Paper No. 1

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMERICAN AIRLINES, INC. AND SOUTHWEST AIRLINES CO.,

Petitioners

v.

INTELLECTUAL VENTURES II LLC,

Patent Owner.

IPR2025-00931
U.S. Patent No. 8,332,844

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 8,332,844

TABLE OF CONTENTS

TABLE OF AUTHORITIES iv

EXHIBIT LIST v

I. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8 1

II. GROUNDS FOR STANDING 5

III. IDENTIFICATION OF CHALLENGES AND RELIEF REQUESTED 5

IV. THE '844 PATENT 6

A. Overview 6

B. Effective Filing Date 10

C. Level of Ordinary Skill 11

V. CLAIM CONSTRUCTION 11

VI. OVERVIEW OF THE ART 12

A. Sapuntzakis (Ex.1005) 14

B. Birse (Ex.1006) 18

C. Holzmann (Ex.1007) 21

D. Federwisch (Ex.1008) 22

VII. GROUND 1: CLAIMS 1-13 ARE OBVIOUS OVER SAPUNTZAKIS AND HOLZMANN 23

A. Motivation to Combine 23

B. Sapuntzakis and Holzmann meet all claim elements. 24

1. Claims 1 and 7 24

2. Claim 2 33

3. Claim 3 35

4. Claims 4 and 11 36

5. Claims 5 and 12 39

6. Claims 6 and 13 40

7. Claim 8 41

8. Claim 9 43

9. Claim 10 44

VIII. GROUND 2: CLAIMS 14-27 ARE OBVIOUS OVER SAPUNTZAKIS AND FEDERWISCH 47

A. Motivation to Combine 47

B. Sapuntzakis and Federwisch meet all claim elements. 48

1. Claims 14, 19, and 23 48

2. Claim 15 52

3. Claim 16 53

4. Claims 17 and 21 53

5. Claims 18, 22, and 27 54

6. Claims 20 and 24 54

7. Claim 25 55

8. Claim 26 57

IX. GROUND 3: CLAIMS 1-13 ARE OBVIOUS OVER BIRSE AND HOLZMANN 58

A. Motivation to Combine 58

B. Birse and Holzmann meet all claim elements. 59

1. Claims 1 and 7 59

2. Claim 2 67

3. Claim 3 68

4. Claims 4 and 11 70

5. Claims 5 and 12 72

6. Claims 6 and 13 73

7. Claim 8 73

8. Claim 9 73

9. Claim 10 74

X. GROUND 4: CLAIMS 14-27 ARE OBVIOUS OVER BIRSE AND FEDERWISCH 74

A. Motivation to Combine 74

B. Birse and Federwisch meet all claim elements. 75

1. Claims 14, 19, and 23 75

2. Claim 15 77

3. Claim 16 77

4. Claims 17 and 21 77

5. Claims 18, 22, and 27 77

6. Claims 20 and 24 78

7. Claim 25 78

8. Claim 26 78

XI. THERE ARE NO DISCRETIONARY FACTORS FAVORING DENIAL OF INSTITUTION 79

A. The *Fintiv* factors either favor institution or are neutral 79

B. An analysis under 35 U.S.C. § 325(d) also favors institution 80

XII. CONCLUSION 81

TABLE OF AUTHORITIES

Cases

Ericsson Inc. v. XR Commc'ns LLC,
IPR2024-00613, Paper 9 (PTAB Oct. 9, 2024) 79

Mylan Pharmaceuticals Inc., v. Regeneron Pharmaceuticals, Inc.,
IPR2022-01226, Paper 22 (PTAB Jan. 11, 2023).....81

Phillips v. AWH Corp.,
415 F.3d 1303 (Fed. Cir. 2005).....11

Sand Revolution II, LLC v. Continental Intermodal Grp.-Trucking LLC,
IPR2019-01393, Paper 24 (PTAB. June 16, 2020) (informative) 79

Sotera Wireless, Inc. v. Masimo Corp.,
IPR2020-01019, Paper 12 (PTAB Dec. 1, 2020) (precedential) 80

Studiengesellschaft Kohle v. Shell Oil Co.,
112 F.3d 1561 (Fed. Cir. 1997).....11

Toyota Motor Corp. v. Cellport Systems, Inc.,
IPR2015-00633, Paper 11 (Aug. 14, 2015) 12

Wellman, Inc. v. Eastman Chemical Co.,
642 F.3d 1355 (Fed. Cir. 2011).....12

Statutes

35 U.S.C. § 325(d) 80

Rules

37 C.F.R. § 42.104(a).....5

37 C.F.R. §42.100 11

EXHIBIT LIST

Exhibit No.	Description of Document
1001	U.S. Patent No. 8,332,844
1002	Prosecution File History of U.S. Patent No. 8,332,844 (Application No. 11/709,477)
1003	Expert Declaration of Erez Zadok, Ph.D.
1004	Erez Zadok, Ph.D. Curriculum Vitae
1005	Sapuntzakis, et al. "Optimizing the Migration of Virtual Computers," Association for Computing Machinery, Special Interest Group on Operating Systems, Operating Systems Review, Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Volume 36, Issue S1, published December 31, 2002 ("Sapuntzakis")
1006	U.S. Patent No. 7,089,300B1, entitled "Method and Apparatus for Administering the Operating System of a Net-Booted Environment," filed on October 18, 1999, and issued August 8, 2006 to Birse, et al. ("Birse")
1007	U.S. Patent Application Publication No. 2005/0283597 A1, entitled "System and Method for Booting Multiple Servers from a Single Operating System Image," filed on June 22, 2004, and published on December 22, 2005 ("Holzmann")
1008	U.S. Patent Application Publication No. 2003/0182313 A1, entitled "System and Method for Determining Changes in Two Snapshots and for Transmitting Changes to Destination Snapshot," filed on March 19, 2002, and published on September 25, 2003 ("Federwisch")
1009	File History for U.S. Patent Application No. 11/026,622 (the "'622 Application", now abandoned)
1010	File History for U.S. Patent Application No. 11/395,816 (the "'816 Application", now U.S. Patent No. 7,721,282)

Exhibit No.	Description of Document
1011	Microsoft Computer Dictionary, 3rd ed., 1997, excerpts ("Microsoft Computer Dictionary")
1012	Abraham Silberschatz and Peter B. Galvin, Operating Systems Concepts, 4th Edition, 1994, Addison-Wesley Publishing, excerpts ("Silberschatz")
1013	File System Design for an NFS File Server Appliance, Dave Hitz et al., Network Appliance, USENIX 1994 ("Hitz")
1014	SnapMirror: File System Based Asynchronous Mirroring for Disaster Recovery, Hugo Patterson et al., Network Appliance, USENIX 2002 ("Patterson I")
1015	U.S. Patent 6,668,264B1 to Patterson et al., Filed Apr. 3, 2001, Issued Dec. 23, 2003, titled "Resynchronization of a target volume with a source volume" ("Patterson II")
1016	U.S. Patent 7,334,095B1 to Fair et al. Filed Apr. 30, 2004 ("Fair")
1017	Internet Small Computer Systems Interface (iSCSI), RFC 3720, IETF, April 2004, https://www.rfc-editor.org/rfc/pdfrfc/rfc3720.txt.pdf
1018	Claim Listing
1019	Complaint, *Intellectual Ventures I LLC et al. v. Southwest Airlines Co.*, No. 7:24-cv-00277 (W.D. Tex. Nov. 2, 2024)
1020	Complaint, *Intellectual Ventures I LLC et al. v. American Airlines, Inc.*, No. 4:24-cv-980-ALM (E.D. Tex. Nov. 2, 2024)
1021	Claim Chart Exhibit to Complaint, *Intellectual Ventures I LLC et al. v. Southwest Airlines Co.*, No. 7:24-cv-277 (W.D. Tex. Nov. 2, 2024)
1022	Screenshots From Internet Archive Related to www.usenix.org
1023	Declaration of Keith D. Harden

Petitioners American Airlines, Inc. ("American") and Southwest Airlines Co. ("Southwest") (collectively "Petitioners") petition for *inter partes* review ("IPR") of claims 1-27 of U.S. Patent No. 8,332,844 (the "'844 Patent") owned by Intellectual Ventures II LLC ("Patent Owner").

I. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8

A. Real Parties-in-Interest Under § 42.8(b)(1)

American is a real party in interest to this petition. American is a wholly owned subsidiary of American Airlines Group, Inc., which is not a real party in interest to this petition. No unnamed entity is funding, controlling, or otherwise has an opportunity to control or direct this petition or American's participation in any resulting IPR.

Southwest is also a real party in interest to this petition. No unnamed entity is funding, controlling, or otherwise has an opportunity to control or direct this petition or Southwest's participation in any resulting IPR.

B. Related Matters Under § 42.8(b)(2)

To the best of Petitioners' knowledge, the '844 Patent has been involved in the following related matters:

- *Intellectual Ventures I LLC et al. v. Liberty Mutual Holding Company Inc., et al.*, No. 23-cv-525 (E.D. Tex. filed November 15, 2023)

- *Intellectual Ventures I LLC et al. v. Comerica Incorporated*, No. 23-cv-524 (E.D. Tex. filed November 15, 2023)
- *Intellectual Ventures I LLC et al. v. JP Morgan Chase Bank, National Association et al.*, No. 23-cv-523 (E.D. Tex. filed November 15, 2023)
- *Intellectual Ventures I LLC et al. v. Nationwide Mutual Insurance Company*, No. 25-cv-632 (N.D. Tex. filed March 15, 2025)
- *Intellectual Ventures I LLC et al. v. The Bank of New York Mellon Corporation*, No. 25-cv-631 (N.D. Tex. filed March 15, 2025)

The actions against Liberty, Comerica, and JP Morgan has been dismissed with prejudice.

The ’844 Patent was also at issue in the following IPR proceeding:

- IPR2025-00202 – terminated/settled prior to institution

The ’844 Patent is also asserted in the following cases:

- *Intellectual Ventures I LLC et al. v. American Airlines, Inc.*, No. 24-cv-980 (E.D. Tex. filed November 2, 2024)
- *Intellectual Ventures I LLC et al. v. Southwest Airlines, Co.*, No. 24-cv-277 (W.D. Tex. filed November 2, 2024)

Petitioners are not aware of any other civil actions or proceedings filed in connection with the ’844 Patent.

C. Lead and Backup Counsel Under § 42.8(b)(3)

Lead Counsel	Backup Counsel
Keith D. Harden (Reg. No. 74,472) kharden@munckwilson.com Munck Wilson Mandala, LLP 2000 McKinney Ave., Ste. 1900 Dallas, Texas 75201 Tel: (972) 628-3600 Fax: (972) 628-3616	Daniel E. Venglarik (Reg. No. 39,409) dvenglarik@munckwilson.com Munck Wilson Mandala, LLP 2000 McKinney Ave., Ste. 1900 Dallas, Texas 75201 Tel: (972) 628-3600 Fax: (972) 628-3616 S. Wallace Dunwoody (Texas Bar No. 24040838 - admission *pro hac vice* to be requested) wdunwoody@munckwilson.com Munck Wilson Mandala, LLP 2000 McKinney Ave., Ste. 1900 Dallas, Texas 75201 Tel: (972) 628-3600 Fax: (972) 628-3616 Michael C. Wilson (Texas Bar No. 21704590 - admission *pro hac vice* to be requested) mwilson@munckwilson.com Munck Wilson Mandala, LLP 2000 McKinney Ave., Ste. 1900 Dallas, Texas 75201 Tel: (972) 628-3600 Fax: (972) 628-3616
	John B. Campbell (Reg. No. 54,665) jcampbell@McKoolSmith.com MCKOOL SMITH, P.C. 303 Colorado Street, Suite 2100 Austin, TX 78701 Tel: (512) 692-8730 Fax: (512) 692-8744 Casey Shomaker (Reg. No. 77,998)

Lead Counsel	Backup Counsel
	cshomaker@mckoolsmith.com McKool Smith, P.C. 300 Crescent Court, Suite 1200 Dallas, TX 75201 Tel: (214) 978-4218 Fax: (214) 978-4044 Emily Tannenbaum (Reg. No. 80,655) etannenbaum@mckoolsmith.com McKool Smith, P.C. 1301 Avenue of the Americas, Fl. 32 New York, NY 10019 Tel: (212) 402-9441 Fax: (212) 402-9444

An executed power of attorney under 37 C.F.R. § 42.10(b) accompanies this Petition.

D. Service Information Under § 42.8(b)(4)

Petitioners consent to electronic service at the above designated Counsel email addresses and at the following email address:

SWA-IV@munckwilson.com.

Service of any documents via post or hand-delivery may be made at: Munck Wilson Mandala LLP, 2000 McKinney Ave., Ste. 1900, Dallas, TX 75201; Telephone: (972) 628-3600; Facsimile: (972) 628-3616.

II. GROUNDS FOR STANDING

Pursuant to 37 C.F.R. § 42.104(a), Petitioners certify the '844 Patent is available for IPR, and that Petitioners are not barred or estopped from requesting an IPR challenging the claims on the grounds identified in this Petition. This Petition is filed within one year of service of the Complaint. Exs.1019,1020.

III. IDENTIFICATION OF CHALLENGES AND RELIEF REQUESTED

Petitioners respectfully request review and cancellation of claims 1-27 on the following prior art and statutory grounds:

Prior Art
Sapuntzakis, et al. "Optimizing the Migration of Virtual Computers," published 12/31/2002 (Ex.1005); Pre-AIA 35 U.S.C. § 102(b)
U.S. 7,089,300 (Birse), issued 8/8/2006 (Ex.1006); Pre-AIA 35 U.S.C. §§ 102(a), (b) and/or (e)
U.S. 2005/0283597 (Holzmann), published 12/22/2005 (Ex.1007); Pre-AIA 35 U.S.C. §§ 102(a), (b) and/or (e)
U.S. 2003/0182313 (Federwisch), published 9/25/2003 (Ex.1008); Pre-AIA 35 U.S.C. §§ 102(a), (b) and/or (e)

Ground	Challenged Claims	Basis
1	1-13	§ 103: Sapuntzakis and Holzmann
2	14-27	§ 103: Sapuntzakis and Federwisch
3	1-13	§ 103: Birse and Holzmann
4	14-27	§ 103: Birse and Federwisch

IV. THE '844 PATENT

A. Overview

The '844 Patent describes clustered computing involving multiple compute nodes with Operating System ("OS") software image management, where nodes are activated by diskless booting protocols or remote software installation to local storage, and where a boot image is required for each node in the cluster – and states that storing a boot image for each server results in wasted storage, and copying the entire master image at boot results in long boot times. Ex.1001, 1:31-67. Ex.1003, ¶¶160-65. The Background discusses branching-store file systems where a read-only "root" image is accessible by all nodes, and changes made by a node to the root image are stored in a "leaf" image unique to that node–a filter merges the leaf images with the root image to provide a cohesive instance of the application environment. Ex.1001, 2:13-24. Ex.1003, ¶¶166-72.

The system, as shown in Fig. 1, includes a general purpose computing system environment (compute node 100). Ex.1001, 4:27-36.

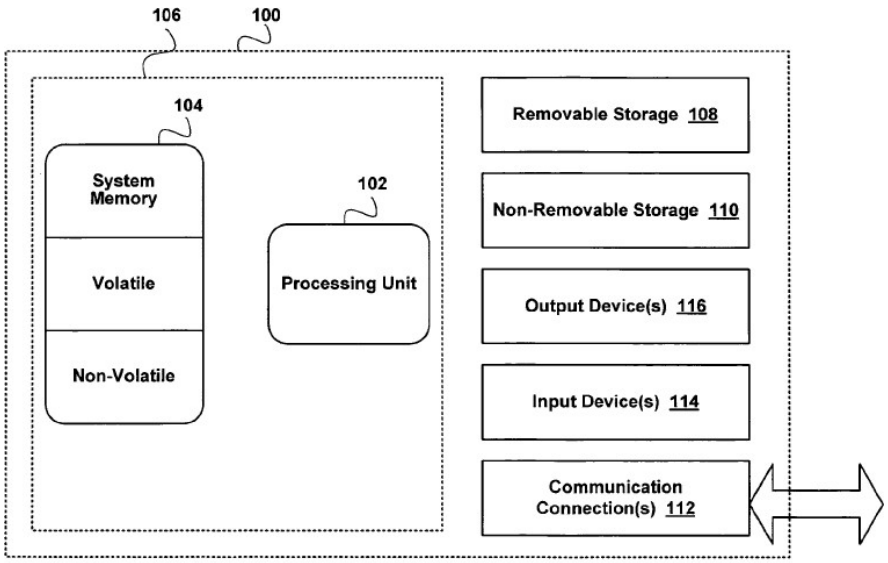


FIG. 1

Fig. 2 diagrams system 200 (e.g., a multi-computer system) having a number of compute nodes 220a-n coupled to a first storage unit 240 and a corresponding second storage unit 250a-n through a corresponding union block device ("UBD") 230a-n. Ex.1001, 5:12-22. To the compute nodes 220a-n, it appears that they have access to their own version of the application environment. However, a separate boot image is not created and stored for each node. Ex.1001, 5:22-26. Ex.1003 ¶¶177-78.

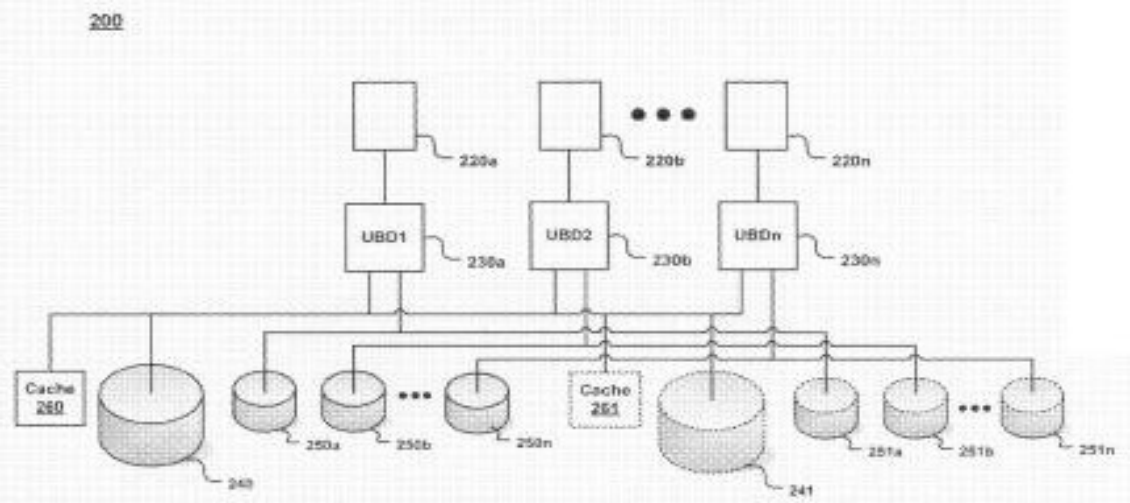


FIG. 2

Instead, a first storage unit 240 stores blocks of the root image. The root image contains data initially common to the compute nodes 220a-n, and is not changed by them. Ex.1001, 5:27-32. Each compute node 220a-n also has access to a corresponding second storage unit 250a-n for storing a leaf image, which in the challenged claims includes "only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of [said/the] root image…" or similar language. *Id.*, 5:33-34, 10:61-65, 11:32-35, 12:4-8, 12:47-50. The first storage unit 240 and second storage units 250a-n may each be contained on separate physical storage devices, on separate logical spaces on the same storage device, or any combination thereof. Ex.1001, 5:34-38. Ex.1003 ¶¶179-81.

A compute node 220a-n mounts its instantiation of the application environment via its respective UBD 230a-n which, in one embodiment, are "effectively low level drivers that operate as an interface between the first and second storage devices and the file system of each compute node 220a-n." *Id.*, 5:59-64. UBDs 230a-n determine what "leaf image" (from the appropriate second storage unit 250) is needed for portions of the image that their respective compute nodes 220a-n have changed, and locate the portions of the image not changed by their respective node (which may reside in the root image or intermediate images comprising versions of the root image, not depicted in Fig. 2). Ex.1001, 6:4-13. UBDs 230a-n also create a new leaf image on a respective second storage unit 250a-n when their respective compute nodes 220a-n make changes to their instantiations of the image (and may also modify the leaf image in response to their respective compute node's access to its instance of the image). *Id.*, 6:13-22. Ex.1003, ¶¶182-87, 200-203.

The blocks of the root image, leaf image, and cache are merged "on the fly" to create the node's image. Ex.1001, 8:13-15. From the node's perspective, the created image appears as one cohesive image even though a separate and complete image is not actually stored for the node. *Id.*, 8:18-55. This saves space without significantly impacting load times. *See id.*, 2:51-57. Ex.1003 ¶¶164, 188-89.

The ’844 Patent claims to further improve speed by caching portions of the root image in a cache 260 (as several nodes in a cluster may often access the same root image data). Ex.1001, 2:59-61, 6:38-43. Because cache capacity is typically limited, older data can be removed (as the amount approaches a threshold) to make way for new data by techniques “known in the art.” *Id.*, 7:1-16. Ex.1003 ¶¶189-90.

The ’844 Patent provides an “added benefit” by which one compute node can index the contents of the (shared) root image and provide the results to the other compute nodes (e.g., by storing results on a shared storage unit or providing the results directly). Ex.1001, 7:23-31. Ex.1003 ¶¶191-199.

B. Effective Filing Date

The asserted claims of the ’844 Patent have a filing date of February 21, 2007. Application No. 11/709,477 (the “’477 Application”), which matured into the ’844 Patent, was filed on February 21, 2007 and was a continuation-in-part of Application No. 11/395,816 (filed March 30, 2006) (now Patent No. 7,721,282) (the “’816 Application”), which was a continuation-in-part of Application No. 11/026,622 (filed December 20, 2004) (now abandoned) (the “’622 Application”).

However, none of the ’844 Patent claims are entitled to either priority date, as the ’816 and ’622 Applications never mention caching or indexing – at least one of which is expressly recited in each claim of the ’844 Patent. Ex.1001, 10:55-14:17; *compare* Ex.1001 6:37-7:44, Figs. 2, 3A, 3B, 7 *with* Ex.1009, 4-40 and Ex.1010, 8-

30. *See Studiengesellschaft Kohle v. Shell Oil Co.*, 112 F.3d 1561, 1564 (Fed. Cir. 1997). Therefore, the asserted claims of the '844 Patent are only entitled to a filing date of February 21, 2007.

C. Level of Ordinary Skill

A POSITA, on or before February 21, 2007, or alternatively on or before December 30, 2004, would have had a bachelor's degree in computer science, computer engineering, or an equivalent degree, and approximately two years of experience working in the fields of networked data storage, file systems, and operating systems. Additional experience might substitute for less education and *vice versa*. Ex.1003 at ¶34.

V. CLAIM CONSTRUCTION

Claim terms are construed according to the *Phillips* standard in an IPR. *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005); 37 C.F.R. §42.100. The analysis in Sections VI-IX establishes that the challenged claims are obvious under any reasonable interpretation under the *Phillips* standard, and thus Petitioners submit that the Board need not resolve any express constructions.[1] *See, e.g., Toyota Motor*

[1] Patent Owner very broadly reads the claims asserted in the litigation against commonly and widely used open-source software known as Docker "containers." *See* Ex.1021.

Corp. v. Cellport Systems, Inc., IPR2015-00633, Paper 11 at 16 (Aug. 14, 2015) (citing *Wellman, Inc. v. Eastman Chemical Co.*, 642 F.3d 1355, 1361 (Fed. Cir. 2011)) (claim construction only required where necessary to resolve dispute). Ex.1003, ¶¶307-309. Should any subsequent discussion of claim interpretation arise here or in the parallel district court proceedings, Petitioners reserve the right to propose constructions and respond to any construction advanced by the Patent Owner, the Board, and/or the Court.

VI. OVERVIEW OF THE ART

Systems and methods for using Internet-connected servers for storing read-only root images and separately storing modified blocks of data (leaf images), caching frequently used data, and merging modified data with original root image data, was known well before the critical date of the ’844 Patent. As explained by Dr. Zadok, servers that communicated over the Internet were well known before the mid-2000s. *See* Ex.1003, ¶¶52-85 (citing Microsoft Computer Dictionary, 3rd ed., 1997 (Ex.1011) (describing various Internet networks, communication protocols, and distributed computing environments used by servers) and RFC-3720 (Ex.1017) (describing protocol on top of TCP used to transport data over Internet/IP networks)).

Systems using snapshots with network access and using copy-on-write (CoW) were also known. For example, Hitz (Ex.1013) was a paper on “File System Design

for an NFS File Server Appliance" that described a "WAFL (Write Anywhere File Layout)" which was "a file system designed specifically to work in an NFS appliance." The "primary focus is on the algorithms and data structures that WAFL uses to implement Snapshots™, which are read-only clones of the active file system. WAFL uses a copy-on-write technique to minimize the disk space that Snapshots consume." Ex.1013, 4. Ex.1003, ¶109.

As further explained by Dr. Zadok, Hitz described that snapshots can be created and deleted as needed, offering space saving and improved reliability. Hitz explained that "snapshots use a copy-on-write technique to avoid duplicating disk blocks that are the same in a Snapshot as in the active file system" and "[o]nly when blocks in the active file system are modified or removed do Snapshots containing those blocks begin to consume disk space." Ex.1013, 6. Ex.1003, ¶¶110-14.

Various other systems used read-only root images with CoW-based systems. *See, e.g.*, Fair (Ex.1016) at 6:18-41, 21:27-31 (describing storing a read-only base snapshot on a source volume and using writable clones), Patterson I (Ex.1014) at 2-3, 8, 12 (describing asynchronous mirrors enabling self-consistent snapshots of data), and Patterson II (Ex.1015) (describing merging/synchronization of a target volume with a source volume where data blocks that are found to not be in the source are copied to a target to replace original source blocks with modified data). Ex.1003, ¶¶115-138.

Caching data was also well known before the '844 Patent as a means to store data in faster memory that is frequently used by computing systems. *See* Ex.1003, ¶¶86-101 (citing Silberschatz (Ex.1012) at 42-43, Figs. 2,6 (describing data storage hierarchies that prioritize use of faster storage systems to maximize cost, speed/latency, etc.) and Ex.1011 at 72,122,151,276,395 (describing use of cache storage for "frequently accessed segments of data")). Ex.1003, ¶¶115-138.

Indexing data and file systems was also well known before the '844 Patent as a means to organize a catalog for finding information more efficiently. *See* Ex.1003, ¶¶86-101 (citing Ex.1011 at 247,505 (describing indexing of file systems to enable searching of data)). Ex.1003, ¶¶115-138.

The cited references detailed below further demonstrate the claims of the '844 Patent were well known and obvious before the '844 Patent.

A. Sapuntzakis (Ex.1005)

Sapuntzakis concerns system architecture for quickly moving the state of a running computer (including its OS, applications and user files on disk, and all data in memory) – called a "capsule" – across a network to another machine. Ex.1005, 2. Sapuntzakis discloses "copy-on-write" ("COW") disk images that track just the updates to the disk-storage component of capsules. *Id*., Ex.1003 ¶¶225-30. Sapuntzakis was available at http://www.usenix.org at least as early as 2004. *See* Exs.1022, 1023. Sapuntzakis is currently still available on the Usenix website at

https://www.usenix.org/legacy/publications/library/proceedings/osdi02/tech/full_papers/sapuntzakis/sapuntzakis.pdf.

As users travel (commute), the system can move their capsules to computers close to them, giving them a consistent environment. Ex.1005, 3. Ex.1003, ¶¶227-28. As capsules are mostly derived from other capsules, and thus differences between related capsules are small relative to their total size, disks in those capsules can be stored efficiently by creating a hierarchy where each child capsule can be viewed as inheriting from its parent – with the differences in disk state between them captured in a separate COW virtual disk. Ex.1005, 6. Ex.1003, ¶¶229-231.

At the hierarchy base is a root disk, which is a complete capsule disk. For example, administrators can install and maintain the same software set on multiple machines by creating one (inactive) capsule and distributing it to all machines. Ex.1005, 6. This approach shares similarities with disk imaging, but the disclosed system allows multiple capsules to exist on the same machine. *Id*., 5. Ex.1003 ¶¶231-32.

All other nodes of the hierarchy represent a COW disk. Each path of COW disks originating from the root in the capsule hierarchy represents a capsule disk. Figure 1 shows an example of a capsule hierarchy (for a university) where the root capsule contains all software available to students, departments may extend

the basic capsules with department-specific software, and student files are stored on networked storage servers. Ex.1005, 6. Ex.1003, ¶¶233-34.

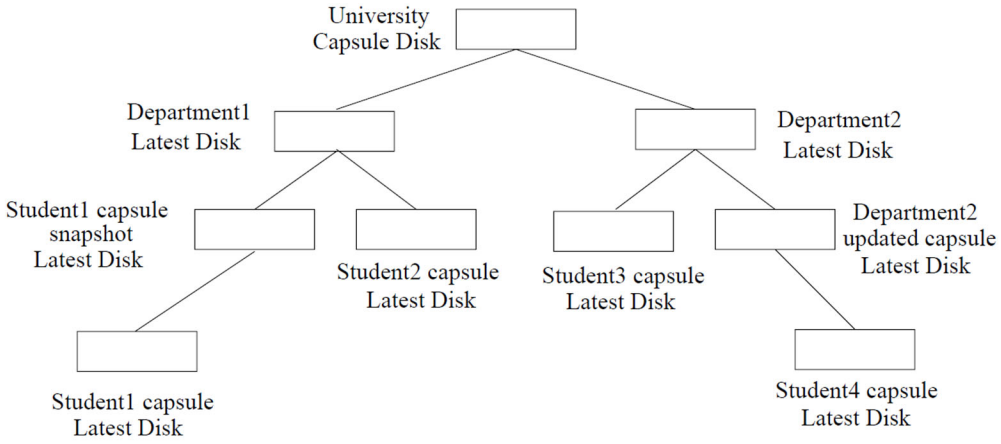


Figure 1: An example capsule hierarchy.

Each COW disk is implemented as a bitmap file and a sequence of extent files (a sequence of blocks of the COW disk). The bitmap file contains one bit for each 16KB block on the disk, indicating whether the block is present in the COW disk. Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Ex.1005, 6. Because the root COW disk contains a copy of all blocks, the search is guaranteed to terminate. *Id.*, 6-7. Ex.1003 ¶¶233-34.

Figure 2 shows an example capsule disk and the chain of COW disks that comprise the capsule disk. Notably, none of the COW disks (from "COW Disk 1" to the "Latest COW Disk") contain any unchanged blocks of the root disk. Nor is

the COW disk hierarchy visible to the user, as they see a normal flat disk as illustrated in the figure. Ex.1005, 7. Ex.1003 ¶¶235-36.

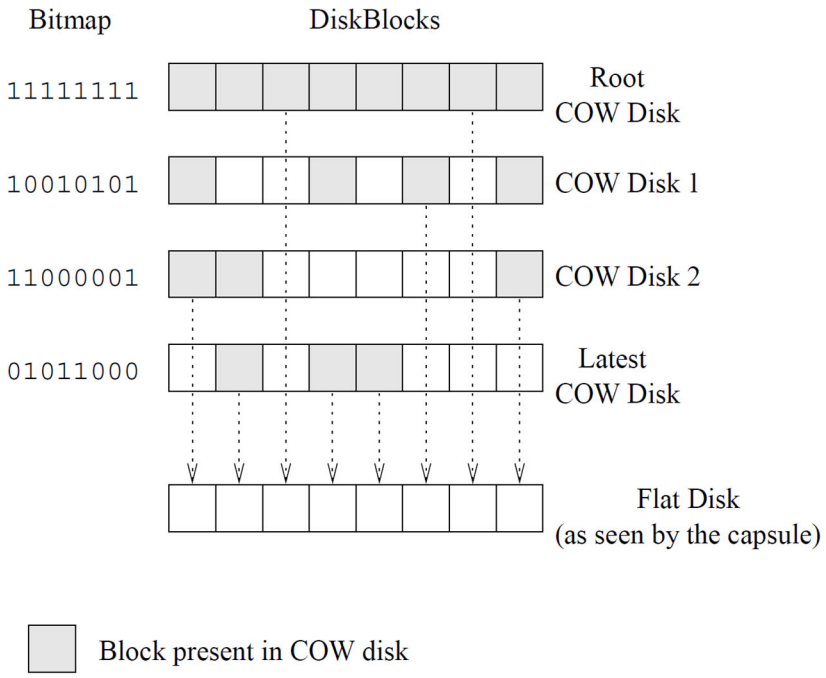


Figure 2: An example capsule disk.

To further reduce the capsule start-up time, the COW disks are read page-by-page on demand, rather than being pre-fetched. Demand paging is useful because COW disks can be extremely large; and, during a typical user session, the working set of disk blocks needed is a small fraction of the total blocks on the disk. Ex.1005, 7. Ex.1003 ¶¶237-38.

The implementation of the system, including demand paging, is shown in Figure 3. Each COW disk can be local or remote, with each remote COW disk having a corresponding local shadow COW disk that contains all locally cached blocks of the remote COW disk. Because the latest COW disk is local, all writes are local.

Reads can be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW. Ex.1005, 7. Ex.1003, ¶¶237-38.

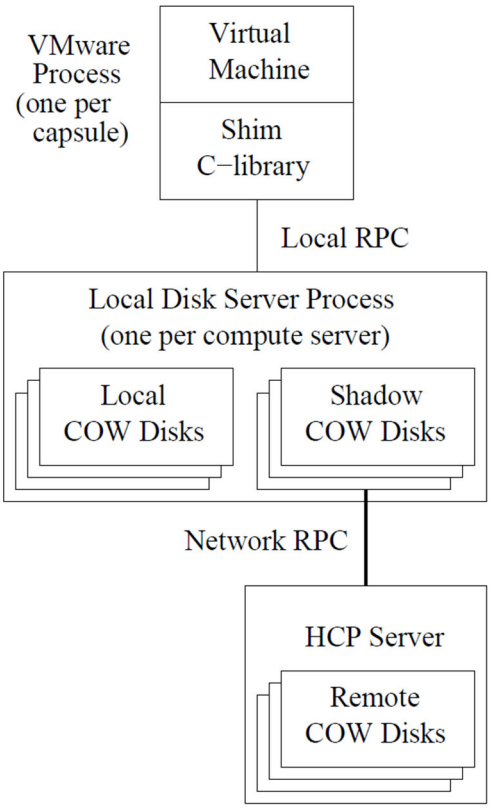


Figure 3: Implementation of capsule disks and demand paging.

B. Birse (Ex.1006)

Birse relates to the provision of an OS in a net-booted environment. Ex.1006, 1:18-21. According to one embodiment, a server in a network computer ("NC") system maintains a copy of the OS that cannot be corrupted by ordinary users of the NC system. Additionally, the NC system may preserve user customizations (*e.g.*, preferences) by maintaining individual, user, storage areas. When an NC client boots

from the network and accesses a stored copy of the OS from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. *Id*., 2:56-65. Advantageously, because of these features, a user may login to any NC client on the network and have the same user experience. *Id*., 2:65-3:3. Ex.1003, ¶¶275-85.

The NC system includes an NC client 150 (*e.g.*, personal computer) coupled to an NC server 170 via data communication line 140. Ex.1006, 4:65-5:8. NC server 170 can include one or more system volumes 174 (including a protected, read-only, master copy of the OS software), one or more application volumes 176, a user registry 178, non-persistent client-specific data 184 (temporary data storage), and persistent user data 186 (long-term storage for user information, *e.g.*, preferences, browser bookmarks, other desktop environment customizations). *Id*., 5:9-29, 6:43-48. Ex.1003, ¶¶286-89.

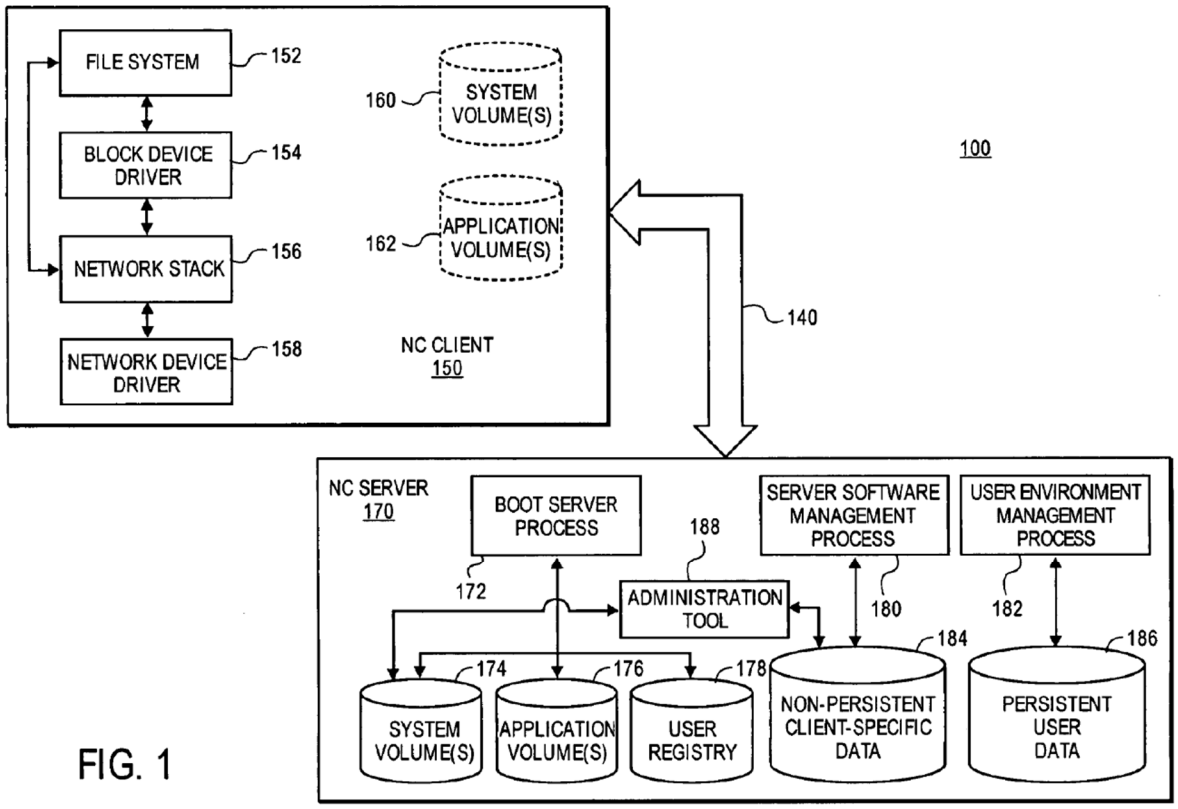


FIG. 1

The NC server also includes a boot server process 172 (managing access to the system volumes 174 and the application volumes 176), a server software process 180 (managing access to the non-persistent client-specific data 184), and a user environment process 182 (maintaining the persistent user data 186). Ex.1006, 5:46-6:34. Ex.1003, ¶¶290-94.

The NC client also includes a file system 152 (logical structures and software routines to control access to and from local and remote storage media), a block device driver 154 (servicing file system read/write requests to the system volume 160 or shadow volumes, discussed below), a network stack 156 (used to access the

data communication link 140), and a network device driver 158. Ex.1006, 7:6-60. Ex.1003, ¶¶295-299.

When an NC client boots from the network and accesses the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Ex.1006, 2:62-65. In one embodiment, one or more shadow volumes is created from the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. *Id*., 7:41-46. Ex.1003, ¶¶300-306.

C. Holzmann (Ex.1007)

Holzmann is directed to a system and method for booting multiple servers or other external devices from a storage area network using a single OS image. Ex.1007, [0017]. In Figure 1, a solid-state disk system 101 is disclosed having an internal architecture of component devices combined to achieve its purposes. Ex.1007, [0017-20]. Ex.1003, ¶¶244-47.

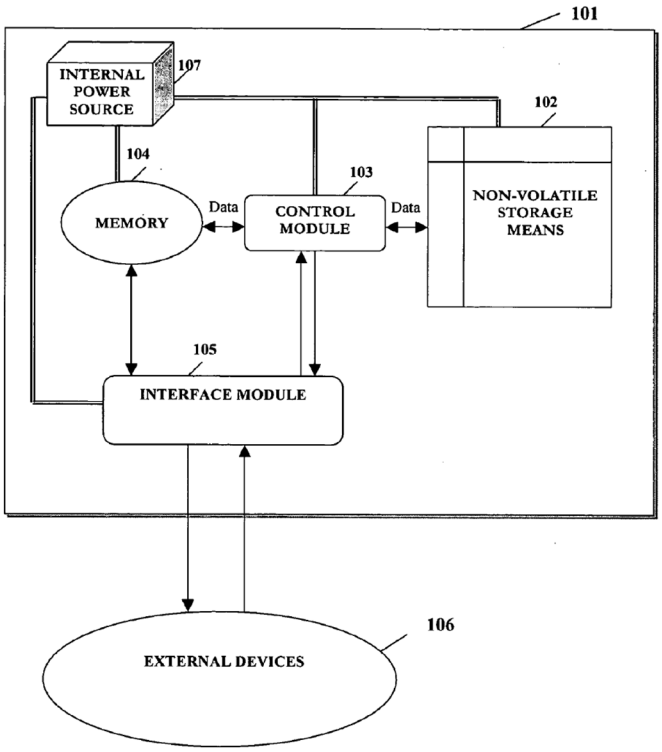


Fig. 1

When a boot request is received from an external device, the control module 103 allocates cache space in the volatile cache memory portion 104 of the disk. The cache is used in the typical manner to optimize future access to data once it has demonstrated some likelihood of being further referenced. As additional servers or other network resources are booted, the cache is used to speed up access to the most commonly used data. Ex.1007, Abstract, [0021], [0024]. Ex.1003, ¶¶248-57.

D. Federwisch (Ex.1008)

Federwisch discloses mirroring changes between source and destination snapshots by comparing two snapshots and identifying differences in "the logical file block index of each snapshot." The system compares the indexes to file block

numbers between two snapshots to identify changed data. The "scanner searches the index for changed base/incremental inode file snapshot blocks, comparing volume block numbers or another identifier…. Hence, scanned blocks at the same index in both snapshots can be efficiently bypassed…." Ex.1008, Abstract, [0071-72]. Ex.1003, ¶¶258-65.

Each snapshot (the "leaf image") has its own index of file blocks. Ex.1008, [0109]. The system performs re-indexing when updating destination snapshots' index maps. *Id*., [0134], [0137]. It uses NVRAM and buffer caches for both snapshots and active file systems. *Id*., [0048], [0053], [0058], [0070]. Ex.1003, ¶¶266-70.

Data is shared ("exported" over a network), thus "allowing source and destination systems of different vendors to share the information." Ex.1008, [0015], [0086], [0120]. Ex.1003, ¶¶271-74.

VII. GROUND 1: CLAIMS 1-13 ARE OBVIOUS OVER SAPUNTZAKIS AND HOLZMANN

A. Motivation to Combine

Both Sapuntzakis and Holzmann are concerned with the management of whole OS images being shared by users on a network, particularly efficient storage/transfer of such images. Both disclose caching blocks of the root/boot disk images to improve efficiency. Further, a POSITA would have been motivated to modify Sapuntzakis with Holzmann's caching techniques, including because, while Sapuntzakis discloses caching, a POSITA would have known there are many

different caching techniques with associated benefits and drawbacks (several disclosed in Holzmann) – and Holzmann details how to cache data for each OS (*e.g.*, Sapuntzakis' virtual machine).

Moreover, caching data from comparatively slow storage devices in faster devices (including solid-state memory) was a well-known technique commonly used for decades before the '844 Patent. Modifying Sapuntzakis with Holzmann's caching teachings would have been simple to a POSITA, and would not have required extensive experimentation or imposed undue burden – especially when both disclose caching. For the same reasons, the combination would have yielded predictable results (as caching is well-understood to improve performance and resource utilization). A POSITA would also have a high expectation of success, as Holzmann merely describes caching in greater detail than Sapuntzakis. Ex.1003, ¶¶336-55.

B. Sapuntzakis and Holzmann meet all claim elements.

1. Claims 1 and 7

Claims 1 and 7 recite substantially overlapping elements and are discussed together. *See* Ex.1018.

> **[1.0]/[7.0]** "**[1.0:A system][7.0:A method]** for providing data to a plurality of compute nodes"

Sapuntzakis discloses a system and method for providing data (capsules) to a plurality of compute nodes (user computers): "Computers and storage in the Collective system act as caches of capsules. As users travel, the Collective can move

their capsules to computers close to them, giving users a consistent environment. Capsules could be moved with users as they commute between home and work. Capsules can be duplicated, distributed to many different machines, and updated like any other data; this can form the basis for administering a group of computers." Ex.1005, 2. Ex.1003, ¶¶311,409.

> **[1.1]/[7.1]** "**[1.1:a first storage unit configured to store][7.1:storing]** blocks of a root image of said compute nodes **[7.1:on a first storage unit]**"

Sapuntzakis discloses storing blocks of a root image (root disk) of said compute nodes (computers) on a first storage unit (remote COW disks on HCP server of Figure 3): "At the root of the hierarchy is a *root disk*, which is a complete capsule disk." Ex.1005, 5.

The root capsule "contains all the software available to all students." Ex.1005, 5. Figures 1-3 (reproduced below) show the root COW disk:

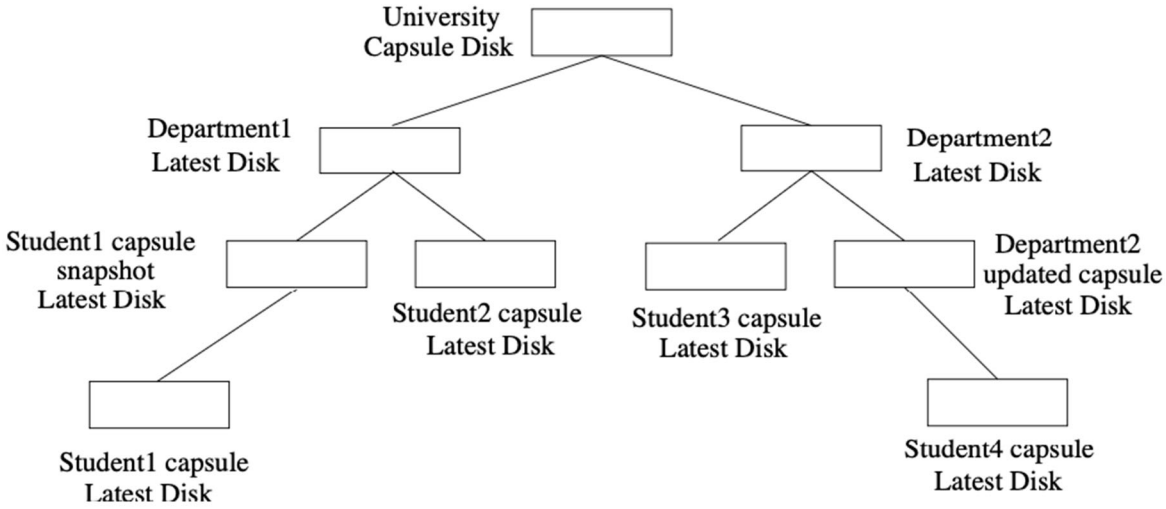


Figure 1: An example capsule hierarchy.

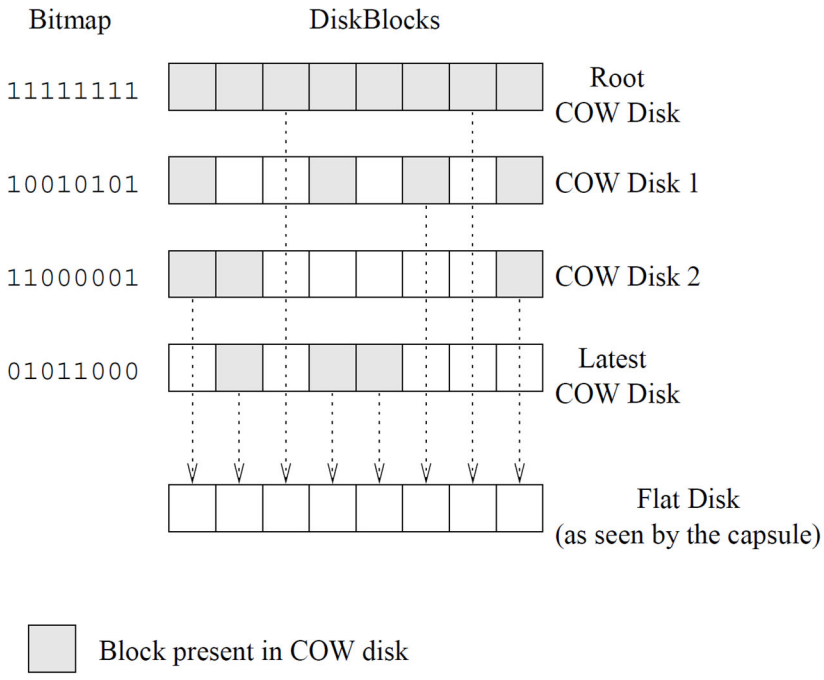


Figure 2: An example capsule disk.

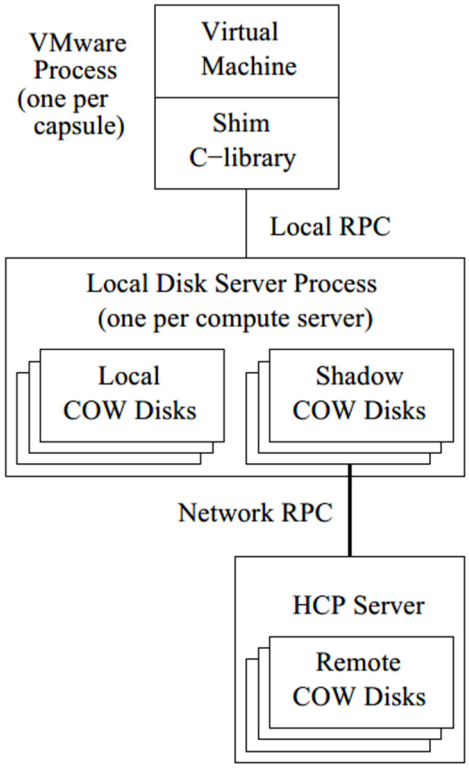


Figure 3: Implementation of capsule disks and demand paging.

A POSITA would have understood that the remote COW Disk images of Figure 3 would have been stored on a first storage unit of the HCP server because it is the central server hosting these images. Ex.1003, ¶¶312-314,410.

> **[1.2]/[7.2]** "[**1.2:a plurality of second storage units configured to store][7.2:storing]** leaf images **[1.2:of][7.2:for]** respective compute nodes **[7.2:on respective second storage units]**, said leaf images including only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of **[1.2:said][7.2:the]** root image, wherein said leaf images of

respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes"

Sapuntzakis discloses storing leaf images (the local COW disk images of Figures 2 and 3) for respective compute nodes on respective second storage units (attached either to the HCP server or the client computer), said leaf images including only additional data blocks not previously contained in said root image (write requests for new data) and changes made by respective compute nodes to the blocks of the root image (write request for changes to operating system data), wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes (only new data and changes are written): "We can store the disks in these capsules efficiently by creating a hierarchy, where each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW) virtual disk." Ex.1005, 5; Ex.1003, ¶¶316,411.

The local COW disk images holding the leaf images would have been stored on respective second storage units attached to the (local) client computers: "Writes to a capsule disk are performed by writing the data to the latest COW disk and updating its bitmap file. Reads involve searching the latest COW disk and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Ex.1005, 5-

6. *See* FIG. 2, showing how only modified blocks of data (leaf images) are stored in COW Disk 1, COW Disk 2, and so on.

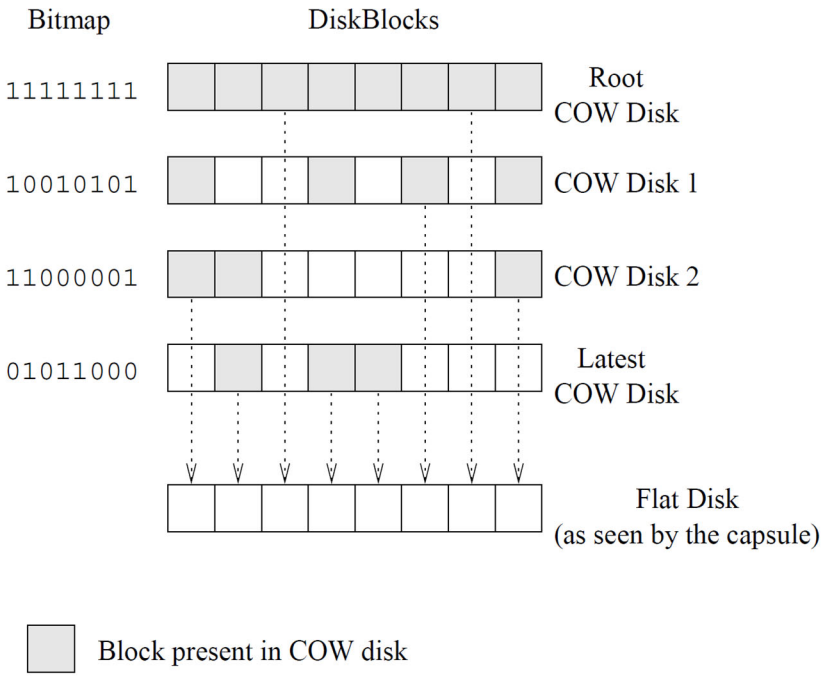


Figure 2: An example capsule disk.

A POSITA would have understood that, because writes to the local (latest) COW disk are described to occur only in these circumstances (when existing blocks are changed or overwritten), the local COW disk will include only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image and will not include blocks of the root image that are unchanged by respective compute nodes. Ex.1003, ¶¶317-320,411.

As explained by Dr. Zadok, such standard operations of CoW snapshots as described in Sapuntzakis were well-known prior to the '844 Patent. *See supra* §VI. Ex.1003, ¶¶107-138.

[1.3]/[7.3] "**[1.3:a cache configured to cache][7.3:caching]** blocks of said root image **[1.3:previously][7.3:that have been]** accessed by at least one of said compute nodes **[7.3:in a cache memory].**"

The Sapuntzakis-Holzmann combination discloses using a cache to cache blocks of said root image (the remote COW disks) that have been previously accessed by at least one of said compute nodes in a cache memory (the shadow COW disks). Ex.1003, ¶¶321-35,412.

Sapuntzakis states: "Each COW disk can either be local or remote. Each remote COW disk has a corresponding local *shadow* COW disk which contains all the locally cached blocks of the remote COW disk." Ex.1005, 6.

Sapuntzakis further states that "[r]eads, on the other hand, could either be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW." Ex.1005, 6.

Sapuntzakis also states "[f]irst, the memory in most systems caches disk blocks." Ex.1005, 7. Ex.1003, ¶¶316-320,411.

Holzmann "is directed to a system and method for booting multiple servers using a single operating system image located on a solid state disk in a storage area network." Ex.1007, [0002].

While referred to as servers in Holzmann, a POSITA would have understood there is no meaningful difference between compute nodes and servers and these

correspond to the compute nodes of the ’844 Patent and the HCP servers and clients in Sapuntzakis. Holzmann’s Figure 1 (reproduced below) illustrates this:

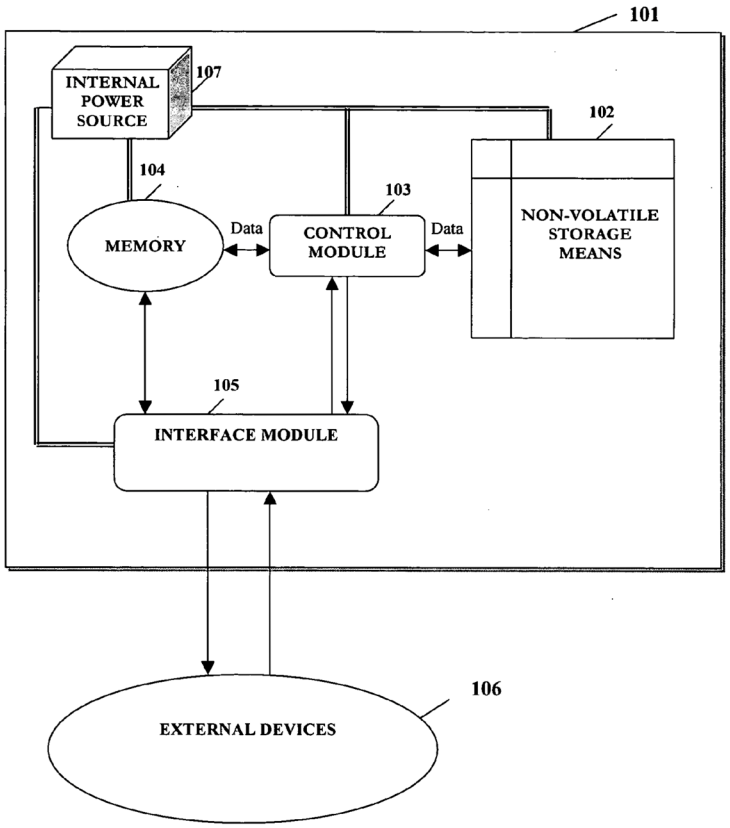


Fig. 1

“The memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106. The memory module 104 is more quickly accessible, and performs read and write processing functions more quickly, than non-volatile or disk memory, such as storage means 102. The memory module 104 preferably comprises at least one random-access memory (RAM) module.” *Id*., [0020]. Ex.1003, ¶¶325-28,412.

A POSITA would have recognized this passage as describing a cache memory because RAM is often used as a fast volatile cache and (SSD) storage devices are used as relatively fast non-volatile caches. Ex.1003, ¶¶86-101,329,412.

"Upon receiving a boot request, the control module 103 loads the required portion of the operating system image (the portion loaded typically depends on the operating system and its boot process) into memory module 104, …. Portions of the cache may be assigned statically for each server or other external devices 106. Alternatively, and preferably, the portions of the cache attributed to each server may be assigned dynamically…." Ex.1007, [0021].

"The external device 106 requests operating system data through the interface module 105. […T]he control module 103 monitors the request for operating system data, and determines whether the particular operating system data block requested by the external devices 106 has previously been loaded into the operating system cache. If the data block has been loaded into the cache, and that data block has not been overwritten…, the requested data block is provided from the cache…." *Id*., [0024].

"If a second server or other external device sends a boot request to control module 103, the module allocates additional space in the cache for a second cache 402…. If additional servers require boot operations, additional cache space 403 can [be] allocated for those servers. By allocating separate cache space for each server

or other network resource, the inventive system prevents corruption of data, overwriting of data, and other harmful actions that would lead to resource malfunction." *Id*., [0033]. Ex.1003, ¶¶330-32,412.

A POSITA would have understood that, when the cache of Holzmann is used to cache blocks of the read-only shared OS image 660 of Sapuntzakis, there is no need to create separate caches for each client because the read-only image is not subject to "corruption of data, overwriting of data, and other harmful actions that would lead to resource malfunction." Ex.1007, [0033]. That is, read-only data can be easily shared with multiple users and they will all see the same consistent data. Ex.1003, ¶333.

Finally, "[i]f the requested data has not been loaded into the cache 501, the control module 103 copies the data from the operating system data stored in memory portion 400 to the cache 501." Ex.1007, [0034]-[0035]. Ex.1003, ¶¶334,412. The Sapuntzakis-Holzmann combination renders claims 1 and 7 obvious. Ex.1003, ¶¶335,412.

2. Claim 2

[2] "wherein said cache is configured to store X most recently accessed blocks of said root image, and wherein X represents a cache threshold value."

Sapuntzakis and Holzmann further disclose said cache being configured to store X most recently accessed blocks of said root image, and wherein X represents

a cache threshold value. Moreover, this limitation describes nothing more than common caching behavior (freeing some items from a cache whose size is by definition limited to make room for new items called a "cache eviction" policy). Ex.1003, ¶414.

Sapuntzakis states that "[o]n a miss, the cache adds the entry to the bucket, possibly evicting an existing entry. Each entry contains a use count that the cache increments on every hit. When adding an entry to the cache, the hash cache chooses a fraction of the entries at random from the bucket and replaces the entry with the lowest use count; this evicts the least used and hopefully least useful entry of the group." Ex.1005, 8.

Holzmann states that "[i]f the requested data has not been loaded into the cache 501, the control module 103 copies the data from the operating system data stored in memory portion 400 to the cache 501." Ex.1007, [0035].

Moreover, "[i]t will be appreciated by those skilled in the art that other cache configurations may be used in accordance with the present invention, such as a set associative cache. Ex.1007, [0037]. "It will be understood by those skilled in the art that still more cache configurations can be used in accordance with the invented system, including additional set-associative caches or fully associative caches." Ex.1007, [0038].

A POSITA would have understood this limitation describes nothing more than common caching behavior (cache management policies, where caches naturally have limited room and must use thresholds to decide what to keep/evict). Ex.1003, ¶¶413-19.

3. Claim 3

[3] "wherein said first storage unit, said second storage units, and said cache are contained within a single storage appliance."

A POSITA would have understood that much of Sapuntzakis is focused on transferring data over comparatively slow connections, such as DSL or even dialup modems. Ex.1005, 2. In settings where communication speed between server and user computers was not the dominating concern (*i.e.*, for students, faculty and administrators working in offices, labs or dorms on the university network), it could be advantageous to store the "local" COW disks (the leaf images) on the same or similar servers as the root images.

The same is true of the cache "shadow" COW disks. Sapuntzakis recognizes that "[w]e expect to find identical blocks of data between disk images and memories, even across different users' capsules['s caches]." Ex.1005, 8. Thus, a shared cache in a centralized storage appliance would be advantageous.

Holzmann discloses a cache mechanism for server drives. "The control module 103 then allocates a sub-portion of the memory portion to be used as a cache

for the operating system. As multiple servers or other external devices 106 are booted, the control module 103 allocates additional space into the cache stored in memory module 104. Portions of the cache may be assigned statically for each server or other external devices 106. Alternatively, and preferably, the portions of the cache attributed to each server may be assigned dynamically, as described further below with reference to FIG. 5." Ex.1007, [0021].

A POSITA would have understood that this would also improve the cost/benefit of the use of high-speed solid state, persistent disk-centered memory caches (of the sort present in certain embodiments of Holzmann). For example, a POSITA would have known that storage systems can be in single storage appliances; where each such appliance supports multiple volumes, multiple base images, multiple snapshots, and multiple caches (both volatile and non-volatile). Thus, Sapuntzakis alone or combined with Holzmann discloses these elements. Ex.1003, ¶¶420-25.

4. Claims 4 and 11

[4]/[11] "**[4:a plurality of union block devices configured to interface between respective compute nodes and said first storage unit, respective second storage units, and said cache to distribute application environments to the compute nodes, wherein said union block devices are configured to create said application environments by]** merging the blocks of said root image with the

blocks of respective leaf images **[11:to create cohesive respective application environments]**."

Sapuntzakis further discloses merging the blocks of said root image (the root COW disk) with the blocks of respective leaf images (the local COW disk and ancestors between it and the root) to create cohesive respective application environments. Moreover, this limitation describes nothing more than common snapshotting behavior (traversing snapshots down to a base image and logically merging all blocks to produce a cohesive image that can be mounted as a file system for applications to use). Ex.1003, ¶391.

"To reduce the start-up time of a capsule, the COW disks corresponding to a capsule disk are read page-by-page on demand, rather than being pre-fetched." Ex.1005, 6.

"Reads involve searching the latest COW disk[, the leaf image,] and its ancestor disks in turn until the required block is found. Since the root COW disk contains a copy of all the blocks, the search is guaranteed to terminate." Ex.1005, 5–6.

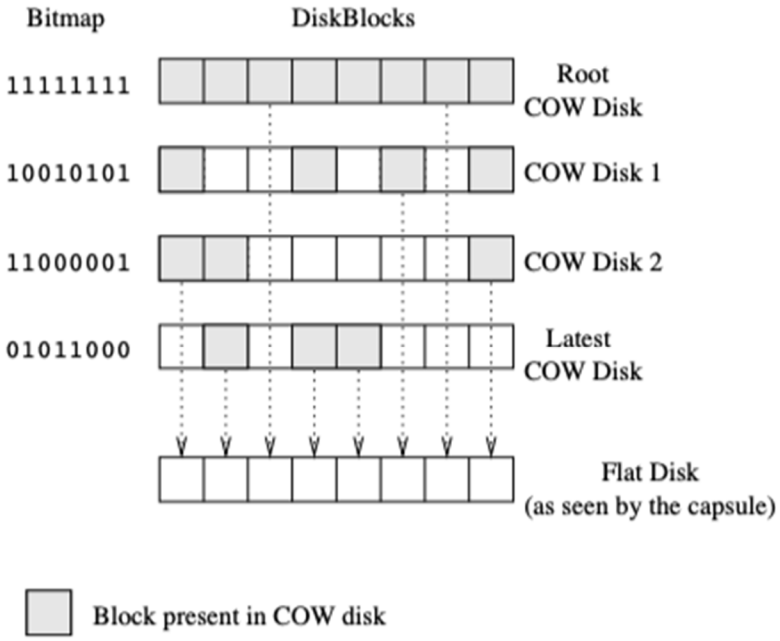


Figure 2: An example capsule disk.

Sapuntzakis, Figure 2

"Note that the COW disk hierarchy is not visible to the VMM, or to the OS and applications inside the capsule; all of them see a normal flat disk as illustrated in the figure." Ex.1005, 6.

A POSITA would have understood that none of the individual COW disks in the hierarchy are visible to the client user; instead, a single cohesive (logical) storage image corresponding to the '844 Patent's merged root and leaf blocks presenting cohesive respective application environments. Ex.1003, ¶¶390-96,426-28.

The UBD is informally defined in the '844 Patent by way of a set of tasks it performs. *See supra* §IV.A. A POSITA would have recognized that such system components as necessary to provide such features, taken together, embody the UBD

of this claim, and that it was well-known for storage systems to use single storage appliances. Ex.1003, ¶¶426-28.

5. Claims 5 and 12

[5]/[12] "**[5:wherein said union block devices comprise low-level drivers for interfacing][12:wherein said merging occurs at an operational level]** between **[5:the]** file systems of **[12:the]** respective compute nodes and said first storage unit, respective second storage units, and said cache **[12:memory]**."

Sapuntzakis discloses said merging occurs at an operational level between file systems of the respective compute nodes and said first storage unit (the root COW disks), respective second storage units (the local COW disks), and said cache memory (the shadow COW disks).

Sapuntzakis states that "[t]he COW disk implementation interfaces with GSX Server through a shim library that sits between GSX Server and the C library. The shim library intercepts GSX Server's I/O requests to disk image files in VMware's 'plain disk' format, and redirects them to a local disk server." Ex.1005, 6.

"The local disk server translates these requests to COW disk requests, and executes the I/O operations against the COW disks." Ex.1005, 6.

"The implementation of the capsule disk system, including demand paging, is shown in Figure 3. The shim library intercepts all of VMware's accesses to plain disks and forwards them to a disk server on the local machine. The disk server

performs a translation from a plain disk access to the corresponding access on the COW disks of the capsule." Ex.1005, 6.

A POSITA would have understood the COW disks in these passages refer to the root (base image) COW disks, the local (leaf) COW disks, and the shadow (cache) COW disks. Moreover, a POSITA would have understood that even without the presence of COW snapshots, operating systems provide a cohesive view of data that logically merges what is in the cache and what is not cached. Ex.1003, ¶¶397-403,429.

6. Claims 6 and 13

[6]/[13] "wherein said first storage unit is read-only."

Sapuntzakis further discloses the first storage unit (storing root disk images and possibly certain intermediate COW disks) being read-only.

"At the root of the hierarchy is a root disk, which is a complete capsule disk…once capsules have children, they become immutable." Ex.1005, 6 (describing Figure 1).

A POSITA would have understood that a working operating system has to be able to write some state (*e.g.*, boot and login records, application activity, swap space, and more), and capsules with children are immutable and cannot be run directly because they are read-only. Ex.1003, ¶¶404-408,430.

7. Claim 8

[8.1] "receiving a read request from at least one of said compute nodes, wherein a first portion of the data requested is currently stored in said cache memory;"

The Sapuntzakis-Holzmann combination discloses receiving a read request from at least one of said compute nodes, wherein a first portion of the data requested is currently stored in said cache memory (the shadow COW disks). Moreover, this limitation describes nothing more than common caching behavior (finding if something is cached in response to a read/write request). Ex.1003, ¶357.

Sapuntzakis states: "[r]eads, on the other hand, could either be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW." Ex.1005, 6.

Holzmann states: "the control module 103 monitors the request for operating system data, and determines whether the particular operating system data block requested by the external devices 106 has previously been loaded into the operating system cache. If the data block has been loaded into the cache, and that data block has not been overwritten subsequent to being loaded into the cache, the requested data block is provided from the cache in memory module 104 to interface module

105 for transmission to the appropriate external device 106." Ex.1007, [0024]. Ex.1003, ¶¶356-61.

[8.2] "providing said first portion of said data to said at least one of said compute nodes from said cache memory."

The Sapuntzakis-Holzmann combination discloses providing said first portion of said data to said at least one of said compute nodes from said cache memory (the shadow COW disks). Moreover, this limitation describes nothing more than common caching behavior (returning a cached item if found). Ex.1003, ¶362.

Sapuntzakis states that "[r]eads, on the other hand, could either be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW." Ex.1005, 6.

Holzmann states that "the control module 103 monitors the request for operating system data, and determines whether the particular operating system data block requested by the external devices 106 has previously been loaded into the operating system cache. If the data block has been loaded into the cache, and that data block has not been overwritten subsequent to being loaded into the cache, the requested data block is provided from the cache in memory module 104 to interface module 105 for transmission to the appropriate external device 106." Ex.1007, [0024].

A POSITA would have understood from this passage that if the requested blocks are cached, they are served from the cache as is common for caches. Ex.1003, ¶¶362-66.

8. Claim 9

[9] "updating said cache memory based on said read request."

The Sapuntzakis-Holzmann combination discloses updating cache memory (the shadow COW disks) based on said read request. Moreover, this limitation describes nothing more than common caching behavior (updating a cache based on reading a new item, called a "cache admission" policy). Ex.1003, ¶368.

Sapuntzakis states "[r]eads…could either be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW." Ex.1005, 6.

Holzmann states "[i]f the requested data has not been loaded into the cache 501, the control module 103 copies the data from the operating system data stored in memory portion 400 to the cache 501." Ex.1007, [0035].

A POSITA would have understood that if the requested blocks are not cached, they are loaded from backing store and stored in the cache for subsequent faster reading from the cache. Ex.1003, ¶¶367-72.

9. Claim 10

[10.0] "wherein a second portion of the data requested is not currently stored in said cache memory and said updating comprises:"

The Sapuntzakis-Holzmann combination discloses processing a request wherein a second portion of the data requested is not currently stored in said cache memory and said updating comprises: caching said second portion in said cache memory. Moreover, this limitation describes nothing more than common caching behavior (requesting a plurality of items from a cache). Ex.1003, ¶373.

Sapuntzakis states "[r]eads, on the other hand, could either be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW." Ex.1005, 6.

Holzmann states "the control module 103 monitors the request for operating system data, and determines whether the particular operating system data block requested by the external devices 106 has previously been loaded into the operating system cache. If the data block has been loaded into the cache, and that data block has not been overwritten subsequent to being loaded into the cache, the requested data block is provided from the cache in memory module 104 to interface module 105 for transmission to the appropriate external device 106." Ex.1007, [0024]. Ex.1003, ¶¶373-78.

[10.1] "caching said second portion in said cache memory;"

The Sapuntzakis-Holzmann combination discloses updating said cache memory (the shadow COW disks) based on said read request. Moreover, this limitation describes nothing more than common caching behavior (updating a cache based on reading a new item, called a "cache admission" policy). Ex.1003, ¶379.

Sapuntzakis states that "[r]eads, on the other hand, could either be local or remote. In the case of a remote read, the disk server requests the block from the shadow COW disk. If the block is not cached locally, it is fetched remotely and added to the shadow COW." Ex.1005, 6.

Holzmann states that "[i]f the requested data has not been loaded into the cache 501, the control module 103 copies the data from the operating system data stored in memory portion 400 to the cache 501." Ex.1007, [0035].

A POSITA would have understood that if the requested blocks are not cached, they are loaded from backing store and stored in the cache for subsequent faster reading from the cache. Ex.1003, ¶¶379-83.

[10.2] "removing the least recently accessed data from said cache memory if the amount of data in said cache memory is above a threshold value."

The Sapuntzakis-Holzmann combination discloses removing the least recently accessed data from cache memory if the amount of data in cache memory is above a threshold value. Moreover, this limitation describes nothing more than

common caching behavior (freeing some items from a cache to make room for new items, called a "cache eviction" policy). Ex.1003, ¶384.

Sapuntzakis states that "[o]n a miss, the cache adds the entry to the bucket, possibly evicting an existing entry. Each entry contains a use count that the cache increments on every hit. When adding an entry to the cache, the hash cache chooses a fraction of the entries at random from the bucket and replaces the entry with the lowest use count; this evicts the least used and hopefully least useful entry of the group." Ex.1005, 8.

Holzmann states that "[i]f the requested data has not been loaded into the cache 501, the control module 103 copies the data from the operating system data stored in memory portion 400 to the cache 501." Ex.1007, [0035].

Moreover, "[i]t will be appreciated by those skilled in the art that other cache configurations may be used in accordance with the present invention, such as a set associative cache. Ex.1007, [0037]. "It will be understood by those skilled in the art that still more cache configurations can be used in accordance with the invented system, including additional set-associative caches or fully associative caches." Ex.1007, [0038].

A POSITA would have understood this limitation describes nothing more than common caching behavior (removing some cached items to make room for others is called a "cache eviction" policy). Ex.1003, ¶¶384-89.

VIII. GROUND 2: CLAIMS 14-27 ARE OBVIOUS OVER SAPUNTZAKIS AND FEDERWISCH

A. Motivation to Combine

Both Sapuntzakis and Federwisch operate at and recognize disk blocks at the block layer (while also recognizing that file systems interoperate and work atop the block layer), disclose/use CoW snapshots (a POSITA would have known CoW techniques enable snapshots), and disclose caching blocks of the root/boot disk images. Further, a POSITA would have been motivated to modify Sapuntzakis with Federwisch's indexing – including because, while Sapuntzakis discloses that data is indexed, it does so only in connection with a hash table that indexes cached entries for faster access. A POSITA building a Sapuntzakis system, and knowing indexing of data items was a well-known technique, would have been motivated to seek additional details on indexing storage systems, files, and caches thereof (supplied by Federwisch). A POSITA would also have known that indexing of data is an age-old technique long utilized in computer systems.

Modifying Sapuntzakis to include Federwisch's indexing would have been simple to a POSITA, and would not have required extensive experimentation or imposed undue burden – including because Sapuntzakis already indexed data objects and a POSITA would not have expected Sapuntzakis to provide details of such a basic concept. For the same reasons, the combination would have yielded predictable results, as indexes are well understood to structure and locate data objects efficiently,

and a POSITA would have had a high expectation of success, because indexing techniques were well understood for decades beforehand. Ex.1003, ¶¶449-471.

B. Sapuntzakis and Federwisch meet all claim elements.

1. Claims 14, 19, and 23

Claims 14, 19, and 23 recite substantially overlapping elements and are discussed together. *See* Ex.1018. Minor differences are shown in bold.

> **[14.0]/[19.0]** "**[14.0:A system][19.0:A method]** for indexing file systems for a plurality of compute nodes"
>
> **[23.0]** "A computer-readable storage medium having instructions stored thereon that, in response to execution by at least one computing device, cause the at least one computing device to:"

As described, *supra* §§VI.A, VI.D, Sapuntzakis and Federwisch both disclose compute nodes and computing devices. As described, *supra* §VI.D, and further below Federwisch discloses indexing file systems. Ex.1003, ¶¶431,476,491.

> **[14.1]/[19.1]** "**[14.1:a first storage unit configured to store][19.1:storing]** blocks of a root image of said compute nodes **[14.1:on a first storage unit]**;"
>
> **[23.1] "**receive data blocks of a file system, said data blocks comprising a root image portion;"

These elements are substantially similar to elements recited in claims 1 and 7 regarding root images. As described, *supra* §VII.B.1, Sapuntzakis discloses these elements. Ex.1003, ¶¶432,477,492.

> **[14.2]/[19.2]/[23.2]** "**[14.2:a plurality of second storage units configured to store][19.2:storing] [14.2/19.2:leaf images of respective compute nodes] [19.2: on respective second storage units][23.2:and leaf image portion]**, said leaf **[14.2/19.2:images][23.2:image portion]** comprising only additional data blocks not previously contained in said root image **[23.2:portion]** and changes made by **[14.2/19.2:respective compute nodes][23.2:said first compute node]** to the blocks of **[14.2/23.2:said][19.2:the]** root image, wherein said **[14.2/19.2:leaf images] [14.2:of][19.2:for] [14.2/19.2:respective compute nodes do][23.2:leaf image portion does]** not include blocks of said root image that are unchanged by **[14.2/19.2:respective compute nodes][23.2:said first compute node]**;"

These elements are substantially similar to elements recited in claims 1 and 7 regarding leaf images. As described, *supra* §VII.B.1, Sapuntzakis discloses these elements. Ex.1003, ¶¶433,478,492.

> **[14.3]/[19.3]** "**[14.3:a plurality of union block devices corresponding to said compute nodes, said union block devices configured to interface between said compute nodes and said first and second storage units to distribute said file systems to said compute nodes, wherein said union block devices are configured to create said file systems by]** merging the blocks of said root image **[14.3:stored on the first storage unit]** with the blocks of respective leaf images stored on respective second storage units **[19.3:to create respective file systems for respective compute nodes]**"

[23.1] **"**wherein said file system is the result of merging said root image portion and said leaf image portion together at the block-level;"

These elements are substantially similar to elements recited in claims 4 and 11 regarding UBDs and merging. As described, *supra* §VII.B.4, Sapuntzakis discloses these elements. Ex.1003, ¶¶439,479,492.

[14.4] "wherein further at least one of said compute nodes is configured to index said root image and provide the indexing results to another of said compute nodes."

[19.4] "receiving indexing results pertaining to said root image from one of said compute nodes; and providing said indexing results to the others of said compute nodes."

[23.4] "index said root image portion; and provide the results of said indexing to a second compute node, wherein said logic encoded in the one or more tangible media comprise computer executable instructions executed by the first compute node."

The Sapuntzakis-Federwisch combination further discloses at least one of said compute nodes being configured to index said root image and provide the indexing results to another of said compute nodes.

Federwisch discloses a "system and method for remote asynchronous replication or mirroring of changes" between "a source file system snapshot" and "a destination replica file system." Ex.1008, Abstract.

Federwisch compares two snapshots "using a scan (via a scanner) of the blocks that make up two versions of a snapshot of the source file system, which

identifies changed blocks in the respective snapshot files based upon differences in volume block numbers identified in a scan of the logical file block index of each snapshot." Ex.1008, Abstract. Thus, every snapshot in a volume has a "logical file block index."

A POSITA would have understood that the scanner of Federwisch could be implemented on any of the compute nodes of Sapuntzakis, as part of the Union Block Device. As described, *supra* §§VI, VIII.A, a POSITA would further have understood that such scanner could be used to index the root image and to provide the index results to other compute nodes, because such indexing and sharing of data over networks was well known. Ex.1003, ¶¶434-37,480-81,493-94.

Sapuntzakis discloses that data is indexed. In Sapuntzakis, "[t]he cache is implemented using a hash table whose size is fixed at creation. We use the first several bits of the hash key to index into the table." Ex.1005, 7. Sapuntzakis's data structure is a hash table that indexes cached entries for faster access. "The hash key indexes into a bucket of entries, currently a memory page in size. On a lookup, the cache does a linear search of the entries in a bucket to check whether one of them matches the hash." Ex.1005, 7.

A POSITA interested in building a system based on Sapuntzakis would have been motivated to seek out additional details on indexing storage systems, files, and caches thereof. This is particularly so because indexing of data items was a well-

known technique. A POSITA would have found Federwisch relevant, which discloses greater details of indexing.

Federwisch uses common indexing to better manage cached data. "If the information is not in memory, the file system layer 450 indexes into the inode file using the inode number to access an appropriate entry and retrieve a volume block number." Ex.1008, [0053].

Federwisch consults its indexes to identify changed and unchanged blocks. "The existence of a changed volume block number at a given offset in the index between the base snapshot inode file and incremental snapshot inode file generally indicates that one or more of the underlying inodes and files to which the inodes point have been changed." Ex.1008, [0071].

Federwisch scans these indexes to identify unchanged blocks. "A scanner 820 searches the index for changed base/incremental inode file snapshot blocks, comparing volume block numbers or another identifier. … Hence, scanned blocks at the same index in both snapshots can be efficiently bypassed, reducing the scan time." Ex.1008, [0072]. Ex.1003, ¶¶438-48,480-81,493-94.

2. Claim 15

[15] "wherein said first storage unit and said second storage units are contained within a single storage appliance."

These elements are substantially similar to elements recited in claim 3. As described, *supra* §VII.B.3, Sapuntzakis discloses these elements. Ex.1003, ¶472.

3. Claim 16

[16] "a plurality of union block devices configured to interface between respective compute nodes and said first storage unit and respective second storage units, said union block devices configured to distribute application environments to the compute nodes, wherein said union block devices are configured to create said application environments by merging the blocks of said root image with the blocks of respective leaf images."

These elements are substantially similar to elements already recited in claim 14, as well as claim 4. As described, *supra* §§VII.B.4, VIII.B.1, Sapuntzakis discloses these elements. Ex.1003, ¶473.

4. Claims 17 and 21

[17]/[21] "**[17:wherein said union block devices comprise low-level drivers for interfacing][21: wherein said merging occurs at an operational level]** between **[17:the][21:respective]** file systems of **[17:respective][21:the]** compute nodes and said first storage unit, **[21:and]** respective second storage units**[17:, and said cache]**."

Petitioners note "said cache" in claim 17 lacks antecedent basis, but, even to the extent claim 17 requires a cache, these elements are substantially similar to

elements recited in claims 5 and 12. As described, *supra* §VII.B.5, Sapuntzakis discloses these elements. Ex.1003, ¶¶474,489.

5. Claims 18, 22, and 27

[18]/[22]/[27] "wherein said **[18/22:first storage unit][27:root image portion]** is read-only."

These elements are substantially similar to elements recited in claims 6 and 13. As described, *supra* §VII.B.6, Sapuntzakis discloses these elements. Ex.1003, ¶¶475,490,514.

6. Claims 20 and 24

[20]/[24] "**[20:storing said indexing results][24:wherein the instructions further cause the at least one computing device to store said results of said indexing]** on a shared storage unit **[24: accessible by said second compute node]**."

A POSITA would have known that sharing storage and files across devices was well known well before the '844 Patent. Indeed, all of the prior art references cited herein, as well as the '844 Patent, rely on central shared storage for various aspects of their respective inventions. This would have made any data (index or otherwise) conveniently accessible over network storage volumes.

Further, Federwisch recognizes the usefulness of sharing files and data in a format-agnostic manner to other systems. "More generally, the format should also be independent of the underlying network protocol or device (in the case of a tape

or local disk/non-volatile storage) protocol and file system—that is, the information is system 'agnostic,' and not bound to a particular operating system software, thereby allowing source and destination systems of different vendors to share the information. The format should, thus, be completely self-describing requiring no information outside the data stream." Ex.1008, [0086].

Being a network appliance, Federwisch supports the ability to share files, snapshots, and whole volumes across the network (*e.g.*, using NFS or CIFS). This sharing is called "an export." The "destination file system is exported read-only the user." *Id*., [0015]. This first snapshot may be an exported snapshot of the replicated snapshot on the destination. *Id*., [0120]. Ex.1003, ¶¶482-88,495.

7. Claim 25

[25] "wherein the instructions further cause the at least one computing device to index said leaf image portion."

A POSITA would have understood that when some data changes on a disk or file system, it is inefficient to scan and re-index an entire system, only to reprocess unchanged data; instead, it is more efficient to index only files that have changed. This is easily achieved because storage and file systems include timestamps that denote when the files/data were last modified. The Sapuntzakis-Federwisch combination discloses instructions further causing at least one computing device to index a leaf image portion.

Federwisch compares the indexes to file block numbers between two snapshots to identify whether data has been changed between the snapshots (as would be accomplished using the CoW techniques provided by the cited references).

"The existence of a changed volume block number at a given offset in the index between the base snapshot inode file and incremental snapshot inode file generally indicates that one or more of the underlying inodes and files to which the inodes point have been changed." Ex.1008, [0071].

Federwisch scans these indexes to identify unchanged blocks: "A scanner 820 searches the index for changed base/incremental inode file snapshot blocks, comparing volume block numbers or another identifier. … Hence, scanned blocks at the same index in both snapshots can be efficiently bypassed, reducing the scan time." Ex.1008, [0072].

A POSITA would have understood these disclosures to mean that in Federwisch each snapshot (corresponding to the '844 Patent's "leaf image") has its own index of file blocks. "More specifically, each entry in the inode map 1400 contains an entry for each inode on the source snapshot. Each inode entry 1402 in the map is indexed and accessed via the source inode number (1404)." Ex.1008, [0109]. Ex.1003, ¶¶496-505.

8. Claim 26

[26] "wherein the instructions further cause the at least one computing device to re-index said file system by re-indexing said leaf image portion and merging the results of said re-indexing of said leaf image portion with said results of said indexing of said root image portion."

A POSITA would have understood that when some data changes on a disk or file system, it is inefficient to scan and re-index an entire system, only to reprocess unchanged files; instead, it is more efficient to index only files that have changed. This is easily achieved because storage and file systems include timestamps that denote when the files were last modified. Lastly, when some data changes, for example in any form of a simple database (such as an index), a POSITA would have understood that the new index data would be easily merged with older data—this would be useful to purge now-stale index entries.

Federwisch discloses a form of re-indexing of incremental snapshots (which the '844 Patent calls "leaf" images). In one example, "the Destination Snapshot A (2002) is now prepared to transfer changes so as to establish a mirror in Destination Snapshot B (2004). … In this case, both the new source and the new destination system have their own inode maps A and B (2012 and 2014), indexed off the old source 2001, and referencing the inodes in their respective trees. … It then treats the new source as the appropriate map index for the new associated inode map 2018. In

the associated map, it stores the new source generation number for the new source index inode number, with each index entry also associated with/mapped to the new destination inode/generation number extracted from the inode map B (2014)." Ex.1008, [0137].

When snapshots in Federwisch are copied from a source to a destination, although the data has not changed, the snapshot's block locations may have changed. Thus, Federwisch updates the destination snapshot's index maps, a form of re-indexing. Ex.1003, ¶¶506-13.

IX. GROUND 3: CLAIMS 1-13 ARE OBVIOUS OVER BIRSE AND HOLZMANN

A. Motivation to Combine

Both Birse and Holzmann are concerned with efficient management of whole OS images and booting systems over a network. While Birse discloses the presence of cache as a feature on some embodiments of the system's server and client computers, it lacks detail on how caching may be applied to its boot images; while Holtzman discloses details of caches for each system/server being booted, as well as different cache configurations. Further, a POSITA would have been motivated to modify Birse with Holzmann's caching techniques, including because, while Birse discloses typical caching, Holzmann offers details of how to cache data for the OS distribution purpose (*e.g.*, Birse's boot images). Moreover, caching data from

comparatively slow storage devices in faster devices (including solid-state memory) was a well-known technique commonly used for decades before the '844 Patent.

Modifying Birse with Holzmann's caching teachings would have been simple to a POSITA, and would not have required extensive experimentation or imposed undue burden – especially when both disclose caching. For the same reasons, the combination would have yielded predictable results (as caching is well-understood to improve performance and resource utilization). A POSITA would also have a high expectation of success, as Holzmann merely describes caching in greater detail than Birse. Ex.1003, ¶¶537-555.

B. Birse and Holzmann meet all claim elements.

1. Claims 1 and 7

[1.0]/[7.0] "**[1.0:A system][7.0:A method]** for providing data to a plurality of compute nodes"

Birse discloses a system and method for providing data (from NC server 170) to a plurality of compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826): "…[T]he NC clients [150] boot over the network by accessing a stored copy of the operating system from the NC server [170]. Additionally, when NC clients want to run an application, they access it from the NC server." Ex.1006, 4:56-60. *See also id.*, 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications). This is illustrated in Birse's Figure 1, reproduced below:

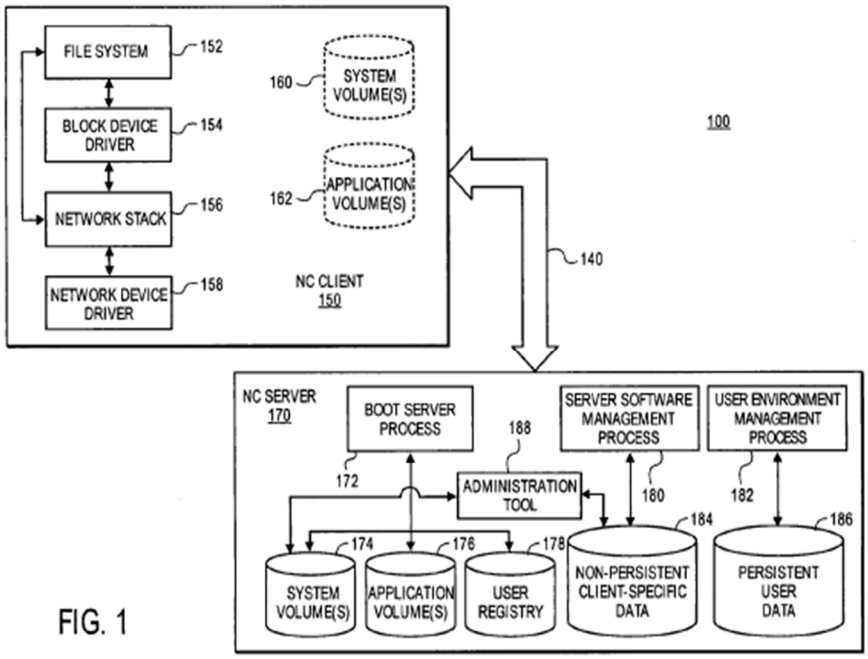


FIG. 1

"FIG. 8 is a simplified block diagram illustrating two NC systems 810 and 820 coupled via the Internet 805." *Id*., 14:8-9. Birse's Figure 8 is reproduced below:

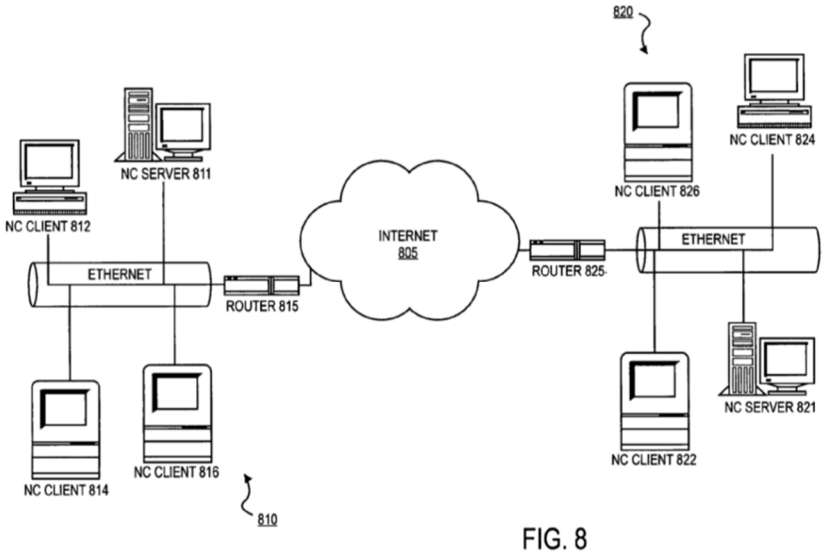


FIG. 8

Further, NC server 170 and NC clients 150 are illustrated by exemplary digital processing system 200 of FIG. 2. As described at 7:63-8:35, they disclose all essential elements of a compute node. *Id.*, 4:27-57. Birse's Figure 2 is reproduced below:

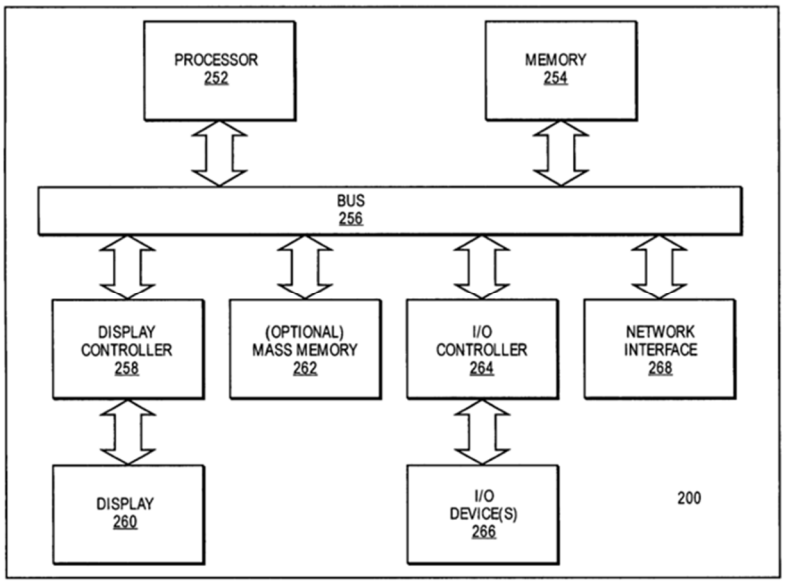


FIG. 2

Ex.1003, ¶¶516-519,579.

[1.1]/[7.1] "**store/storing** blocks of a root image…"

Birse discloses storing blocks of a root image (system volumes 174, 660, 922, 924) of said compute nodes (multiple NC servers 170) on a first storage unit (one or more hard drives 610, logically separate in NC folders/directories 630, 640): "The NC server 170 includes one or more system volumes 174…. The system volumes

174 include a protected, read-only, master copy of the operating system software." Ex.1006, 5:10-14.

"[T]he NC server 170 contains a read-only core system volume image 922 and a read-write user system volume image 924 …. The core system volume 922 preferably contains those parts of the system that do not need to be written back to during system operation. One goal of the core system volume 922 is to provide all the system components that are mandatory for system operation…. The user system volume 924 contains all the user-configurable system components, including preferences and all the system additions installed by application software, such as application-installed extensions and libraries." *Id*., 14:27-34, 14:41-44.

"[T]he system volumes 174…may be independently distributed across multiple server computer systems." *Id*., 6:59-63.

"FIG. 6 is a simplified block diagram illustrating the layout of an exemplary hierarchical directory structure that may be used by an NC server [170] according to one embodiment…. [A] hard drive 610 of the NC server 170… includes an NC folder (directory) 620. Within the NC folder 620 are an NC admin folder 630 and an NC shared data folder 640." *Id*., 12:30-38. Birse's Figure 6 is reproduced below:

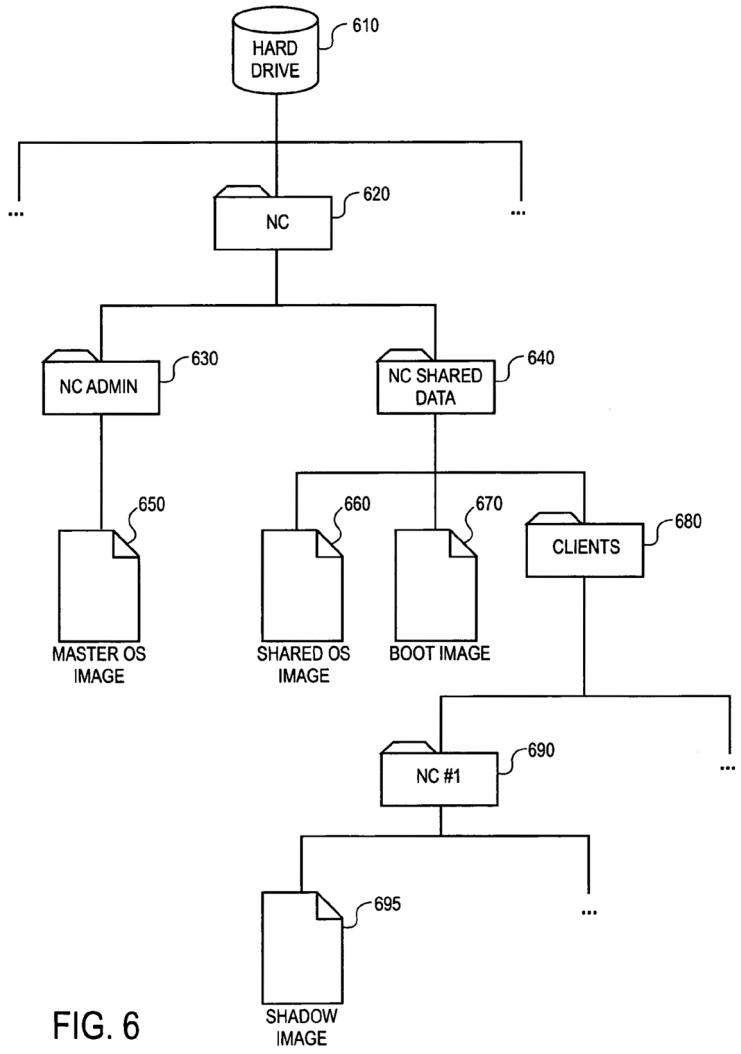


FIG. 6

Birse discloses that the file 660, storing the shared OS image (within the folder 640), is a first storage unit storing blocks of a root image. "The shared operating system image 660 is a read-write version of the read-only master operating system image 650 and may be mounted by the NC client 150 as part of step 395." *Id*., 12:44–47.

The shared system volume 660 (a.k.a. system volume 174, 922, 924, 1010) is stored as blocks as shown in Fig. 10, reproduced below:

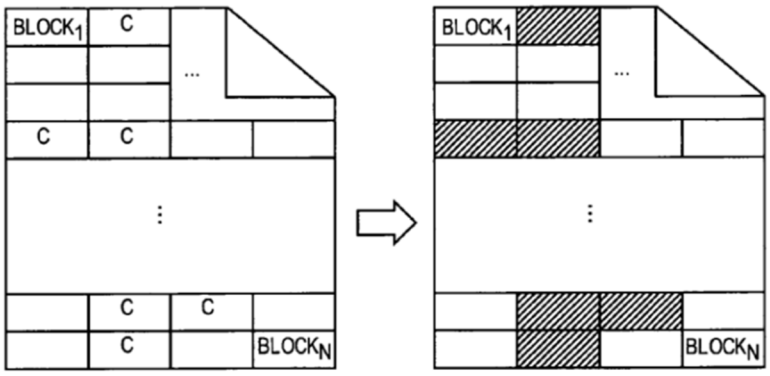


FIG. 10

Ex.1003, ¶¶520-26,580.

[1.2]/[7.2] "store/storing] leaf images…"

Birse discloses storing leaf images (shadow volumes/images 930, 695) for respective compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826) on respective second storage units (NC client folders 690 #1…n), said leaf images including only additional data blocks not previously contained in said root image (write requests for new data) and changes made by respective compute nodes to the blocks of the root image (write request for changes to operating system data), wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes (only new data and changes are written): "In this example, the NC clients are numbered from 1 to N and the clients folder 680 includes a folder for each NC client. For example, an NC #1 folder 690 is an area for storing client-specific information, such as a shadow image 695, corresponding to NC client #1." Ex.1006, 12:54–59.

"The block device driver 154 services file system read and write requests to the system volumes 160. In one embodiment, the block device driver 154 appears to the file system 152 to be a standard hard drive device driver … [but is] configured to service the read and write requests of the file system 152 by accessing the NC server 170. As will be discussed further below, in one embodiment, one or more shadow volumes [695], may be created for the NC client 150 and stored with the non-persistent client-specific data 184 for the purpose of storing user preferences and/or changes to the operating system. Therefore, some mechanism is needed to direct reads and writes to the appropriate volume, e.g., the shadow volumes or the system volumes 174. According to one embodiment, the block device driver 154 has knowledge of the separate shadow volumes and system volumes 174 and redirects read and write requests to the appropriate volume. Alternatively, server-side logic may perform the redirection locally." *Id*., 7:35-53; *see also id.*, 11:30-32.

"FIG. 4 is a flow diagram illustrating how NC client [150] write requests directed to the system volumes 160 are handled…. [A]t step 410, the file system 152 generates a write request directed to the system volumes 160. The write request is received by the block device driver 154 and redirected by the block device driver 154 at step 420 by translating it into a write request directed to the user's shadow volume on the NC server 170. At step 430, the NC server 170 stores information associated with the write request in the shadow volume associated with the NC client

150. In this manner, the file system 152 need not be aware that attempted modifications to the system volumes 160 are recorded instead in a client-specific shadow volume residing on the NC server 170." *Id*., 10:63-65, 11:6-17. Birse's Figure 4 is reproduced below:

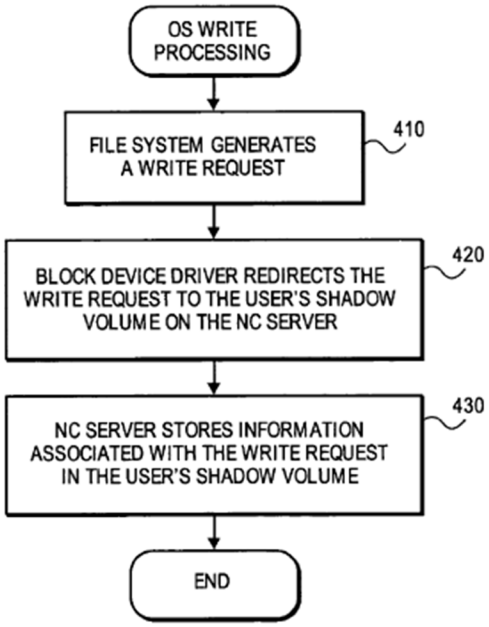


FIG. 4

The "shadow volume contains only those portions of the operating system that have been modified by the user rather than a complete copy of the operating system as modified." *Id*., 11:49-52.

In another embodiment, the shared OS image 660 is divided into two parts: "[t]he NC server 170 also creates a shadow system volume 930 for each connected

NC client. The shadow system volume 930 shadows the user system volume 924 by storing modifications that are made to the user system volume 924." *Id*., 14:49-53.

Moreover, "… the shadow system volume 930 is used by the block device driver 154 of the NC client 150 to implement a 'copy-on-write' storage scheme in which modifications (writes) directed to the user system volume 930 are instead copied to the shadow system volume 930." *Id*., 14:56-61. Ex.1003, ¶¶527-33,581.

[1.3]/[7.3] "cache/caching blocks of said root image…"

The Birse-Holzmann combination discloses caching blocks of said root image (system volumes 174, 660, 922, 924) that have been accessed by at least one of said compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826) in a cache memory. Ex.1006, 4:56-60, 5:9-29, 6:43-48; Ex.1007, Abstract, [0021], [0024]. *See also supra* §§VI.C, VII.B.1 (describing Holzmann's caching disclosures). Ex.1003, ¶¶534-36,582

As described, *supra* §IX.A, it would have been obvious to a POSITA to combine the cache system of Holzmann with the system of Birse to improve the performance of Birse. Ex.1003, ¶¶534-36,582.

2. Claim 2

[2] "store X most recently accessed blocks…"

As also described, *supra* §VII.B.2, this limitation describes nothing more than common caching behavior (freeing some items from a cache whose size is by

definition limited, to make room for new items and Holzmann states that "[i]f the requested data has not been loaded into the cache 501, the control module 103 copies the data from the operating system data stored in memory portion 400 to the cache 501." Ex.1007, [0035].

Moreover, "[i]t will be appreciated by those skilled in the art that other cache configurations may be used in accordance with the present invention, such as a set associative cache." Ex.1007, [0037]. "It will be understood by those skilled in the art that still more cache configurations can be used in accordance with the invented system, including additional set-associative caches or fully associative caches." Ex.1007, [0038].

A POSITA would have understood this limitation describes nothing more than common caching behavior (cache management policies, where caches naturally have limited room and must use thresholds to decide what to keep/evict). Ex.1003, ¶¶583-588.

3. Claim 3

[3] "…within a single storage appliance."

Birse discloses the first storage unit (holding the core system volumes and the user system volume), and the second storage units (holding the shadow system volumes) contained within a single storage appliance: "FIG. 6 is a simplified block diagram illustrating the layout of an exemplary hierarchical directory structure that

may be used by an NC server according to one embodiment of the present invention." Ex.1006, 2:24–27.

"According to the embodiment depicted, a hard drive 610 of the NC server 170 or other mass storage device associated with the NC server 170 includes an NC folder (directory) 620." Ex.1006, 12:30–37. Birse's Figure 6 is reproduced below:

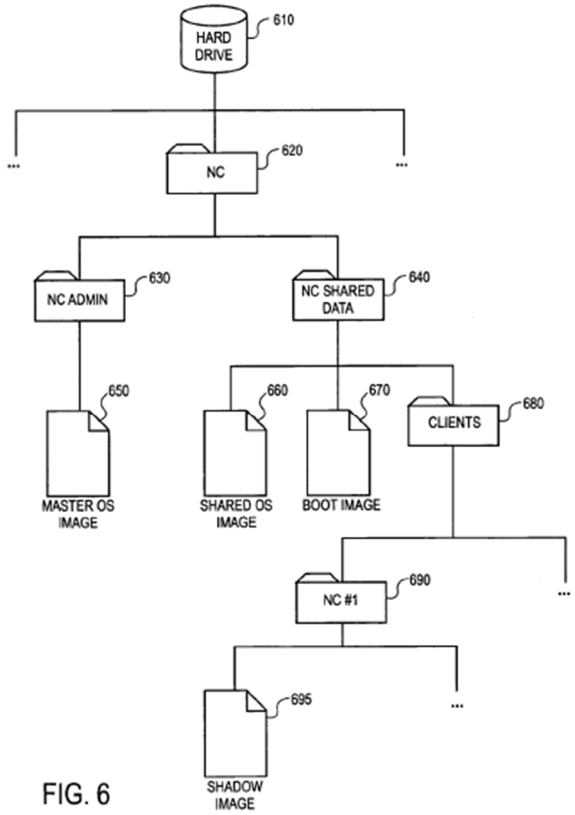


FIG. 6

As already described, *supra* §VII.B.3, Holzmann discloses a cache mechanism for server drives. Ex.1007, [0021].

It would have been obvious to a POSITA to implement the cache system of Holzmann on the storage device of Birse holding the first and second storage units. This would also improve the cost/benefit of the use of high-speed solid-state,

persistent disk-centered memory caches (of the sort present in certain embodiments of Holzmann). Ex.1003, ¶¶589-593.

4. Claims 4 and 11

[4]/[11] "…merging the blocks of said root image with the blocks of respective leaf images…"

Birse discloses merging the blocks of said root image (blocks of the operating system software stored on system volumes 174, 660, 922) with the blocks of respective leaf images (695) to create cohesive respective application environments (dynamically merged system environment): "In one embodiment, the shadow system volume 930 is not visible to the user. Consequently, the user system volume 924 will appear to be the location where all writes are directed. For example, if a user modifies a preference, that preference will appear to be written to the preference file located in the preferences folder on the user system volume 924." Ex.1006, 14:61-67.

"When an NC client [150] boots from the network and accesses the operating system from an NC server [170], the user's preferences are dynamically merged with the system environment provided to the NC client [150]." *Id*., 9:2-5. Also, "[a]dvantageously, since the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client [150]

as it boots from the network, the user may login to any NC client [150] on the network and have the same user experience." *Id*., 9:5-10.

"Referring now to Fig. 5, NC client [150] read request processing will now be described according to one embodiment of the present invention. … At step 510, the file system 152 generates a read request directed to the system volumes 160." At "step 520, the read request is received by the block device driver 154 and a determination is made whether the read request specifies a portion of the operating system that has been modified by the user or whether the read request specifies a portion of the operating system that remains unchanged." If "the read request corresponds to a portion of the operating system that has not been modified by the user, the processing continues with step 530." However, "if the read request corresponds to a portion of the operating system that has been previously modified by the user, then processing continues with step 540." At "step 530, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the system volumes 174." At "step 540, the block device driver directs the NC server 170 to retrieve the information associated with the read request from the user's shadow volume." Thus, "[i]n this manner, the file system 152 need not be aware of the particular mechanism used to coordinate user modifications with the original operating system as contained in the system volumes 174." *Id*., 11:39-41, 11:55-12:10. Birse's Figure 5 is reproduced below:

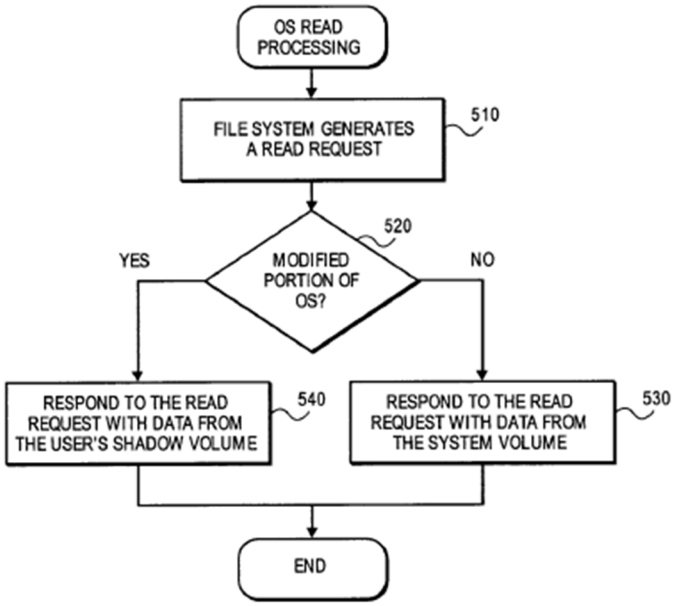


FIG. 5

Ex.1003, ¶¶567-71,595-597.

Additionally, a POSITA would have understood that such system components, taken together, embody the UBD of this claim. The term "UBD" is informally defined in the '844 Patent by way of a set of tasks it performs. *See supra* §IV.A. A POSITA would have recognized that such system components as necessary to provide such features, taken together, embody the Union Block Device of this claim, and that it was well-known for storage systems to use single storage appliances. Ex.1003, ¶¶596-597.

5. Claims 5 and 12

[5]/[12] "low-level drivers/at an operational level between file systems…"

Because the block device drivers (154) of Birse are already designed to merge blocks being read from multiple sources into one cohesive image using the algorithm of Figure 5, it would have been obvious to a POSITA to combine Birse with Holzmann to extend the block device drivers' function to incorporate blocks stored in the cache, as well. That is because cached blocks are just as easily accessible to an operating system or storage system as blocks stored on any storage device such as a hard disk. Ex.1003, ¶¶572-75,598.

6. Claims 6 and 13

[6]/[13] "…first storage unit is read-only."

Birse further discloses the first storage unit being read-only: "The system volumes 174 include a protected, read-only, master copy of the operating system software." Ex.1006, 5:13-14. Ex.1003, ¶¶576-78,599.

7. Claim 8

[8.1]-[8.2] "receiving a read request…a first portion…stored in said cache memory…providing said first portion…from said cache memory."

The Birse-Holzmann combination as described above discloses these elements. *See supra* §VII.B.7 (describing the additional disclosures of Holzmann that meet these elements). Ex.1003, ¶¶556-562.

8. Claim 9

[9] "updating said cache memory based on said read request."

The Birse-Holzmann combination described above disclose these elements. *See supra* §VII.B.8 (describing the additional disclosures of Holzmann that meet these elements). Ex.1003, ¶563.

9. Claim 10

[10.0]-[10.2] "…second portion…not currently stored in said cache memory…caching said second portion…removing…least recently accessed data from said cache memory if…above a threshold value."

The Birse-Holzmann combination described above discloses these elements. *See supra* §VII.B.9 (describing the additional disclosures of Holzmann that meet these elements). Ex.1003, ¶¶564-566.

X. GROUND 4: CLAIMS 14-27 ARE OBVIOUS OVER BIRSE AND FEDERWISCH

A. Motivation to Combine

Both Birse and Federwisch operate at and recognize disk blocks at the block layer (while also recognizing that file systems interoperate and work atop the block layer), disclose/use CoW snapshots (a POSITA would have known CoW techniques enable snapshots), and disclose caching blocks of the root/boot disk images. Further, a POSITA would have been motivated to modify Birse with Federwisch's indexing – including because, while Birse suggests that the boot images are indexed and organized through boot file selection and a registry, a POSITA building a Birse system, and knowing that indexing of data items was a well-known technique, would

have been motivated to seek additional details on indexing storage systems, files, and caches thereof (supplied by Federwisch). A POSITA would also have known that indexing of data is an age-old technique long utilized in computer systems. Ex.1003, ¶¶617-638.

Modifying Birse to include Federwisch's indexing would have been simple to a POSITA, and would not have required extensive experimentation or imposed undue burden – including because Birse likely already supported the use indexing tools and a POSITA would not have expected Birse to provide details of such a basic concept. For the same reasons, the combination would have yielded predictable results, as indexes are well understood to structure and locate data objects efficiently, and a POSITA would have had a high expectation of success, because indexing techniques were well understood for decades beforehand. Ex.1003, ¶¶639-40.

B. Birse and Federwisch meet all claim elements.

1. Claims 14, 19, and 23

[14.0]/[19.0] "system/method for indexing…"

[23.0] "A computer-readable storage medium…"

As described, *supra* §§VI.B, VI.D, Birse and Federwisch both disclose compute nodes and computing devices. As described, *supra* §VI.D, and further below, Federwisch discloses indexing file systems. Ex.1003, ¶¶600,645,661.

[14.1]/[19.1] "store/storing blocks of a root image…;"

[23.1] "receive…data blocks comprising a root image portion;"

These elements are substantially similar to elements recited in claims 1 and 7 regarding root images. As described, *supra* §IX.B.1, Birse discloses these elements. Ex.1003, ¶¶601,646,662.

[14.2]/[19.2]/[23.2] "store/storing leaf images/leaf image portion…"

These elements are substantially similar to elements recited in claims 1 and 7 regarding leaf images. As described, *supra* § IX.B.1, Birse discloses these elements. Ex.1003, ¶¶602,647,662.

[14.3]/[19.3] "…merging the blocks of said root image with the blocks of respective leaf images…"

[23.1] "…merging said root image portion and said leaf image portion…"

These elements are substantially similar to elements recited in claims 4 and 11 regarding UBDs and merging. As described, *supra* § IX.B.4, Birse discloses these elements. Ex.1003, ¶¶605,648,662.

[14.4] "…index said root image and provide the indexing results…"

[19.4] "receiving indexing results…and providing said indexing results…"

[23.4] "index said root image portion; and provide the results of said indexing…"

As described, *supra* §X.A, a POSITA would have been motivated to combine Birse and Federwisch to include Federwisch's indexing in Birse's system. *See supra*

§VIIIB.1 (providing applicable disclosures of Federwisch already described herein). Ex.1003, ¶¶603-616,649-50,663-64.

2. Claim 15

[15] "…within a single storage appliance."

These elements are substantially similar to elements recited in claim 3. As described, *supra* §IX.B.3, Birse discloses these elements. Ex.1003, ¶641.

3. Claim 16

[16] "a plurality of union block devices… merging the blocks of said root image with the blocks of respective leaf images."

These elements are substantially similar to elements already recited in claim 14, as well as claim 4. As described, *supra* §§IX.B.4, X.B.1, Birse discloses these elements. Ex.1003, ¶642.

4. Claims 17 and 21

[17]/[21] "low-level drivers/at an operational level between file systems…"

Petitioners note that "said cache" in claim 17 lacks antecedent basis, but, even to the extent claim 17 requires a cache, these elements are substantially similar to elements recited in claims 5 and 12. As described, *supra* §§ IX.B.5, Birse discloses these elements. Ex.1003, ¶¶643,659.

5. Claims 18, 22, and 27

[18]/[22]/[27] first storage unit/root image portion "is read-only."

These elements are substantially similar to elements recited in claims 6 and 13. As described, *supra* §§IX.B.6, Birse discloses these elements. Ex.1003, ¶¶644,660,684.

6. Claims 20 and 24

[20]/[24] storing/store indexing results "on a shared storage unit"

As described, *supra* §X.A., a POSITA would have been motivated to combine Birse and Federwisch to include Federwisch's indexing in Birse's system. *See supra* §VIII.B.6 (providing applicable disclosures of Federwisch already described herein). Ex.1003, ¶¶651-58,665.

7. Claim 25

[25] "…index said leaf image portion."

As described, *supra* §X.A, a POSITA would have been motivated to combine Birse and Federwisch to include Federwisch's indexing in Birse's system. *See supra* §VIII.B.7 (providing applicable disclosures of Federwisch already described herein). Ex.1003, ¶¶666-75.

8. Claim 26

[26] "…re-index said file system by re-indexing said leaf image portion and merging the results…"

As described, *supra* §X.A, a POSITA would have been motivated to combine Birse and Federwisch to include Federwisch's indexing in Birse's system. *See supra*

§VIII.B.8 (providing applicable disclosures of Federwisch already described herein). Ex.1003, ¶¶676-83.

XI. THERE ARE NO DISCRETIONARY FACTORS FAVORING DENIAL OF INSTITUTION

A. The *Fintiv* factors either favor institution or are neutral

Institution should not be denied because all factors favor institution or, at the very least, are neutral. Factor 1 is neutral because neither party has requested a stay. *See, e.g.*, *Sand Revolution II, LLC v. Continental Intermodal Grp.-Trucking LLC*, IPR2019-01393, Paper 24 at 7 (PTAB. June 16, 2020) (informative).

Factor 2 is neutral because, although no trial date is yet set in IV's case against American, the schedule jointly submitted to the district court would provide for a trial more than eighteen months after the filing date of this petition. And although trial is set for July 27, 2026, in the case against Southwest, that date often changes. For example, the most recent time-to-trial data show that the median time to trial for jury trials in the Western District of Texas is 34.9 months. Table T-3—U.S. District Courts–Trials Statistical Tables For The Federal Judiciary (December 31, 2024), U.S. Courts, https://www.uscourts.gov/data-news/data-tables/2024/12/31/statistical-tables-federal-judiciary/t-3 (last visited March 28, 2025). The Board has instituted IPR based on other petitions with similar facts. *Ericsson Inc. v. XR Commc'ns LLC*, IPR2024-00613, Paper 9, 33-34 (PTAB Oct. 9, 2024).

Factor 3 favors institution because the parties have expended limited resources in the litigations and Petitioners acted expeditiously—*i.e.*, by filing the Petition less than six months of the original complaints and well before the start of the of the claim construction process, in the early stages of fact discovery, and before expert discovery. Exs.1019, 1020. *Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-01019, Paper 12 at 16-17 (PTAB Dec. 1, 2020) (precedential).

Factor 4 favors institution because Petitioners challenge all claims of the '844 Patent, although IV asserts only a subset in the litigation. Exs.1019, 1020, 1021. Petitioners also stipulate that, if the Petition is instituted, Petitioners will not pursue in the District Court litigation any ground raised or that could have been reasonably raised in an IPR. *Sotera,* Paper 12 at 18.

Factor 5 favors institution because institution will promote judicial efficiency and integrity by reducing the number of issues to be addressed in the parallel litigations against American and Southwest. *Sand Revolution*, Paper 24 at 12.

Factor 6 favors institution because Petitioners have set forth a strong case of unpatentability. *Sand Revolution*, Paper 24 at 13.

B. An analysis under 35 U.S.C. § 325(d) also favors institution

None of the references in the above-grounds were substantively addressed or even cited during prosecution of the '844 Patent, and nothing suggests their teachings relied on herein—much less the combination of those teachings as informed by a

POSITA's knowledge and expert testimony—were considered by the Examiner. *Mylan Pharmaceuticals Inc., v. Regeneron Pharmaceuticals, Inc*., IPR2022-01226, Paper 22 at 27-29 (PTAB Jan. 11, 2023).

While U.S. 6,751,658 (Haun) (which shares a common specification with Birse) was briefly referenced by the Examiner during prosecution of the '844 Patent, it was never relied upon. Ex.1002, 64,66. At the same time, a different Examiner assigned to the "grandparent" '622 Application repeatedly and successfully cited Haun to support rejection of all similar claims (resulting in abandonment). Ex.1009, 94,146,205,263,295.

XII. CONCLUSION

Claims 1-27 are unpatentable under Pre-AIA §103. Petitioners request cancellation of claims 1-27.

Dated: April 30, 2025

Respectfully submitted,

By: /Keith D. Harden/
Keith D. Harden
Registration No. 74,472

CERTIFICATE OF WORD COUNT UNDER 37 C.F.R. § 42.24(d)

The undersigned certifies that the word count of the foregoing PETITION FOR *INTER PARTES* REVIEW, excluding the table of contents, table of authorities, exhibit list, mandatory notices, certificate of service, certificate of word count, and any appendix of exhibits or claim listing, is 13,901.

Dated: April 30, 2025

/Keith D. Harden/
Keith D. Harden
Registration No. 74,472

CERTIFICATE OF SERVICE

The undersigned certifies that a copy of the foregoing PETITION FOR *INTER PARTES* REVIEW and all exhibits identified herein were served on April 30, 2025 at the following address of record for the subject patent:

Brandon R. Theiss
Volpe Koenig
30 South 17th Street, 18th Floor
Philadelphia, PA 19103

via FedEx Overnight

Courtesy Copy via electronic mail to Patent Owner's Litigation Counsel:

Jonathan K. Waldrop
Kasowitz Benson Torres LLP
333 Twin Dolphin Drive, Suite 200
Redwood Shores, CA 94065
jwaldrop@kasowitz.com
IntellectualVentures@kasowitz.com

Dated: April 30, 2025

/Keith D. Harden/
Keith D. Harden
Registration No. 74,472