modifying or accessing data on the" computers. Calder, [0079]. *See also supra* §VII.E.1.b.iii (demonstrating that Schmidt-Tormasov-Calder's VCE-capsules are *secure environments*). That would have further insulated applications sets in Schmidt-Tormasov-Calder's VCE-capsules from the effects of application sets in other VCE-capsules, and thus is another reason why each Schmidt-Tormasov-Calder's VCE-capsule would have met each party's construction of *secure containers of application software*. EX1106, 5.

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

[0300] As I discussed above (§VII.E.1.c.i.(2), *container* definition part [**4**]), each of Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have had one or more *applications* that would have been *executable* in the VCE-capsule.

[0301] Based on Schmidt-479, each "[VCE-]capsule comprises one or more processes and their associated system environment." Schmidt-479, [0038]. Each of those processes would have been an ***application* program** in ***execution*** in the VCE-capsule. My opinion in that regard is corroborated by EX1121, which discloses, consistent with a POSA's background knowledge, that "[a] *process* is a dynamic object that is a program in execution." EX1121, 43 (emphasis original); EX1121, 42 ("[W]e call the executing program a *process*.") (emphasis original). The applications that would have been executed in a VCE-capsule are examples of programs, and thus executing those applications would have resulted in various processes being executed on the server.

[0302] Based on Schmidt-479, 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].

[0303] Tormasov's Figure 1 (to which I have added coloring below) demonstrates that VCEs *comprise one or more executable application programs*, like an http web-server application (blue) or Ftp (file-transfer-protocol) server application (green) that users can *execute* in the VCE, just as they were interacting with a conventional "full-featured operating system." Tormasov, [0023].

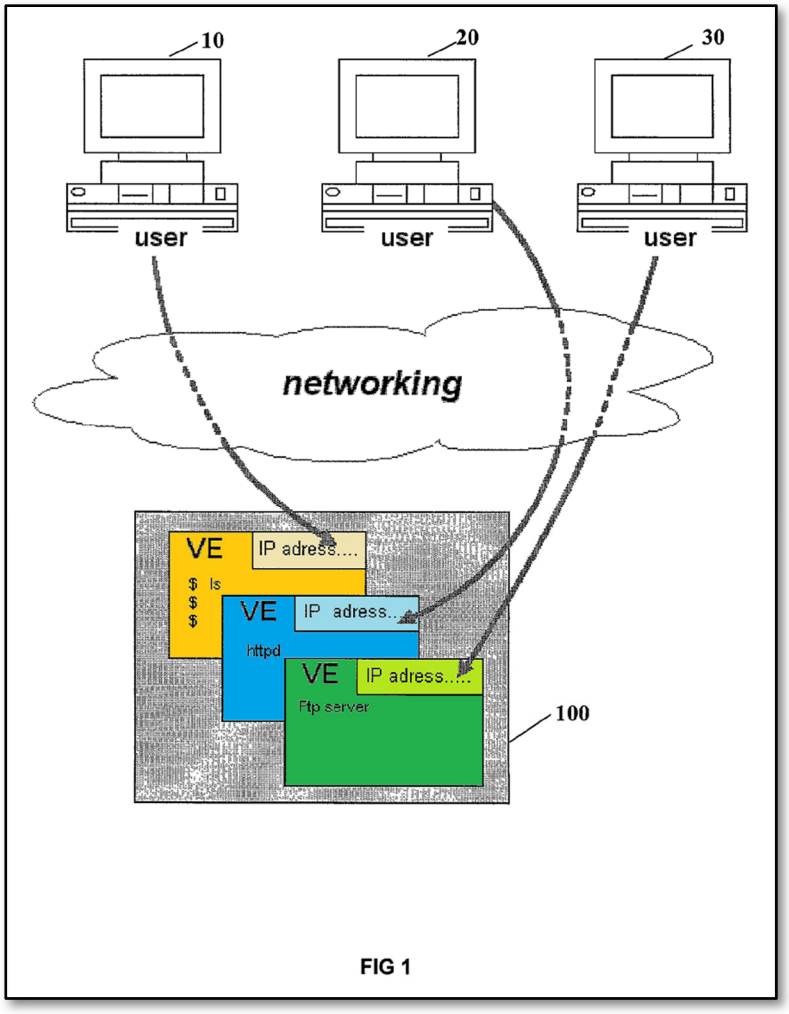


FIG 1

[0304] Thus, based on Schmidt-479's and Tormasov's disclosures discussed above, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would have *each comprised one or more of the executable applications*.

[0305] Based on Calder, each VCE-capsule in Schmidt-Tormasov-Calder would have also included "modified ***application*** binaries" (executable application programs) and modified *system files* so that the VCE-capsule (*container*) and the *applications* in each VCE-capsule (*container*) could be *executed* on servers running OSs that were native **and** non-native to the VCE-capsule's (*container's*) application(s). Calder, [0098]

[0306] That is an additional reason why Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would have *each comprised one or more of the executable applications*.

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

[0307] 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 '814 patent recognizes (as POSAs knew) that conventional prior art OSs provided (or at least typically provided) *system files*.

[0308] In Schmidt-Tormasov and Schmidt-Tormasov-Calder, each VCE-capsule would have been "functionally equivalent to a computer with a full-featured" OS (Tormasov, [0023]) and would have had its own "root file system" (Tormasov, [0030]). Like the conventional prior art OSs the '814 patent relies on, each VCE-capsule's local file system would have had its own *set of associated system files required to execute* the VCE-capsule *application* (and any additional applications therein).

[0309] For example, based on EX1122 (cited in Tormasov at [0035]), POSAs knew Tormasov's local file systems include (or in an obvious implementation would have included) "cop[ies]" of "the files that" the application(s) (e.g., "ftpd") "needs," including various library files ("lib" files in Unix's "lib" directory) and configuration files (in Unix's "etc" directory). EX1122, 122-123. I note that the '814 patent provides "example[s]" of *system files* in the "shared libraries" and "configuration files, [] provided with the [Linux] OS distribution," that are provided in the same "lib" and "etc" directories that EX1122 refers to. *See* EX1001, 2:55-3:1

[0310] My opinion that POSA would have understood that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used in their servers would have comprised a *kernel and local system files* is corroborated by EX1029. EX1029 discloses that typically the

"minimum…OS components **include[] a *kernel***, hardware drivers, **and *system files***." EX1029, [0027].

[0311] My opinion that POSA would have understood that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used in their servers would have comprised a *kernel and local system files* is corroborated by EX1023. EX1023 discloses that Windows-based OSs have "***system files*** which form the ***kernel*** part of the operating system." EX1023, [0023]. EX1023 further notes that these OSs have "customized **configuration files** [examples of *system files* in the '814 patent]" and "***system files*** of the operating system." EX1023, [0027].

[0312] My opinion that POSA would have understood that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used in their servers would have comprised a *kernel and local system files* is corroborated by EX1011. EX1011 discloses that "[e]very windows application is registered in a **windows *kernel*** with an application name." EX1011, [0032].

[0313] My opinion that POSA would have understood that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used in their servers would have comprised a *kernel and local system files* is corroborated by EX1067. EX1067 discloses that conventional "**operating**

system files" include "[t]he operating system ***kernel***, which forms the operating system's core program." EX1067, 9:12-20.

[0314] My opinion that POSA would have understood that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used in their servers would have comprised a *kernel and local system files* is corroborated by EX1064. EX1064 discloses:

> **A traditional operating system is logically layered and divided into two main portions: the *kernel*** and user programs.[16] The kernel interfaces with and controls the hardware, and also provides the user programs with a set of abstract system services called system calls. The kernel runs at a kernel level, where it can execute privileged operations and allows the kernel to have full control over the hardware as well as user level programs. This centralization provides an environment where all programs share the underlying hardware in a coordinated fashion.

EX1064, 1:53-62.

---

[16] Though EX1064 calls the "two main portions" "the kernel [or 'kernel level'] and user **programs**," POSAs conventionally refer to the two main portions of a conventional OS as "kernel **mode**" and "user **mode**." E.g., 1034, [0041] ("Kernel mode is a privileged processor mode that allows system code to have direct access to hardware resources."), [0047] ("A user mode is a mode of the host microprocessor that does not allow code to directly access hardware resources.")

[0315] My opinion that POSA would have understood that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used in their servers would have comprised a *kernel and local system files* is corroborated by EX1062. EX1062 discloses that "[m]any computing platforms **support standard shared libraries**. For example, the **Windows platform supports Dynamic Link Libraries (DLLs)**, the **Unix platform supports Shared Libraries**, and the **Macintosh platform supports Code Fragments**." EX1062, 3:37-40. As I discussed above, libraries are examples of *system files* identified in the '814 patent. Thus, EX1062 corroborates my opinion that conventional prior art OSs like those discussed in Schmidt-479, Tormasov, and Calder had *local system files* (e.g., shared libraries).

[0316] My opinion that POSA would have understood that the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used in their servers would have comprised a *kernel and local system files* is corroborated by EX1094. EX1094 discusses some of the "standard directories that will be found in many UNIX [OS]," which include:

> /bin This directory contains the basic system commands.
>
> **/etc** This directory contains **system configuration files** and programs used for administrating the system.
>
> **/lib** This directory contains the **system libraries**.

/tmp This directory is used to store temporary files.

/usr/bin This directory contains the commands that are not stored in /bin.

/usr/man This directory contains manual pages for programs

/usr/local This directory contains local programs that were installed by the system administrator (sysadmin) and were not included with the original system. In particular, /usr/local/bin contains local command files (binaries), and /usr/local/man contains local manual pages.

/home The actual directory location varies from system to system, but somewhere on the system will be a location where all of the users' home directories are located.

EX1094, 3:34-63. The files conventionally found in the /etc and /lib directories of a conventional prior art Unix OS file system are examples of *system files* identified in the '814 patent. *See* EX1001, 2:55-3:13 (identifies example "'*system' files*" in /lib and /etc directories).

[0317] Tormasov's VCEs "allow[] a root user" "to make file modifications and **configure** their own local **parameters** of the" OS. Tormasov, [0030]. Schmidt-479 similarly says (as POSAs knew) it was "desirable" to allows "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].

[0318] In the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used, those modifications/configurations would conventionally be made in configuration files, which are additional *system file required to execute the applications* in the VCE-capsule. For example, POSAs knew as part of the background knowledge that "[t]ypically,…a computer application is instantiated…in the form of one or more application processes…." EX1061, 1:12-16, 2:28-33. EX1061 further corroborates that "[t]ypically, when" an OS "instantiate[s] an application," it will "locate[]…and then process[]…the **configuration file[] for the application**…." EX1061, 2:53-3:5. Thus, when an application executed in VCE-capsule, the server would have conventionally processed one or more **configuration files** (an example of *system files*) corresponding to the application(s).

[0319] Moreover, as I discussed above (§VII.D.2), based on Calder's teachings, Schmidt-Tormasov-Calder's VCE-capsules would have also included modified *system files* (e.g., libraries and configuration files) that would have been *required to execute the one or more applications* in the VCE-capsule on native **and** non-native OS. Calder, [0077], [0082], [0094], [0101].

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

[0320] As I discussed above (§VI.B.7), Google construes *local kernel residing permanently on one of the servers* as "local kernel in one of the server's

memory that is not lost when power is removed from it" (EX1106, 4; EX1102, 12); VirtaMove argues that "[n]o construction is necessary" and that the "plain and ordinary meaning" should apply (EX1106, 4; EX1103, 13).

[0321] Tormasov's Figure 2 (which I have annotated with coloring below) shows VCEs *using* "a single hardware system with a single operating system [*local*] ***kernel*** [red]" *on one of the servers*. Tormasov, [0017]. Tormasov explicitly says that the VCEs on each server VCEs "share [*use*] the same ***kernel*** of the [***local***] operating system" *residing on the server*. Tormasov, [0023]-[0025].

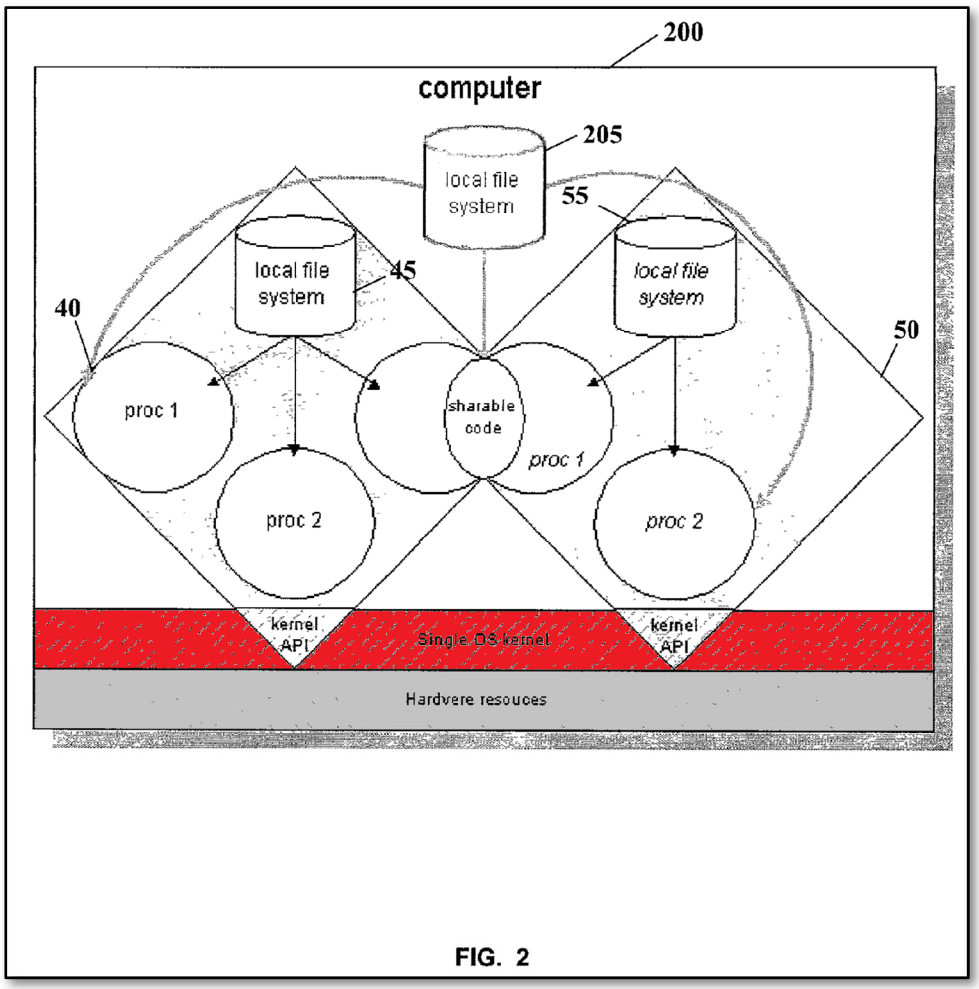


FIG. 2

See also Tormasov, [0029] ("The implementation of **the *kernels*** of the operating system with the properties necessary for this invention **carry out the separation of the personal computer users** not on the level of hardware but on the level of the namespace and **on the basis of access limitations implemented inside the *kernels* of the operating system**.").

[0322] A conventional *local kernel* (red) in the prior art that would have been used by each prior art OS on a servers executing VCE-capsules in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have *resided permanently* in memory of the *server* (a hardware resource (gray) above) because the kernel would not have been lost when power was removed from the server (e.g., in an obvious, typical implementation in the prior art the *kernel* would have been stored in non-volatile memory). That obvious implementation meets Google's construction "*local kernel residing permanently on one of the servers*" and VirtaMove's "plain and ordinary meaning" construction of that term. EX1106, 4.

[0323] My opinion that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's local kernel would not have been lost (or that POSAs would have had motivation for it not to be lost as a typical feature of prior art kernels) when power is removed from the corresponding server is corroborated by EX1121, which is cited in Tormasov's "BIBLIOGRAPHY." *See* Tormasov, [0034]. EX1121

discloses that the OS, and thus its associated *kernel*, "is the first program run on a computer when the computer boots up," which further corroborates my opinion that each conventional OS (and its *kernel*) in a Schmidt-Tormasov server and Schmidt-Tormasov-Calder server is stored in non-volatile memory, and remains present in that memory even when power is removed from the server. EX1121, 2.

[0324] My opinion that Schmidt-Tormasov's and Schmidt-Tormasov-Calder's local kernel would not have been lost (or that POSAs would have had motivation for it not to be lost as a typical feature of prior art kernels) when power is removed from the corresponding server is corroborated by EX1091. EX1091 discloses as background information:

> In general, the ability to store and access data is critical for computers. For example, **when turned on**, a computer (e.g., a personal computer ("PC")) accesses, and **prepares (or "boots") the operating system from its local persistent storage device (e.g., "hard disk"**). Once the booting is finished, the contents of the hard disk are accessible and available to the user. The contents of the hard disk (also referred to as the hard disk's "disk image" or "data image") define the user's personalized environment: **the operating system (such as Windows 98 SR-2, Linux, etc.)**, the software applications (such as word processors, spreadsheet programs, web browsers, etc.), the data files (such as documents, spreadsheets, images, or cookies), and any additional customization (such as whether a particular web browser,

such as Netscape Navigator or Internet Explorer, is automatically launched when an HTML file is accessed).

EX1091, 1:23-38. Thus, EX1091 corroborates that in a conventional computing device, the OS (and its associated local kernel) are stored in a persistent storage device (i.e., a device whose contents are not lost when power is removed from the server) so that it can be loaded whenever the computer is turned on from a powered-off state.

[0325] To the extent VirtaMove's "plain and ordinary meaning" construction of *local kernel residing permanently on one of the servers* (EX1106, 4) includes (at least) at least a kernel that is not lost when power is removed from the server, then Schmidt-Tormasov and Schmidt-Tormasov-Calder also meet this limitation under VirtaMove's construction. *See also* EX1103, 13. Moreover, as I discussed above, the VCE-capsules would have used conventional *local kernels* for conventional OSs (e.g., Windows- and Unix-based OSs), which are the same types of conventional OSs discussed in the '814 patent (e.g., EX1001, 7:33-35). Thus, to the extent VirtaMove's construction includes the same conventional OSs and *local kernels* that the patent discloses, that is another reason why Schmidt-Tormasov and Schmidt-Tormasov-Calder meet this limitation under VirtaMove's construction.

[0326] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have been *for use with a local kernel residing permanently on one of the servers*, as claimed, under both parties' constructions.

[0327] Moreover, based on Calder, Schmidt-Tormasov-Calder's VCE-capsules (*supra* §VII.D.2) would have "execute[d] [i.e., used] an interrupt call on the [local] operating system ***kernel*** of a client computer," which would have been used by the servers in Schmidt-Tormasov-Calder to execute VCE-capsule on non-native OSs in the system. *See* Calder, [0150].

[0328] That is an additional reason Schmidt-Tormasov-Calder's VCE-capsules would have been *for use with a local kernel residing permanently on one of the servers*, as claimed, under both parties' constructions.

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,"

[0329] As I discussed above (§VI.B.5), Google construes *at least some of the plurality of different operating systems* as "at least two or more of the plurality of different operating systems" (EX1102, 11); VirtaMove argues that "[n]o construction is necessary" and that the "plain and ordinary meaning" applies (EX1103, 12).

[0330] As I discussed above (§VII.E.1.c.iii, [**1A**]), each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule (*container*) would have

comprised a *set of associated system files*, like libraries **and** configuration files for execution on the plurality of different OSs in the system. Tormasov, [0026]; Schmidt-479, [0048]-[0052]; Calder, [0077], [0082], [0094], [0101].

[0331] Moreover, as I also discussed above (*see* §VII.E.1.c.iv), each VCE-capsule on one server would have used the same OS's *local kernel*, and thus *system files* within each VCE-capsule would have been *compatible with* the *local kernel of at least* that server's "single ***operating system***." Tormasov, [0017].

[0332] Furthermore (*see* §VII.E.1.a.i, [**1PREA**]), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have been compatible with at least two different OSs in the system, and thus their *system files* would have also been compatible with *the plurality of different* OSs under the parties' construction of "*at least some*…" (EX1100, 4; EX1101, 2).

[0333] For example, Schmidt-479 discloses that capsules can be "migrated to a **different instance of an operating system** on a different machine." Schmidt-479, [0044]. And another published patent application by Schmidt inventors (EX1116; "Schmidt-629") discloses that Schmidt's capsules can be moved to "a **different operating system**"). EX1116, [0015]. Another published patent application by Schmidt inventors (EX1009; "Schmidt-529") discloses that the "capsule is entirely self-contained and can be suspended in secondary storage, [and] arbitrarily bound to different machines and **different operating systems**."

EX1009, [0018]. Thus, EX1116 and EX1009 corroborate my opinion that Schmidt-479's capsule were executable on *different operating system* (or had reason to be with a reasonable expectation of success in doing so).

[0334] Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1B**] under Google's constructions. To the extent VirtaMove's plain-meaning construction of *at least some of the plurality of different operating systems* also includes (at least) at least two or more of the plurality of different operating systems in the system, then Schmidt-Tormasov and Schmidt-Tormasov-Calder also meet this limitation under VirtaMove's construction.

[0335] Moreover, based on Calder, Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would have had modified copies of system files to permit execution of the VCE-capsule's one or more applications on "any non-native [OS] platform," and thereby utilize non-native kernels that would have responded to those system calls. Calder, [0101].

[0336] That is another reason why Schmidt-Tormasov-Calder meet [**1B**] under (at least) Google's constructions.

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

[0337] As I discussed above (§VII.E.1.c.i.(2), [**1A**]) for the '814 patent's "container" definition **part [4]**) and reiterate below, Schmidt-Tormasov's

and Schmidt-Tormasov-Calder's VCE-capsules (which are *containers of application software*) would have *excluded a kernel*.

[0338] Schmidt-479's Figure 3 (which I have annotated below) shows the server's OS (red), and thus the OS (and its associated OS *kernel*) are stored outside the capsules. *See also* Schmidt-479, [0016], [0038] (describing the various files and data inside a capsule, but not stating or suggesting that a kernel would be inside a capsule); *see* EX1086, 2:53-55 (noting that, as POSAs knew, "the operating system is commonly called the system ***kernel*** [], or just the ***kernel***")). Thus, Schmidt-479 demonstrates that the VCE-capsules in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have shared a server's local kernel, and would have included (and would not have needed to include) a kernel.

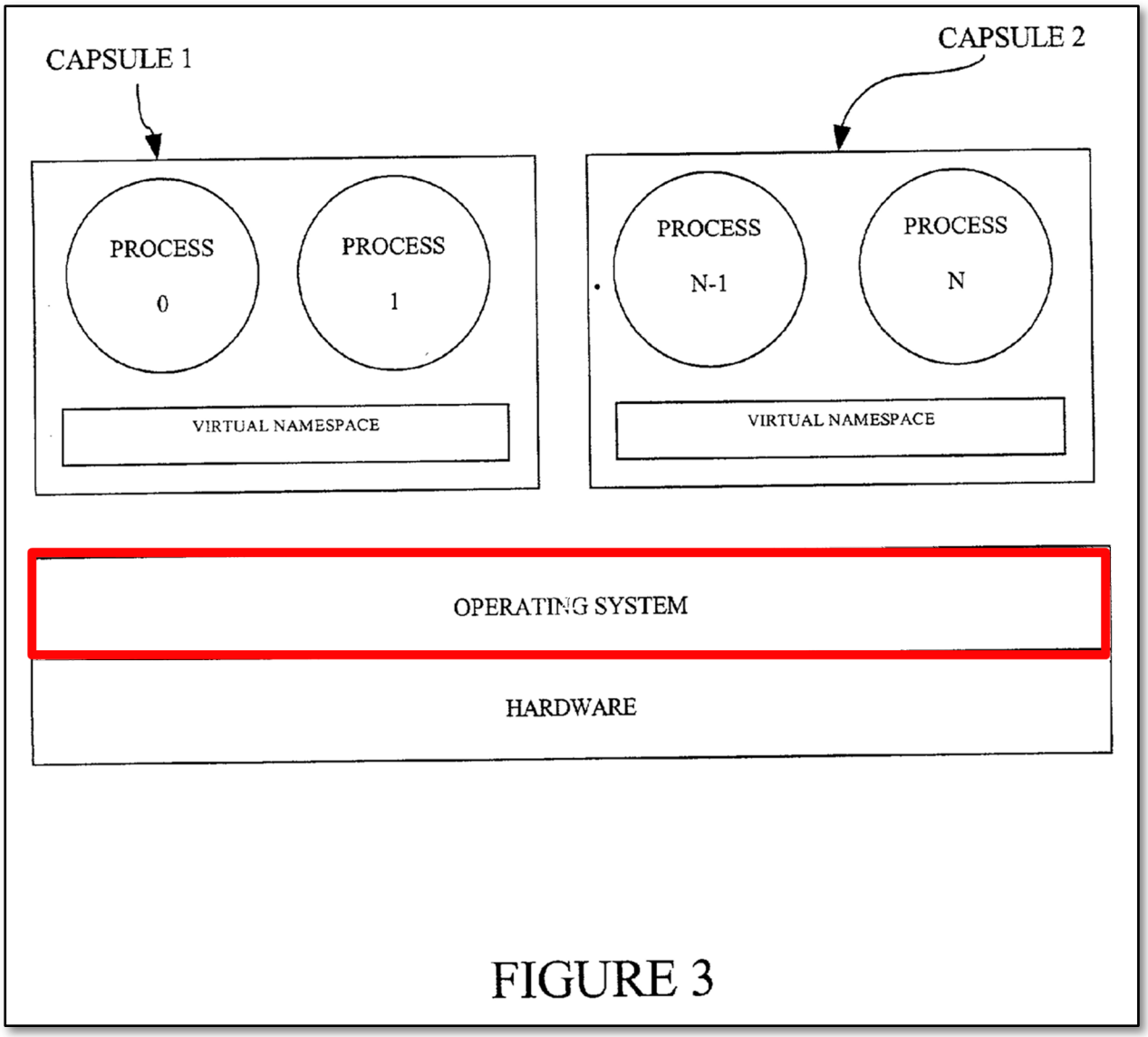


FIGURE 3

[0339] Similar to Schmidt-479 discussed above, Tormasov's Figure 2 (which I have annotated below), illustrates (and reinforces) that the VCEs that would have been in the VCE-capsules include (1) application programs including one or more processes and (2) local file systems for use in executing the one or more processes, but do not comprise (i.e., *exclude*) a kernel (red), which remains outside the VCEs on the server.

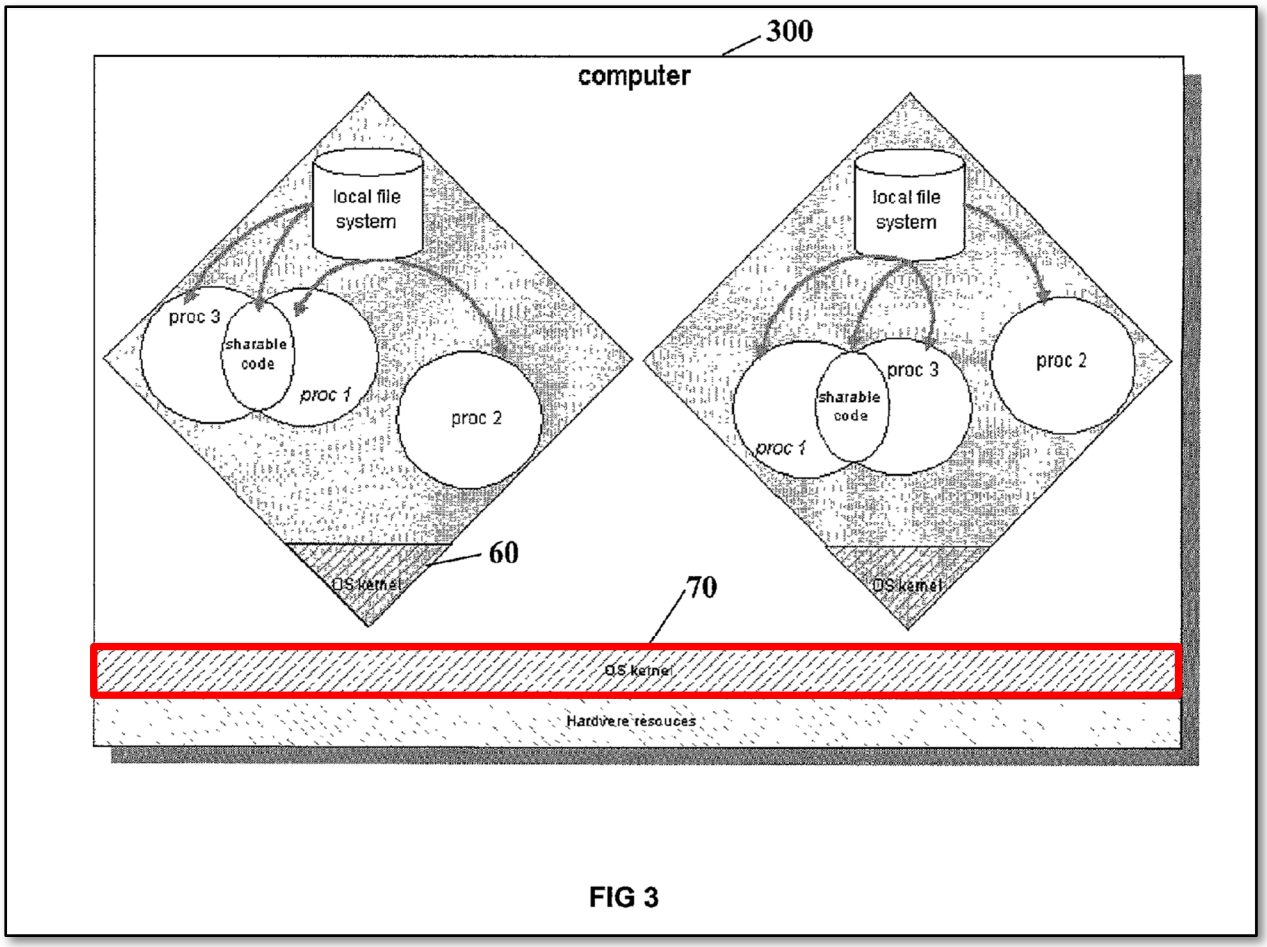


FIG 3

Rather than include a *kernel* within each (or any) VCE, Tormasov explains that when applications are executing in VCEs they "share" resources provided by the local kernel:

> All the processes inside the virtual computing environment are the common processes of the operating system and all the **resources** inherent to each virtual computing environment **are shared** in the same way as typically happens inside an **ordinary single *kernel* operating system.**

Tormasov, [0025]. That is, the VCEs (and application installed and processes executing therein) that are on one server share the server's "single [i.e. one] ***kernel*** operating system" (Tormasov, [0025]) rather than require distinct kernels for each VCE. That understanding is reinforced by Tormasov's background information, which (as I discussed above, §VII.B), expressly distinguishes Tormasov's VCEs from prior-art approaches that required "**several** operating system ***kernels***" on one machine when executing multiple, independent computing environments that permitted user to, e.g., install applications and modify *system files* without influencing each other (e.g., a virtual-machine technique). *See* Tormasov, [0014]-[0016]. That is, Tormasov's VCEs permitted multiple, independent computing environments on one machine with just one shared OS kernel (that was maintained outside the VCEs).

[0340] Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder's VCE-capsules (*containers of application software*) *would have* also *excluded a kernel*, and thus meet limitation [**1C**]. *See also supra* §§VII.D.1-2 (where I discuss the Schmidt-Tormasov-Calder and Schmidt-Tormasov-Calder combinations); §VII.E.1.c.i.(2) ([**1A**], discussing the '814 patent's container-definition part [**4**]).

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,"

[0341] The '814 patent says "[t]he term '*within a container*'" "mean[s] 'associated with a container.'" *See* EX1001, 2:32-34.

[0342] Tormasov's Figure 2 (which I have annotated below) shows that each *server* comprises a "common *file system*" (pink) *that remains resident on the server*. Tormasov, FIG. 2, [0026]. POSAs understood that that file system comprises (or in a typical, obvious implementation thereof for a conventional prior art OS would have comprised) *associated local system files* needed to execute the OS and to execute applications that are not installed to VCEs (and thus execute without accessing a VCE's local file system).

[0343] My opinion that the conventional OSs that would have been used in Schmidt-Tormasov and Schmidt-Tormasov-Calder for its servers would have comprised (or had reason to comprise) *local system files* is corroborated by EX1029. EX1029 discloses that the "minimum…OS components **include[] a *kernel***, hardware drivers, **and *system files***." EX1029, [0027].

[0344] Moreover, Tormasov's Figure 2's illustrates a "kernel API" (Application Program Interface). POSAs understand that a kernel API is (or in an obvious implementation would have been) used, *inter alia*, to make system calls to

the server's OS, which are activated and processed by libraries (*system files*) *that remain resident on the server*.

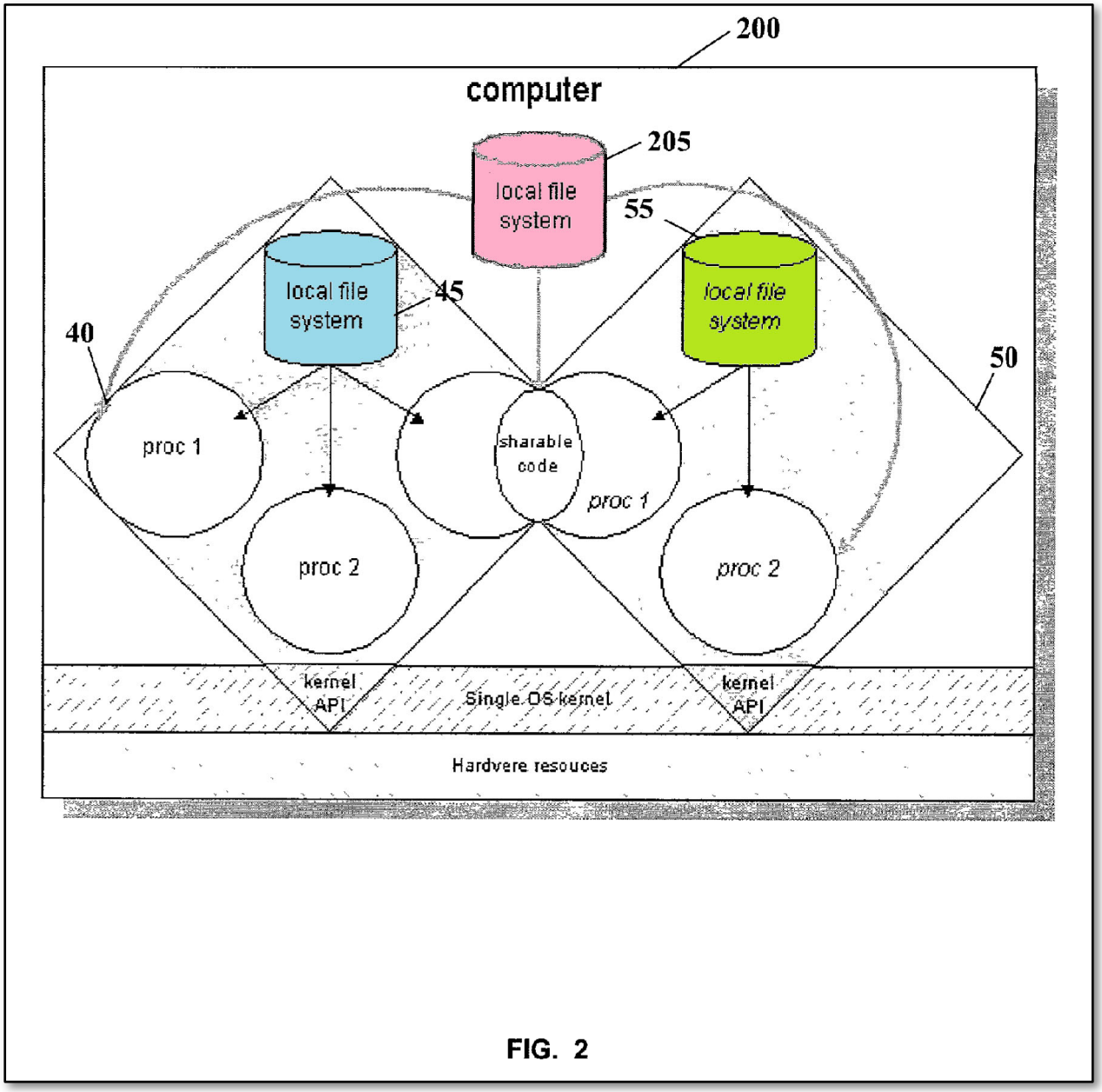


FIG. 2

[0345] My opinion that in conventional, prior art OSs, a server's OS kernel was typically and conventionally used by applications to make system calls

to the server's OS is corroborated by EX1086. EX1086 discloses in connection with its Figure 1 that:

> FIG. 1 depicts a computer based file system 100 which operates under control of a UNIX® operating system 110, shown using a high-level architecture layer diagram. (UNIX is a registered trademark of UNIX System Laboratories, Inc. (hereinafter USL)). The layer diagram includes a user level 120, a kernel level 130, and a hardware level 140. The user level 120 interfaces to a user 101 enabling access to the file system 100 to obtain the desired file stored in memory (e.g., disk 180).
>
> The user level 120 includes the user programs 121 and libraries 122. The hardware level 140 provides the operating system 110 with basic services needed by computer 100. **The kernel level 130 interacts directly with the hardware level 140 providing common services to user level 120 programs and insulating them from hardware idiosyncrasies**. Viewing the system as a set of layers, **the operating system is commonly called the system kernel 130, or just the kernel, emphasizing its isolation from user programs**. Because user programs are independent of the underlying hardware, it is easy to move them between UNIX systems running on different hardware. The general description of the well-known operation of a UNIX operating system is derived from Chapter 2 of the book entitled "The Design of the UNIX Operating System" by Maurice J. Bach.
>
> **The system call interface 131 represents the border between user level 120 (user programs 121 and program libraries 122) and the kernel level 130. System call interface** converts user program calls

into UNIX system calls. System calls look like ordinary function calls in C programs, and libraries map these function calls to the primitives needed to enter the operating system in a well-known manner. The set of system calls includes those that interact with the file system driver 132 and those that interact with the process control subsystems 133. The file system driver 132 manages files, allocating file space, controlling access to files, and retrieving data for users. **Processes interact with the file system driver 132 via a specific set of system calls, such as open (to open a file for reading or writing), close, read, write, star (query the attributes of a file), chown (change the record of who owns the file) and chmod (change the access permissions of a file)**.

EX1086, 2:37-3:11. Thus, EX1086 corroborates my opinion that a conventional OS kernel (e.g., the kernel of a conventional Unix-based OS) is accessed by applications via an interface (like an API) to obtain services from the underlying OS (e.g., to open, read, or write to a file).

[0346] As discussed above (§VII.E.1.c.iii, [**1A**]), each VCE-capsule (*container*) installs *associated system files* to a "local file system" (light blue and green above), which are *stored in memory* and *are utilized* by each VCE-capsule's application(s) (as illustrated by the arrows running from the local file systems to application processes). Tormasov, [0026]. EX1122 (which is cited in Tormasov at [0035]), further corroborates that POSAs knew that (or would have had motivation to have) Tormasov's local file systems include "cop[ies]" of "the files that" the

application(s) (e.g., "ftpd") "needs," including various library files ("lib" files in Unix's "lib" directory) and configuration files (in Unix's "etc" directory). EX1122, 122-123. I note that the '814 patent provides "example[s]" of "shared libraries" and "configuration files, [] provided with the [Linux] OS distribution," that are provided in the same "lib" and "etc" directories that EX1122 refers to. *See* EX1001, 2:55-3:13.

[0347] As I discussed above (§VII.B), Tormasov allows for "installation of programs….and other ***applications***" inside a VCE, where the applications are only accessible to that VCE where the application is installed. 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 would have understood that when an application is installed to a VCE, the application in the VCE uses the *associated* local *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). (That is the reason why local *system files*, like libraries and configuration files are copied into a VCE – to permit the user to interact with the VCE as if they were using "a full-featured operating system," Tormasov, [0023], "without noticing" any other users on the same physical server,

"just as if they [each] worked on totally separate computers with no associated hardware," Tormasov, [0024]).

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

[0349] As I discussed above (§VII.A), Schmidt-479 discloses that "the default set of file system view mappings" in each capsule "includes standard system directories, such as /bin for applications, **/lib for system libraries**, /usr/man for manual pages, etc." Schmidt-[0051]. Those files that are mapped into each capsule's "file system view" (e.g., highlighted blue and green above) include, *inter alia*, copies of *the associated local system files that remain resident on the server* (highlighted pink above; e.g., the system libraries in /lib). POSAs understood that they *remain resident on the server* because, at least initially, those same files are mapped to every capsule on the server. *See* Schmidt-479, [0051] ("all capsules [] access standard software and data in the system"). For example, Schmidt-479 explains that "[a]lthough capsule owners are free to modify their file system views, it is not desirable for them to modify the underlying local system-level files (for which they may not have permission). Thus, mounts of directories such as /lib are made read-only." Schmidt-479, [0052].

[0350] However, when a capsule modifies one of those files, e.g., a "user[] wish[es] to modify a directory such as /lib," the system will "make a **copy** in their view and remove the map to the underlying file system." Schmidt-479, [0052]. That "underlying file system" includes, *inter alia*, the *system files* (e.g., libraries) that *remain resident on the server*. Each capsule will have (at least) copies of any modified files, which is "termed copy-on-write mode" – a well-known functionality in prior art OS where copies of files are only made when they are modified (written to).

[0351] As an alternative to a copy-on-write mode, POSAs would have known that an alternative would have been a straight "copy" mode – wherein **all** files in the underlying file system would have been copied into each capsule. *See, e.g.*, EX1077, 122 (which I discuss further below, §VII.E.1.g). POSAs would have understood that that would have been a simpler way to provide each capsule with the files (including the *system files*) needed to execute the applications inside the capsule. The tradeoff of increased simplicity at the expense of the more-efficient copy-on-write mode would have been a conventional, well-understood tradeoff for POSAs.

[0352] For example, as EX1077 corroborates, using a straight-copy approach to creating a chroot environment by copying *system files* into the environment, as would have been done in Schmidt-Tormasov and Schmidt-

Tormasov-Calder VCE-capsules, allows for "the sandbox [chroot'd] environment to be moved to other similar platforms with little or no customization." EX1077, 122. EX1077 says that, for example, "in a company environment [they] can create the Netscape sandbox in one machine and then **copy** it to the machines of all the other employees" (i.e., the *associated local system files* from which these copies are made *remain resident on the* server). EX1077, 122. EX1077, notes, "[h]owever, copying all these files is wasteful and more importantly, we will need to keep track of all the duplicates and update them every time we upgrade the application, or the operating system." EX1077, 122. In other words, POSAs knew that a straight-copying approach to creating sandboxed/chroot'd environments by copying *system files* into the environment may not have always been the most efficient technique, but, as EX1077 discloses, it was a known technique, with benefits over an alternative, like a copy-on-write approach – specifically, a straight copying approach would have been simpler on the front-end (by simply copying rather than observing and copying on edited versions) and, as EX1077 states, straight copying allows for moving the environment (and the application and system files therein) to another machine "with little or no customization," consistent with Schmidt-479's goal of being able to move its similar capsules to different servers in a multi-server system.

[0353] Moreover, whether or not a straight copy or copy-on-write mode were utilized, as I've discussed above (§VII.A), Schmidt-479 discloses that its capsules can be suspended and stored so that they could be moved and/or started or restarted on any "binary compatible machine" in a multi-server system. *See* Schmidt-479, [0016]. As I discussed above (§VII.E.1.c.i.(2), §§VII.E.1.c.ii-iii), POSAs understood that Schmidt-479 is (or in an obvious implementation thereof would be) copying into each stored capsule the one or more applications being executed in the capsule and associated system files required to execute the application(s). The application binaries and system files that are / would have been stored in each capsule is / would have been a copy or modified copy of the application binaries and *local system files* of the server that the capsule was originally created on.

[0354] Thus, based on the disclosures in Tormasov and Schmidt-479 that I discussed above, *some or all of the associated system files within a* Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule (*container*) that would have been *stored in memory* would have been *utilized in place of the associated local system files that* would have *remained resident on the serve*r.

[0355] Thus, *system files* in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) meet limitation [**1D**].

[0356] Based on Calder, Schmidt-Tormasov-Calder's VCE-capsules include additional, modified system files – e.g., "libraries" and "configuration files" (Calder, [0082]) – that *are utilized in place of* (at least some) *local system files* to allow VCE-capsules (and their application(s)) to execute on non-native OSs. Calder, [0082]. For example, rather than use a local configuration file (an example of a *system file*), a Calder-modified application uses in its place a "modified configuration file[]." Calder, [0082].

[0357] That is another reason why, *system files* in Schmidt-Tormasov-Calder's VCE-capsules (*containers*) meet limitation [**1D**].

g. [1E] "wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server,"

[0358] As I discussed above (§VII.E.1.f, [**1D**]), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's servers would have comprise respective "local file system[s]" (Tormasov, FIG. 2, [0026]), and each of those local file system would have comprised comprise *system files* (*see* Tormasov, [0026]-[0031]; Schmidt-479, [0048], [0051]; Calder, [0082]).

[0359] When a VCE-capsule's application(s) were native to the server's local OS, at least some of the VCE-capsule's *system files* would have been *copies of* the server's *local system files* (e.g., identical libraries used to make that OS's system calls), which meets limitation [**1E**] under the apparent claim scope

VirtaMove applies in related district court litigation. *See* EX1079, 30 (VirtaMove arguing limitation 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).

[0360] For example, as I have discussed above, EX1122 (which is cited in Tormasov's "BIBLIOGRAPHY" at [0035]) discloses when "using chroot() to restrict users' access," as in Tormasov (*see infra* §VII.E.1.i, [**1G**]), "you need" "***cop[ies]***" of "the files that" the application(s) "needs," including various library ***system files***. *See* EX1122,122-23

[0361] Based on Schmidt-479, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would have been installed to and/or restarted on any server in the system by having "portions of" a first server's "underlying file system" "map[ped, i.e., ***copied***]…into their namespace." Schmidt-479, [0045]-[0046], Schmidt-479, [0052] ("mounts of directories such as /lib" are "***cop[ied]***" into each user's view of the file system); *see also* Tormasov, [0018] (VCEs are "separate[ed]" based on their respective "namespace[s]").

[0362] Schmidt-479 also says that "the 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 teaches a "***copy***-on-write

mode of file access to support the modification of read-only or ***system files***," where editing those files creates a *copy* in the user's environment (Schmidt-479, [0046]).

[0363] As EX1077 corroborates, using a straight-**copy** approach to creating a chroot environment by copying *system files* into the environment, as would have been done in Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules, would have allowed for "the sandbox [chroot'd] environment to be moved to other similar platforms with little or no customization." EX1077, 122. EX1077 says that, for example, "in a company environment [they] can create the Netscape sandbox in one machine and then **copy** it to the machines of all the other employees" (i.e., the *associated local system files* from which these copies are made *remain resident on the* server). EX1077, 122. EX1077, notes, "[h]owever, copying all these files is wasteful and more importantly, we will need to keep track of all the duplicates and update them every time we upgrade the application, or the operating system." EX1077, 122.

[0364] In other words, POSAs knew that a straight-copying approach to creating sandboxed/chroot'd environments by copying *system files* into the environment may not have always been the most efficient technique, but, as EX1077 discloses, it was a known technique, with benefits over an alternative, like a copy-on-write approach – specifically, a straight copying approach would have been simpler on the front-end (by simply copying rather than observing and

copying on edited versions) and, as EX1077 states, straight copying allows for moving the environment (and the application and system files therein) to another machine "with little or no customization," consistent with Schmidt-479's goal of being able to move its similar capsules to different servers in a multi-server system.

[0365] As I discussed above, (§VII.E.1.f, [**1D**]), whether using Schmidt's copy-on-write option or a straight-copy option, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have *stored copies or modified copies of the associated local system files that remain resident on the server* as Schmidt-479 taught (or at least suggested) as a way to permit capsules to be stored and started / restarted on any computing device in a system.

[0366] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder meet limitation [**1E**].

[0367] Moreover, based on Calder's teachings, Schmidt-Tormasov-Calder's VCE-capsules would have included include "**modified**" system files (e.g., "**modified** libraries [and], **modified** configuration files," Calder, [0082]) that would have been based on "**cop[ying]** specific information from the **existing system registry**" (Calder, [0121]). That is, Calder's modified system files are (or had reason to be) *modified copies of the associated local system files that remain resident on the server*.

[0368] That is another reason Schmidt-Tormasov-Calder meets limitation [**1E**].

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

[0369] Tormasov discloses that user's accessing VCEs "can work with the same [server] hardware node [] without noticing each other, just as if they worked on totally seperate computers with no associated hardware." Tormasov, [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**." Tormasov, [0026]. As a result, a VCE's "processes are never visible to other" VCEs "running on the same computer." Tormasov, [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, so "changes done in the file system in one" VCE "do not influence the file systems in the other" VCEs. Tormasov, [0030]-[0031]. *See also* Schmidt-479, [0052].

[0370] Because Tormasov disclsoes that (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** from processes running in other VCEs on the same computer, and (iii) VCE's can

be stopped and started independently, POASs understood that *the application software* within a VCE-capsule (*container*) *cannot be* (or had reason not to be) *shared between the plurality of secure containers of application software*.

[0371] Conversely, if the application software were shared between the VCE-capsules, a configuration of an application in one VCE would affect a configuration of that application in another VCE; similarly, if the application software were shared, starting and stopping VCEs and the applications therein would have an effect on the same applications executing in another VCE on the same computer. Those effects do not occur because, as I noted above, Tormasov discloses that VCEs are "**completely isolated from each other**." Tormasov, [0026].

[0372] Moreover, as I discussed above and further below (§VII.E.1.i, [**1G**]), based on similar teachings in Schmidt-479 and Tormasov, each VCE-capsule would have had a respective, unique root directory where the VCE-capsule's applications would have been installed, which teaches or suggests to POSAs (and thus at a minimum renders obvious) that applications would not have been shared across VCE-capsules (because the applications would have been installed and confined to different root directories). Tormasov, [0030]-[0031]; Schmidt-479, [0052].

[0373] POSAs would have also had motivation to prevent sharing application software between VCE-capsules as a way to protect applications installed in VCE-capsules from corruption by other VCE-capsules (e.g., on the same machine). Calder's teaches how to do this by using a well-known "sandbox directory" technique that includes "all of the files for the application packages." Calder, [0135]. As I discuss further below (§VII.E.1.i), each sandbox directory has a unique "virtual" root directory that appears to be the host's root directory, which teaches or at least suggests that applications cannot be shared across sandbox directories.

[0374] My opinion that applications installed to their own sandbox directory (such as would have been provided to each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule) would have resulted in *application software cannot be shared between the plurality of secure containers of application software* is corroborated by EX1075. EX1075 discloses that "sandbox[ed]" applications have "access only to the constrained environment and **cannot corrupt software applications outside it**, i.e. beyond the sandbox boundary." EX1075, 2:9-15.

[0375] Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder meet *the application software (within a* VCE-capsule *(container)) cannot be shared*

between the plurality of secure containers of application software, and meet limitation [**1F**].

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."

[0376] As I discussed above (§VI.B.9), in the District Court litigation Google argues that limitation [**1G**] is indefinite because "the claim language does not indicate to **which** 'operating system' th[e] comparison should be made" to determine whether "each of the containers [] unique root file system" must be "different from **an** operating system's root file system." EX1102, 14. VirtaMove argues that "[n]o construction is necessary" and the "plain and ordinary meaning applies." EX1103, 15. VirtaMove further argues that "[t]he point of the claim is that each claimed container will have its own root file system rather than sharing a root file system with *any* operating system's root file system. If a container shared its root file system with even one operating system, then it would not be unique." EX1105, 8 (emphasis original); *see also* EX1103, 8 (VirtaMove arguing: "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).

[0377] As I discuss below, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) each would have had a *unique root*

file system that would have been different from each other and from the root file system of every server in the world, and thus I understand that the Board may find that that meets [**1G**] regardless of the uncertain boundaries of what OS (or set of OSs) the claim is referring to.

[0378] Based on Tormasov, each VCE-capsule "has a **completely independent *root file system***." Tormasov, Abstract. As Tormasov's FIG. 2 (which I have added color to below) illustrates, "[e]ach of…two [VCEs] 40, 50 has its own ***unique file system*** 45, 55 [blue and green] and each [VCE] can also see the common file system 205 [red]," which is the server computer's *operating system's root file system*. Tormasov, [0026].

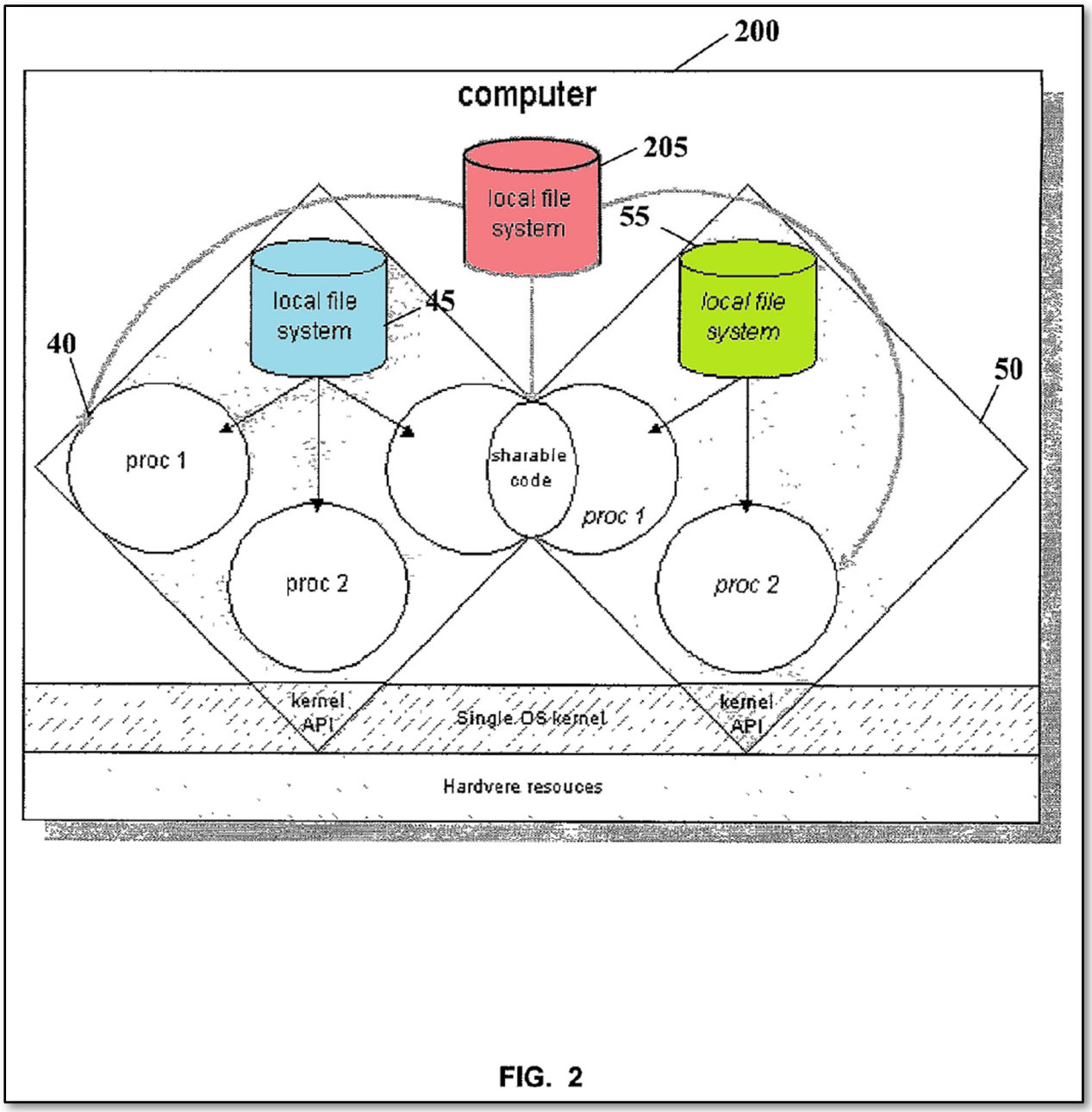


FIG. 2

[0379] Thus, each VCE's local *root file system* (e.g., 45, 55 above) is different from the server OS's *root file system* 205, which meets 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]. (I further note thought that Tormasov is not only providing a chroot environment for VCEs, but is specifically providing each VCE its own, respective chroot file system.) POSAs would have known that that changed root environment could be implemented using a chroot system call, as in Schmidt-479, as I discuss below.

[0380] Moreover, based on Tormasov, a VCE's "***root file system***…allows a root user" of the VCE-capsule "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 above) access the server 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 the VCE's *root file system different from*) the server computer's *operating system's root file system* 205. Tormasov, [0029]-[0031]. This also meets VirtaMove's interpretation of [**1G**]. *See* EX1103, 9.

[0381] As I discussed above in this section, Google explained in district court that limitation [**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. However, as I discuss below, 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 limitation [**1G**] regardless of the uncertain boundaries of what OS (or set of OSs) the claim is referring to.

[0382] For example, each VCE-capsule's "***root file system*** *is* [] never visible to other" VCE-capsules "running on the same computer." Tormasov, [0018], [0030] (each VCE's "root file system **is independent** and is also **never visible** to other" VCEs). Tormasov thus "allows the ***root*** user of every" VCE-capsule "to perform file **modifications** and local operating system parameters configuration." Tormasov, [0018]. As a result, each root user of a VCE "perceive[s]" itself having "obtained full network root access to a common computer" (when, in fact, a VCE's root user only has full access to the VCE's *root file system*). Tormasov, [0023]. Thus, as users make changes to their VCE-capsules (e.g., changing a configuration file, like the default fonts, font sizes, application startup order, icons, etc.) and/or install applications to their VCE-capsules (which typically would install various *system files*, like shared libraries) users will be making changes to the layers local *file system* that will make the files

with a VCE-capsule's *root file system unique* and *different from* the *root file system* of every other OS in the world.

[0383] Additionally, Schmidt-479 capsule technique (which would have been used in Schmidt-Tormasov/Schmidt-Tormasov-Calder, *see* §VII.D) 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]. *See also* Schmidt-479, [0013] (explaining that a capsule will continue to "a user to **customize their computing environment**, for instance **by** arranging and **naming their file system**"), [0020] ("**a capsule** might be created **for an untrusted guest** user by populating the namespace with reference to a limited set of resources"), [0052] ("it is not desirable" for user "to modify the underlying local-level files (for which they may not have permission)").

[0384] As I noted above, in the district court litigation, VirtaMove alleges that a container assigned to a "**chroot**" file system meets limitation [**1G**]. EX1079, 40-51 (emphasis original). For example, VirtaMove's contentions quote a document that says, in part, "[t]he original purpose of the cgroup, **chroot**, and namespace facilities in the kernel was to protect applications from noisy, nosey, and messy neighbors." EX1079, 40 (emphasis original). VirtaMove appears to have bolded "chroot" to emphasize that that is the functionality that VirtaMove alleges meets limitation [**1G**].

[0385] Thus, based on either Schmidt-479's chroot technique for capsules or Tormasov's similar technique for VCEs, under VirtaMove's apparent construction / claim scope in the district court litigation, each VCE-capsule's *root files system* is (or POSAs had motivation to make those *root file systems*) different *from* each other and *from* the *root file system* of every OS in the system, and thus are *unique root file systems*, as claimed. Given that chroot functionality was well-known in the art (and expressly disclosed in Schmidt-479, as I discussed above), POSAs would have had a reasonable expectation of success in doing so.

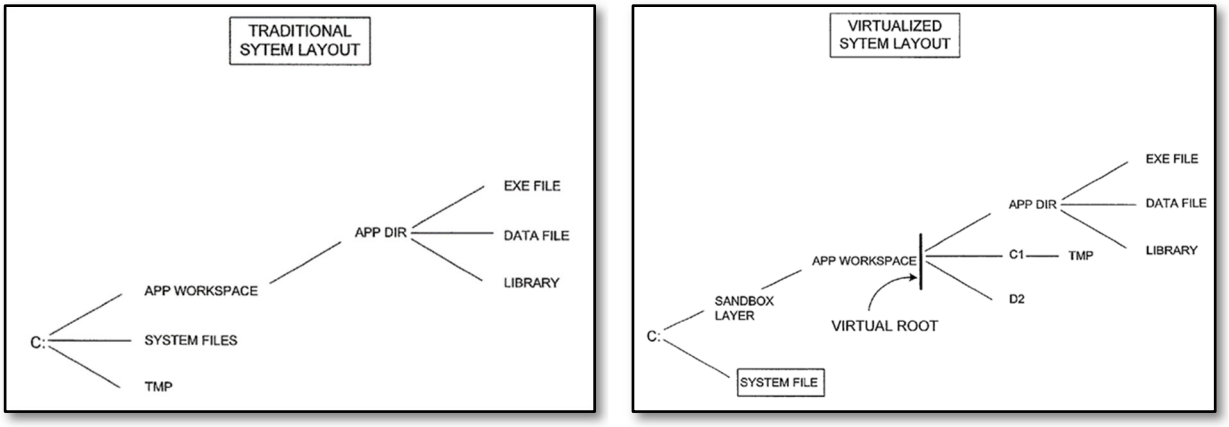
[0386] Additionally, Tormasov also provides "**separation** of user processes" in different VCEs "on the level of kernel objects/resources **namespace**." Tormasov, [0018], [0029]. That also appears to meet limitation [**1G**] under VirtaMove's apparent claim scope. *See* EX1079, 40, 45 (quoting documentation discussing uses of namespaces to isolate computing environments).

[0387] Thus, Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1G**] under VirtaMove's apparent claim scope in the two ways discussed above.

[0388] Additionally, as I discussed above and further below, Schmidt-Tormasov and Schmidt-Tormasov-Calder renders obvious having a VCE-capsule's (*container's*) root that would have been different from **every** OS root in the world, based on Calder's teachings discussed below. That obvious implementation would have been within the scope of limitation [**1G**] even though the precise, outer

boundary on the scope of limitation [**1G**] is unclear. That is, to the extent not taught by Tormasov or Schmidt-479, POSAs also had reason to provide each VCE-capsule (*container*) with *a unique root file system that is different from* the *root file system* of every server's OS in the world, based on Calder's teachings discussed below.

[0389] Calder's Figure 46 (below left) shows a "traditional system layout"; Figure 47 (below right) shows a "virtualized system layout":



In the traditional layout (above left), an application can "access a DOS prompt for the ***root*** directory C:," and see "three folders located within the ***root*** directory": "APP WORKSPACE," "SYSTEM FILES," and "TMP." Calder, [0241]. In Calder's "virtualized system layout," inside the "sandbox directory" is "a virtual **root directory** [that] provides the directory structure." Calder, [0241]. When Calder-modified application "request[]" "to access[] the subdirectory 'C:\TMP,' the interception module would rename the file to its corresponding location in the

sandbox directory C:\SANDBOX_LAYER\AP[P]_WORKSPACE\ TMP," which is "done transparently to the application." Calder, [0241].

[0390] Thus, when Schmidt-Tormasov-Calder's VCE-capsules would have used Calder's modified system calls, the application(s) in each VCE-capsule thinks would have had access to the OS's root file system ("C:\") but, in fact, would have been accessing *a unique root* directory (and thus *unique file system*)– "C:\SANDBOX_LAYER_AP[P]_WORKSPACE."

[0391] Thus, for the reasons discussed above, Schmidt-Tormasov and Schmidt-Tormasov-Calder meet limitation [**1G**] under VirtaMove's apparent claim scope and are within the outermost possible boundary of the scope of limitation [**1G**] even though the precise outer boundary is unclear.

* * *

[0392] Thus, for the reasons I discussed above (§§VII.E.1.a-i), in my opinion, POSAs would have found claim 1 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations under both parties' constructions.

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

[0393] As part of their background knowledge in the art, POSAs knew that the conventional prior art OS that would have been used in the servers in

Schmidt-Tormasov and Schmidt-Tormasov-Calder – like Windows-, Unix-, and Linux-based OSs, *see supra* §VII.E.1.a.i – would have had at least one *execution file associated otherwise for starting applications* in an environment provided by the OS, *e.g.*, *execution files* for starting particular applications (potentially in a particular order) upon start-up of the OS.

[0394] My opinion in that it would have been conventional for the conventional OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used to have an *execution file associated therewith for starting applications* is corroborated by EX1041. EX1041 discloses as background information:

> **Various application programs and system components** can also read **their own configuration and initialization files when they start up**. For example, the **WINDOWS™ program** developed and sold by the Microsoft Corporation, Redmond, Washington is a well-known graphic interface program which **has its own initialization files**. **When this program starts, it reads five initialization files**, some of which contain parameters that specify the appearance of a user interface, **a list of application programs to start automatically**, and the size and location of a swap file.

EX1041, 2:8-17.

[0395] My opinion in that it would have been conventional for the conventional OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used to have an *execution file associated therewith for starting applications* is

further corroborated by EX1027. EX1027 discloses what it terms "compartments" that had "**startup and shutdown scripts** [that] are responsible for inserting rules, creating routing-tables, mounting filesystems…and other per-service initialization steps." EX1027, [0070]-[0073].

[0396] My opinion in that it would have been conventional for the conventional prior art OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used to have an *execution file associated therewith for starting applications* is further corroborated by EX1013. EX1013 discloses as background information:

> **The selection of what applications to load and the order in which the applications are loaded by the operating system** ("the start-up sequence") is determined by **examining one or more start-up data structures maintained by the operating system**.

EX1013, [0003].

[0397] POSAs understood that the conventional initialization files (EX1041), start-up data (EX1013), and startup scripts (EX1027) discussed above for use in conventional, prior art OS are, or at least typically would have been processed by, *execution files for starting the application(s) identified in those* files/data/script. For example, Windows has an *execution file* that reads the initialization files that EX1041 refers to so as to determine which "application programs to start automatically." EX1041, 2:8-17.

[0398] As I discussed above (§VII.B, §VII.E.1.a.i), Tormasov discloses servers that provide multiple, independent virtual **computing environments** ("VCEs") that are each "functionally equivalent to a computer with [a] full-featured operating system." Tormasov, [0023]. Because each VCE is functionally equivalent to a full-featured OS, each VCE installed to and executing on a server can execute independent application-based services (e.g., independent websites and/or independent FTP-server applications). Tormasov, [0024], FIG. 1 (showing an httpd application and an ftpd application in different VCEs on one server)). Tormsov further discloses that users may also "stop and **start** the virtual computing environment **in the same manner as with a common operating system**." Tormasov, [0025].

[0399] POSAs would thus have found it obvious to use a conventional *execution file* to *start* a Schmidt-Tormasov VCE-capsule and a 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 I discussed above.

[0400] Additionally, VirtaMove alleges claim 2 is met by an "image configuration" that "includes information such as application arguments, environments, etc." EX1079, 51-55. Each of Tormasov's VCEs is an "*environment* that is functionally equivalent to a computer with full-featured

operating system" that allows for "command shells, compilation and installation of programs, configuration of network services," etc. Tormasov, [0023]-[0024]. Based on Schmidt-479's teachings, each VCE-capsule would also include "[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.

[0401] My opinion in that regard is corroborated by EX1084. EX1084 discusses "Portable Storage Devices Containing Layers" (unique computing environments like Tormasov's VCEs) stored on, for example, "Flash memory devices" or "CD-ROM or CD-RW media." EX1084, [0145]-[0146]. The portable media device "includes software components for **automatically installing layered system drivers** contained in driver package 1006 to a Windows-type architecture." EX1084, [0147]. When portable media with layers is recognized by a computing device, the computer "attempt[s] to detect an 'autorun' configuration element, for example an **executable file** with a particular name or a script or other configuration file **specifying instructions to execute**." EX1084, [0148]. "The purpose of the

installation application is to **install layered system drivers to the host operating system and to enable layers** located on the media device." EX1084, [0149].

[0402] "The installation application may then proceed to **enable the layers** stored to the portable device." EX1084, [0150]. "[T]he installation application may **activate all layers** stored to the device." EX1084, [0150]. As a result, "**the specified default application is executed or started**." EX1084, [0150]. "Alternatively, the executable started in step 1518 might include a window, icon or other widget to signal termination of the use of the portable media device and layers stored thereon." EX1084, [0151]. "Other auto-loading layered devices may be constructed for other operating systems, or for several operating systems for the same device, by following the principles given above." EX1084, [0153].

[0403] Thus, in my opinion, POSAs would have found claim 2 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations in any of the ways I discussed above.

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

[0404] As I discussed above (§VII.E.2, claim 2), POSAs had motivation to have *an execution file associated* for starting the one or more applications. As the evidence I cited above demonstrates (as I discuss again

below), those *execution files* typically (and beneficially) *included instructions related to an order in which executable applications will be executed*. For example, Tormasov's example https web-server application (Tormasov, FIG. 1) often required supporting applications (e.g., database applications) to start before the web-server application could execute properly.

[0405] My opinion in that it would have been conventional for the conventional OSs that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have used to have an *execution file* that *includes instructions related to an order in which executable applications within will be executed* regard is corroborated by EX1041. EX1041 discloses as background information:

> **Various application programs and system components** can also read **their own configuration and initialization files when they start up**. For example, the **WINDOWS™ program** developed and sold by the Microsoft Corporation, Redmond, Washington is a well-known graphic interface program which **has its own initialization files**. **When this program starts, it reads five initialization files**, some of which contain parameters that specify the appearance of a user interface, **a list of application programs to start automatically**, and the size and location of a swap file.

EX1041, 2:8-17.

[0406] My opinion in that it would have been conventional to have an *execution file* that *includes instructions related to an order in which executable*

applications within will be executed regard is further corroborated by EX1027. EX1027 discloses what it terms "compartments" that had "**startup and shutdown scripts** [that] are responsible for inserting rules, creating routing-tables, mounting filesystem…and other per-service initialization steps." EX1027, [0070]-[0073].

[0407] My opinion in that it would have been conventional to have an *execution file* that *includes instructions related to an order in which executable applications within will be executed* regard is further corroborated by EX1013. EX1013 discloses as background information:

> **The selection of what applications to load and the order in which the applications are loaded by the operating system** ("the start-up sequence") is determined by **examining one or more start-up data structures maintained by the operating system**.

EX1013, [0003].

[0408] My opinion in that it would have been conventional for the conventional to have an *execution file* that *includes instructions related to an order in which executable applications within will be executed* regard is further corroborated by EX1021 discloses:

> For example, **many applications use Microsoft Word as an engine for doing Mail Merge or document creation functionality**. **The application must know about the installation or presence of Word and be able to tap into its functions**.

EX1021, [0020]. Thus, in order for a program to use Word for the Mail Merge and/or document creation functionality, Word must be activated before the merge could occur and thus POSAs would have had reason to include an execution file that executed Word before executing any application that relied on Word for supporting mail-merge functionality.

[0409] Thus, in my opinion, POSAs would have found claim 3 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

4. Claim 4: "[C]laim 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 [[1A]].

[0410] As discussed above (§VII.E.1.c.i.(2)), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules are transportable *containers* of software applications stored in a memory accessible to the servers before being installed to a server. *See supra* §VII.E.1.c.i.

[0411] POSAs would have understood that for VCE-capsules to be stored and communicated to servers, it would have (at lesat) been obvious that the *application(s) and system files* within the VCE-capsules (*containers*) that are *required for association with* the VCE-capsules would have been (or had reason to have been) *pre-identified prior to said storing step* in limitation [**1A**].

[0412] Because the applications and files within (i.e., *associated with*) a VCE-capsule are stored in a transportable package that is in existence **before** installation of the VCE-capsule to the computing device, the VCE-capsule includes *pre-identified applications and system files required for association with the one or more containers prior to said storing step*.

[0413] My opinion in that regard is corroborated by EX1084, which was filed on December 30, 2004 and claims priority to a provisional application (EX1085) filed on December 30, 2003.

[0414] EX1084 discusses "Portable Storage Devices Containing Layers," stored on, for example, "Flash memory devices" or "CD-ROM or CD-RW media." EX1084, [0145]-[0146]. The portable media device "includes software components for **automatically installing layered system drivers** contained in driver package 1006 to a Windows-type architecture." EX1084, [0147]. When portable media with those layers is recognized by a computing device, the computer "attempt[s] to detect an 'autorun' configuration element, for example an **executable file** with a particular name or a script or other configuration file **specifying instructions to execute**." EX1084, [0148]. "The purpose of the installation application is to **install layered system drivers to the host operating system and to enable layers** located on the media device." EX1084, [0149].

[0415] "The installation application may then proceed to **enable the layers stored to the portable device**." EX1084, [0150]. Thus, the layers (and applications and system files within the layer) were pre-identified and prior to being stored to the portable media device and associated with the layer(s) on the storage device. The same would have been true as an obvious implementation of the system files within each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule.

[0416] Based on Calder, at least some of the VCE-capsules that are stored on memory that is accessible to the servers is *associated with pre-identified* applications and associated system files, like modified libraries and configuration files *required to* execute the applications on servers running non-native OSs. Calder, [0082]. Based on Calder, those VCE-capsules (and their application(s))are transmitted and installed on memory accessible to those servers (Calder, [0073]), which is another way Schmidt-Tormasov-Calder meets claim 4. (If the application and associated system files in the VCE-capsules were not *pre-identified* and *associated with the one or more* VCE-capsules (*containers*) then those applications would not have been executable on servers running non-native OS because the required *system files* would not have typically been present in on a server with anon-native OS).

[0417] Thus, in my opinion, POSAs would have found claim 4 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

5. Claim 5: "[C]laim 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."

[0418] Tormasov "separate[es]" VCEs "on the level of the namespace" (Tormasov, [0029]), which is a container specific identity assigned to that particular container. POSAs would have had reason to modify at least some of the associated system files in plural containers to provide an association" with the unique locators (namespaces) so a VCE's application could identify/communicate with an application isolated in another VCE.

[0419] Thus, in my opinion, POSAs would have found claim 5 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

6. Claim 6: "[C]laim 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."

[0420] Tormasov's Figure 1 (which I have added color to below) shows that each VCE is *assigned* its own a *unique* "***IP address***" (a *unique associated identity*). For example, as illustrated below, user 20 accesses the VE

executing an httpd application at one *IP address* (light blue) – which is the *IP address* at which other computers would have accessed the http server (e.g., web site) hosted by that VCE (and thus is the *IP address assigned to* that VCE) – and user 30 accesses VE executing the Ftp server at a different *IP address* (light green) – which is a different IP address at which other computers would have accessed the ftp server hosted by that VCE (and thus is a different *IP address assigned* to that VCE). That is, Thus, based on Tormasov, Schmidt-Tormasov and Schmidt-Tormasov-Calder meets claim 6.

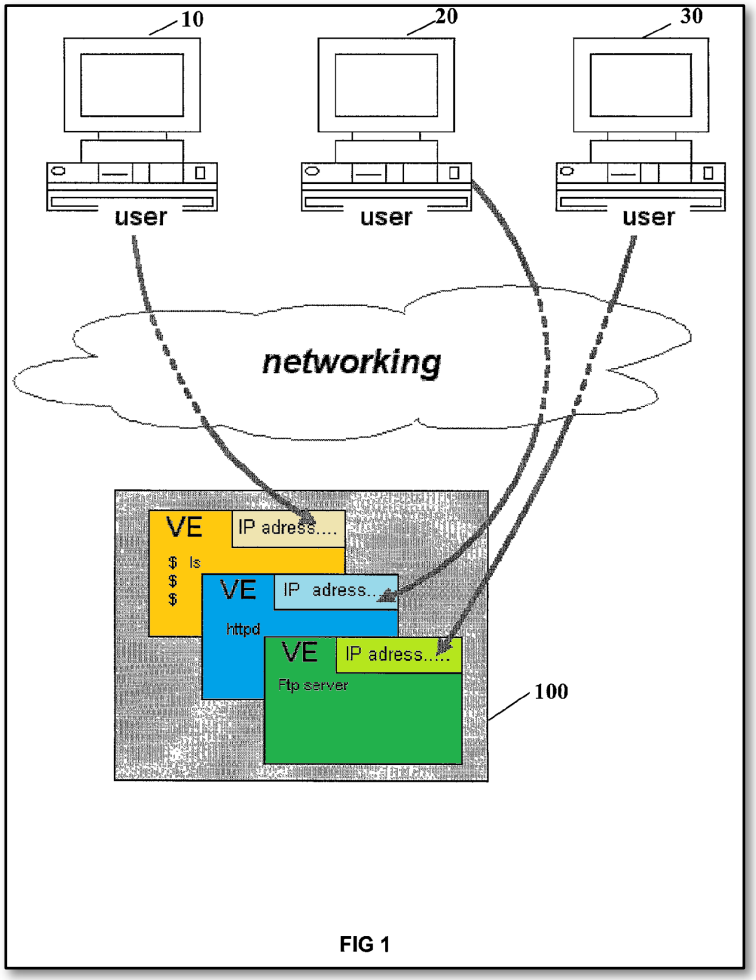


FIG 1

[0421] Additionally and relatedly, Tormasov sought to allow different end-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 suggests and motivates, and thus renders obvious) *assigning* to each VCE *a unique identity* that *includes at least* an *IP address*. That is because, for example, different web servers were conventionally (and beneficially) assigned different IP addresses and/or different DNS names (i.e., host names). Similarly, different network services would require different MAC addresses when the network was based on assigning nodes different MAC addresses (e.g., conventional Ethernet networks).

[0422] My opinion that, based on their background knowledge, POSAs would have understood that an *IP address* and/or *host name* would have been a conventional, unique identifier for web servers in the prior art is further corroborated by EX1130. EX1130 discloses as background information:

> **Every host and router on the Internet has an *IP address***. The Domain Name System (DNS) is often used to map ***host names*** to these ***IP addresses***. By way of example, a client typically sends a DNS query to a DNS server which includes a ***host name*** and an indication that an ***IP address*** is requested. The DNS server then returns an ***IP address*** **associated with the *host name***. It would therefore be desirable if a DNS server were designed to accomplish load distribution compatible with the Internet and DNS.

EX1130, 2:6-14.

[0423] My opinion that, based on their background knowledge, POSAs would have understood that an *IP address* and/or *host name* would have been a conventional, unique identifier for web servers in the prior art is further corroborated by EX1131. EX1131 discloses as background information:

> In general, all the machines on the Internet can be categorized into two types: servers and clients. Typically, machines that provide services (like Web servers, FTP servers, Email servers, etc.) are servers. Servers are loaded with the appropriate software in order to allow them to perform their intended services. Machines that request information from servers are typically called clients. **In order to differentiate between machines on the network, each machine is given a unique address called an *IP address***.
>
> The IP address is a thirty-two bit number that is normally expressed as 4 octets in a dotted decimal number (e.g., 192.168.1.101). Each of the octets can have values between 0 and 255 (2^8 possibilities per octet).

> **When a client connects to the Internet, the client is assigned an *IP address*** through their Internet Service Provider (ISP) for the duration of the connection. Conversely, the IP addresses of servers are relatively static, and do not change very often.
>
> Because it is difficult for clients to remember IP addresses, and because IP addresses need to change, most servers on the Internet possess domain names (e.g., "www.whoknowz.com") to help users reach their intended servers without remembering strings of numbers. **Name servers, used in the domain name system (DNS), map the human-readable names [e.g., host names] into *IP addresses* to help clients reach their destinations**. When a client enters a domain name, the browser (via a resolver) extracts the domain name and passes it to a name server, which will return the correct ***IP address*** **to the associated site**. The Domain Name System is comprised of a distributed database and name servers that access that database.

EX1131, 1:27-56.

[0424] Based on Schmidt-479 (*supra* §§VII.D.1-2), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have also been assigned "unique" "tokens" that would have been "bound to different physical resources on any host in the system" (Schmidt-479, [0054]), including "Internet addresses" (Schmidt-479, [0055]), which suggests (if not teaches) *assigning* each capsule *a unique IP address*.

[0425] And another published patent application by Schmidt inventors (EX1116; "Schmidt-629") explicitly discloses "assign[ing]" capsules "a ***unique*** locator, such as an ***IP address***." EX1116, [0015]-[0016].

[0426] In my opinion, POSAs would have also had motivation to assign each VCE-capsule additional unique identifiers – e.g., *host name* and *MAC address* – to support communication protocols that used those unique identities (e.g., MAC address for Ethernet networks; hostname for remote access).

[0427] My opinion that, based on their background knowledge, POSAs would have understood that a *host name* was a conventional, beneficial unique identifier in the prior art is further corroborated by EX1039. EX1039 discloses as background information that a "***hostname*** **is a** ***unique*** **name by which a computing device may be** ***identified*** on a network…[and thereby] simplify access to computing devices." EX1039, [0002].

[0428] My opinion that, based on their background knowledge, POSAs would have understood that a host name was a conventional, beneficial unique identifier in the prior art is further corroborated by EX1019. EX1019 discloses "virtualization of local area networks (LANs) or the virtualization of I/O storage" so that:

> processing resources may be deployed rapidly and easily through software via configuration commands, e.g., from an administrator,

> rather than through physically providing servers, cabling network and storage connections, providing power to each server and so forth.

EX1019, [0012]. EX1019 further discloses "**assign[ing] virtual *MAC addresses* to each processor node**" so "processing resources may be deployed rapidly and easily through software via configuration commands." EX1019, [0023].

[0429] My opinion that, based on their background knowledge, POSAs would have understood that an *IP address* and/or *host name* would have been a conventional, beneficial unique identifier for web servers in the prior art is further corroborated by EX1130. EX1130 discloses as background information:

> **Every host and router on the Internet has an *IP address***. The Domain Name System (DNS) is often used to map ***host names*** to these ***IP addresses***. By way of example, a client typically sends a DNS query to a DNS server which includes a ***host name*** and an indication that an ***IP address*** is requested. The DNS server then returns an ***IP address*** **associated with the *host name***. It would therefore be desirable if a DNS server were designed to accomplish load distribution compatible with the Internet and DNS.

EX1130, 2:6-14.

[0430] My opinion that, based on their background knowledge, POSAs would have understood that an IP address and/or host name would have been a conventional, beneficial unique identifier for web servers in the prior art is further corroborated by EX1131. EX1131 discloses as background information:

In general, all the machines on the Internet can be categorized into two types: servers and clients. Typically, machines that provide services (like Web servers, FTP servers, Email servers, etc.) are servers. Servers are loaded with the appropriate software in order to allow them to perform their intended services. Machines that request information from servers are typically called clients. **In order to differentiate between machines on the network, each machine is given a unique address called an *IP address***.

The IP address is a thirty-two bit number that is normally expressed as 4 octets in a dotted decimal number (e.g., 192.168.1.101). Each of the octets can have values between 0 and 255 (2^8 possibilities per octet). **When a client connects to the Internet, the client is assigned an *IP address*** through their Internet Service Provider (ISP) for the duration of the connection. Conversely, the IP addresses of servers are relatively static, and do not change very often.

Because it is difficult for clients to remember IP addresses, and because IP addresses need to change, most servers on the Internet possess domain names (e.g., "www.whoknowz.com") to help users reach their intended servers without remembering strings of numbers. **Name servers, used in the domain name system (DNS), map the human-readable names [e.g., host names] into *IP addresses* to help clients reach their destinations**. When a client enters a domain name, the browser (via a resolver) extracts the domain name and passes it to a name server, which will return the correct ***IP address*** **to the associated site**. The Domain Name System is comprised of a distributed database and name servers that access that database.

EX1131, 1:27-56.

[0431] Thus, in my opinion, POSAs would have found claim 6 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0432] The '814 patent mentions the term "mount points" but does not explain what it means or provide an example of a mount point. EX1001, 11:17-18. POSAs understood that, conventionally, "[a] mount point simply allows a drive volume to be mounted at some point within an existing directory tree structure." EX1056, [0008]. For example, EX1018 discloses that:

> **"Mounting" a file system means taking a file system and mapping it to an existing directory in the file tree, called the <u>mount point</u>**. Once a file system is mounted at a given mount point, the file tree of that file system is accessed as if it is contained in the directory serving as the mount point. **Mount points are typically empty directories**, as the contents of a directory serving as a mount point will be inaccessible while the new file system is mounted there.

EX1018, [0044].

[0433] As I discussed above (§VII.E.1.i, [**1G**]), each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule would have had a local file

system and unique root file system (using a so-called chroot (change-root) environment). Schmidt-479 discloses that an example way "[t]o create [a] file system view from the configuration **files**" has "***mount points*** [that] are established for each of the required directories." 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 had reason to *modify* similar configuration files (*system files*) *to define container specific mount points associated with the* Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules (*containers*) as an obvious, conventional way to mount each VCE-capsule's local file-system view whenever a VCE-capsule was installed to a server.

[0434] Thus, it would have been obvious for Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules to meet *modifying at least some of the system files to define container specific mount points associated with the container*.

[0435] Thus, in my opinion, POSAs would have found claim 7 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

8. Claim 8: "[C]laim 1, wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers."

[0436] Based on Tormasov, each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule comprises *one or more applications and associated system files* and are executed on (and thus *retrieved from*) a "server [*computer*] system" (Tormasov, [0017]) executing "multiple" VCEs (Tormasov, [0027]).

[0437] As I discussed above (§VII.A, §§VII.D.1-2), based on Schmidt-479, VCE-capsules were designed to be suspended and stored for moving to a new server and/or starting a capsule for the first time on a new server or restarting a previously suspended and stored capsule on the same server. Schmidt-479, [0016]. As Schmidt-479's Figure 3 illustrates, each server has a plurality of capsules executing thereon (e.g., two capsules in Figure 3's example). Schmidt-479, [0044]. Thus, when one of those capsules was suspended and stored and started / restarted on a server, that capsule (and the *one or more applications and associated system files* therein) would have been *retrieved from a computer system having a plurality of* capsules (i.e., the original server executing the capsule before it was stored). In the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations, it would have been the VCE-capsules (*secure containers*) that would have been

retrieved from a computer system having a plurality of VCE-capsules (*secure containers*).

[0438] As I discussed above (§§VII.D.1-2, §VII.E.1.a.i), POSAs would have had motivation to implement Schmidt-Tormasov and Schmidt-Tormasov-Calder in a network for a large organization (e.g., large corporation or academic or government institution) that would have conventionally comprised several servers running OSs that differ. Thus, POSAs would have also had reason to retrieve VCE-capsules from a computer system storing the VCE-capsules (*secure containers*).

[0439] In my opinion, POSAs would have recognized that it would have been efficient and conventional (and thus obvious) for those different VCE-capsules installed across a large network to be retrievable from a centralized computer system.

[0440] My opinion that it would have been efficient and conventional to retrieve applications (and thus Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules) from a centralized computer system is further corroborated by EX1084. EX1084 discloses:

> When first directed to server 1200 browser displays a first screen as depicted in FIG. 11A. In this example, **server 1200 stores layers related to the business of "Acme. Inc.", which are made available to persons to the internal Acme local area network**.

EX1084, [0162]-[0166]. Thus, EX1084 discloses an example of applications being stored on a server system for a corporation, where applications were available to other computers in the system (which, in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have been servers).

[0441] My opinion that POSAs would have recognized that it would have been efficient and conventional to retrieve applications (and thus Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules) from a centralized computer system is further corroborated by EX1014. EX1014 discloses as background information a prior art feature in UNIX-based OSs called "TELNET," which "is based on human interface over a communication protocol, allowing remote operation of a machine in a console mode, as if the user was operating the machine locally." EX1014, [0005]. Using a feature like TELNET, "[t]he IT administrator" can "have a full control over the installation procedures from his own console or computer, wherever the remote physical situation of the PC client." EX1014, [0006]. EX1014 further disclose that using a console like that provides "a very simple and effective way for controlling the setup procedure of **a software package**, **stored on a server**, and which are installed within a remote client computer, elsewhere in the LAN." EX1014, [0015].

[0442] My opinion that POSAs would have recognized that it would have been efficient and conventional to retrieve applications (and thus Schmidt-

Tormasov and Schmidt-Tormasov-Calder VCE-capsules) from a centralized computer system is further corroborated by EX1015. EX1015 discloses as background information:

> [M]ore and more individuals are **acquiring software by downloading it from remote server computers** connected to the client computers through the Internet. Additionally, **companies and organizations are distributing software** to their users **across their local area networks**.

EX1015, [0004].

[0443] Indeed, Calder discloses "at least **one server that transmits application packages to the member computers**." Calder, [0075]. That too would have suggested (and provided a reasonable expectation of success in) having Schmidt-Tormasov's and Schmidt-Tormasov-Calder's servers retrieve their layers *from a computing system having a plurality of secure containers* (e.g., a remote server repository for Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules).

[0444] Based on Calder, Schmidt-Tormasov-Calder would have had "at least **one server that transmits application packages [within VCE-capsules] to the member computers**." Calder, [0075]. That would have motivated and suggested to POSAs (and provided a reasonable expectation of success in) having Schmidt-Tormasov-Calder's servers retrieve VCE-capsules (*containers* of applications) *from a computing system having a plurality of secure* VCE-capsules.

[0445] Moreover, Tormasov discloses a server computer executing multiple VCEs. Tormasov, [0017], [0027], FIG. 1. When those VCEs are executing, each VCE (and *the one or more applications and associated system files* therein) are *retrieved from a computer system having a plurality of secure containers* (the server that is executing those VCEs). Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (and *the one or more applications and associated system files* therein) would have similarly been *retrieved from* the server *computer system* executing multiple VCE (and thus also *having a plurality of secure containers*).

[0446] Thus, in my opinion, POSAs would have found claim 8 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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."

[0447] Calder teaches to modify an application package to control" use of computer resources, like "***memory*** usage," "***network bandwidth*** used," and "***disk usage***." Calder, [0125]-[0126]. The "grant[ed] allocation" of resources is "predicated" (i.e., pre-assigned) using "heuristics" (e.g., "amount of virtual memory currently being consumed"). Calder, [0126].

[0448] Thus, POSAs would have understood Calder teaches allocating to any suitable application package *hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* based on that information.

[0449] POSAs would have recognized that these disclosures in Calder could have been beneficially applied to any suitable applications (like VCE-capsules in Schmidt-Tormasov and Schmidt-Tormasov-Calder), with or without applying Calder's technique for modifying application files to execute on non-native OS. (That is, POSAs would have recognized that pre-allocating system resources to one or more applications would have been beneficially functionality whether or not the application package was executable on non-native OSs.)

[0450] Thus, in my opinion, POSAs would have had motivation to apply Calder's resource-allocation teachings (and other conventional resource-allocation functionality discussed below) to Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules, so that *server hardware resource usage* would be allocated to (and thus *associated with*) VCE-capsules (and the applications executing therein) *prior to the applications within the* VCE-capsules (*containers*) *being executed*. As I discussed above (§§VII.E.1.c.i.(2)-(3), §§VII.E.1.h-i), VCE-capsules shared a server's hardware (and other critical) "***resources***" would have been isolated and unaware of competition for those resources. *See, e.g.*, Tormasov,

[0025]-[0026]. That would have reinforced a POSA's motivation to *pre-allocate* resources to VCE-capsules (and the applications installed therein) before executing those applications to ensure that VCE-capsules would have had sufficient resources to execute each of their one or more applications on a server executing multiple VCE-capsules.

[0451] Thus, based on Calder and a POSA's background knowledge, POSAs would have had motivation and a reasonable expectation of success in having Schmidt-Tormasov and Schmidt-Tormasov-Calder allocate to their VCE-capsule applications *hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* that *is associated with at least some of the containers prior to the applications within the containers being executed*, as claimed.

[0452] Moreover, conventional OSs like those that would have been used in Schmidt-Tormasov and Schmidt-Tormasov-Calder (e.g., Unix-based OS) included commands for allocation *server hardware resources prior to executing applications* (which require *hardware resources*). For example, conventional Unix-based OS include the command "quota," which "implements the policy for **restricting the disk capacity** used by each user." EX1132, [0009]-[0011] ("the UNIX quota implements the policy for restricting the disk capacity used by each user").

[0453] My opinion that POSAs would have found it conventional (and thus would have been motivated with a reasonable expectation of success) to track *server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* and that POSAs would have had motivation to *associate* those resources *with at least some of the containers prior to the applications within the containers being executed* to ensure that each VCE-capsule's applications had sufficient resources and/or to allocate resources to a higher-priority application / VCE-capsule is further corroborated by EX1036. EX1036 discloses that:

> For computer systems operating multiple workloads, **it is <u>desirable to specify how computer system resources</u>, such as central processing unit (CPU) resources <u>are allocated</u> to the workloads**. As used herein, a 'workload' refers to any process or group of processes that accesses computer system resources.

EX1036, 1:16-26.

[0454] My opinion that POSAs would have found it conventional (and thus would have been motivated with a reasonable expectation of success) to track *server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* and that POSAs would have had motivation to *associate* those resources *with at least some of the containers prior to the applications within the containers being executed* to ensure that each

VCE-capsule's applications had sufficient resources and/or to allocate resources to a higher-priority application / VCE-capsule is further corroborated by EX1063.[17] EX1063 discloses that:

> Different customers have different virtual server needs. For example, a company A providing large quantities of data and information to its employees and customers will want to ensure that its virtual servers are always available to perform a large number of tasks. Company A may be willing to pay a premium for a guaranteed high quality of service, with high server availability and large amounts of processing power

[17] I note that EX1063 refers to "a virtual server," however POSAs would understand that EX1063 is not referring to a virtual machine (or virtual machine-like) technology, as the physical server in EX1063 is only executing a single operating system and single operating system kernel, and there is no indication or suggestion that there are multiple OS kernels executing on the physical server executing virtual servers. *See, e.g.*, EX1063, 4:41-46 (discussing that "the physical host to request the correct quality of service from **the operating system kernel** of the physical host"). But even if EX1063's virtual servers on one physical server utilized separate OS kernels, EX1063's disclosure quoted below would still corroborate my opinion that it was known to track a physical server's resources and associate them to different applications/environments executed by the same physical server.

always on-call. By contrast, a small individual B who merely uses his virtual server for back-up file storage space has very different quality of service requirements. Customer B needs (and wishes to pay for) only a limited amount of storage space to be available on an intermittent basis.

When servicing the needs of multiple customers having different needs, it is desirable to provide a virtual server that is dynamic, not static, in its allocation of resources. A customer's virtual server is typically assigned a fixed level of resources, corresponding to either a fixed percentage of the capacity of a particular physical host…. However, customers may be unable to anticipate the exact amount of resources they will require, and a static assignment of a particular resource allocation limit may not allow the virtual server system to adapt to changing customer needs.

Instead of requiring customers to select a static level of resources, **a better resource allocation model is structured along the lines of electricity pricing—a customer receives what he needs, and he pays for what he receives**. Referring back to a previous example, small customer B may initially request a very low level of resources. However, should his new home business suddenly expand, he may quickly bump up against the limit of the server resources he originally requested. In this case, it would be preferable if customer B's virtual server resources were able to automatically, dynamically adjust to his increased resource needs.

Thus it is desirable to provide a system and method for a virtual server capable of providing quality of service guarantees for a customer, which is also capable of adjusting the quality of service based upon changing customer demand. **It is desirable for such a system to dynamically adjust the physical host resources allocated to a virtual server**.

SUMMARY OF THE INVENTION

The present invention dynamically adjusts the quality of service guarantees for virtual servers based upon the resource demands experienced by the virtual servers. Virtual servers having individual quality of service guarantees are distributed among a group of physical hosts. Each physical host's resources are allocated among the physical host's resident virtual servers. The resources allocated to a particular virtual server may be dynamically adjusted in response to changing virtual server resource needs.

Occasionally, a physical host executing a virtual server may not have additional resources to allocate to a virtual server requiring increased resources. In this instance, a virtual server hosted by the overloaded physical host is transferred to another physical host with sufficient resources.

In one embodiment, **a dynamic resource configuration module monitors resource denials received by virtual servers and determines if a virtual server is overloaded based upon the resource denials**. A resource denial may refer to any request by the virtual server that cannot be immediately serviced, such as a denial of a request to

create a file or a network packet delay. If the resource denials received by a particular virtual server exceed a pre-specified limit, the virtual server is considered overloaded and a request is made for additional resources.

The resource usage of the physical hosts within the system is monitored. A load-balancing function is performed to select the appropriate physical host when a virtual server transfer becomes necessary. A virtual server is transferred between physical hosts with minimal impact upon the operation of the virtual server.

EX1063, 1:52-2:62.

[0455] My opinion that POSAs would have found it conventional (and thus would have been motivated with a reasonable expectation of success) to track *server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* and that POSAs would have had motivation to *associate* those resources *with at least some of the containers prior to the applications within the containers being executed* to ensure that each VCE-capsule's applications had sufficient resources and/or to allocate resources to a higher-priority application / VCE-capsule is further corroborated by EX1027. EX1027 discloses an example of "error-prevention checks that may be performed by the compartment management utilities," which "**include checking for**

compartment name duplication and checking that sufficient memory is available." EX1027, [0176].

[0456] My opinion that POSAs would have found it conventional (and thus would have been motivated with a reasonable expectation of success) to track *server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* and that POSAs would have had motivation to *associate* those resources *with at least some of the containers prior to the applications within the containers being executed* to ensure that each VCE-capsule's applications had sufficient resources and/or to allocate resources to a higher-priority application / VCE-capsule is further corroborated by EX1037. EX1037 discloses an example where:

> [E]ach user application 170 can be controlled through **explicit <u>allocation of resources</u>** owned by the protection domain, by the partition operating system 160, **<u>to prevent the applications from affecting the operation of the entire system</u>.**

EX1037, 6:29-38.

[0457] EX1076 discloses that "[a]nother area of great interest" for its technique – which uses so-called "jails" to isolate applications, similar to Schmidt-Tormasov's and Schmidt-Tormasov-Calder's isolation of applications in VCE-capsule – "is the ability to **limit the impact of one jail on the CPU resources**

available for other[s]," which "would require that the jail of a process play a rule in its scheduling parameters." EX1076, 13.

[0458] My opinion that POSAs would have found it conventional (and thus would have been motivated with a reasonable expectation of success) to track *server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* and that POSAs would have had motivation to *associate* those resources *with at least some of the containers prior to the applications within the containers being executed* to ensure that each VCE-capsule's applications had sufficient resources and/or to allocate resources to a higher-priority application / VCE-capsule is further corroborated by EX1083. EX1083 discloses:

> Resource Aggregation Units
>
> **To <u>manage the resources of a group of processes</u> in the aggregate**, the group of processes must be transformed into a single unit. The group of processes are transformed into such a unit when a compute capsule is created. Then, the compute capsule is promoted to the status of a "first-class object" within the operating system. A first-class object can have its resources managed, and security policies for its member processes can be implemented. This enables the system to treat groups of processes as single units for accounting and resource management decisions, which has three important effects. First, **resources available anywhere in the system can be directly assigned to a particular capsule**. This may require migrating a capsule between machines,

thereby **dynamically load-balancing the system**. Second, **users may demand a guaranteed level of resources or performance isolation from others. This can be achieved by assigning fixed shares of resources to capsules**. Third, **capsules maybe subjected to policy constraints, such as restrictions or quotas on their resource allocations**, or restrictions on network access, certain file system operations, or resource or server access.

FIG. 1 is a flowchart which describes the **allocation of system resources to compute capsules** according to an embodiment of the present invention. At step 100, a compute capsule is promoted to first-class object status. At step 110, it is determined what resources need to be assigned to the compute capsule. **Resources include memory use, processor attention, or the use of a device, for instance**. **Once the resources that need to be assigned are determined, then it is determined at step 120 if the capsule must migrate to a new machine to receive the needed resources**. If it does, the capsule migrates at step 130, and the required resources are assigned at step 140. If the capsule does not need to migrate (i.e., the needed resources are available on the current machine), then the resources are assigned at step 140.

Resource Aggregation

Once a resource aggregation unit, such as a capsule, is created, various resource management operations can be performed on the unit. FIG. 2 is a flowchart showing the process of aggregating resources to compute capsules according to an embodiment of the present

invention. At step 200, a compute capsule is created. Then, at step 210, the compute capsule is promoted to first class object status. Thereafter, some form of resource management operation is performed on the compute capsule at step 220.

Some examples of the types of resource management operations that can be performed on compute capsules are shown in FIGS. 3-6. FIG. 3 is a flowchart showing how **a guaranteed share of resources can be assigned to a compute capsule according to an embodiment of the present invention**. At step 300, a compute capsule is created. Then, at step 310, the compute capsule is promoted to first class object status. Thereafter, **a guaranteed share of the resources is assigned to the compute capsule at step 320. Resources include, for instance, memory, processor attention, and a plurality of other types of resources as well**.

EX1083, [0035]-[0040].

[0459] My opinion that POSAs would have found it conventional (and thus would have been motivated with a reasonable expectation of success) to track *server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation* and that POSAs would have had motivation to *associate* those resources *with at least some of the containers prior to the applications within the containers being executed* to ensure that each VCE-capsule's applications had sufficient resources and/or to allocate resources to

higher-priority application / VCE-capsules is further corroborated by EX1094. EX1094 discloses an example where:

> **The UNIX operating system keeps track of all programs running in the system and allocates resources, such as disks, memory, and printer queues, as required. UNIX allocates resources so that, ideally, each program receives a fair share of resources to execute properly.** UNIX doles out resources using two methods: scheduling priority and system semaphores. **Each program is assigned a priority level. Higher priority tasks (like reading and writing to the disk) are performed more regularly**. User programs may have their priority adjusted dynamically, upwards or downwards, depending on their activity and the available system resources. System semaphores are used by the operating system to control system resources. A program can be assigned a resource by getting a semaphore by making a system call to the operating system. When the resource is no longer needed, the semaphore is returned to the operating system, which can then allocate it to another program.

EX1094, 2:5-21.

[0460] POSAs would have had a reasonable expectation of success because, as the cited corroboration above demonstrates, pre-allocating resources to applications/processes was conventional by the early-2000s. Doing so in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have required only a POSA's ordinary skill, as it would have required conventional computer-programming skill

to *associate server information with at least some of the* VCE-capsules (*containers*) *prior to the applications within the* VCE-capsules (*containers*) *being executed*.

[0461] Thus, in my opinion, POSAs would have found claim 9 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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."

[0462] Schmidt-479 teaches that "it is **not desirable**" to permit users sharing one computing device "to modify the underlying local system-level files (for which they may not have permission" (e.g., "/lib for system libraries"). Schmidt-479, [0051]-[0052]. To prevent that unwanted modification of local-level files, 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 (*e.g.*, because capsules are prevent from "modify[ing] the underlying local system-level files," Schmidt-479, [0051]-[0052]).

[0463] Similarly, Tormasov teaches another way of preventing applications from having *access to system files or applications in other*

environments. In Tormasov, *application* "processes [*executing* applications] in different" VCEs are "**completely isolated** from each other" (Tormasov, [0026]) and "are **never visible to other**" VCEs "running on the same computer" (Tormasov, [0018]). Similar to Schmidt-479's chroot, Tormasov's VCEs have an "independent" "root file system" comprising respective *system files*, as I discussed above (§VII.E.1.i, [**1G**]). Thus, a VCE's *application has no access to system files or applications in other* VCEs. *See also supra* §VII.E.1.c.i.(3) (where I demonstrate VCE-capsules meet the claimed *secure containers of application software*).

[0464] Thus, Schmidt-479 and Tormasov each teach, or at a minimum suggest and motivate (and thus render obvious), techniques for preventing *applications residing within* VCE-capsules (*containers*) 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. Indeed, if VCE-capsule's "root user" were permitted to access the server OS's system files, it would have contradicted Tormasov's desire for VCEs to have "no[] influence" on each other (Tormasov, [0031]) and to be able to freely stop and start VCEs (Tormasov, [0025]). It would have also contradicted Schmidt-479's statement that it was "not desirable" to allow users "to modify the underlying local ***system*-level *files***" of the *operating system*. Schmidt-479, [0052].

[0465] Additionally, Schmidt-479 and Tormasov separate applications using namespaces. Schmidt-479, [0041]; Tormasov, [0018], [0029]. EX1015 discloses assigning a "'namespace'" to "**isolate** software packages **from one another** even if the same component is used by multiple packages" ([0080]) 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**" ([0086]).

[0466] Accordingly, based on their separation into different namespaces, applications executed within Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have had (or in an obvious implementation thereof would have had) *no access to system files or applications in other* VCE-capsules (*containers*) *or to system files within the operating system during execution thereof*.

[0467] I note that VirtaMove's infringement contentions identify "namespaces" as a procedure for "isolat[ing] processes in an environment." EX1079, 59.

[0468] My opinion that POSAs would have found it conventional and beneficial based on their background knowledge to ensure that applications (such as applications in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules) *have no access to system files or applications in other containers or to*

system files within the operating system during execution thereof is further corroborated by EX1076. EX1076 discloses "tak[ing] advantage of the **existing chroot(2) behaviour to limit access to the file system name-space** for jailed processes" so that applications/processes in a "jail" are "bound to a particular file system root" and "files outside of the jail file system root are protected." EX1076, 5.

[0469] My opinion that POSAs would have found it conventional and beneficial based on their background knowledge to ensure that applications (such as applications in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules) *have no access to system files or applications in other containers or to system files within the operating system during execution thereof* is further corroborated by EX1077. EX1077 discloses a technique for "Sandboxing Applications" so "applications are **run with the chroot operation that limits their access to the FMAC managed filesystem**" and thus "will have access only to special sanitized versions of the system files." EX1077, 120 (explaining that its "File Monitoring and Access Control (FMAC) tool," "implements a filesystem that mirrors the system's existing file system," similar to Tormasov's VCEs and Schmidt-479's capsules).

[0470] Thus, in my opinion, POSAs would have found claim 10 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0471] Though claim 11 refers to *descriptors of containers*, the '814 patent's specification does not describe a *descriptor of* a *container*.

[0472] Schmidt-479 discloses a system where data (e.g., files) used by capsules are *stored in a remote network* (e.g., *in network file storage*). Schmidt-479, [0063]-[0070]. Calder similarly discloses application "files can be ***stored* remotely** on separate machines." Calder, [0139]. Based on those disclosures, POSAs would have had motivation to implement Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) to *include files stored* remotely on separate machines (i.e., *in network file storage*) to provide the well-known benefits of remote file storage (e.g., remote access and sharing of data).

[0473] My opinion that it would have been conventional and beneficial in the prior art to have *files stored in network file storage* is corroborated by EX1092. EX1092 discloses as background information that:

> **Distributed file systems allow networked computers** (referred to as clients) **to access remote *storage* devices as if the devices were on a local file system**. These file systems **allow for sharing of data among**

> **networked clients**. Additionally, a user **can access the networked data** from other networked computers **in the same way she accesses it from her own computer**. **This type of network file sharing <u>is becoming increasingly prevalent</u>** as the computing industry becomes more network centric.

EX1092, 1:21-30.

[0474] Another Schmidt-related reference (Schmidt-629; EX1116) discloses a remote "[c]apsule directory service" that manages and "keep[s] track of" capsule "unique locators" (e.g., their "IP address[es]"). EX1116, [0057]. In my opinion, POSAs would have had reason to store those and other *parameters* of Schmidt-Tormasov and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) in a similar remote directory (e.g., to facilitate management by an administrator).

[0475] POSAs would have also had reason to store *parameters forming descriptors of containers stored in a separate location* in Schmidt-Tormasov and Schmidt-Tormasov-Calder to facilitate retrieval, management, and storage of the VCE-capsules that would have been stored on remote (i.e., separate) network locations, for the reason discussed above. Some obvious parameters that POSAs would have used would have been parameters that identified the name, type, applications within, and storage location of available VCE-capsules.

[0476] Thus, in my opinion, POSAs would have found claim 11 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

12. Claim 12: "[C]laim 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."

[0477] Though claim 12 refers to *merging* and *parameters*, the '814 patent's specification does not use the term "*merging*" and does not explain what it means by "*parameters*" or provide examples thereof.

[0478] As I noted above (§VII.E.8, claim 8), an obvious implementation of Schmidt-Tormasov and Schmidt-Tormasov-Calder would have retrieved VCE-capsules (*containers*) from storage of a network server. POSAs would have also had reason to merge retrieved files with the parameters discussed above for claim 11 (§VII.E.11), *to affect* claim 1's step of storing (§VII.E.1.c.i, [**1A**]). For example, POSAs would have had reason to have the *parameters forming descriptors of containers* (§VII.E.11, claim **11**) include information about where each VCE-capsule should have been stored. The VCE-capsule's name, mount point, and storage device storing the VCE-capsule would have been exemplary parameters that would have been *merged* with *the files stored in network storage to affect the step of storing in claim 1*" (e.g., parameter identifying where the VCE-capsule is being storage).

[0479] Thus, in my opinion, POSAs would have found claim 12 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

13. Claim 13: "[C]laim 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."

[0480] As I discussed above (§§VII.D.1-2, §VII.E.1.a.i), the conventional prior art OSs that would have been used in the servers in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have included Windows-based OS and conventional non-Windows OSs (e.g., Unix-based and Linux-based OSs). POSAs knew those prior art OSs typically and conventionally included tools for measuring/tracking *processes related to an application* being executed by the OS, including *tracking statistics for resource allocation and for monitoring the status of the application(s)*.

[0481] For example, EX1114 (Tormasov-815), which is related to Tormasov and was filed by Tormasov's inventors (plus one additional inventor), teaches as background information (and thus corroborates it was a POSA's 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].

[0482] Similarly, as discussed above (§§VII.D.1-2, §VII.E.1.a.i), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's servers run conventional OSs like Windows- and Unix-based OS, which POSAs knew as part of their background knowledge of the art included tools for measuring/tracking processes related to an application, including tracking statistics for resource allocation and monitoring the status of application(s).

[0483] My opinion that functionality for *storing the history of when processes related to applications are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application* was conventional and beneficially used in the conventional prior art OSs used in Schmidt-Tormasov and Schmidt-Tormasov-Calder is corroborated by EX1033. EX1033 discloses:

> **There exists known performance monitoring software in various forms. Among them is software known as PerfMon software, which is commercially available from Microsoft Corporation. PerfMon software is a utility which, among other things, can provide an indication of the utilization of the computer's central processor unit (CPU) and memory unit**. PerfMon software operates by sampling. That is, it **tracks continuous data by monitoring a machine and looking at its behavior. It can track the free space on a disk, monitor network usage, and so on**, but it cannot gather event-based information, such as what function was most recently started.

There also exist known tools called profilers. These look at a single executing software application and try to understand its performance. They do this either by monitoring the program (in a similar way to PerfMon software), or else they hook into the program they are monitoring and generate "events" each time a program subcomponent (function) commences or completes.

…

The performance analysis component analyzes the collected data and creates a call tree by pairing events (e.g. Call and Return) and ordering them using temporal ordering and heuristics. The result is a presentation of the call tree in a Gantt style view with any Perfmon (or other dynamic) data displayed adjacent to or overlying the displayed call tree.

…

FIG. 19 illustrates this concept. Here, for example, the user made a time selection in the performance view window 556 (representing PerfMon data) over a period of time where CPU behavior was in question. The animated application model or process diagram 552 highlights the entities/processes involved in the selection. **The event log window 554 highlights all events in the specified time range, part of which represent a call tree**.

EX1033, 2:62-3:18, 34:33-39, 36:53-65.

[0484] My opinion that functionality for *storing the history of when processes related to applications are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application* was conventional and beneficially used in the conventional prior art OSs used in Schmidt-Tormasov and Schmidt-Tormasov-Calder is corroborated by EX1055. EX1055 discloses:

> The determination may also use, for example, **a Windows® application program called SYSMON.EXE** which accesses **the Performance Registry of Windows® and allows monitoring of system performance and resources.**

EX1055, 5:35-39.

[0485] My opinion that functionality for *storing the history of when processes related to applications are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application* was conventional and beneficially used in the conventional prior art OSs used in Schmidt-Tormasov and Schmidt-Tormasov-Calder is corroborated by EX1060. EX1060 discloses:

> The number of concurrent clients was gradually increased while the scripts were cycled through the various applications. Thus **multiple test runs were conducted and additional sets of Performance Monitor (PERFMON) log files were produced to verify reproducibility**.

EX1060, 5:57-61.

[0486] My opinion that functionality for *storing the history of when processes related to applications are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application* would have been conventional and beneficially used in the conventional prior art OSs used in Schmidt-Tormasov and Schmidt-Tormasov-Calder is corroborated by EX1094. EX1094 discloses:

> The UNIX operating system keeps track of all programs running in the system and allocates resources, such as disks, memory, and printer queues, as required. UNIX allocates resources so that, ideally, each program receives a fair share of resources to execute properly. UNIX doles out resources using two methods: scheduling priority and system semaphores. Each program is assigned a priority level. Higher priority tasks (like reading and writing to the disk) are performed more regularly. User programs may have their priority adjusted dynamically, upwards or downwards, depending on their activity and the available system resources. System semaphores are used by the operating system to control system resources. A program can be assigned a resource by getting a semaphore by making a system call to the operating system. When the resource is no longer needed, the semaphore is returned to the operating system, which can then allocate it to another program.
>
> …

> UNIX **maintains** a great deal of information about the files that it manages. For each file, the file system **keeps track** of the file's size, location, ownership, security, type, creation time, modification time, and access time. All of this information **is maintained automatically by the file system** as the files are created and used. UNIX file systems reside on mass storage devices such as disk drives and disk arrays. UNIX organizes a disk into a sequence of blocks. These blocks are usually either 512 or 2048 bytes long. The contents of a file are stored in one or more blocks which may be widely scattered on the disk

EX1094, 2:5-21, 4:20-30.

[0487] My opinion that POSAs would have understood functionality for *storing the history of when processes related to applications are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application* to have been conventional and beneficially for conventional prior art OSs is corroborated by EX1120. EX1120 discloses that one system call in the Unix operating system was called "times," which

> returns the elapsed real time in clock ticks from an arbitrary fixed time in the recent past, and fills *tbuf* with account information: [such as] CPU time spend in user mode[.]

EX1120, 450 (emphasis original). Thus, *tbuf* would include a stored history of clock ticks in various modes, which is a *tracking statistic* of when applications in user mode where consuming CPU resources.

[0488] My opinion that that POSAs would have found it to be conventional and beneficial to track and maintain a history of computing resources in the conventional prior art OSs used in Schmidt-Tormasov and Schmidt-Tormasov-Calder is further corroborated by EX1121. EX1121 discloses a definition of "process descriptor" for OSs as:

> A process descriptor is a data structure in an operating system that contains all the information the operating system needs to **record about the process**. This information includes:…how much processor time and **other resources** the process has used….

EX1121, G-14. Thus, EX1121 corroborates my opinion that POSAs would have found it to be conventional for the conventional prior art OSs used in Schmidt-Tormasov and Schmidt-Tormasov-Calder to maintain a *stored history of when processes related to applications are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application.*

[0489] In my opinion, POSAs would have had motivation and a reasonable expectation of success to use those conventional performance-monitoring and tracking tools in Schmidt-Tormasov and Schmidt-Tormasov-Calder to track the history of processes in VCE-capsules for the same reason POSAs had to measure process history in conventional computer systems. For example, POSAs would have recognized the benefit in measuring and maintaining a history of computing resources so as to determine how to allocate computing

resources and make decisions on the purchase of future equipment, maintenance, and support staff.

[0490] My opinion that POSAs would have found it to be conventional and beneficial to track and maintain a history of computing resources is further corroborated by EX1093. EX1093 discloses:

> The present invention relates to the monitoring of server performance in a network like the Internet. **System administrators, as well as information technology (IT) executives, generally like to know the productivity of their systems, and want to accurately forecast future needs for upgrading or expanding existing systems. Capital expenditures must be budgeted according to changes in server resource requirements, based on key server operating parameters such as CPU utilization, memory availability, I/O usage, and permanent storage (disk) capacity**. This objective has become very difficult to achieve, due to extremely complex environments of hundreds or thousands of networked servers running on many different platforms, including UNIX and its variations (such as AIX, HP-UX, System V, and Mach), Windows NT, OS/2, NetWare, etc. (UNIX is a trademark of UNIX System Laboratories; AIX is a trademark of International Business Machines Corp (IBM); HP-UX is a trademark of Hewlett-Packard Co.; System V is a trademark of American Telephone & Telegraph Co.; Mach is a trademark of NeXT Computers; Windows NT is a trademark of Microsoft Corp.; OS/2 is a trademark of IBM; and NetWare is a trademark of Novell Corp.). In addition to these platforms, some servers are managed by specialized systems,

such as SAP (a suite of bundled business software including an accounts receivable program, a materials management program, a scheduling program, etc.).

…

The foregoing objects are achieved in a method of **monitoring a computer network, generally comprising the steps of collecting resource data from a plurality of network nodes, analyzing the resource data to generate historical performance data,** and reporting the historical performance data to another network node. In one implementation, the plurality of network nodes are servers operating on different platforms, and the collecting step includes the step of gathering resource data using separate programs having different application programming interfaces for the respective platforms. The analyzing step can generate daily, weekly, and monthly historical performance data, based on a variety of resources including **CPU utilization, memory availability, I/O usage, and permanent storage capacity.** The report may be constructed using a plurality of documents related by hypertext links. The hypertext links can be color-coded in response to at least one performance parameter in the historical performance data surpassing an associated threshold. An action list can also be created in response to at least one performance parameter in the historical performance data surpassing an associated threshold.

EX1093, 1:66-3:6.

[0491] My opinion that POSAs would have found it to be conventional and beneficial to track and maintain a history of computing resources is further corroborated by EX1121. EX1121 discloses a definition of "efficiency" for OSs as:

> Efficiency is a goal of a resource manager. An efficient use of resources is one where the least amount of the resource is used to accomplish the stated goal. Operating systems worry about efficient use of processor time and memory space.

EX1121, G-5. Thus, EX1121 corroborates my opinion that POSAs would have had motivation (with a reasonable expectation of success) to monitor resources in Schmidt-Tormasov and Schmidt-Tormasov-Calder to develop algorithms for efficient use of resources.

[0492] Thus, in my opinion, POSAs would have found claim 13 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

14. Claim 14

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

[0493] As I discussed above (§VII.E.4, claim 4), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have contained pre-identified applications and system files stored therein.

[0494] Based on Schmidt-479, VCE-capsules would have been "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 would have been *created prior to storing* the capsule *in* the *memory* of one storage device from which it would have, subsequently, been *stored in memory* accessible to a destination server that would have executed the VCE-capsule after retrieving it.

[0495] Similarly, users of Tormasov's VCEs can log-in to a server providing multiple VCEs. Tormasov, FIG. 1, [0017]. For that teaching, POSAs would have understood (or at least have had motivation for) having the VCE-capsules (and applications therein) *created prior to storing in memory* so they would have been available to users when accessing the server. Tormasov further discloses that users "may stop and start the [VCE] in the same manner as with a common operating system" (Tormasov, [0025]), from which POSAs would have understood the VCEs are stored in memory (or at least would have had motivation for that storing), where the VCEs could, on demand, be stored to that memory when they were stopped and retrieved from that memory when they were started.

[0496] Another publication by one or more Tormasov inventors that concerns aspects of implementations of Tormasov's VCE (Tormasov-633; EX1111) similarly discloses "computers with a set of **installed** virtual

environments." EX1111, 5:38-41. That teaching corroborates that POSAs understood VE/VCEs (and applications and system files contained therein) are created before they would have been stored to memory accessible to the server. (At a minimum, EX1111 would have further motivated and provided a reasonable expectation of success in doing that.)

[0497] Thus, POSAs would have found limitation [**14PRE**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

[0498] Calder discloses "transferr[ing]" of pre-packaged applications "from a server to the client." Calder, [0078]. That would have provided additional teaching for POSAs to have any suitable application package, like Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules, *created* and stored *prior to the step of storing* them *in memory* accessible to the servers that would have retrieved the package (VCE-capsules (*containers*) in Schmidt-Tormasov and Schmidt-Tormasov-Calder) from a server.

[0499] That is another reason why POSAs would have found limitation [**14PRE**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0500] To create a capsule, Schmidt-479 *runs* applications processes and related system files and stores them in the capsule, which can be started and/or restarted on the same or a different server. Schmidt-479, [0044]-[0048].

[0501] My opinion that executing a Schmidt-Tormasov and Schmidt-Tormasov-Calder application in a VCE-capsule would have been *running an instance of an* associated *service on a server* is corroborated by EX1040. EX1040 discloses:

> For a suite of ***applications*** developed for a common software platform (e.g., an application server platform), each ***application*** within the suite **typically includes a group of *services*** (e.g., content management ***services***, workflow services, publishing ***services***, search and query services) **that are common amongst the *applications* in the suite**.
>
> As the software platform does not allow **the *services* of one *application* to be shared by another**, these ***services*** must be coded for and included in each ***application*** written for the software platform, even though **multiple *applications* use common *services***.

EX1040, 1:14-25.

[0502] My opinion that executing a Schmidt-Tormasov and Schmidt-Tormasov-Calder application in a VCE-capsule would have been *running an*

instance of an associated *service on a server* is corroborated by EX1099. EX1099 discloses:

> In general, **an application could be viewed as a collection of distributed services**. For example, Hotmail would be an application comprised of multiple services where each service performs a different function.

EX1099, [0104].

[0503] As I discussed above (§VII.E.1.b.ii-iv), applications in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have related to *a service* provided by an application (e.g., an httpd (web-server) service). *See, e.g.*, Tormasov, FIG. 2 (executing various services in VCEs); Schmidt-479, [0010] (examples of applications for execution is capsules include "a word processor, a web browser," or an application for "an interactive multimedia simulation"), [0077] ("Examples of ***services*** include X11/Unix ***services***, archived or live audio or video ***services***, Windows NT ***service***, Java program execution ***service*** and others.").

[0504] Based on Schmidt-479 and/or POSAs' background knowledge, POSAs would have had reason to create Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules by *running an instance of* a VCE-capsule and its application(s) *on a server* before encapsulating them so as, as I explain below (§VII.E.14.c, [**14B**]) to identify the system files required to provide that service,

and to test the VCE-capsule (and the applications therein) to ensure compatibility with various server hardware-OS combinations in the system.

[0505] For example, to create a VCE-capsule for a website application service, POSAs would have had reason to run programs for that service, which, as I explain below (§VII.E.14.c, [**14B**]) would have identified system files required to execute the website-server application(s) associated with that service. Additionally, POSAs would have wanted to create VCE-capsules by *running an instance of a service on a server* in order to test the VCE-capsule (and the applications therein) to ensure their compatibility with each server's hardware configuration and OS / OS-kernel, for the plurality of severs in the system.

[0506] My opinion that it would have been conventional and beneficial (and that POSAs would have had a reasonable expectation of success) to test an application on a target computing platform (like a server) *prior to* storing the application package in long-term storage device memory (e.g., a DVD for distribution to servers) or in a memory accessible to individual servers executing the layer is further corroborated by EX1129. EX1129 discloses:

> Producers of software products **generally test their products extensively to ensure proper operation of the products on several different various operating systems and computer models**.

EX1129, 1:11-20.

[0507] My opinion that it would have been conventional and beneficial (and that POSAs would have had a reasonable expectation of success) to test an application on a target computing platform (like a server) *prior to* storing the application package in long-term storage device memory (e.g., a DVD for distribution to servers) or in a memory accessible to individual servers executing the layer is further corroborated by EX1031. EX1031 discloses:

> Pre Validate Hardware Configuration of Network Element (5):
>
> **There may be dependencies between the software modules to install and the hardware configuration of the network element (2) to upgrade**. E.g. the software may require that a special I/O processor is installed or a higher stepping of the main processor.
>
> **Software/hardware dependencies <u>should be checked prior to initiating</u> any further upgrade activities, since they will ultimately force the upgrade to backtrack**. A step of the recipe is designed to check these software/hardware dependencies for the software modules included in the package distributed.
>
> **The <u>check is performed to assure that the new software is loadable</u> in the current hardware configuration**. The software configuration states the HW compatibility window for the delivered software. The compatibility window will be compared with the installed HW revision.
>
> The check function will return OK/NOK status. **If the check fails, the recipe will not continue unless the operator overrides the result**.

> The operator is warned by the system that overruling of the negative result may lead to traffic disturbances.

EX1031, [0064]-[0068].

[0508] My opinion that it would have been conventional and beneficial (and that POSAs would have had a reasonable expectation of success) to test an application on a target computing platform (like a server) *prior to* storing the application package in long-term storage device memory (e.g., a DVD for distribution to servers) or in a memory accessible to individual servers executing the layer is further corroborated by EX1020. EX1020 discloses:

> For instance, administrators who manage the installation of software on a large number of computer systems **often must keep track of each computer system's hardware configuration <u>to ensure that the hardware configuration is adequate to support the software that is to be installed</u>.**

EX1020, [0007].

[0509] Thus, POSAs would have found limitation [**14A**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0510] As I discussed above (§VIII.E.14.b, [**14A**]), based on Schmidt-479, POSAs would have had reason 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. Identifying what *files are being used* by an application would have also been a conventional way to determine files required to execute the VCE-capsule and its application(s). Doing that *determines which files are being used* by the application, as claimed.

[0511] Thus, POSAs would have found limitation [**14B**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

[0512] In addition, POSAs would have had additional motivations to create VCE-capsules by first *running (e.g., using) an instance of a service on a server* to *determine* the system files *being used* and thus determine which *system files* were needed to execute the VCE-capsule's application(s).

[0513] My opinion that it was would have been obvious to POSAs with a reasonable expectation of success to isolate applications (like Schmidt-Tormasov and Schmidt-Tormasov-Calder would have done with their VCE-capsules) by having copies or modified copies of the *system files* required to run the applications is corroborated by EX1077, which discloses a technique for "Sandboxing Applications" in a chroot environment – much like Schmidt-479, Tormasov, and Calder – by "[r]unning the application" to identify required system files:

[O]ur approach for sandboxing applications is based on the following steps:

1. **Run the application** with known benign input to create access logs **specifying its file access behavior.**

2. Use the logs to create **an access list** that can be morphed into a typical chroot environment (like the one used by ftpd). The user can augment this list based on the particular requirements of the application.

3. Create the sandbox as a chroot environment based on the access list.

4. Run the application with untrusted input in the sandboxed environment thus denying it access to unauthorized files.

…

To support the approach we outlined above, we designed the File Monitoring and Access Control (FMAC) tool. … Initially **applications are run** with FMAC in passive mode to create the access log (step 1 of our approach).

…

We ran Netscape Communicator version 4.75 with the FMAC system in passive mode and we extracted, using FMAC in passive mode, **the file hierarchy shown in Figure 1**.

We separated the files into two categories, ***system* files** that are common to all installations (e.g. the /Netscape hierarchy) and *user* files that are personal files

accessible by the user. The former category is typically read-only and consists of files that are not owned by the user. These are the files that will be handled by the access control part of FMAC as we will see later on:

```
/bin/
      /sh
  /dev/
      ...
  /etc/
      /group
      /kerberosIV
              /krb.conf
      /localtime
      /login.conf
      /networks
      /pwd.db
      /resolv.conf
      /spwd.db
  /usr/
      /X11R6
          /bin
          /lib
              /X11
                  /XKeysymDB
                  /app-defaults
                  /locale
                      ...
      /bin
          /su
      /lib
          ...
      /libexec
          /ld.so
      /local
          /netscape
                  /netscape
      /share
          /zoneinfo
                 ...
  /users
      /bob
          ...
  /var/
      /X11
          /app-defaults
      /db
          /kvm_bsd.db
      /mail
              /bob
      /run
          /ld.so.hints
```

Figure 1: Files required by the Netscape Communicator sandbox

…

If we do not wish to use the FMAC tool for the active phase, we will **need to create an actual file hierarchy by <u>copying</u> all the files included in Figure 1 to another part of the filesystem**. We would then run the chroot command as in the earlier example.

This approach relies only on the standard system tools for the active phase. **It also allows the sandbox environment to be moved to other similar platforms with little or no customization**. Thus, in a company environment we **can create the Netscape sandbox in one machine and then <u>copy</u> it to the machines of all the other employees**.

EX1077, 120-122 (italicized emphasis original).

[0514] The above is an additional reason why POSAs would have found limitation [**14B**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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."

[0515] As discussed above (§VII.E.1.c.i.(2), §§VII.E.1.c.ii-iii), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have included 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). *See* Schmidt-479, [0050]-[0053] (describing a "***copy*-on-write** mode" where files in the filesystem, which would include *system files*, are copied into a capsule whenever they were modified by the capsule); EX1077, 120-122 (which I excerpted and annotated above). Each encapsulated VCE (and applications and system files therein) would have been a *second instance* of that data beyond the computer's local first instance or another VCE's instance, as claimed. *See also* EX1079, 67 (VirtaMove arguing claim is met by storing "different instance[s]" of the alleged *containers*).

[0516] Thus, POSAs would have found limitation [**14C**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

* * *

[0517] Thus, for the reasons discussed above (§§VII.E.14.a-d), in my opinion, POSAs would have found claim 14 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

15. Claim 15

a. [15A] "[C]laim 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;"

[0518] Schmidt-Tormasov and Schmidt-Tormasov-Calder meet limitation [**15A**] for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder meet claim 14 (§VII.E.14) and claim 6 (§VII.E.6).

[0519] Thus, in my opinion, POSAs would have found limitation [**15A**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0520] As I discussed above (§VII.E.1.a.i, §VII.E.1.c.i), Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would have been *installed on* "***server*** system[s]." Tormasov, [0017]; Schmidt-479, [0065]-[0067]; Calder, [0073]-[0078].

[0521] Thus, in my opinion, POSAs would have found limitation [**15B**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0522] In my opinion, POSAs would have had reason to *test the applications and files within* each Schmidt-Tormasov and Schmidt-Tormasov-

Calder VCE-capsule (*container*) to ensure the proper operation of each VCE-capsule (and application(s) therein) on the several different various operating systems and computer models that would have been present in the system.

[0523] My opinion that it would have been conventional to test the *applications and files within* each VCE-capsule (which would have been implemented in various server devices running different OSs) is also corroborated by EX1129. EX1129 discloses as background information that:

> Producers of software products **generally test their products extensively to ensure proper operation of the products on several different various operating systems and computer models**.

EX1129, 1:11-20.

[0524] Thus, in my opinion, POSAs would have found limitation [**15C**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

* * *

[0525] Thus, for the reasons discussed above (§§VII.E.15.a-c), in my opinion, POSAs would have found claim 15 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

16. Claim 16

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

[0526] Schmidt-Tormasov and Schmidt-Tormasov-Calder meet limitation [**16A**] for the same reason Schmidt-Tormasov and Schmidt-Tormasov-Calder meet limitation [**14A**] (*supra* §VII.E.14.b).

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

[0527] The '814 patent describes a "skeleton" file system as "the OS provided system files before an application is installed." EX1001, 10:41-47.

[0528] Schmidt-479 can create capsule files using a "'copy-on-write mode,'" where capsules "modify 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 would serve *as a* VCE-capsule (*container*) *starting point* when creating VCE-capsules. To prevent conflicts between *system files* in different VCE-capsules, POSAs also had reason to use *a skeleton set of system files* to serve *as a container starting point* when "capturing" applications so the skeleton files would be *installed into that set of files* to avoid creating conflicts with the server's base OS's local copy of those *system files*.

[0529] My opinion in that regard is corroborated by EX1077. As I discussed above (§§VII.E.1.g-h), EX1077 discloses a technique to reduce security risks by sandboxing applications with "all the files needed for executing the application" by "copying all the files included in Figure 1" of EX1077 – which were application system files – "to another part of the filesystem." EX1077, 119-122. Thus, *system files* from which those copies were made served as *a skeleton of system files* to use as a *starting point* for *installing applications into that set of files* to determine which files were needed to run the application in a sandboxed/chroot'd environment (like those that would have been used in Schmidt-Tormasov and Schmidt-Tormasov-Calder, *see supra* §VII.E.1.i).

[0530] Thus, POSAs would have found limitation [**16B**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

* * *

[0531] Thus, for the reasons discussed above (§§VII.E.16.a-b), in my opinion, POSAs would have found claim 16 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

17. Claim 17

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

[0532] As I discussed above (§VII.E.1.b.ii), Schmidt-Tormasov and Schmidt-Tormasov-Calder would have *installed a service* (e.g., an httpd (web-server)) *on a target server selected from one of the* system's *plurality of servers*. Tormasov, FIG. 1.

[0533] POSAs would have also had reason to *install a service on a target server* to make it available to computers accessing the server, which was conventional by the early-2000s.

[0534] POSAs would have had a reasonable expectation of success because installing services on a server was conventional by the early-2000s.

[0535] My opinion that it would have been conventional and beneficial to install applications and related services on a server by selecting the *target server* for the installation is corroborated by EX1014. EX1014 disclose an example where:

> The IT manager is therefore given **a very simple and effective way for controlling the setup procedure of a software package, stored on a server, and which are installed within a remote client computer, elsewhere in the LAN**.

EX1014, [0015].

[0536] My opinion that it would have been conventional and beneficial to install applications and related services on a server by selecting the *target server* for the installation is corroborated by EX1020. EX1020 disclose:

> In FIG. 3, **from the deployment console 110 the user requests from the IPS 120 a list of all the images in the image database and a list of all installation engines known to the IPS**. The installation engines known to the IPS are maintained in a database that is accessible to the IPS. **The information stored in the database that is associated with a particular installation engine includes, but is not limited to, the hostname and the network address, such as IP address, of the intelligent installation server at the installation engine**. In one embodiment of the invention, the database is created by manually entering the information for each installation engine into the database.

EX1020, [0069]-[0078].

[0537] Thus, POSAs would have found limitation [**17A**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

b. [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."

[0538] Tormasov discloses that its VCEs and associated software provide a way 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].

[0539] Thus, POSAs would have had reason 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 GUIs (e.g., supporting drag-and-drop installation operations using icons).

[0540] My opinion that it was known to use a GUI having icons associated with a service (e.g., an application for a service) and icons associated with a computing device for hosting the applications and executing the service to cause an application to be distributed to and installed to the server is further corroborated by EX1014. EX1014 discloses a GUI showing the different "clients comprised within the network" and "the list of the different software packages which are present and available on server [] and the distribution of those between the different PC clients." EX1014, [0030]. EX1014 further discloses using the GUI to install to computers a "software package" residing on a server "[b]y dragging an icon corresponding to one software, and dropping it onto the visual icon representative of one PC client." EX1014, [0031].

[0541] My opinion that it was known to use a GUI having icons associated with a service (e.g., an application for a service) and icons associated with a computing device for hosting the applications and executing the service to cause an application to be distributed to and installed to the server is further corroborated by EX1047. EX1047 discloses:

> **In graphical user interfaces employing an object-oriented programming paradigm, application programs are typically represented to a user by an icon displayed within a window on a computer screen, one icon for each application program that can be run. Execution of an application program is initiated by selecting its corresponding icon**, most often using a pointing device such as a mouse. When an application program is selected, a message is sent to the corresponding application program object, indicating that the application program object is to invoke certain of its methods. For example, if a word processing program is selected, the methods contained within the application program object may include starting the word processing program. The user may also "drag" icons from one area of the screen to another, or from one window to another using the mouse. The user may even "drag" one icon representative of an application object and "drop" that icon on top of another. This "object-object" interaction will result in a combination of application objects. For example, if a word processing document icon is dropped upon a word processing program icon, the object-object interaction results in starting the word processing program and causing that program to open the word processing document. This is possible because both the word

processing program and the word processing document have been represented as compatible application program objects. **Hence, the icons in the object-oriented programming paradigm allow the user to graphically control various computer system elements and the interrelationships between computer system elements.**

EX1047, 1:39-67.

[0542] My opinion that it was known to use a GUI having icons associated with a service (e.g., an application for a service) and icons associated with a computing device for hosting the applications and executing the service to cause an application to be distributed to and installed to the server is further corroborated by EX1020. EX1020 discloses:

> **The deployment console provides a user interface to the overall software duplication and installation process**. From the deployment console, **a user can select a source computer system, create an image of that source computer system, deploy that image to an installation engine, select a plurality of target computer systems associated with the installation engine and direct the installation engine to install a copy of the software system, represented in the deployed image, onto each of the selected target computer systems**.
>
> …
>
> The deployment console is a software interface that interfaces with the IPS. **This software interface may be a Graphical User Interface (GUI), such as Win32, a web browser, or other form of interface**.

> In the case of a GUI, the interface runs on a computer system and preferably connects over a network to the IPS using a socket interface. In the case of a browser, the browser runs on a computer system and preferably acts as a web client that connects over a network to a web server that is part of the IPS.

EX1020, [0024], [0046].

[0543] My opinion that it was known to use a GUI having icons associated with a service (e.g., an application for a service) and icons associated with a computing device for hosting the applications and executing the service to cause an application to be distributed to and installed to the server is further corroborated by EX1109. EX1109 discloses:

> **Generally described, computer networks can include a collection of a number of computers, such as server computer systems**, to perform various functions. In one application, a large computer network can include more than one thousand networked server computers. Such a computer network is generally referred to as an enterprise network.
>
> In some typical enterprise networks, several of the server computer systems may be remote from one another. For example, a corporation may maintain an enterprise network in which a set of server computer systems are located in North America, a set of server computer systems are located in Asia and another set of server computer systems are located in Europe. In one aspect, a system administrator can manage

each individual server computer by utilizing an attached display and keyboard/mouse. **However, as will be easily understood by one skilled in the relevant art, managing an enterprise network by individually manipulating each server is extremely inefficient, if not impossible**. For example, it would be very difficult for a single system administrator to implement a simultaneous change requiring a manipulation of each server's input devices on all multiple server computers within an enterprise network, **especially if the server computers were remote from one another**.

To mitigate the need for individually manipulating each individual server computer, some enterprise networks utilize a centralized control to remotely manage each individual server in the enterprise network. For example, some enterprise networks utilize a graphical control in which individual servers are managed by manipulating a graphical user interface ("GUI"). In such an embodiment, each server is represented as individual graphical icons within the GUI. Accordingly, the individual server icon may be selected by the system administrator to display various server properties and/or to allow the system administrator to mange [sic] server from the centralized control.

While a common user interface facilitates centralized control of a group of servers, the conventional approach remains deficient for situations in which a common action is to be implemented in a number of servers within the enterprise network. For example, a system administrator may wish to initialize one or more capacity planning measures in each of the database servers within an enterprise network. In the conventional

> centralized control GUI, the system administrator would then have to individually select each of the database servers and then issue and activate the measurement through the centralized user interface. Additionally, if the system administrator would want to slightly modify the measurement or repeat the same set of capacity planning measurement at a different time, the entire initialization process may have to be repeated for each of the servers. Accordingly, this deficiency is further accentuated for larger scale enterprise networks.

EX1109, [0002]-[0005].

[0544] POSAs would have reasonably expected success because, as demonstrated above, using GUIs to configure servers was conventional and Tormasov includes functionality for "administration of a remote node on a data storage network." Tormasov, [0011]-[0012].

[0545] Thus, POSAs would have found limitation [**17B**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

* * *

[0546] Thus, for the reasons discussed above (§§VII.E.17.a-b), in my opinion, POSAs would have found claim 17 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

18. Claim 18: "[C]laim 17 wherein the target server and the graphical user interface are at remote locations."

[0547] Tormasov teaches to use its VCEs and associated software to provide a way to reduce "time and effort setting up and administering" software packages on a "remote node." Tormasov, [0011]-[0012]. In my opinion, that teaching would have suggested and motivated to POSAs to allow, and a reasonable expectation of success in allowing, system administrators at remote locations (e.g., corporate/IT headquarters) to configure target servers (*e.g.*, in different cities) in the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations using a GUI.

[0548] POSAs would have recognized that doing so would have beneficially allowed system administrators to configure and maintain servers remotely, rather than require administrators to present at each target server, which could be located in different cities, states, or countries in a large organization like the large corporations that Schmidt-Tormasov and Schmidt-Tormasov-Calder would have been implemented in (*see supra* §§VII.D.1-2, §VII.E.1.a.i).

[0549] My opinion that POSAs would have found it conventional and beneficial based on their background knowledge to configure and install applications on target computing devices using a GUI at a remote location is corroborated by EX1014. EX1014 discloses using a GUI showing the different "clients [computers] comprised with the network" and "the list of the different

software packages which are present and available on server [] and the distribution of those between the different PC clients. EX1014, [0030].

[0550] My opinion that POSAs would have found it conventional and beneficial based on their background knowledge to configure and install applications on target computing devices using a GUI at a remote location is corroborated by EX1020. EX1020 discloses:

> **The deployment console provides a user interface to the overall software duplication and installation process**. From the deployment console, **a user can select a source computer system, create an image of that source computer system, deploy that image to an installation engine, select a plurality of target computer systems associated with the installation engine and direct the installation engine to install a copy of the software system, represented in the deployed image, onto each of the selected target computer systems**.
>
> …
>
> The deployment console is a software interface that interfaces with the IPS. This software interface may be a Graphical User Interface (GUI), such as Win32, a web browser, or other form of interface. In the case of a GUI, the interface runs on a computer system and preferably connects over a network to the IPS using a socket interface. In the case of a browser, the browser runs on a computer system and preferably acts as a web client that connects over a network to a web server that is part of the IPS.

…

One benefit of the present invention is that it enables the duplication of a software system onto a number of target computer systems without having to manually install each piece of software that comprises the software system on each target computer system.

EX1020, [0024], [0046], [0108].

[0551] My opinion that POSAs would have found it conventional and beneficial based on their background knowledge to configure and install applications on target computing devices using a GUI at a remote location is corroborated by EX1109. EX1109 discloses:

Generally described, computer networks can include a collection of a number of computers, such as server computer systems, to perform various functions. In one application, a large computer network can include more than one thousand networked server computers. Such a computer network is generally referred to as an enterprise network.

In some typical enterprise networks, **several of the server computer systems may be remote from one another**. For example, a corporation may maintain an enterprise network in which a set of server computer systems are located in North America, a set of server computer systems are located in Asia and another set of server computer systems are located in Europe. In one aspect, a system administrator can manage

each individual server computer by utilizing an attached display and keyboard/mouse. **However, as will be easily understood by one skilled in the relevant art, managing an enterprise network by individually manipulating each server is extremely inefficient, if not impossible**. For example, it would be very difficult for a single system administrator to implement a simultaneous change requiring a manipulation of each server's input devices on all multiple server computers within an enterprise network, **especially if the server computers were remote from one another**.

To mitigate the need for individually manipulating each individual server computer, some enterprise networks utilize a centralized control to remotely manage each individual server in the enterprise network. For example, some enterprise networks utilize a graphical control in which individual servers are managed by manipulating a graphical user interface ("GUI"). In such an embodiment, each server is represented as individual graphical icons within the GUI. Accordingly, the individual server icon may be selected by the system administrator to display various server properties and/or to allow the system administrator to mange [sic] server from the centralized control.

While a common user interface facilitates centralized control of a group of servers, the conventional approach remains deficient for situations in which a common action is to be implemented in a number of servers within the enterprise network. For example, a system administrator may wish to initialize one or more capacity planning measures in each of the database servers within an enterprise network. In the conventional

> centralized control GUI, the system administrator would then have to individually select each of the database servers and then issue and activate the measurement through the centralized user interface. Additionally, if the system administrator would want to slightly modify the measurement or repeat the same set of capacity planning measurement at a different time, the entire initialization process may have to be repeated for each of the servers. Accordingly, this deficiency is further accentuated for larger scale enterprise networks.

EX1109, [0002]-[0005].

[0552] Thus, in my opinion, POSAs would have found claim 18 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

19. Claim 19: "[C]laim 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."

[0553] 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

[0554] If a *different computing platform* includes a different physical computer, then Schmidt-Tormasov and Schmidt-Tormasov-Calder meet claim 19 in the same way they meet claim 18, as I discussed (§VII.E.18).

[0555] If *different computing platform* means the platform and target server have different OSs and/or hardware, POSAs would have had reason to do so because large organizations like those that would have used Schmidt-Tormasov and Schmidt-Tormasov-Calder's VCE-capsules typically had multiple OS-families and/or OS-versions on their servers, which often had different hardware configurations. *See supra* §§VII.D.1-2, §VII.E.1.a.i (where I discuss POSAs reason to have servers with different OSs in the Schmidt-Tormasov and Schmidt-Tormasov-Calder systems).

[0556] My opinion that POSAs would have found it conventional and beneficial based on their background knowledge to configure and install applications on target computing devices using a GUI installed on a computing platform that is a different computing platform from the target server is corroborated by EX1020. EX1020 discloses:

> **The deployment console provides a user interface to the overall software duplication and installation process**. From the deployment console, **a user can select a source computer system, create an image of that source computer system, deploy that image to an installation engine, select a plurality of target computer systems associated with the installation engine and direct the installation engine to install a copy of the software system, represented in the deployed image, onto each of the selected target computer systems**.
>
> …

> The deployment console is a software interface that interfaces with the IPS. **This software interface may be a Graphical User Interface (GUI), such as Win32, a web browser, or other form of interface**. In the case of a GUI, the interface runs on a computer system and preferably connects over a network to the IPS using a socket interface. In the case of a browser, the browser runs on a computer system and preferably acts as a web client that connects over a network to a web server that is part of the IPS.

EX1020, [0024], [0046]. Because the GUI discussed in EX1020 can be a web-browser-based GUI, POSAs understood that the OS running on the computing device with the installation GUI can be a different OS than the target computing device.

[0557] Thus, in my opinion, POSAs would have found claim 19 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

20. Claim 20: "[C]laim 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."

[0558] In my opinion, POSAs would have had motivation for limitation [**17B**]'s *associating* step (*supra* §VII.E.17.b)– associating a unique icon representing a service with an unique icon representing a server – to include *relatively moving the unique icon representing the service to the unique icon*

representing a server because that would have been a conventional, convenient drag-and-drop type method of using a GUI.

[0559] My opinion that conventional GUI functionality included drag-and-drop operations to install software to a computer based on *relatively moving* one icon to another icon is further corroborated by EX1014. EX1014 discloses using a GUI to install on computers a "software package" that is residing on a server "[b]y dragging an icon corresponding to one software, and dropping it onto the visual icon representative of one PC client." EX1014, [0031].

[0560] My opinion that conventional GUI functionality included drag-and-drop operations to install software to a computer based on *relatively moving* one icon to another icon is further corroborated by EX1047. EX1047 discloses:

> **In graphical user interfaces employing an object-oriented programming paradigm, application programs are typically represented to a user by an icon displayed within a window on a computer screen, one icon for each application program that can be run**. Execution of an application program is initiated by selecting its corresponding icon, most often using a pointing device such as a mouse. When an application program is selected, a message is sent to the corresponding application program object, indicating that the application program object is to invoke certain of its methods. For example, if a word processing program is selected, the methods contained within the application program object may include starting the word processing program. The user may also "drag" icons from one

area of the screen to another, or from one window to another using the mouse. **The user may even "drag" one icon representative of an application object and "drop" that icon on top of another. This "object-object" interaction will result in a combination of application objects.** For example, if a word processing document icon is dropped upon a word processing program icon, the object-object interaction results in starting the word processing program and causing that program to open the word processing document. This is possible because both the word processing program and the word processing document have been represented as compatible application program objects. **Hence, the icons in the object-oriented programming paradigm allow the user to graphically control various computer system elements and the interrelationships between computer system elements**.

EX1047, 1:39-67.

[0561] My opinion that conventional GUI functionality included drag-and-drop operations to install software to a computer based on *relatively moving* one icon to another icon is further corroborated by EX1109. EX1109 discloses:

To complete the initiation, the user manipulates the desired action icon with a graphical interface device, such as a mouse, for the purpose of depositing it on the selected server icon. Specifically, the user utilizes a "drag and drop" technique in which the action icon is selected, moved across the screen display until it overlaps the desired server group and then deselected. FIG. 8 is exemplary of the screen display 120 of FIG. 7 illustrating the graphical manipulation of

> the action icon 126 to a selected server grouping icon. **Drag and drop techniques are well known in the relevant art and will not be discussed in further detail. One skilled in the relevant art will appreciate that alternative methods for selecting a server icon may be practiced with the present invention, including the selection of a server icon and drag and dropping it on the selected action icon**.

EX1109, [0048].

[0562] Thus, in my opinion, POSAs would have found claim 20 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

21. Claim 21: "[C]laim 20 further comprising starting a distributed software application."

[0563] Claim 21 refers to a *distributed software application*. A conventional *distributed software application* is an application that is "spread over several computers that are connected via a network." EX1025, [0038].

[0564] As I discussed above (§VII.B, §§VII.E.1.b.i-ii), Tormasov discloses an example of running an httpd application in VCE (*see* Tormasov, FIG. 1). POSAs understood that httpd – which indicates a Hypertext Transfer Protocol Daemon program running on a server that acts a permanent server process that serves web content in response to requests for that content – is an example of a *distributed software applications* as there are multiple clients accessing the web server. *See* EX1027, [0185] (discussing the "web server daemon" "httpd");

EX1034, [0059] (noting that servers and clients "communicating with one another over a communications medium" "over a remote network such as the Internet" is commonly "providing ***distributed*** functionality").

[0565] In my opinion, POSAs would have also included other conventional distributed software applications in one or more of Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsule as a common, beneficial implementation of the types of applications that would have been used in those system (e.g., web-server and ftp-server applications).

[0566] My opinion in that regard is corroborated by EX1034, which discloses existing "Distributed Computing Frameworks or Architectures" for ***distributed software applications***:

> Exemplary ***Distributed*** **Computing Frameworks or Architectures**
>
> **Various *distributed* computing frameworks have been and are being developed in light of the convergence of personal computing and the Internet**. Individuals and business users alike are provided with a seamlessly interoperable and **Web-enabled interface for *applications* and computing devices, making computing activities increasingly Web browser or network-oriented**.
>
> For example, **MICROSOFT®'s .Net platform includes servers, building-block services, such as Web-based data storage and downloadable device software**. Generally speaking, the .Net platform provides (1) the ability to make the entire range of computing devices

> work together and to have user information automatically updated and synchronized on all of them, (2) increased interactive capability for Web sites, enabled by greater use of XML rather than HTML, (3) online services that feature customized access and delivery of products and services to the user from a central starting point for the management of various applications, such as e-mail, for example, or software, such as Office .Net, (4) centralized data storage, which will increase efficiency and ease of access to information, as well as synchronization of information among users and devices, (5) the ability to integrate various communications media, such as e-mail, faxes, and telephones, (6) for developers, the ability to create reusable modules, thereby increasing productivity and reducing the number of programming errors and (7) many other cross-platform integration features as well.

EX1034, [0074]-[0076].

[0567] In my opinion, POSAs would have also had reason to include a *distributed software application* in Schmidt-Tormasov and Schmidt-Tormasov-Calder because using a *distributed software application* would have been a common, beneficial implementation for applications in computer systems like Schmidt-Tormasov and Schmidt-Tormasov-Calder, which would have had multiple servers and client computers, as I discussed above (§§VII.D.1-2, §VII.E.1.a.i).

[0568] Calder techniques "relate[] to ***distributed*** **computing**" ([0002]), with files "stored remotely **on separate machines**," where calls to a local "kernel

to perform [a] file operation" are "communicated over the network 130 to another computer or the server 120" ([0139]). Based on Calder's teachings, POSAs would have had further motivation (and a reasonable expectation of success) to include starting a *distributed software application* in a Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule.

[0569] My opinion that *starting a distributed software application* was conventional and beneficial is further corroborated by EX1025. EX1025 discloses

> **"Distributed" is a description of a software system architecture that locates components of a software application on separate, but networked computers.** This is in contrast to having all the components of a software application residing on the same computer. **A distributed software application is spread over several computers that are connected via a network**.

EX1025, [0038].

[0570] My opinion that *starting a distributed software application* was conventional and beneficial is further corroborated by EX1028. EX1028 discloses that "**[i]t is common** modern software practice **to develop *software applications* which are *distributed***." EX1028, [0005].

[0571] My opinion that *starting a distributed software application* was conventional and beneficial is further corroborated by EX1014. EX1014 disclose

installing a "software package [that] may be located in different locations or shared resources within the NT domain." EX1014, [0046].

[0572] Thus, in my opinion, POSAs would have found claim 21 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0573] Claim 22 refers to *updating a console*. The '814 patent refers to using a console but does not explain what it means by "console" or illustrate an example of a console. Thus, POSAs understood the patent to be relying on a conventional, prior art console. POSA knew that "**[c]omputing devices typically include a *console*** that is used **to control the computing device** manually…and provide communications…between an operator and the…operating system." EX1022, [0004].

[0574] EX1022's ***console*** teaches (or at least suggests) a so-called graphical console, which is also commonly referred to as a dashboard, that conventional *console*s for conventional server OSs would have had *resident on the server* and *updated* when a *service* was installed, activated, and/or updated. My opinion in that regard is further corroborated by EX1032, which discloses an example of a "***console*** application," which "allows network administrators to

configure network services engines [] network services switches [] and, thus, operation of web services network []":

> The ***console*** **application facilitates configuration of all data required for routing, security and user management**. Network services switches 56 depend on instances of network services engine 52, 54 for routing table and other associated data. Network services switches also depend on engines 52, 54 to provide functionality allowing for maintenance, configuration and support of web services network 40. For example, embodiments of network services switch 56 do not store routing table data in persistent memory, but depend on network services engine 52, 54 to provide this information on demand and/or by push updates, as required. Accordingly, configuration of network services switch 56 entails providing a computer network address of a network services engine 54 with which it communicates for routing table and other configuration data

EX1032, [0025].

> **The *console* application provides an interface facilitating access to functionality for network administrators and other users of the web services network** enabled by the present invention. For example, console application 90 provides the user interface through which web services are configured, registered, and discovered, by which access to those services may be controlled, by which network services engines 54 and switches 56 are configured, and by which the user rights of those participating in the network may be administered. In one embodiment,

the *console* application further provides access to service activity reporting capabilities and customer service functionality.

Console application 90 presents the user interface by which authenticated end-users interact with and/or configure the web services network of the present invention. In one embodiment, ***console*** application 90 is a web-based application **executed on a *server*** or other computing device operably connected to the computer network environment. In another embodiment, the ***console*** application may reside on a desktop computer 22 associated with a network administrator. As discussed above, the functionality allowing for maintenance, configuration, support and other administrative tasks associated with web services network 40 is, in one embodiment, available as web services residing on one or more instances of network services engines 52 and/or 54. In one embodiment, a set of SNMP interfaces allow access to such administrative functionality without the use of ***console*** application 90. To implement an end-user command, ***console*** application 90 generates an appropriate service action request and transmits it to a network services engine 52 or 54. ***Console*** application 90 is further operative to allow for configuration of all nodes associated with a web services network concurrently or individually.

EX1032, [0032]-[0033].

[0575] Thus, POSAs would have had motivation to use a conventional *console* in Schmidt-Tormasov and Schmidt-Tormasov-Calder so that their systems

would have had the beneficial capability to *update a console on the selected target server with information indicating that the service is resident on the selected target server*, as claimed, in order to configure the server's console to recognize and properly execute the service installed and executing thereon.

[0576] Thus, in my opinion, POSAs would have found claim 22 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

23. Claim 23: "[C]laim 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."

[0577] In my opinion, POSAs would have had motivation (and thus it would have been obvious) for Schmidt-Tormasov and Schmidt-Tormasov-Calder 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*. POSAs would have wanted to test the target server before distribution and installation of software to the server so as to avoid wasting time and resources and potentially damage servers by attempting to install VCE-capsules that were incompatible with the target server and/or to target servers that did not have sufficient resources (e.g., memory, storage space, etc.) to support the VCE-capsule's applications.

[0578] My opinion that it would have been conventional and beneficial to test a target computing platform (like a server) *prior to distributing and installing* the application software on the computing platform is further corroborated by EX1044. EX1044 discloses remote-installation of software programs to a computing device but first "determin[ing] whether the hardware of the remote computer system is capable of supporting the remote installation" and "whether sufficient disk space exists on the remote computer system for the installation." EX1044, 2:25-32.

[0579] My opinion that it would have been conventional and beneficial to test a target computing platform (like a server) *prior to distributing and installing* the application software on the computing platform is further corroborated by EX1031. EX1031 discloses:

> Pre Validate Hardware Configuration of Network Element (5):
>
> **There may be dependencies between the software modules to install and the hardware configuration of the network element (2) to upgrade**. E.g. the software may require that a special I/O processor is installed or a higher stepping of the main processor.
>
> **Software/hardware dependencies <u>should be checked prior to initiating</u> any further upgrade activities, since they will ultimately force the upgrade to backtrack**. A step of the recipe is designed to check these software/hardware dependencies for the software modules included in the package distributed.

> **The check is performed to assure that the new software is loadable in the current hardware configuration**. The software configuration states the HW compatibility window for the delivered software. The compatibility window will be compared with the installed HW revision.
>
> The check function will return OK/NOK status. **If the check fails, the recipe will not continue unless the operator overrides the result**. The operator is warned by the system that overruling of the negative result may lead to traffic disturbances.

EX1031, [0064]-[0068].

[0580] My opinion that it would have been conventional and beneficial to test a target computing platform (like a server) *prior to distributing and installing* the application software on the computing platform is further corroborated by EX1020. EX1020 discloses:

> For instance, administrators who manage the installation of software on a large number of computer systems **often must keep track of each computer system's hardware configuration to ensure that the hardware configuration is adequate to support the software that is to be installed.**

EX1020, [0007].

[0581] Thus, in my opinion, POSAs would have found claim 23 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0582] Tormasov permits "several different ***users***" (Tormasov, [0024]) to "connect[] to a server system" to execute VCEs that are related to *the service* (e.g., an httpd (web-server) *service*), as I discussed above (§VII.B, §§VII.E.1.b.ii-iii).

[0583] To facilitate system-administration tasks that Tormasov describes, POSAs would have had reason to *create a user account for the service* that the user may access. Doing so would have been a conventional system-administrator function.

[0584] My opinion that it would have been conventional for system-administrators to have the ability to *create a user account for* access to a *service* offered by a computing device is corroborated by EX1032. EX1032 discloses:

> **The console application also facilitates creation and management of user accounts**. Each user account has associated therewith a set of access privileges to the functionality available over the console application or other administrative tool. **An IT administrator may have privileges to create and modify user accounts and entities. An administrative end-user associated with a subscriber may also have the ability to modify or create user accounts within the subscriber realm.** An administrative end-user associated with a publisher may have access privileges to register or delete services or actions within the publisher's realm. In one embodiment, a user account further includes

> access privilege information relating to specific services or actions available to the corresponding end-user. **The console application supports similar capabilities** available with respect to entities: searching, browsing, **creation of new accounts**, editing existing accounts, and deletion of obsolete accounts.

EX1032, [0041]-[0042].

[0585] Thus, in my opinion, POSAs would have found claim 24 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

25. Claim 25: "[C]laim 17, further comprising the step of installing files specific to the selected application on the selected server."

[0586] Tormasov permits "***installation*** of programs…and other ***applications***" into a *server's* VCEs. Tormasov, [0024]. The *application* (and *files specific to the selected application* that are within the VCE) are thereby *installed on the selected* computing device (which would have been a *server* in Schmidt-Tormasov and Schmidt-Tormasov-Calder). Similarly, starting a Schmidt-Tormasov and Schmidt-Tormasov-Calder's VCE-capsule would have *installed files specific to the selected application on the selected server*.

[0587] Thus, in my opinion, POSAs would have found claim 25 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0588] Based on Tormasov, Schmidt-Tormasov and Schmidt-Tormasov-Calder's VCE-capsules would have implemented "***access*** restrictions enforce[d] inside the" server's OS kernel. Tormasov, [0018]. POSAs understood that includes (or conventionally would have included) *setting the file access permissions to allow an individual user to access the one of the applications to be distributed* and installed in the VCE-capsule.

[0589] My opinion that it would have been conventional in computing systems to *set file access permissions to allow a user to access the one of the applications to be distributed* is corroborated by EX1065. EX1065 discloses:

> **Most conventional computing systems operate using an operating system that provides certain basic access control mechanisms. Using such conventional access control mechanisms, a computer systems manager or administrator (a person responsible for maintaining the computer systems) can configure, for example, an operating system such as Unix in a computer system to control how that computer system allows users to access various directories and files maintained under control of the operating system**.
>
> In an operating system such as Unix, **access control is frequently provided via access control lists (ACLs) and/or by file and directory permission settings. ACLs are lists of users with associated permissions (e.g., read, write) that a systems manager can create**

> **and associate with a file and/or a directory in a filesystem**. For instance, an ACL might list all users for a file that have read access to the file, write access to the file, and execute access to the file. W**hen a user requests access to a particular file, the operating system checks the ACL associated with that file to determine if the user requesting access is listed for the type of access that is requested (e.g., read, write or execute)**. If the user is listed for the requested access, the operating system allows the requested access. Otherwise, a protection violation occurs and the operating system denies user access as requested.

EX1065, 1:33-56.

[0590] Thus, in my opinion, POSAs would have found claim 26 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

27. Claim 27

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

[0591] Tormasov's users "may **stop**" VCEs "in the same manner as with a common operating system." Tormasov, [0025]. That stopping includes ir at least suggests *de-installing* the VCE (and the *service* it provides) *from the server*. Thus, based on Tormasov, Schmidt-Tormasov and Schmidt-Tormasov-Calder would have included functionality for *de-installing a service from a server*.

[0592] Thus, POSAs would have found limitation [**27A**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0593] As I discussed above (§VII.E.17.b, [**17B**]), POSAs would have had motivation to install a service in a VCE-capsule by dragging and dropping an icon for the service into an icon/folder representing the target server. POSAs similarly would have had motivation to *display the icon representing the service* and *display the icon representing the server on which the service is installed* to provide a user-friendly way to identify which services are installed on each Schmidt-Tormasov and Schmidt-Tormasov-Calder server.

[0594] My opinion that conventional GUI functionality included drag-and-drop operations to install software to a computer based on *relatively moving* one icon to another icon is further corroborated by EX1047. EX1047 discloses:

[0595] EX1047 discloses:

> **In graphical user interfaces employing an object-oriented programming paradigm, application programs are typically represented to a user by an icon displayed within a window on a computer screen, one icon for each application program that can be run. Execution of an application program is initiated by selecting its corresponding icon**, most often using a pointing device such as a mouse. When an application program is selected, a message

is sent to the corresponding application program object, indicating that the application program object is to invoke certain of its methods. For example, if a word processing program is selected, the methods contained within the application program object may include starting the word processing program. **The user may also "drag" icons from one area of the screen to another, or from one window to another using the mouse. The user may even "drag" one icon representative of an application object and "drop" that icon on top of another**. This "object-object" interaction will result in a combination of application objects. For example, if a word processing document icon is dropped upon a word processing program icon, the object-object interaction results in starting the word processing program and causing that program to open the word processing document. This is possible because both the word processing program and the word processing document have been represented as compatible application program objects. **Hence, the icons in the object-oriented programming paradigm allow the user to graphically control various computer system elements and the interrelationships between computer system elements.**

EX1047, 1:39-67.

[0596] My opinion that conventional GUI functionality included drag-and-drop operations to install software to a computer based on *relatively moving* one icon to another icon is further corroborated by EX1109. EX1109 discloses:

To complete the initiation, the user manipulates the desired action icon with a graphical interface device, such as a mouse, for the

> **purpose of depositing it on the selected server icon. Specifically, the user utilizes a "drag and drop" technique in which the action icon is selected, moved across the screen display until it overlaps the desired server group and then deselected**. FIG. 8 is exemplary of the screen display 120 of FIG. 7 illustrating the graphical manipulation of the action icon 126 to a selected server grouping icon. **Drag and drop techniques are well known in the relevant art and will not be discussed in further detail. One skilled in the relevant art will appreciate that alternative methods for selecting a server icon may be practiced with the present invention, including the selection of a server icon and drag and dropping it on the selected action icon**.

EX1109, [0048].

[0597] Thus, POSAs would have found limitation [**27B**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0598] In my opinion, POSAs would have had reason for Schmidt-Tormasov and Schmidt-Tormasov-Calder to *utilize 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* as a user-friendly way to de-install services from target servers, which was a conventional administrator feature.

[0599] My opinion that it would have been conventional to utilize icons in a GUI-based OS to initiated de-installation of application programs (and corresponding services) is further corroborated by EX1095. EX1095 discloses:

> **Objects such as documents, applications, and OS components are represented by icons.** While the primary means of interfacing with these types of systems are the menus, there are limited visual facilities that allow operation of the system purely via manipulation of the objects themselves. For example, **Apple's Macintosh System 7 and MicroSoft's [sic] Windows 95 allows the user to delete files by moving the icons representative of the files to a delete function object called "Trash" or the "Recycle Bin"**.

EX1095, 8:4-13.

[0600] My opinion that it would have been conventional to utilize icons in a GUI-based OS to initiated de-installation of application programs (and corresponding services) is further corroborated by EX1109. EX1109 discloses:

> **To complete the initiation, the user manipulates the desired action icon with a graphical interface device, such as a mouse, for the purpose of depositing it on the selected server icon. Specifically, the user utilizes a "drag and drop" technique in which the action icon is selected, moved across the screen display until it overlaps the desired server group and then deselected**. FIG. 8 is exemplary of the screen display 120 of FIG. 7 illustrating the graphical manipulation of the action icon 126 to a selected server grouping icon. **Drag and drop techniques are well known in the relevant art and will not be**

> **discussed in further detail. One skilled in the relevant art will appreciate that alternative methods for selecting a server icon may be practiced with the present invention, including the selection of a server icon and drag and dropping it on the selected action icon**.

EX1109, [0048].

[0601] Thus, POSAs would have found limitation [**27C**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

* * *

[0602] Thus, for the reasons discussed above (§§VII.E.27.a-c), it is my opinion that POSAs would have found claim 27 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0603] As I discussed above (§§VII.E.27.b-c), POSAs would have had reason to *separate* the *icon representing the service from the icon representing the server* for de-installing services from the Schmidt-Tormasov and Schmidt-Tormasov-Calder servers. POSAs would have also had reason to allow for *separating* the server and service icons as a user-friendly way to de-install the service, using a conventional drag-and-drop operations that was a conventional feature of GUI-based system administration. *See supra* §§VII.E.27.b-c (limitations [**27B**]-[**27C**]) (citing corroboration).

[0604] My opinion that it would have been conventional in a GUI-based OS to de-install applications by manipulating icons is further corroborated by EX1109. EX1109 discloses:

> To complete the initiation, the user manipulates the desired action icon with a graphical interface device, such as a mouse, for the purpose of depositing it on the selected server icon. Specifically, the user utilizes a "drag and drop" technique in which the action icon is selected, moved across the screen display until it overlaps the desired server group and then deselected. FIG. 8 is exemplary of the screen display 120 of FIG. 7 illustrating the graphical manipulation of the action icon 126 to a selected server grouping icon. **Drag and drop techniques <u>are well known</u> in the relevant art** and will not be discussed in further detail. **One skilled in the relevant art will appreciate that alternative methods for selecting a server icon may be practiced with the present invention, including the selection of a server icon and drag and dropping it on the selected action icon**.

EX1109, [0048].

[0605] Thus, in my opinion, POSAs would have found claim 28 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0606] As I discussed above with corroboration (§VII.E.23, claim 23), POSAs would have had reason to test whether the selected server would have been a valid platform for installing the service in the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations. For similar reasons, POSAs would have had reason to *test whether the selected server is a valid computing platform for de-installation of the service*, as claimed, to prevent corrupting the server by improper de-installation of a VCE-capsule (and the service(s) it would provide) and so as not to waste time attempting to de-install a VCE-capsule that should or would not have been installed or was not installed despite the attempt to de-install the service.

[0607] Thus, in my opinion, POSAs would have found claim 29 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

30. Claim 30: "[C]laim 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."

[0608] Schmidt-479 discloses a "virtual desktop system architecture" where capsules are presented to end-user devices from a server. Schdmit-479, [0073]-[0074]. POSAs would have understood that as the server executed a service, the service would at times have changed local files and thereby create a

copy of those *data file changes specific to the service*, which are *copied back to a storage medium from which the data file changes originated prior to installation*.

[0609] In my opinion, POSAs also had reason to store those changes *back to a storage medium from which the data file changes originated prior to installation*, as claimed, to allow any data file changes to be retrievable for later user.

[0610] Thus, in my opinion, POSAs would have found claim 30 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

31. Claim 31

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

[0611] As I discussed above (§§VII.E.1.a-b, [**1PREA**]-[**1PREB**]), Schmidt-Tormasov and Schmidt-Tormasov-Calder would have provided a system for installing VCE-capsules (*containers*) of applications to a computing system comprising, *inter alia*, a plurality of servers running different OSs. As I also discussed above (§VII.E.1.b.iv, §VII.E.1.c.i.(2) (patent's "container" definition **part** [**4**])), a VCE-capsule's applications would have been *for performing a plurality of tasks* – e.g., tasks related to a website server application – that would have *comprised a plurality of processes*.

[0612] Thus, in my opinion, POSAs would have found limitation [**31PRE**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the reasons I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious limitations [**1PREA**]-[**1PREB**] and [**1A**].

b. [31A] "a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers, "

[0613] For the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1A**] (§VII.E.1.c), the VCE-capsules in Schmidt-Tormasov and Schmidt-Tormasov-Calder would have been *secure containers of associated files accessible to, and for execution on, one or more servers*.

[0614] Thus, in my opinion, POSAs would have found limitation [**31A**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious limitation [**1A**].

c. [31B] "each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers,"

[0615] As I discussed above (§§VII.E.1.c.i. (2)-(3), [**1A**]), application sets in Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules

(*containers*) would have been insulated from effects of other application sets (i.e., they would have been *mutually exclusive of* each *other*). Similarly, Schmidt-Tormasov's and Schmidt-Tormasov-Calder 's VCE-capsule's applications would have been isolated from other VCE-capsule's applications on the server.

[0616] Thus, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) would have been *mutually exclusive of the other, such that read/write files within a container* could *not* have *been shared with other containers*.

[0617] As I also discussed above (§VII.E.1.i, [**1G**]), each VCE-capsule would have had its own unique root directory that would have been mutually exclusive of the root directories of other VCE-capsules. *See* Schmidt-479, [0052]; Tormasov, [0018], [0023], [0030]; Calder, [0138], [0241]. The VCE-capsule's read/write files would have been located under the VCE-capsule's mutually exclusive root and thus could *not* have *been shared with other* VCE-capsules (*containers*).

[0618] Thus, in my opinion, POSAs would have found limitation [**31B**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder meet limitation [**1A**] (specifically, part [**2**] of the parties' construction of *container*). *See supra* §VII.E.1.c.i.(2).

d. [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;"

[0619] Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**31C**] for the same reason they meet claim 6 that I discussed above (§VII.E.6).

[0620] Thus, in my opinion, POSAs would have found limitation [**31C**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious claim 6.

e. [31D] "wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes,"

[0621] Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**31D**] for the same reason Schmidt-Tormasov and Schmidt-Tormasov-Calder meet part [**4**] of the patent's definition of a "*container*" that I discussed above (§VII.E.1.c.i.(2), [**1A**]).

[0622] Thus, in my opinion, POSAs would have found limitation [**31D**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious limitation [**1A**] (specifically, part [4] of the patent's definition of a *container*). *See supra* §VII.E.1.c.i.(2).

f. [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,"

[0623] Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules would have had *associated system files for use in executing the one or more processes* of the applications installed to VCE-capsules, for the reasons discussed above (§VII.E.1.c.iii, [**1A**]). For the same reason Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**1D**]-[**1E**] (*supra* §§VII.E.1.f-g), Schmidt-Tormasov and Schmidt-Tormasov-Calder meet *the associated system files in a* VCE-capsule (*container*) *are files that are copies of files or modified copies of files that remain as part of the operating system*.

[0624] Thus, in my opinion, POSAs would have found limitation [**31E**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious limitations [**1A**] (§VII.E.1.c.iii), [**1D**]-[**1E**] (§§VII.E.1.f-g).

g. [31F] "each container having its own execution file associated therewith for starting one or more applications,"

[0625] Schmidt-Tormasov and Schmidt-Tormasov-Calder meet [**31D**] for the same reason Schmidt-Tormasov and Schmidt-Tormasov-Calder meet part

[**5**] of the patent's definition of a "container" (*supra* §VII.E.1.c.i.(2), [**1A**])) and claim 2 (*supra* §VII.E.2).

[0626] Thus, in my opinion, POSAs would have found limitation [**31F**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious limitation [**1A**] (specifically, part [5] of the patent's definition of a *container*) and claim 2.

h. [31G] "in operation, each container utilizing a kernel resident on the server and"

[0627] When *in operation*, each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule (*container*) would have *utilized a kernel resident on the server* for the reasons discussed above that each VCE-capsule would have been *for use with a local kernel residing permanently on one of the servers* (§VII.E.1.c.iv, [**1A**]).

[0628] Thus, in my opinion, POSAs would have found limitation [**31G**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious limitation [**1A**] (§VII.E.1.c.iv).

i. [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,"

[0629] Each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule (*container*) would have *used a kernel in an underlying operation system in which it is running*, as I discussed above (limitations [**31G**], [**1A**]). *See supra* §VII.E.1.c.iv ([**1A**]), §VII.E.31.h ([**31G**]). As I also explained above, each VCE-capsule would have been *absent its own kernel* for the same reason Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules meet part [**4**] of the patent's definition of a "container" (§VII.E.1.c.i.(2), [**1A**]) and limitation [**1C**] (§VII.E.1.e).

[0630] Because a VCE-capsule (*container*) would not have included a kernel, the VCE-capsule would have *exclusively used* the base OS's kernel of the server where the VCE-capsule would have been installed, for the reasons I discussed above (§VII.E.1.c.i.(2) ("container" definition **part** [**4**]), §VII.E.1.c.iv ([**1A**]).

[0631] Thus, in my opinion, POSAs would have found limitation [**31H**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations for the same reason I discussed above that Schmidt-Tormasov and Schmidt-Tormasov-Calder render obvious limitation [**1A**] (specifically, part

[4] of the patent's definition of a *container*) and limitation [**1C**]. *Supra* §VII.E.1.c.i.(2) ([**1A**]), §VII.E.1.e ([**1C**]).

j. [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."

[0632] 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 functionality teaches or suggests having a *run time module for monitoring system calls from applications associated with one or more* Schmidt-Tormasov and Schmidt-Tormasov-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* Schmidt-265 (EX1118), [0061], [0066], [0072] (describing capsule creation/re-starting via *system calls*).

[0633] That is one way by which Schmidt-Tormasov and Schmidt-Tormasov-Calder would have rendered obvious limitation [**31I**].

[0634] Moreover, Schmidt-Tormasov-Calder incorporates Calder's teaching of translating system calls so they could have been understood on native and non-native OSs. Calder, [0101]; *supra* §VII.C, §VII.D.2. As Calder teaches, some run-time software module (e.g., an "interception module") will *monitor*

system calls from the applications in order to translate them to the system call format / syntax required by the base OS on the server, and thus *provide control of the one or more applications*. Calder, [0086]-[0087].

[0635] That is another way by which Schmidt-Tormasov-Calder would have rendered obvious limitation [**31I**].

[0636] Thus, in my opinion, POSAs would have found limitation [**31I**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

* * *

[0637] Thus, for the reasons discussed above (§§VII.E.31.a-j), it is my opinion that POSAs would have found claim 31 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

32. Claim 32: "[C]laim 31, further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources."

[0638] As I discussed above (§VII.E.9, claim 9), POSAs would have had motivation to pre-allocate system resources to Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsules. POSAs similarly would have had motivation (and thus would have found it obvious) to include *a scheduler comprising values related to an allotted time in which processes within a* VCE-

capsule (*container*) *may utilize predetermined resources*, as a conventional way to have assigned a server's resources to its applications.

[0639] My opinion that it would have been conventional to allocate utilization of predetermined computing resources based on an allotted time is further corroborated by EX1120, which is cited by Tormasov in its "BIBLIOGRAPHY" (*see* Tormasov, [0033]). EX1120 discloses that a conventional system call in a Unix-based OS is utime, which "sets the access and modification times of the specified file according to the value of ***times*** [a parameters provided to the utime system call]." EX1120, 452 (italics original).

[0640] My opinion that it would have been conventional to allocate utilization of predetermined computing resources based on an allotted time is further corroborated by EX1034. EX1034 discloses:

> Beginning with the first time-sharing system in the mid-1960s, operating systems (OSs) have implemented numerous methods of allowing multiple applications to share computational resources of a computer without knowledge of one another. By **allocating small 'time slices' to each application, and interrupting when a 'time slice' has expired**, a computer can present each application with the illusion that it is running alone on the computer.

EX1034, [0003].

[0641] My opinion that it would have been conventional to allocate utilization of predetermined computing resources based on an allotted time is further corroborated by EX1094. EX1094 discloses:

> The UNIX operating system keeps track of all programs running in the system and allocates resources, such as disks, memory, and printer queues, as required. UNIX allocates resources so that, ideally, each program receives a fair share of resources to execute properly. **UNIX doles out resources using two methods: <u>scheduling priority</u>** and system semaphores. Each program is assigned a priority level. Higher priority tasks (like reading and writing to the disk) are performed more regularly. User programs may have their priority adjusted dynamically, upwards or downwards, depending on their activity and the available system resources. System semaphores are used by the operating system to control system resources. A program can be assigned a resource by getting a semaphore by making a system call to the operating system. When the resource is no longer needed, the semaphore is returned to the operating system, which can then allocate it to another program.

EX1094, 2:5-21.

[0642] Thus, in my opinion, POSAs would have found claim 32 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

33. Claim 33: "[C]laim 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 [sic]."

[0643] As I discussed above (§VII.E.32, claim 32), Schmidt-Tormasov and Schmidt-Tormasov-Calder would have rendered obvious a *run time module* that *intercepts system calls* from reaching the host's kernel and redirects them into a VCE-capsule. Schmidt-479, [0052], [0059]; Calder, [0139].

[0644] As I discussed above (§VII.E.6, claim 6), POSAs would have had motivation to assign each VCE-capsule (*container*) *at least one of the IP address, the host name and the Mac_Address*.

[0645] Schmidt-479 discloses a *run-time module* that monitors "where interface objections are named," including in "system calls." Schmidt-479, [0059]. An exemplary named interface object in Schdmit-479 are communication interfaces (e.g., an interface for communication over IP-based or MAC-based networks). Schdmidt-479, [0070].

[0646] POSAs would have known that a conventional way to implement those conventional unique identifiers would have been for the *run time module* that *intercepts system calls from any of the plurality of containers* to

replace the host's identifier with the container's identifier (or vice-versa) to ensure the VCE-capsules would have *run independently of one another without contention*, as Schmidt-479 and Tormasov intended. Schmidt-479, [0054]; Tormasov, [0030]. For example, when a VCE-capsule would have made a system call to request the server's IP address, Schmidt-Tormasov's and Schmidt-Tormasov-Calder's run-time module would have provided the VCE-capsule's assigned, unique IP address back to the VCE-capsule and applications therein (e.g., so the applications in the VCE-capsule could include the IP address as the source address in an IP packet sent by the application).

[0647] Thus, in my opinion, POSAs would have found claim 33 to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

34. Claim 34

a. [34A] "[C]laim 31, wherein the run time module performs: monitoring resource usage of applications executing;"

[0648] Schmidt-Tormasov and Schmidt-Tormasov-Calder meets limitation [**34A**] for the same reasons I discussed above that POSAs would have had reason to allocate resources to Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules (*containers*) (§VII.E.9, claim 9) and track/monitor *resource usage of applications executing* in VCE-capsules (§VII.E.13, claim 13).

[0649] Thus, in my opinion, POSAs would have found limitation [**34A**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

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

[0650] POASs knew that conventional *applications* run in *user mode* and make *system calls to kernel mode from user mode to kernel mode* (e.g., to access the server's hardware resources).

[0651] My opinion in that regard is corroborated by EX1120 (cited in Tormasov at [0033]), which discloses:

> The execution of user processes on UNIX systems is divided into two levels: **user** and **kernel**. When a process executes a **system call**, the *execution mode* of the process changes from ***user mode*** **to** ***kernel mode***: the operating system executes and attempts to service the user request, returning an error code if it fails.

EX1120, 15 (italicized emphases original).

[0652] As I discussed above (§VII.E.1.c.i.(2) ("container" definition **part** [**4**]), §VII.E.1.c.iv, [**1A**]), applications in Schmidt-Tormasov's and Schmidt-Tormasov-Calder's VCE-capsules installed to one server share the OS's kernel, and system calls made by applications in a VCE-capsule would have been either (i) intercepted and redirected into the VCE-capsule (e.g., if a requested system file is

within the VCE-capsule) or (ii) passed to the Base OS's kernel (e.g., if server hardware is required and/or a system file is not present in the VCE-capsule). Tormasov, FIG. 1; Schmidt-479, [0052], [0059]; Calder, [0139].

[0653] Schmidt-479 discloses that "a chroot ***system call*** 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]. Schmidt-479 further discloses that "[a] hash table lookup is performed at all points within the system where interface objects are named, including system entry points (e.g., ***system calls***)…. As new system objects are created, they are assigned capsule-local names and entered into the hash tables for future lookups." Schmidt-479, [0059]. POSAs understood that the above-discussed *system call* that re-assigns a capsule's root and use of a table lookup in response to *system calls* teaches (or suggests) *intercepting systems calls to kernel mode*, *made by the at least one respective application within a* capsule, *from user mode to kernel mode* as intercepting the *system calls* from the capsule's applications would have been a convention way to re-assign the root and recognize the need to use the hash lookup table for assigned capsule-local names.

[0654] Thus, in my opinion, POSAs would have found limitation [**34B**] to have been over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

[0655] Calder discloses that its techniques for translating system calls (which would have been used in Schmidt-Tormasov-Calder) operates via an "***interception*** module [that] ***intercepts system calls*** that are made by the ***application*** program" to the *kernel*, which are (or conventionally would have been) *made by the respective application from user mode to kernel mode*. Calder, [0076]; *see also* Calder, [0074]-[0076].

[0656] That is another reason why POSAs would have found limitation [**34B**] to have been over Schmidt-Tormasov-Calder.

c. [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."

[0657] In my opinion, POSAs would have had reason and it would have been obvious with a reasonable expectation of success to *compare the monitored resource usage of the at least one respective application with the resource limits* in Schmidt-Tormasov and Schmidt-Tormasov-Calder that I discussed above (§VII.E.32, claim 32) and only *forwarded the system calls to a kernel on the basis of the comparison*. POSAs would have wanted to do so to ensure that the allocated resource limits would not be violated if the system call were passed to the kernel without first comparing the monitored resources to resource limits. As I discussed above (§VII.E.1.a.i ([**1PREA**]), §§VII.E.8-9

(claims 8-9)), the conventional prior OSs that would have been used in Schmidt-Tormasov and Schmidt-Tormasov-Calder (e.g., Windows-, Unix-, and Linux-based OS) included functionality that *monitored resource usage* allocated to a user (and thus to the *applications* executed by the user) and that only *forward system calls* by the user's applications *to the kernel based on comparing the monitored resource and the resource limits* of the user. *E.g.*, EX1132, [0009]-[0011] (noting that Unix includes the command "quota," which "implements the policy for **restricting the disk capacity** used by each user").

[0658] Thus, in my opinion, POSAs would have found limitation [**34C**] to have been obvious over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

* * *

[0659] Thus, for the reasons discussed above (§§VII.E.34.a-c), it is my opinion that POSAs would have found claim 34 to have been over the Schmidt-Tormasov and Schmidt-Tormasov-Calder combinations.

F. Grounds 3-4: Mapping of the Combinations of Grounds 1-2 in Further View of Schmidt-629 to Claims 1-34

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

[0660] Schmidt-629 is a U.S. Patent Application Publication that was filed on March 22, 2001 (two months after Schmidt-479), and published on July 18, 2002. Brian Schmidt, the sole inventor listed on the face of Schmidt-479, is

also listed as an inventor of Schmidt-629. Schmidt-479 and Schmidt-629 both identify and "incorporate[]…by reference" the same "co-pending U.S. patent application….assigned to the assignee of the present application," that "described how a group of active processes and their associated state could be represented in a form that made it possible to halt the active processes, to move them to a different binary compatible machine, or to suspend them on disk for later revival on the same or a different machine." Schmidt-479, [0006]; Schmidt-629, [0005].

[0661] Schmidt-629 and Schmidt-479 also describe complementary (and overlapping) aspects of the same capsule technique. *Compare, e.g.*, chmdit-629, [0033]-[0035] *with* Schmidt-479, [0038]-[0040].

[0662] As I discussed above (*e.g.*, §VII.A), Schmidt-479 discusses techniques for "provid[ing] virtual namespaces for active computing environments," using "compute capsule[s that] serve[] to represent and encapsulate an active computing environment." Schmidt-479, [0016]. Schmidt-479 teaches techniques to all "the encapsulation to be suspended and revived on any binary compatible machine." Schmidt-479, [0016].

[0663] Schmidt-629 similarly teaches techniques for use with Schmidt's "compute capsule." Schmidt-629, [0015]. In particular, Schmidt-629 teaches to implement each "capsule [to] ha[ve] a unique locator, such as an IP address, assigned to it." Schmidt-629, [0015]. Having a unique locator like an IP

Address beneficially supports having the capsule "moved to a different machine having potentially a different operating system or on different network and maintain the open network connections it had prior to the migration." Schmidt-629, [0015]. Schmidt-629 says that "[a]ll users are assigned their own capsule at the time of log in, i.e. session creation," and each "capsule communicates with other capsules in the form of packets." Schmidt-629, [0016]. The "target capsule" (i.e., the destination capsule for a communication) "may lie on the same host network or on a different one." Schmidt-629, [0016]. "The underlying system knows the location of the target capsule by looking at the locator (e.g., IP address) of the target capsule." Schmidt-629, [0016]. Alternatively, "the underlying system wraps the original packet in another packet that uses the IP address of the host network on which the target capsule currently lies as the final destination instead of the locator of the target capsule." Schmidt-629, [0016]. For example, "all applications address incoming packets by their network address regardless of the location of the target capsule." Schmidt-629, [0017].

[0664] Using Schmidt-629's techniques, a user of a capsule "can now migrate to another machine on the same or different network, and the processes that were open before the migration will now be open displaying the same information on the new machine." Schmidt-629, [0056]. That is because, as

discussed above, a "unique locator, such as an IP address and unique network locator address, is assigned to a capsule." Schmidt-629, [0057].

2. Description of the Grounds 3-4 Combinations

[0665] As I discussed above, Ground 1 (§VII.D.1), places inside Schmidt-479's capsule a Tormasov VCE (and applications installed thereto) and Ground 2 (§VII.D.2) modifies Ground 1's VCE-capsules based on Calder to allow a VCE-capsule to be executed on both native- and non-native OSs running on a server. In Grounds 1-2, the VCE-capsules are stored for installation on servers operating in a system a multi-server system, where each server runs a respective one of a conventional, prior art OS (e.g., one server runs Windows-95, another server runs Windows-98, another server runs a Unix-based OS, etc.), and where the servers may be part of different local area networks. For example, some servers could be in location providing applications and services for one group of people and other servers could be located in a network in another location providing the same and different applications and services for another group of people that are part of the same overall organization, like different corporate or academic departments.

[0666] To facilitate the installing VCE-capsules to different servers in a large organization and/or moving VCE-capsules among those different servers, POSAs would have been motivated and reasonably expected success based on the

features of Schmidt-629 discussed above (§VII.F.1) to assign to each Schmidt-Tormasov and Schmidt-Tormasov-Calder VCE-capsule a unique locator identifier, such as an IP address, to beneficially support moving a capsule to among the "different machine[s] having potentially a different operating system or on different network" in the Schmidt-Tormasov and Schmidt-Tormasov-Calder systems. Schmidt-629, [0015].

[0667] That is, in the **Ground 3** combination, each Schmidt-Tormasov (Ground 1) VCE-capsule is assigned a unique locator like an IP address or other conventional unique locator, based on Schmidt-629's teachings; and in the **Ground 4** combination, each Schmidt-Tormasov-Calder (Ground 2) VCE-capsule is similarly assigned a unique locator like an IP address or other conventional unique locator, based on Schmidt-629's teachings.

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

[0668] In my opinion, claims 1-4, claims 7-30, independent claim 31's limitations other than [**31C**] (discussed below, §VII.F.4.c.ii), and the limitations added by (dependent) claims 32-34 (discussed below, §VII.F.4.d) are rendered obvious by Schmidt-479's and Tormasov's teachings (with or without Calder's teachings) in the Grounds 3-4 combinations (discussed *supra* §VII.F.2) for the same reasons those claims / limitation are met by the Grounds 1-2 combinations. *Supra* §VII.E.1-4 (claims 1-4), §VII.E.7-30 (claims 7-30), §VII.E.31.a-c (claim 31,

limitations [**31PRE**]-[**31B**]), §VII.E.31.e-j (claim 31, limitations [**31D**]-[**31I**]), §VII.E.32-34 (claims 32-34).

4. Mapping of the Grounds 3-4 Combinations (the Ground 1-2 Combinations in Further View of Schmidt-629) to Claims 1-34

a. 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."

[0669] The Grounds 3-4 combinations (*see supra* §VII.F.2) render obvious claim 2 in the same way the Ground 1-2 combinations, respectively, render obvious claim 2, as I discussed above (§VII.E.2). Based on Schmidt-629 (§VII.F.1), the Grounds 3-4 combination would have further *modified at least some of the associated system files in* the *plural* VCE-capsules (*containers*) discussed above (§VII.F.2), *to provide an association with a* VCE-capsule *specific identity* (e.g., an IP address) *assigned to* each *particular* VCE-capsule (*container*), as claimed (*see supra* §VII.F.2 (discussing the Grounds 3-4 combinations)).

[0670] Thus, in my opinion, the Grounds 3-4 combinations meet the limitation added by Claim 5 and thus would have rendered claim 5 obvious to a POSA.

b. 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."

[0671] The Grounds 3-4 combinations (*see supra* §VII.F.2) render obvious claim 2 in the same way the Ground 1-2 combinations, respectively, render obvious claim 2, as I discussed above (§VII.E.2). Based on Schmidt-629 (§VII.F.1), the Grounds 3-4 combination would have further *assigned a unique associated identity to each of a plurality of the* VCE-capsules (*containers*)*, wherein the identity includes at least one of IP address, host name, and MAC address*, as claimed (*see supra* §VII.F.2 (discussing the Grounds 3-4 combinations)). For example, as discussed above (§VII.F.1), Schmidt-629 expressly teaches assigning to a capsule a *unique IP address* as a *unique identity* for a capsule and based on that teaching, POSAs had motivation to assign to each of the Ground 3-4 VCE-capsules a *unique IP address*, for the reasons I discussed above (§VII.F.2).

[0672] Thus, in my opinion, the Grounds 3-4 combinations meet the limitation added by Claim 6 and thus would have rendered claim 6 obvious to a POSA.

c. Claim 31:

i. Limitations [31PRE]-[31B], [31D]-[31H]

[0673] The Grounds 3-4 combinations (*see supra* §VII.F.2) render obvious limitations [**31PRE**]-[**31B**], [**31D**]-[**31H**] in the same way the Ground 1-2 combinations, respectively, render obvious those limitations, as I discussed above. *See supra* §VII.E.31.a-c (claim 31, limitations [**31PRE**]-[**31B**]), *supra* §VII.E.31.e-j (claim 31, limitations [**31D**]-[**31I**]).

ii. Limitation [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"

[0674] Based on Schmidt-629 (§VII.F.1), the VCE-capsules (*containers of files*) in the Grounds 3-4 combinations (*see supra* §VII.F.2) would *each have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a MAC_address*. For example, as discussed above (§VII.F.1), Schmidt-629 expressly teaches assigning to a capsule a *unique IP address* as a *unique identity* for a capsule and based on that teaching, POSAs had motivation to assign to each of the Ground 3-4 VCE-capsules a *unique IP address*, for the reasons I discussed above (§VII.F.2).

[0675] Thus, in my opinion, the Grounds 3-4 combinations meet limitation [**31C**].

* * *

[0676] Thus, for the reasons discussed above (§§VII.F.4.c.i-ii), it is my opinion that POSAs would have found claim 31 to have been obvious over the Ground 3-4 combinations discussed above (§VII.F.2).

d. Claims 32-34

[0677] Claims 32 and 34 depend from claim 31, which the Ground 3-4 combinations render obvious for the reasons I discussed above (§VII.F.4.c). The limitations added by claims 32 and 34 are met by the Ground 3-4 combinations (*supra* §VII.F.2) in the same way the Ground 1-2 combinations, respectively, meet those limitations, as I discussed above (§VII.E.32, §VII.E.34).

[0678] Claim 33 depends from claim 32. The limitation added by claims 33 is met by the Ground 3-4 combinations (*supra* §VII.F.2) in the same way the Ground 1-2 combinations, respectively, meet the limitation added by claim 33, as I discussed above (§VII.E.33).

[0679] Thus, in my opinion, the Ground 3-4 combinations would have rendered obvious claims 32-34.

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

I declare under penalty of perjury that the foregoing is true and correct.

Dated: Jan 31, 2025



Samrat Bhattacharjee

VIII. 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.

APPENDIX – NON-PATENT LITERATURE

APPENDIX TABLE OF CONTENTS

I. OVERVIEW OF SEVERAL RESOURCES....3

A. Internet Archive....3

B. ACM and ACM Digital Library....5

II. OPINIONS REGARDING PUBLIC ACCESSIBILITY....6

A. Public Accessibility of the Kamp Article (EX1076)....6

B. Public Accessibility of the Prevelakis Article (EX1077)....10

C. Public Accessibility of Microsoft's Computer Dictionary 5th Edition (EX1088)....17

D. Public Accessibility of the Bach Book (EX1120)....19

E. Public Accessibility of the Crowley Book (EX1121)....21

F. Public Accessibility of the Eckel Book (EX1122)....23

1. I understand from counsel that a printed publication qualifies as publicly accessible as of the date it was disseminated or otherwise made available such that any member of the public interested in the subject matter could locate it through the exercise of reasonable diligence.

2. While I understand that the determination of public accessibility under the foregoing standard rests on a case-by-case analysis of the facts particular to an individual publication, I also understand that a printed publication is rendered "publicly accessible" if it is cataloged and indexed by a library such that a member of the public interested in the subject matter a could locate it exercising reasonable diligence (i.e., I understand that cataloging and indexing by a library in a manner that permits a member of the public interested in the subject matter to locate the publication is sufficient, though there are other ways that a printed publication may qualify as publicly accessible, such as being indexed and catalogued in an online database). One manner of sufficient indexing is indexing according to subject matter category. I understand that the cataloging and indexing by a single library of a single instance of a particular printed publication is sufficient, even if the single library is in a foreign country. I also understand that the cataloging and indexing of information that would guide a person interested in the relevant subject matter to the printed publication, such as the cataloging and indexing of an abstract

for the printed publication, is sufficient to render the printed publication publicly accessible.

3. I understand that a document may also be publicly accessible if it is catalogued and indexed in an online digital database such that a person interested in the relevant subject matter could locate the document in the database exercising reasonable diligence. For example, the ACM Digital Library is an example of a digital database that is well-regarded among the academic community and considered to be accurate and trustworthy source of information regarding the scientific and academic publications catalogued in the database. As I discuss in Appendix-§I.B below, I have personally used that and similar databases and, in my opinion, POSAs would have been aware of this database (and similar databases like IEEEXplore) and would have used it to search for and retrieve publications in the art of management and deployment of server applications and related fields at the time of the alleged invention.

4. I understand that routine business practices, such as general library cataloging and indexing practices, can be used to establish an approximate date on which a printed publication became publicly accessible. I further understand that the date on which digital databases, like the ACM Digital Library, indicate a document was published and/or added to the database can be used to establish a date on which the document became publicly accessible.

5. In my opinion, POSAs would have been engaged in research, learning through study and practice in their field, and possibly through formal instruction, the bibliographic resources relevant to their research. In my opinion, in the early-2000s POSAs would have had access to a vast array of long-established print and online resources providing indexing information, abstracts, and full-text services for publications in the field of computing.

6. In my opinion, POSAs would have been able to locate the material discussed in this declaration as well as any additional documents I discuss in the sections below on their own or with the assistance of a research librarian, with relative ease using the tools and resources described in the sections below.

I. OVERVIEW OF SEVERAL RESOURCES

A. Internet Archive

7. Exhibits 1133-1145 were captured by the Internet Archive at the indicated dates and made browsable by the Internet Archive's Wayback Machine. The Internet Archive's Wayback Machine is well known to researchers and the general public as a service that provides archives of webpages as they existed on dates in the past. I am personally familiar with the Internet Archive and its Wayback Machine from my own use of the service in my work and research. It is well known that the Internet Archive collects and archives captures of webpages using web crawlers, which are software programs that surf the Web, visiting

websites and automatically storing copies of the webpages and other files represented by the URLs that the crawler visits. Crawlers are also used by Internet search engines (e.g., Google, Bing, etc.) to access webpages to be provided as search results, as is also well known to researchers and the general public. When the Internet Archive's crawlers visit a URL on a particular date, they store a copy of that webpage or file at the URL as it existed on that date. Users of the Wayback Machine can later access that archived capture of the URL to view and interact with the webpage or retrieve a file as it existed on that past date.

8. It is well known to users of the Internet Archive's Wayback Machine, and I am personally aware from my own use of the Wayback Machine, that the date on which an archived webpage was captured by the Internet Archive's crawlers can be observed in the URL that the Internet Archive assigns to the archived capture (what the Internet Archive calls an "extended URL"). When an archived capture of a webpage is viewed through the Wayback Machine on the Internet Archive's website (http://web.archive.org), the URL of the archived capture is http://web.archive.org/web, followed by numbers indicating the year, month, day, and time at which the webpage was captured by the Internet Archive's crawlers, followed by the original URL of the archived webpage. For example, a Wayback Machine URL of http://web.archive.org/web/20111231171057/http://www.uspto.gov:80/ is the

Internet Archive's archived capture of the USPTO's homepage (http://www.uspto.gov:80/) captured on December 31, 2011 (2011/12/31), at 17:10:57 (5:10 p.m. and 57 seconds).

B. ACM and ACM Digital Library

9. The Association for Computing Machinery (ACM) is "the world's largest educational and scientific computing society, delivers resources that advance computing as a science and a profession."[1] According to the ACM's official homepage, there are "more than 100,000 members" of the ACM.[2] The ACM's website links to a digital database (currently called the ACM Digital Library), which is "the world's most comprehensive database of full-text articles and bibliographic literature covering computing and information technology. This renowned repository includes the complete collection of ACM publications plus an extended bibliographic database of core works in computing from scholarly publishers."[3] I have personally used the ACM Digital Library to retrieve copies of

[1] https://www.acm.org/about-acm (last visited January 16, 2025).

[2] https://www.acm.org/about-acm/about-the-acm-organization (last visited January 16, 2025).

[3] https://www.acm.org/publications/digital-library (January 16, 2025); *see also* https://dl.acm.org/ (January 16, 2025).

articles or presentations published in journals in the field of computing and information technology and in my opinion POSA at the time of the alleged invention would have been aware of the ACM and the ACM Digital Library and could have retrieved an article or presentation published by the ACM or publishers indexed by the ACM Digital Library by exercising reasonable diligence (e.g., by obtaining a copy of an article from a library, by being a member of or subscriber of the publication, or by navigating to the ACM Digital Library).[4]

II. OPINIONS REGARDING PUBLIC ACCESSIBILITY

A. Public Accessibility of the Kamp Article (EX1076)

10. EX1076 (which I refer to herein as "Kamp") is titled "Jails: Confining the omnipotent root." Based on the evidence I discuss below, it is my opinion that Kamp is prior art to the challenged patent (at least) because it was publicly accessible more than one year before the earliest priority date cited on the face of the challenged patent (September 15, 2002) by being disseminated or otherwise made available to POSAs exercising reasonable diligence.

11. According to the face of EX1076, Kamp was published as part of the "The FreeBSD Project." According to a footnote on the bottom of EX1076, Kamp

[4] *See, e.g.*, August 15, 2000 Wayback Machine archive of the ACM Digital Library, at https://web.archive.org/web/20000815053844/http://www.acm.org/dl/.

"was sponsored by http://www.servetheweb.com/ and donated to the FreeBSD Project for inclusion in the FreeBSD OS. FreeBSD 4.0-RELEASE was the first release including this code." According to the Semantic Scholar database,[5] Kamp was "Published 2000."[6]

12. As I discuss below, a copy of Kamp also appears to have been publicly accessible from the NLUUG Unix User Group – The Netherlands' website more than one year before the earliest possible priority date of the Asserted Patent. I understand from counsel that a POSA may use other publicly accessible information as "research aids" as a roadmap to locating additional prior art in the same or related field.

[5] According to the Semantic Scholars website, "Semantic Scholar provides free, AI-driven search and discovery tools, and open resources for the global research community." *See* https://www.semanticscholar.org/about (last checked January 16, 2025).

[6] https://www.semanticscholar.org/paper/Jails%3A-confining-the-omnipotent-root-Kamp-Watson/a659692ed5031c5410668d644a60d7806bb2efbb?year%5B0%5D=2001&year%5B1%5D=2003&sort=relevance&page=2 (last checked January 16, 2025).

13. For example, the NLUUG's website (as archived by the Internet Archive on June 4, 2002) states as follows:

> The NLUUG (UNIX User Group - The Netherlands) is the association of professional UNIX users in the Netherlands.
>
> Since the late seventies, the NLUUG has brought together the community of UNIX systems administrators, programmers, researchers and IP network professionals.
>
> The primary goal of the NLUUG is to extend the application of, and knowledge about, UNIX and related open systems.[7]

14. In my opinion, POSAs in the field of the challenged patent would have had reason to navigate to the NLUUG's website, to find information about publications or, in some cases, copies of publications relating to the field of the challenged patent.

15. EX1141 is the Internet Archive's archived capture of the NLUUG's homepage, http://www.nluug.nl/, as it existed on July 22, 2001, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20010722200036/http://www.nluug.nl/. In the middle of the NLUUG's homepage, under the heading "Conferences," there is text with hyperlinks stating "More information about upcoming and previous events can be

[7] https://web.archive.org/web/**20020604031**649/http://www.nluug.nl/folder/index-uk.html (last checked January 16, 2025) (emphasis added to emphasize the archival date of this website by the Internet Archive).

found in the calender [sic] of events." Clicking on the "previous" events navigates to http://www.nluug.nl/events/indexOld.html. EX1142 is the Internet Archive's archived capture of that webpage as it existed on August 3, 2001, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20010803092814/http://www.nluug.nl/events/indexOld.html.

16. As shown on EX1142, there is a hyperlink to the "Sane 2000, 'Second International SANE Conference', May 22-25, 2000 Maastricht The Netherlands,'" reachable by clicking on the text "SANE 2000." Clicking on that text navigates to http://www.nluug.nl/events/sane2000/index.html. EX1143 is the Internet Archive's archived capture of that webpage as it existed on August 12, 2001, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20010812213235/http://www.nluug.nl/events/sane2000/index.html.

17. As shown on EX1143, there is a hyperlink to "SANE 2000 papers,'" reachable by clicking on that text. Clicking on that text navigates to http://www.nluug.nl/events/sane2000/papers.html. EX1144 is the Internet Archive's archived capture of that webpage as it existed on June 20, 2001, as reflected in the Internet Archive's extended URL:

https://web.archive.org/web/20010620043826/http://www.nluug.nl/events/sane2000/papers.html.

18. As shown on EX1144, there is a hyperlink to "Jails: Confining the omnipotent root (*PDF*), by Poul-Henning Kamp,'" reachable by clicking on the hyperlink text. Clicking on that text navigates to http://www.nluug.nl/events/sane2000/papers/kamp.pdf, which is a copy of the "Jails: Confining the omnipotent root" article in a PDF form. EX1145 is a screen shot of a portion of the Internet Archive's archived capture of that PDF as it existed on September 2, 2000, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20000902205700/http://www.nluug.nl/events/sane2000/papers/kamp.pdf.

19. Kamp (EX1076) is a copy of the PDF retrieved from that link.

20. Based on the above evidence, I conclude that Kamp would have been publicly accessible, via (at least) the NLUUG's website, to a POSA exercising reasonable diligence by September 2, 2000, or at least by August 12, 2001.

B. Public Accessibility of the Prevelakis Article (EX1077)

21. EX1077 (which I refer to herein as "Prevelakis") is titled "Sandboxing Applications." Based on the evidence I discuss below, it is my opinion that Prevelakis is prior art to the challenged patent (at least) because it was publicly accessible more than one year before the earliest priority date cited on the

face of the challenged patent (September 15, 2002) by being disseminated or otherwise made available to POSAs exercising reasonable diligence.

22. According to the face of EX1077, Prevelakis was published in the June 25-30, 2001 issue of the conference proceedings of the *FREENIX Track 2001 USENIX Annual Technical Conference*, that occurred in Boston, Massachusetts, USA. Page 3 of EX1077's PDF bears the text "© 2001 by The USENIX Association," indicating it was published and copyrighted by The USENIX Association in 2001 and thus publicly accessible by the year 2001. EX1077's Page 2 bears what appears to be a stamp from a library, indicating the date when EX1077 was received by that library (August 2001).

23. Additionally, the ACM Digital Library (*see supra* I.B) includes an entry for an article bearing the same title as Prevelakis, published in the same proceedings of the USENIX Annual Technical Conference discussed above, at the same pages of the proceedings indicated in EX1077 (i.e., pages 119 to 126).[8] The ACM Digital Library indicates that that article was "Published: 25 June 2001."[9]

[8] *See* https://dl.acm.org/doi/10.5555/647054.715767 (last checked January 16, 2025).

[9] *Id.*

24. Based on the above evidence, I conclude that Prevelakis would have been publicly accessible to a POSA exercising reasonable diligence at least by the end of 2001.

25. As I discuss below, a copy of Prevelakis also appears to have been publicly accessible from the USENIX Association's website more than one year before the earliest possible priority date of the Asserted Patent. I understand from counsel that a POSA may use other publicly accessible information as "research aids" as a roadmap to locating additional prior art in the same or related field.

26. For example, as the USENIX Association's website states:

> The USENIX Association is a 501(c)(3) nonprofit organization, dedicated to supporting the advanced computing systems communities and furthering the reach of innovative research. **We are known for organizing conferences and publishing research**, but our greatest strength lies in building communities in computing systems. We represent the interests of our communities in a variety of ways, including our professional affiliation with the Computing Research Association.
>
> **Since 1975, USENIX has brought together the community of engineers, researchers, and practitioners** working on the cutting edge of the computing world. **USENIX conferences have become the essential meeting grounds for the presentation and discussion of the most advanced information on the development of all aspects of computing systems**.[10]

[10] *See* https://www.usenix.org/about (last checked January 16, 2025).

27. In my opinion, POSAs in the field of the challenged patent would have had reason to navigate to the USENIX website, to find information about publications or, in some cases, copies of publications relating to the field of the challenged patent.

28. EX1133 is the Internet Archive's archived capture of the USENIX Association's homepage, http://www.usenix.org, as it existed on November 14, 2001, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20011114235021/http://www.usenix.org:80/. On the left of USENIX Association's homepage, there is a set of "SHORTCUTS." Selecting "Conference Proceedings" navigates to USENIX's "Proceeding" webpage at http://www.usenix.org/publications/library/proceedings/. EX1134 is the Internet Archive's archived capture of that webpage as it existed on November 15, 2001, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20011115072346/http://www.usenix.org/publications/library/proceedings/.

29. As shown on EX1134, there is a hyperlink to the "2001 USENIX Annual Technical Conference, Boston, MA," reachable by clicking on that text. Clicking on that text navigates to http://www.usenix.org/publications/library/proceedings/usenix01/, which provides information about the 2001 USENIX Annual Technical Conference. EX1135 is

the Internet Archive's archived capture of that webpage as it existed on November 15, 2001, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20011115173912/http://www.usenix.org/publications/library/proceedings/usenix01/.

30. As shown on EX1135, there is a hyperlink to the "Conference Proceedings," including a hyperlink to "FREENIX Track Refereed Papers," reachable by clicking on that text. Clicking on that text navigates to http://www.usenix.org/publications/library/proceedings/usenix01/freenix01/technical.html, which provides a list of the "REFEREED PAPERS" for the "FREENIX Track: 2001 USENIX Annual Technical Conference." EX1136 is the Internet Archive's archived capture of that webpage as it existed on November 18, 2001, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20011118162406/http://www.usenix.org/publications/library/proceedings/usenix01/freenix01/technical.html.

31. As shown on EX1136, there is a hyperlink to a paper from "Friday, June 29, 2001" titled "Sandboxing Applications," by "Vassilis Prevelakis, University of Pennsylvania, and Diomidis Spinellis, Athens University," reachable by clicking on that text. Clicking on that text navigates to http://www.usenix.org/publications/library/proceedings/usenix01/freenix01/prevelakis.html, which provides an "Abstract" of that article. EX1137 is the Internet

Archive's archived capture of that webpage as it existed on January 12, 2002, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20020112043538/http://www.usenix.org/publications/library/proceedings/usenix01/freenix01/prevelakis.html.

32. I note that page 6 of the PDF of EX1077 similarly lists Prevelakis under the heading "Friday, June 29," with the same title and authors (and associated universities) as shown in EX1137. *See also* EX1077 at 119. I further note that the text of the "Abstract" in EX1137 matches the text of the "Abstract" on page 119 of Prevelakis in EX1077 (i.e., the first page of the article).

33. As shown on EX1137, there are hyperlinks to "View the full text of this paper in HTML form, and PDF form," each of which is reachable by clicking on the text that says "HTML" and "PDF," respectively.

34. Clicking on the "HTML" text shown on EX1137 navigates to http://www.usenix.org/publications/library/proceedings/usenix01/freenix01/full_papers/prevelakis/prevelakis_html/index.html, which provides the text of the "Sandboxing Applications" article in an HTML website. EX1138 is the Internet Archive's archived capture of that webpage as it existed on March 29, 2003, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20030329030611/http://www.usenix.org/publications/

library/proceedings/usenix01/freenix01/full_papers/prevelakis/prevelakis_html/index.html. The bottom of EX1138 states as follows (italic original):

> ***This paper was originally published*** *in the Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, June 25-30, 2001, Boston, Masssachusetts [sic], USA*
>
> ***Last changed: 21 June 2001*** *bleu [sic]*

35. I have compared the text of EX1138 with the copy of Prevelakis in EX1077. Except for some differences in formatting, the text and figures of EX1138 and EX1077 appear to be identical.

36. Clicking on the "PDF" text shown on EX1137 navigates to http://www.usenix.org/publications/library/proceedings/usenix01/freenix01/full_papers/prevelakis/prevelakis.pdf, which a copy of the "Sandboxing Applications" article in aa PDF form. EX1139 is a screen shot of a portion of the Internet Archive's archived capture of that PDF as it existed on January 19, 2004, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/20040119150721/http://www.usenix.org/publications/library/proceedings/usenix01/freenix01/full_papers/prevelakis/prevelakis.pdf.

37. EX1140 is a copy of the PDF retrieved from that link. The cover page of EX1140 states that it was published in connection with the same "Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference," in "Boston, Massachusetts, USA" on "June 25-30, 2001" that is indicated on the

cover of EX1077. Like EX1077, page 1 of EX1140 bear the text "© 2001 by The USENIX Association," indicating it was published and copyrighted by The USENIX Association in 2001 and thus publicly accessible by the year 2001.

38. I have compared the text of EX1140 with the copy of Prevelakis in EX1077. Except for some differences in formatting on the cover page of EX1140 and the footers at the bottom of EX1077, the text and figures of EX1138 and EX1077 appear to be identical.

39. Based on the above evidence, I conclude that Prevelakis would have also been publicly accessible, via the USENIX Association's website, to a POSA exercising reasonable diligence by January 12, 2002, or at least by March 29, 2003.

C. Public Accessibility of Microsoft's Computer Dictionary 5th Edition (EX1088)

40. EX1088 (which I refer to herein as "Microsoft Dictionary") includes excerpts from a book titled "Microsoft Computer Dictionary Fifth Edition." *See* EX1088, pdf. page 1. Based on the evidence I discuss below, it is my opinion that Bach is prior art to the challenged patent (at least) because it was publicly accessible more than one year before the earliest priority date cited on the face of the challenged patent (September 15, 2002) by being disseminated or otherwise made available to POSAs exercising reasonable diligence.

41. Page 2 of EX1088's PDF bears the text "Copyright © 2002 by Microsoft Corporation" and that it was "PUBLISHED BY Microsoft Press A

Division of Microsoft Corporation," indicating it was copyrighted and published by Microsoft in 2002 and thus publicly accessible by the year 2002.

42. Page 2 of EX1088's PDF further includes an "ISBN" number, which I understand "uniquely identifies your book, and facilitates the sale of your book to bookstores (physical and digital) and libraries."[11]

43. According to the U.S. Library of Congress's online catalog, a book bearing the same name, publisher, and ISBN number as EX1088 was "Published/Created" "c2002."[12]

44. According to the ACM Digital Library, a book bearing the same name and publisher as EX1088 was "Published: 01 May 2002."[13]

---

[11] *See* https://www.isbn.org/ (last checked January 16, 2025).

[12] https://catalog.loc.gov/vwebv/search?searchArg1=0735614954&argType1=all&searchCode1=KNUM&searchType=2&combine2=and&searchArg2=&argType2=all&searchCode2=GKEY&combine3=and&searchArg3=&argType3=all&searchCode3=GKEY&year=1525-2025&fromYear=&toYear=&location=all&place=all&type=all&language=all&recCount=25 (last checked January 16, 2025).

[13] https://dl.acm.org/doi/10.5555/515434 (last checked January 16, 2025).

45. Based on the above evidence, I conclude that the Microsoft Computer Dictionary would have also been publicly accessible, via (at least) its publisher Microsoft Press or a library (e.g., the Library of Congress), to a POSA exercising reasonable diligence at least by May 2002.

D. Public Accessibility of the Bach Book (EX1120)

46. EX1120 (which I refer to herein as "Bach") includes excerpts from a book titled "THE DESIGN OF THE UNIX® OPERATING SYSTEM," by Maurice J. Back. *See* EX1120, pdf. page 2. Based on the evidence I discuss below, it is my opinion that Bach is prior art to the challenged patent (at least) because it was publicly accessible more than one year before the earliest priority date cited on the face of the challenged patent (September 15, 2002) by being disseminated or otherwise made available to POSAs exercising reasonable diligence.

47. Page 3 of EX1120's PDF bears the text "Copyright © 1986 by Bell Telephone Laboratories, Incorporated" and that it was "Published by Prentice-Hall, Inc.," indicating it was copyrighted by Prentice-Hall, Inc. and published in 1986 and thus publicly accessible by the year 1986. Page 3 of EX1120's PDF further states that "[t]he Publisher offers discounts of this book when ordered in bulk quantities," and provides contact information for obtaining a copy of the book.

48. Page 3 of EX1120's PDF further includes an "ISBN" number, which I understand "uniquely identifies your book, and facilitates the sale of your book to bookstores (physical and digital) and libraries."[14]

49. According to the U.S. Library of Congress's online catalog, a book bearing the same name, author, publisher, and ISBN as EX1120 was "Published/Created" "c1986."[15]

50. I further note that EX1010 (Tormasov) – a U.S. Patent Application Publication that published on September 5, 2002 – includes a "BIBLIOGRAPHY" that cites the following, which has the same author, title, publisher, and ISBN number as shown in EX1120 and on the Library of Congress's online catalog. *See* EX1010, [0033].

51. I additional note that EX1086 – which is an issued U.S. Patent that issued on May 2, 1995 – discloses that "[t]he general description of the well-known operation of a UNIX operating system" discussed in EX1086 "is derived from Chapter 2 of the book entitled 'The Design of the UNIX Operating System'

[14] *See* https://www.isbn.org/ (last checked January 16, 2025).

[15] https://catalog.loc.gov/vwebv/holdingsInfo?searchId=22140&recPointer=0&recCount=25&searchType=0&bibId=1941862 (last checked January 16, 2025).

by Maurice J. Bach (EX1086, 2:58-61), which is the same title and author listed on the face of EX1120.

Based on the above evidence, I conclude that Bach would have also been publicly accessible, via (at least) its publisher Prentice-Hall or a library (e.g., the Library of Congress), to a POSA exercising reasonable diligence by 1987, or at least by September 5, 2002.

E. Public Accessibility of the Crowley Book (EX1121)

52. EX1121 (which I refer to herein as "Crowley") includes excerpts from a book titled "OPERATING SYSTEMS *A Design-Oriented Approach*," by Charles Crowley (University of New Mexico). *See* EX1121, pdf. page 1. Based on the evidence I discuss below, it is my opinion that Crowley is prior art to the challenged patent (at least) because it was publicly accessible more than one year before the earliest priority date cited on the face of the challenged patent (September 15, 2002) by being disseminated or otherwise made available to POSAs exercising reasonable diligence.

53. Page 1 of EX1121's PDF includes text of what appears to be a library stamp from the "J. ROBERT VAN PELT LIBRARY" "MICHIGAN TECHNOLOGY UNIVERSITY" "HOUGHTON, MICHIGAN."

54. Page 2 of EX1121's PDF bears the text "© Richard D. Irwin, a Times Mirror Higher Education Group, Inc., company, 1997" indicating it was published

and copyrighted by the Richard D. Irwin company in 1997 and thus publicly accessible by the year 1997. Page 2 of EX1121's PDF further includes an "ISBN" number, which I understand "uniquely identifies your book, and facilitates the sale of your book to bookstores (physical and digital) and libraries."[16]

55. Page 3 of EX1121's PDF bears a stamp with the date "APR 18 2002."

56. According to the U.S. Library of Congress's online catalog, a book bearing the same name, author, publisher, and ISBN number as EX1121 was "Published/Created" "1997."[17]

57. I further note that EX1010 (Tormasov) – a U.S. Patent Application Publication that published on September 5, 2002 – includes a "BIBLIOGRAPHY" that cites the following, which has the same author, title, publisher, and ISBN

---

[16] *See* https://www.isbn.org/ (last checked January 16, 2025).

[17] https://catalog.loc.gov/vwebv/search?searchArg=crowley+operating+systems++a+design-oriented+approach&searchCode=GKEY%5E*&searchType=1&limitTo=none&fromYear=&toYear=&limitTo=LOCA%3Dall&limitTo=PLAC%3Dall&limitTo=TYPE%3Dall&limitTo=LANG%3Dall&recCount=25 (last checked January 16, 2025).

number discussed above for EX1120 and on the Library of Congress's online catalog. *See* EX1010, [0034].

58. Based on the above evidence, I conclude that Bach would have also been publicly accessible, via (at least) its publisher Irwin or a library (e.g., the Library of Congress), to a POSA exercising reasonable diligence by 1997, or at least by September 5, 2002.

F. Public Accessibility of the Eckel Book (EX1122)

59. EX1122 (which I refer to herein as "Eckel") includes excerpts from a book titled "Building a Linux Internet Server," by George Eckel & Chris Hare. *See* EX1122, pdf. page 1. Based on the evidence I discuss below, it is my opinion that Eckel is prior art to the challenged patent (at least) because it was publicly accessible more than one year before the earliest priority date cited on the face of the challenged patent (September 15, 2002) by being disseminated or otherwise made available to POSAs exercising reasonable diligence.

60. Page 2 of EX1122's PDF (native page ii) bears the text "Copyright © 1995 by New Riders Publishing" indicating it was published and copyrighted by New Riders Publishing in 1995 and thus publicly accessible by the year 1995. Page 2 of EX1122's PDF (native page ii) further includes an "ISBN" number,

which I understand "uniquely identifies your book, and facilitates the sale of your book to bookstores (physical and digital) and libraries."[18]

61. I further note that EX1010 (Tormasov) – a U.S. Patent Application Publication that published on September 5, 2002 – includes a "BIBLIOGRAPHY" that cites the following, which has the same author, title, publisher, and ISBN number as shown in EX1120 and on the Library of Congress's online catalog. *See* EX1010, [0035].

62. As I discuss below, an article describing Eckel also appears to have been publicly accessible from the Linux Journal's website more than one year before the earliest possible priority date of the Asserted Patent. I understand from counsel that a POSA may use other publicly accessible information as "research aids" as a roadmap to locating additional prior art in the same or related field.

63. For example, the Linux Journal's website (as archived by the Internet Archive on May 8, 1999) describes itself as "the Premier Linux Magazine."[19] In my opinion, POSAs in the field of the challenged patent would have had reason to

[18] *See* https://www.isbn.org/ (last checked January 16, 2025).

[19] *See* https://web.archive.org/web/19990508005124/http://www.ssc.com/lj/ (last checked January 16, 2025).

navigate to the Linux Journal's website, to find information about publications or, in some cases, copies of publications relating to the field of the challenged patent.

64. Page 1 of EX1146 is the Internet Archive's archived capture of the Linux Journal's homepage, http://www.ssc.com/lj/, as it existed on May 8, 1999, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/19990508005124/http://www.ssc.com/lj/. On the left of Linux Journal's homepage, there is hyperlink text to "Previous Issues." Clicking that navigates to a webpage at http://www.ssc.com/lj/mags.html.

65. Pages 2-4 of EX1146 is the Internet Archive's archived capture of that webpage as it existed on April 18, 1999, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/19990418013008/http://www.ssc.com/lj/mags.html.

66. As shown on page 3 of EX1146, there is a hyperlink to "Issue #23, March 1996 (Focus: Linux Distributions Compared)," reachable by clicking on the "Issue #23, March 1996" text. Clicking on that text navigates to http://www.ssc.com/lj/issue23/index.html, which provides the "Table of Contents -- Linux Journal #23 -- March 1996." Pages 5-6 of EX1146 is the Internet Archive's archived capture of that webpage as it existed on April 24, 1999, as reflected in the Internet Archive's extended URL:

https://web.archive.org/web/19990424144757/http://www.ssc.com/lj/issue23/index.html.

67. On page 6 of EX1146, there is a hyperlink to "Book Review: Building a Linux Internet Server," reachable by clicking on the "Book Review:" text. Clicking on that text navigates to http://linuxjournal.com/issue23/blisrev.html.

68. Pages 7-8 of EX1146 is the Internet Archive's archived capture of that webpage as it existed on December 2, 1998, as reflected in the Internet Archive's extended URL: https://web.archive.org/web/19981202103832/http://linuxjournal.com/issue23/blisrev.html. Pages 7-8 of EX1146 are a "Book Review" of a book having the same title, authors, publisher, and ISBN number discussed above for EX1122.

69. Based on the above evidence, I conclude that Eckel would have also been publicly accessible, via (at least) its publisher New Riders Publishing, to a POSA exercising reasonable diligence by May 8, 1999, or at least by September 5, 2002 (the date of Tormasov's publication).