IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Pradip Kulkarni et al.	§	Attorney Docket No.: IVI844
	§	
U.S. Patent No. 8,332,844	§	Customer No.: 165774
	§	
Issue Date: December 11, 2012	§	
	§	
Filing Date: February 21, 2007	§	
	§	
For: ROOT IMAGE CACHING AND	§	
INDEXING FOR BLOCK-LEVEL	§	
DISTRIBUTED APPLICATION	§	
MANAGEMENT	§	

REQUEST FOR *EX PARTE* REEXAMINATION OF U.S. PATENT 8,332,844

Mail Stop "*Ex Parte* Reexam"
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Pursuant to the provisions of 35 U.S.C. §§ 301-307, Unified Patents, LLC ("Unified" or "Requester") hereby requests an *ex parte* reexamination of claims 1-27 (the "Challenged Claims") of U.S. Patent 8,332,844 (the "'844 patent," EX1001), which issued on December 11, 2012, based on U.S. Patent Application 11/709,477, filed February 21, 2007, and claiming priority through two continuations-in-part applications to December 30, 2004. The '844 patent is currently assigned to Intellectual Ventures II LLC ("IV II" or "Patent Owner"). The assignment is recorded in the U.S. Patent and Trademark Office ("USPTO") at reel/frame 065470/0127.

This Request presents prior art references and analyses that are noncumulative of the prior art that was before the Examiner during the original prosecution of the '844 patent and speak directly to the reasons the original claims were originally allowed. As demonstrated herein, the

Challenged Claims are invalid over these references. Unified therefore requests that an order for reexamination and an Office Action rejecting claims 1-27 be issued.

Ex Parte Patent Reexamination Filing Requirements

Pursuant to 37 C.F.R. § 1.510(b)(1), statements pointing out at least one substantial new question of patentability based on material, non-cumulative reference patents and printed publications for the Challenged Claims (claims 1-27) of the '844 patent are provided in Sections IV and V of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(2), reexamination of the Challenged Claims (claims 1-27) of the '844 patent is requested, and a detailed explanation of the pertinence and manner of applying the cited references to the Challenged Claims is provided in Sections IV and V of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(3), copies of every patent or printed publication relied upon or referred to in the statement pointing out each substantial new question of patentability or in the detailed explanation of the pertinence and manner of applying the cited references are provided as Exhibits 1001-1020 (with Exhibits 1005-1008 forming the basis for the SNQs presented herein) of this Request.

Pursuant to 37 C.F.R. §1.510(b)(4), a copy of the '844 patent is provided as Exhibit 1001 of this Request, along with a copy of any disclaimer, certificate of correction, and reexamination certificate issued corresponding to the patent as provided in the prosecution history included as Exhibit 1002.

Pursuant to 37 C.F.R. § 1.510(b)(5), the attached Certificate of Service indicates that a copy of this Request, in its entirety, has been served on Patent Owner at the following address of record for Patent Owner, in accordance with 37 C.F.R. § 1.33(c):

3624 – Volpe Koenig
30 South 17th Street, 18th Floor
Philadelphia, PA 19103

Also submitted herewith is the fee set forth in 37 C.F.R. § 1.20(c)(1).

Pursuant to 37 C.F.R. § 1.510(b)(6), Unified hereby certifies that the statutory estoppel provisions of 35 U.S.C. § 315(e)(1) and 35 U.S.C. § 325(e)(1) do not prohibit Unified from filing this *ex parte* patent reexamination request.

TABLE OF CONTENTS

Table of Exhibits v

I. INTRODUCTION 1

II. CLAIMS FOR WHICH REEXAMINATION IS REQUESTED 2

III. U.S. PATENT 8,332,844 2

A. Overview 2
1. Indexing at the Block Layer 8
2. Union Block Device (UBD) 10

B. Prosecution History 11
1. The '477 Application 11
2. The '622 Application 14

C. The Priority Date 14

D. Level of Ordinary Skill in the Art 15

E. Claim Construction 15

IV. THE PRIOR ART REFERENCES, ARGUMENTS, AND EVIDENCE PRESENT SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY 16

A. Listing of Prior Art Patents and Printed Publications 16

B. *Sapuntzakis, Birse, Holzmann,* and *Federwisch* Present Substantial New Questions of Patentability 18

V. DETAILED EXPLANATION OF THE PERTINENCE AND MANNER OF APPLYING THE PRIOR ART REFERENCES TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED 19

A. SNQ1: *Sapuntzakis* in view of *Holzmann* Presents a Significant New Question of Patentability for Challenged Claims 1-13 20
1. Overview of SNQ1 20
2. Overview of *Sapuntzakis* 20
3. Overview of *Holzmann* (EX1007) 24
4. Combination of *Sapuntzakis* and *Holzmann* 26
5. Claim Chart Providing Detailed Analysis 27

B. SNQ2: *Sapuntzakis* in view of *Federwisch* Presents a Significant New Question of Patentability for Challenged Claims 14-27 47
1. Overview of SNQ2 47
2. Overview of *Sapuntzakis* 48
3. Overview of *Federwisch* 48
4. Combination of *Sapuntzakis* and *Federwisch* 49
5. Claim Chart Providing Detailed Analysis 50

C. SNQ 3: *Birse* in view of *Holzmann* Presents a Significant New Question of Patentability for Challenged Claims 1-13 72
1. Overview of SNQ1 72
2. Overview of Birse 73
3. Overview of Holzmann 75

4. Combination of *Birse* and *Holzmann* 75
5. Claim Chart Providing Detailed Analysis 77

D. SNQ4: *Birse* in view of *Federwisch* Presents a Significant New Question of Patentability for Challenged Claims 14-27 106
1. Overview of SNQ4 106
2. Overview of *Birse* 107
3. Overview of *Federwisch* 107
4. Combination of *Birse* and *Federwisch* 107
5. Claim Chart Providing Detailed Analysis 108

VI. SECONDARY CONSIDERATIONS 138

VII. DISCLOSURE OF CONCURRENT LITIGATION, REEXAMINATION, AND RELATED PROCEEDINGS 138

VIII. DISCRETIONARY DENIAL IS NOT WARRANTED 139

IX. CONCLUSION 140

TABLE OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 8,332,844 B1, to Kulkarni, et al. (the "'844 patent")
1002	File History for U.S. Patent No. 8,332,844
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," *Operating Systems Review*, 36(SI), 377-390, (December, 2002) ("*Sapuntzakis*")
1006	U.S. Patent No. 7,089,300 B1 to Birse, et al. ("*Birse*")
1007	U.S. Patent Application Publication No. 2005/0283597 A1 to Holzmann ("*Holzmann*")
1008	U.S. Patent Application Publication No. 2003/0182313 A1 Federwisch, et al. ("*Federwisch*")
1009	File History for U.S. Patent Application No. 11/026,622
1010	File History for U.S. Patent Application No. 11/395,816
1011	Microsoft Press Computer Dictionary, 3rd ed., 1997 (excerpts)
1012	Silberschatz, et al., Operating Systems Concepts, 4th Edition, 1994, Addison-Wesley Publishing (excerpts)
1013	Hitz et al., "File System Design for an NFS File Server Appliance," Network Appliance, USENIX 1994
1014	Patterson et al., "SnapMirror®: File System Based Asynchronous Mirroring for Disaster Recovery," Network Appliance, USENIX 2002
1015	U.S. Patent No. 6,668,264 B1 to Patterson, et al.
1016	U.S. Patent No. 7,334,095 B1 to Fair, et al.
1017	Internet Small Computer Systems Interface (iSCSI), RFC 3720, IETF, April 2004, *available at* https://www.rfc-editor. org/rfc/pdfrfc/rfc3720.txt.pdf
1018	Infringement Contentions in *Intellectual Ventures v. Liberty Mutual*, No. 23-cv-525 (E.D. Tex.)
1019	Infringement Contentions in *Intellectual Ventures v. Comerica*, No. 23-cv-524 (E.D. Tex.)
1020	Declaration of Ingrid Hsieh-Yee, Ph.D., with its Attachments (Appendices A-C, EX1005, and EX1005A-1005M)

I. INTRODUCTION

The '844 patent describes technology for "root image caching and indexing for block-level distributed application management." EX1001, 2:36-38. However, the references to caching and indexing were merely added to differentiate the '844 patent from two prior parent applications (that mention neither indexing or caching, and to which the '844 patent improperly claims priority). Instead, the '844 patent disclosure is largely directed to systems/methods for distributing an application environment to a compute node using a "root/leaf" system of application environment storage ("storing blocks of a root image on a first storage unit and storing blocks of leaf images on respective second storage units") – to speed creation of and access to boot images. EX1001, 2:38-40, 51-55.

As the prosecution history makes clear, the asserted novelty of claims 1-27 (the "Challenged Claims") was limiting the leaf images to "additional data blocks not previously contained in the root image and changes made by respective compute nodes to the blocks of the root image." EX1001, 2:39-42. However, that architecture was well-known before December 30, 2004, and certainly before February 21, 2007 (the '844 patent's earliest possible priority date).

This Request, supported by the declaration of Dr. Erez Zadok[1,2] (who has over 30 years of

---

[1] Exhibit 1003 is a copy of the Declaration of Dr. Erez Zadok that was filed in IPR2025-00202. In the Declaration, references to Grounds 1A and 1B correspond to SNQ1 and SNQ2 and references to Grounds 2A and 2B correspond to SNQ3 and SNQ4, respectively.

[2] Dr. Zadok's Declaration includes citations to Exhibits 1011-1017 to corroborate his statements regarding POSA knowledge. Exhibit 1011 provides definitions of relevant terms such as "LAN," "IP address," "packet," etc. from the 1997 Microsoft Computer Dictionary. *See, e.g.,* Decl. at Section VII. Exhibit 1012 is a textbook by Silberschatz et al. that provides support for the knowledge a POSA would have had about caching technologies. *See, e.g.,* Decl. at Section VII, D. Exhibit 1013 is a technical report by Hitz et al. that provides support for the knowledge a POSA would have had about copy-on-write (COW) technologies. *See, e.g.,* Decl. at Section VII.E. Exhibit 1014 is a paper about SnapMirror by Patterson et al. that provides support for knowledge a POSA would have had about COW and snapshot technologies, and, in particular, Netapp appliances and SnapMirror. *See, e.g.,* Decl. at Section VII.E. Exhibit 1015 is a patent to Patterson et al. that also provides support for knowledge a POSA would have had about COW and snapshot technologies. *See, e.g.,* Decl. at Section VII.E. Exhibit 1016 is a patent to Fair et al. that provides even further support for knowledge a POSA would have had about COW and snapshot technologies. *See, e.g.,* Decl. at Section VII.E. Exhibit 1017 is a request for comments that provides support for knowledge a POSA would have had about networked and distributed systems. *See, e.g.,* Decl. at Section VII.C.

experience), demonstrate that all 27 claims are unpatentable.

II. CLAIMS FOR WHICH REEXAMINATION IS REQUESTED

Of Challenged Claims 1-27, claims 1, 7, 14, 19, and 23 are independent. Exemplary claim 1 of the '844 patent is provided below.

> **Claim 1.** *A system for providing data to a plurality of compute nodes, the system comprising:*
>
> *a first storage unit configured to store blocks of a root image of said compute nodes;*
>
> *a plurality of second storage units configured to store leaf images of respective compute nodes, 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 said 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; and*
>
> *a cache configured to cache blocks of said root image previously accessed by at least one of said compute nodes.*

III. U.S. PATENT 8,332,844

A. Overview[3]

The '844 patent describes "technology for, among other things root image caching and indexing for block-level distributed application management." EX1001, Abstract. This "involves storing blocks of a root image on a first storage unit and storing blocks of leaf images on respective second storage units [which] … include additional data blocks not previously contained in the root image and changes made by respective compute nodes to the blocks of the root image." EX1001, Abstract. A POSA would have recognized that these leaf images correspond to storage snapshots as described by Dr. Zadok in Section VII.E of his declaration. Decl. at Section VII.E.

---

[3] This overview follows the overview provided in Dr. Zadok's declaration. Decl. at Sections VII.A-C. Dr. Zadok also provides a helpful overview of the technology that is referenced here to put the '844 patent's disclosure in context and illuminates how a POSA would have understood the '844 patent's disclosure. Decl. at Section VII.

The '844 patent then describes common "caching blocks of the root image that have been accessed by at least one compute node." EX1001, Abstract; *see also* Decl. at Section VII.D (describing the well-known and ubiquitous techniques for caching at the time).

Finally, the '844 patent describes generating and "receiving indexing results pertaining to the root image from one compute node and providing the results for other compute nodes." EX1001, Abstract; *see also* Decl. at Section VII.F (explaining the well-known indexing of data and files at the time).

The '844 patent describes a background where, "as the internet has expanded and computers have multiplied, the need for clustered computing such as High Performance Computing (HPC) has increased" and that "[c]lustered computing involves multiple compute nodes, usually a server grid, that work together to achieve a common task." EX1001, 1:31-44. The '844 patent describes the trend towards global file system management: "[i]n large-scale systems such as this, a trend in software deployment is to centralize data management on a globally accessible file system with stateless computing nodes." EX1001, 1:31-44.

That led to the need to manage individual nodes' boot software and/or boot images. "A common example of this is Operating System (OS) software image management, where the compute nodes are activated with the distributed application environment by either diskless booting protocols or remote software installation to local storage. Under this architecture, a boot image is required for each compute node in the cluster." EX1001, 1:31-44.

OS boot images contain the OS kernel, libraries, applications, and more. "The boot image necessarily contains the kernel; it may additionally contain the application software that is intended to be run on the compute node." EX1001, 1:31-44.

The '844 patent explains a concern with managing so many boot images, many of which may be (nearly or) identical. "The primary concern in clustered computing is low cluster bring-up time. The software that provides the boot images for the cluster typically stores a master boot image. It may then either pre-create clones of this master image for each such server, or it may create them 'on the fly.'" EX1001, 1:49-67.

Thus, "[c]reating a boot image on the fly involves copying the entire contents of the master image, which are typically in the range of 5-15 GB." EX1001, 1:49-67. This can be slow. "Even with a significant amount of bandwidth by today's standards, this method will result in a large bring-up time." EX1001, 1:49-67.

One proposed solution is that "Pre-creating a boot image for each server is advantageous from the point of view of cluster bring-up time." EX1001, 1:49-67. But such a solution could waste disk/storage space "since one often does not know in advance how many servers will ever be booted, this scheme may result in wasted disk space." EX1001, 1:49-67. Thus, "both [methods] suffer from the same major problem—updating the boot image(s) for the cluster is cumbersome, as it means updating a number of copies of the boot image." EX1001, 1:49-67.

The '844 Background section then discusses how "[i]n a branching store file system, a read-only base image (or 'root' image) of the application environment is created." EX1001, 2:13-24. "The root image is accessible by all compute nodes in the cluster" and "[c]hanges made by a compute node to the root image are stored in a 'leaf' image unique to that compute node." (i.e., a snapshot). EX1001, 2:13-24.

To make this work, a software driver, component, or filter "operates between the compute nodes and the file system(s), which merges the changes recorded on the leaf images with the root image and delivers the result to the appropriate compute node." EX1001, 2:13-24. The benefit is that, "[f]rom the point of view of the compute node, it is running its own unique and cohesive instance of the application environment." EX1001, 2:13-24. Nevertheless, "[w]hile this system allows for creation of boot images on the fly without severely diminishing bring-up time, a separate version of the system must be created for each unique operating system because data is stored at the file system level (i.e., on a 'per file basis')." EX1001, 2:13-24.

Therefore, the '844 patent argues that "migrating a computing cluster from one operating system to another is much more complicated than simply installing a new root image containing the new OS." EX1001, 2:13-24.

After discussing and admitting to this known prior art, the '844 patent describes an example system that includes "a general purpose computing system environment, such as compute node 100." EX1001, 4:27-29.

"[C]ompute node 100 typically includes at least one processing unit 102 and memory 104" (e.g., RAM, ROM, and/or flash memory). EX1001, 4:29-36. The system may also have additional common features/functionality, such additional removable 108 and/or non-removable 110 storage (e.g., disks, tapes), communications connections 112, input devices (e.g., keyboard) 114, and output devices (e.g., monitor) 116. EX1001, 4:36-5:10; *see also* Decl. Section § VII.A. "Memory 104, removable storage 108, and non-removable storage 110 are all examples of computer storage

media" which "includes, but is not limited to, RAM, ROM, flash memory, DVDs, …, magnetic storage devices, or "any other medium which can be used to store the desired information and which can be accessed by compute node 100." EX1001, 4:46-55, Fig. 1 (reproduced below).

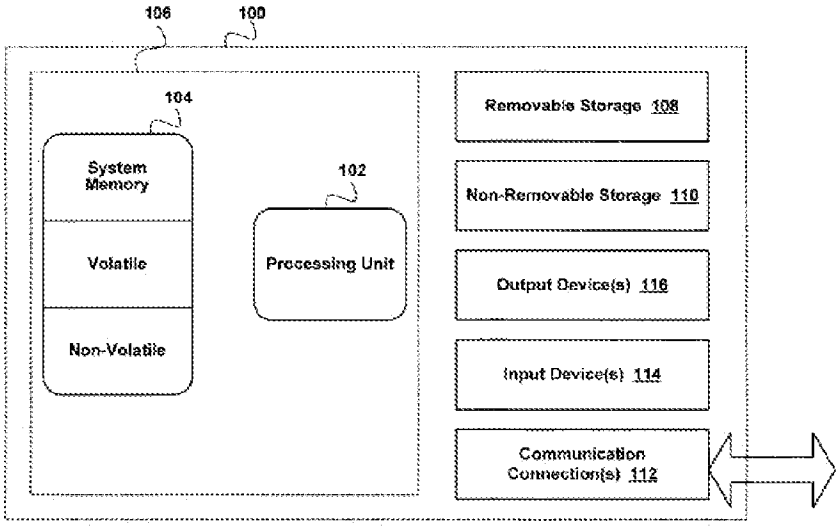


FIG. 1

'844 patent, Figure 1

The '844 patent then describes components of the system. "FIG. 2 is a diagram of a system 200 for root image caching and indexing in a block-level distributed application environment, in accordance with various embodiments of the present disclosure. In one embodiment, system 200 is implemented in a multi-computer system, such as an HPC cluster. In one embodiment, the application environment includes an operating system. In other embodiments, the application environment may contain other applications." EX1001, 5:12-26, Fig. 2 (reproduced below).

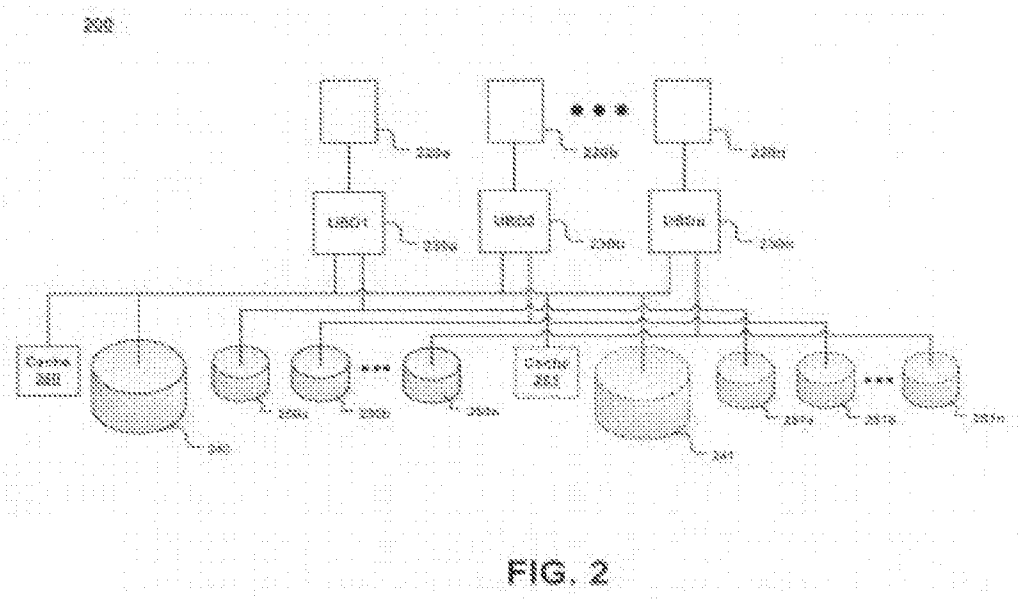


FIG. 2

'844 patent, Figure 2

It then describes how the nodes and the storage they use. "System 200 has a number of compute nodes 220a-n coupled to first storage unit 240 and a corresponding second storage unit 250a-n though a corresponding union block device (UBD) 230a-n. To the compute nodes 220an, it appears that they have access to their own version of a distributed application environment. However, a separate and complete boot image is not created and stored for each compute node 220a-n." EX1001, 5:12-26. Of note is the disclosed "UBD", which is described in more detail in Section III.A.2 below.

Instead of having separate boot images for each node, "a first storage unit 240" "store[s] blocks of a root image of an application environment. The root image contains data initially common to the compute nodes 220a-n" and "is not changed by the compute nodes 220an." EX1001, 5:27-32.

In one embodiment, compute nodes 220a-n have read-only access to the first storage unit 240. Each compute node 220a-n also has a corresponding second storage unit 250an 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. EX1001, 5:33-34, 10:61-65, 11:32-35, 12:4-8, 12:47-50.

In one embodiment, "each compute node 220a-n has a corresponding second storage unit

250a-n for storing a leaf image. 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." EX1001, 5:33-58.

"The leaf image may contain blocks of new data, blocks of changed data, or other blocks of data unique to the individual compute node." EX1001, 5:33-58. Thus, "a leaf image will describe the changes made by the respective compute node 220a-n to its instance of the application environment." EX1001, 5:33-58.

One main reason why the challenged claims were allowed is the addition of "only"[4] in "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. EX1001, 5:33-34, 10:61-65, 11:32-35, 12:4-8, 12:47-50; Section III.B.1, *infra*; EX1002, 166-71. A POSA would have understood this to claim snapshots as described by Dr. Zadok. Decl. at Section VII.E.

The '844 patent states that "[w]ith respect to changes to the root image, only the specific blocks that are changed are stored in the leaf image. … One compute node (e.g., compute node 220a) desires to make a change to this file which involves a modification of only a few specific blocks of the file (e.g., blocks 4-9). In this example, only the modified blocks (e.g., blocks 4-9) will be stored in the compute node's leaf image (e.g., leaf image stored on second storage device 250a) plus some small overhead." EX1001, 5:33-58. 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. EX1001, 5:34-38.

Next, The '844 describes a "union block device" (UBD) that logically merges blocks between a root image and one or more lead images, just as snapshotting systems do. This UBD and its relationship to commonly known snapshots is detailed by Dr. Zadok in his declaration. (Decl. at Section VIII.C), and UBDs are further addressed below in Section III.A.2.

The '844 patent describes an embodiment that uses caching to improve performance.

---

[4] This claim language narrower than the specification, which states that the leaf image "includes, but is not limited to, new data blocks for the compute node and blocks of the root image that the compute node has changed." EX1001, 7:64-66.

"[T]he present disclosure provides for caching of portions of the root image. Because several compute nodes in a cluster may often access the same data (e.g., same drivers, same library files, etc.) on the root image, tremendous speed improvements can be realized by caching such data in cache 260." EX1001, 6:38-49. A POSA would have known that caching can be added to many systems to improve performance and has been in operation for many years, as explained by Dr. Zadok. Decl. at Section VII.D.

Moreover, the '844 patent describes an embodiment where "the cache may be contained within a single storage appliance, along with the first storage unit 240 and the second storage units 250a-n." EX1001, 6:38-49. A POSA would have known that sharing any data (volatile or non-volatile caches or otherwise) was also well known in the art. Decl. at Section VII.C.

Finally, the '844 patent describes an embodiment "for indexing the root image" because "it is beneficial for each compute node to have access to an index of its file system." EX1001, 7:18-25. A POSA would have known that indexing of any data was also well known in the art. Decl. at Section VII.F. The '844 patent further describes how indexing can be shared with multiple nodes: again a POSA would have known how to share data across the network. Decl. at Section VII.C. The '844 patent's indexing disclosures are described in more detail in Section III.A.1 below.

1. Indexing at the Block Layer

The '844 patent claims "indexing" but does not provide sufficient details as to how the indexing is performed. Indexing, however, was well-known in the art, and a POSA would have known that there are multiple forms in which indexes can be organized as Dr. Zadok describes. Decl. at Section VII.H.

The '844 patent further discloses that that indexes can be shared with others. "It is appreciated that the indexing results can be provided to the other compute nodes in a number of ways. For example, the results may be stored on a shared storage unit (such as, but not limited to, cache 260), the results may 40 provided directly to the other compute nodes, etc." EX1001, 7:37-41.

The '844 patent does not say how such index data can be shared with others. A POSA, however, would have known that data can be easily shared across the network using block-layer (e.g., iSCSI) or file-layer protocols (e.g., NFS). Decl. at Section VII.B.1.

For any program to be able to index files at the file system layer, it would require access to

a fully mountable, complete file system. As described above (see also Decl. at Section VII.B), this requires access to a base image plus one or more snapshots that provide a complete view of the file system at a point. A POSA would have also understood that merely taking some subset of blocks (e.g., an incremental snapshot) does not provide a valid format file system; an incremental "leaf" snapshot alone cannot even be mounted or accessible as a file system because critical data structures that make up files, folders, and names are missing.

The '844 patent describes that indexing takes place at the logical block layer, not at the file system layer, for four reasons.

First, the '844 patent states that file systems are "agnostic" to the fact that their lower-level blocks and sectors are spread around a root image plus one or more leaf images (i.e., snapshots). "Step 520 involves the compute node indexing the root image portion of its file system. In such a case, the compute node is to some extent agnostic to the fact that its file system is divided between a root image portion and a leaf image portion." EX1001, 9:48-52. A POSA would have understood this passage to mean that indexing takes place at either the root image or one of the leaf images. Decl. ¶196.

Second, the '844 patent states that it indexes only the leaf image (an incremental snapshot) independently from file systems. "Further down the road, the benefits of indexing the root image separately from the leaf image are realized when the compute node requires that its file system be re-indexed. At step 560, the compute node re-indexes its file system by re-indexing its corresponding leaf image portion with the previous indexing results of the root portion. Thus, by operating at the block level, embodiments of the present disclosure provide file system and operating system independent systems and methods for distributing an application environment to a compute node." EX1001, 9:62-10:4. A POSA would have understood that this can only take place by indexing at the block layer. Decl. ¶197. Indeed, it would be impossible to index files in a leaf image (incremental snapshot) because it is not even mounted as a file system volume. Decl. ¶197.

Third, the '844 patent reinforces the purported benefit of being able to quickly index only the leaf image because it is small. "Thereafter, the compute nodes 220a-n need only be concerned with indexing their respective leaf image, which are relatively small compared to the root image." EX1001, 7:31-33. This further reinforces that the idea that the '844 patent's intended indexing is of the data blocks/sectors of the root/leaf images, which can only take place at the block layer.

Fourth, the '844 patent's claim 19 claims in its preamble "a method for indexing file systems," but its actual limitation not once mentions a file, file name, or file system structure. Rather, it mentions "data blocks" of root and/or leaf images. A POSA would have understood that the indexing here operates at the block/image level; that is, when the preamble states "indexing file system," a POSA would have understood it to be nothing more than an outcome born out of the fact that file systems a logically structured above block devices. Stated differently, every file that a user or application can access (e.g., by name), eventually translates to one or more disk blocks at the block layer; thus, it is natural to discuss (indexed) blocks that relate to files and hence their file systems, even when actual indexing operations take place at the block layer. Decl. at Section VII.B.

Requester submits that for at least these reasons, a POSA would have understood the claimed indexing to operate in this way, e.g., at the logical block layer. Moreover, for the reasons presented here, the broadest reasonable interpretation encompasses this understanding of the claims (e.g., any attempt by the Patent Owner to interpret the claims to exclude this understanding would not be consistent with the broadest reasonable interpretation of the claims).

2. Union Block Device (UBD)

The '844 patent does not define what a UBD is. The patent describes UBDs as "effectively low-level drivers," i.e., "In one embodiment, UBDs 230a-n 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. … Because UBDs 230a-n operate below the file system, they are concerned merely with the blocks of data themselves, rather than files they form." EX1001, 5:60-6:1. But there are many storage drivers reside below file systems. Decl. at Section VII.B.

UBDs are said to "determine what leaf image (from the appropriate second storage unit 250) is needed for portions of the application environment that their respective compute nodes 220a-n have changed. UBDs 230a-n also locate the portions of the application environment that are not changed by their respective compute nodes 220a-n. These portions may reside in the root image. There may also be intermediate images (not depicted in FIG. 2) comprising versions of the root image from which a compute node's instance of the application environment is derived." EX1001, 6:4-8. A POSA would have understood that this is no more than common traversal of snapshots. Decl. at Section VII.E.

Moreover, "UBDs 230a-n may also modify the leaf image in response to their respective compute node's access to its instance of the application environment. For example, upon receiving a write request from their respective compute nodes for a sector X, the UBDs 230an will create an appropriate persistent mapping for sector X and then write sector X onto their respective second storage units 250a-n, where sector X can then be modified." EX1001, 6:20-27. Again, a POSA would have understood that the UBD functionality disclosed here is nothing more than typical copy-on-write (CoW). Decl. at Section VII.E.

Lastly, UBDs logically merge different root and leaf images, which are nothing more than snapshots. It is also natural to extend them to integrate data stored in a cache, which is yet another sort of snapshot in that it contains a subset of the data from the backing store (see Decl. at Section VII.D). Thus "Step 330 involves merging the blocks of the root image, the blocks of the leaf image, and the relevant blocks of the cache, if any, to create the application environment. In other words, the merging occurs at an operational level between the file system of a compute node and the first storage unit, the corresponding second storage unit, and the cache. Once the application environment has been created, it will appear to the compute node as one cohesive image rather than a base image plus its additions, deletions, and modifications. To the compute node, it appears that it has access to its own unique version of an application environment. However, a separate and complete boot image is not actually stored for the compute node." EX1001, 8:13-25. A POSA would have recognized this merging functionality to be no different from what prior art snapshot systems already disclosed. Decl. at Section VII.E.

B. Prosecution History

Application No. 11/709,477 (the "'477 Application"), which matured into the '844 patent, was a continuation-in-part of Application No. 11/395,816 (now Patent No. 7,721,282) (the "'816 Application"), which was a continuation-in-part of Application No. 11/026,622 (now abandoned) (the "'622 Application").

1. The '477 Application

The '477 Application's five independent claims were originally open-ended concerning the content of leaf images – reciting they comprised "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. EX1002, 25-30. Claims 14-27 were rejected (1/5/2010)

as anticipated by Publication No. 2003/0126242 A1 ("Chang"). Claims 1-13 (all others) were rejected as obvious over Chang and Patent No. 6,101,576 ("Kobayashi"). Kobayashi was cited only for caches; and the Examiner found it would have been obvious to a person of ordinary skill in the art ("POSA") to have combined the "detailed image/data cache" of Kobayashi with the "server/client image assess and remotely stored boot/OS system" of Chang "for the benefit of increasing the speed of data transfers from the host to client in data streaming or transfer operations as frequently accessed data is kept in [cache] memory…[versus] slower disk sources." EX1002, 53-64.

In response (4/5/2010), Applicants amended the independent claims to all include a "wherein" limitation of the general form:

> 7. (Currently amended) A method for providing data to a plurality of compute nodes, comprising:
>
> storing blocks of a root image of said compute nodes on a first storage unit;
>
> storing leaf images for respective compute nodes on respective second storage units, said leaf images comprising additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, <u>wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes</u>; and
>
> caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.

EX1002, 73-78. Applicants stated this overcame the rejections. EX1002, 82-88. The findings regarding Kobayashi's cache and motivation to combine were not contested. Nor were any dependent limitations raised. EX1002, 82-88.

The amended claims were rejected (5/25/2010) on the same grounds (with additional citations to Chang addressing the "wherein" amendments). EX1002, 102-25.

Applicants then further amended the independent claims (8/31/2010) to all include the term "only" in the following manner:

> 7. (Currently amended) A method for providing data to a plurality of compute nodes, comprising:
>
> storing blocks of a root image of said compute nodes on a first storage unit;
>
> storing leaf images for respective compute nodes on respective second storage units, said leaf images comprising <u>only</u> additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of 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; and
>
> caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.

EX1002, 166-71. Applicants stated these amendments overcame the rejections because the "client image copies" in Chang (leaf images) include "blocks of the root image that are unchanged by a compute node (e.g., the common OS and application blocks for storing the base boot image)." EX1002, 176-82. Again, Applicants never contested Kobayashi's cache or motivation to combine. Nor were any dependent limitations raised.[5] EX1002, 176-82.

The third Office Action (3/12/2012) never mentioned Chang.[6] Instead, the Examiner rejected claims 19-27 as anticipated by Patent No. 7,870,106 ("Nguyen"), claims 14-18 as obvious over Nguyen, and claims 1-13 as obvious over Nguyen and Kobayashi. Again, Kobayashi was only cited for caches, and the Examiner found it would have been obvious to a POSA to have combined the "detailed image/data cache" teachings of Kobayashi with the "server/client image assess and remotely stored boot/OS system" teachings of Nguyen. EX1002, 192-206.

On 7/11/2012, Applicants traversed these rejections, stating Nguyen was not prior art under § 103(a) given alleged common ownership with the pending application. EX1002, 224, 300. Applicants also amended several claims, including substitution of "including" for "comprising" in two independent claims. EX1002, 239-45. Again, Applicants did not contest findings regarding

---

[5] Applicants also stated a UBD is "a driver," that "drivers serving as interfaces between [OS] file systems…and associated storage devices are well known in the art;" and that a POSA "would understand how a driver may serve as [such] an interface." EX1002, 172-73.

[6] Applicants' prior arguments were found "moot," as they did not apply to the current references. EX1002, 206.

the cache of Kobayashi or the motivation to combine. Nor did they raise any dependent limitations. EX1002, 221-25.

Thus, the sole purported point of novelty for the '844 patent claims is the limitation, in each independent claim, that the leaf images only comprise/include "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 corresponding language in claim 23).

2. The '622 Application

During prosecution of the "grandparent" '622 Application, a different Examiner repeatedly rejected all proposed claims as anticipated (4/13/2009) by Chang and obvious (8/24/2009, 3/15/2010, 8/13/2010, 12/6/2010) over Chang and Patent No. 6,751,658 ("Haun"). EX1009, 50-63, 92-109, 144-167, 203-25, 259-85.

Indeed, Applicants (who differed, in part, from the '477 Applicants) ultimately abandoned the '622 Application with no claims ever allowed. EX1009, 294-95. The '622 Application was abandoned, despite the same attorney (at the same time) offering similar amendments/arguments during prosecution of both the '622 and '477 Applications (including addition of the word "only" to several claims).[7] *Id.* 183-99.

Further, while the '622 and '477 Applications were co-pending, the '477 Application Examiner (unlike the '622 Application Examiner) never relied on Haun. EX1002. Indeed, the former mentioned Haun only once (1/5/2010), and then only to state that Haun (and another reference) were prior art of record and were "not relied upon but is considered pertinent to applicant's disclosure." EX1002, 64.

C. The Priority Date

The '477 Application was filed on February 21, 2007, and claimed priority to the '816 Application (filed 3/30/2006), which claimed priority to the '622 Application (filed

---

[7] For example, the '622 Application claim 1 was amended (on 6/15/2010) to recite a "plurality of second file collections <u>separate from the root file system, each</u> comprising <u>only</u> modified portions of said distributed applications for respective ones of said plurality of compute nodes…." *Id.* 184.

12/20/2004). 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 challenged claim. EX1001, 10:55- 14:17; *compare* EX1001, 6:37-7:44, Figs. 2, 3A, 3B, 7 *with* EX1009, Figs. 1, 7 *and* EX1010, Figs. 1, 3. *See Studiengesellschaft Kohle v. Shell Oil Co.*, 112 F.3d 1561, 1564 (Fed. Cir. 1997).

Applicants even admitted that "indexing and indexing results are not discussed anywhere within Nguyen" (which matured from a continuation-in-part of the '622 Application). EX1002, 223.

Nevertheless, this Request does not challenge the claimed priority date of December 20, 2004, because all of the prior art presented herein predates the earliest claimed priority. *See, e.g.,* Section IV (reviewing the prior art predating even the earliest claimed priority date).

D. Level of Ordinary Skill in the Art

A POSA for the '844 patent, as of the earliest claimed priority date (December 30, 2004), would have had: (1) a bachelor's degree in computer science, computer engineering, or an equivalent degree; and (2) at least two years of experience working in the field of networked data storage, file systems, and operating systems. Additional education could substitute for less experience and *vice versa*. Decl. ¶34-36.

Dr. Zadok, who provides the declaration attached as Exhibit 1003 of this Request, was qualified as a POSA on the earliest possible priority date. Decl. ¶36; EX1004.

E. Claim Construction

"During patent examination, the pending claims must be 'given their broadest reasonable interpretation consistent with the specification.'" MPEP § 2111. The '844 patent has not yet expired; thus, the broadest reasonable interpretation standard applies. MPEP § 2258. The present Request presents the following claim analysis in a manner that is consistent with the broadest reasonable interpretation consistent with the specification.

Unified submits that an express construction[8] is not required for any term at least because

---

[8] Unified does not concede that the claims are capable of express construction, are definite, or that there is written description support for the terms. Nevertheless, Unified's request is proper,

prior art presented herein renders the claims unpatentable under the broadest reasonable construction.[9]

Unified notes that the Dr. Zadok's declaration considers the claims "interpreted according to their 'ordinary and customary meaning,'" i.e., consistent with the *Philips* standard. Decl. ¶306; *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005). As the broadest reasonable interpretation encompasses narrower interpretation under the *Philips* standard, Dr. Zadok's opinions show the claims are unpatentable under either standard, i.e., that the claims are also unpatentable under the broadest reasonable interpretation standard.

IV. THE PRIOR ART REFERENCES, ARGUMENTS, AND EVIDENCE PRESENT SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

A. Listing of Prior Art Patents and Printed Publications

Reexamination of claims 1-27 of the '844 is requested in view of the following references:

- ***Sapuntzakis* (EX1005)** – Sapuntzakis, et al., "Optimizing the Migration of Virtual Computers," *Operating Systems Review*, 36(SI), 377-390, published December, 2002.[10]

---

however, because as long as the claims are not "impossible for the Board" to analyze, the Board can review the "patentability of the claims on section 102 and 103 grounds." *Samsung Elecs. Am. Inc. v. Prisua Eng. Corp.*, 948 F.3d 1342, 1355 (Fed. Cir. 2020); *Intel Corp. v. Qualcomm Inc.,* Nos. 2020-1828, 2020-1867, slip op. at 18-21, 20 (Fed. Cir. Dec. 28, 2021), *available at* https://cafc.uscourts.gov/opinions-orders/20-1828.OPINION.12-28-2021_1885998.pdf (A patentability analysis is only precluded when the claims indefiniteness "renders it logically impossible for the Board to reach such a decision").

[9] Notably, Patent Owner very broadly reads its own claims, asserting them in the litigation against commonly and widely used open-source software known as Docker/Kubernetes "containers." *See* EX1018-EX1019.

[10] As detailed in the declaration of Dr. Ingrid Hsieh-Yee (EX1020), *Sapuntzakis* is the electronic copy provided by the USENIX Association of the *Sapuntzakis* article, which was published in the *Proceedings of the 5th Symposium on Operating Systems Design and Implementation.* EX1020, ¶¶29-36, EX1005; *see also* EX1005A-1005B attached to Dr. Hsieh-Yee's declaration. The symposium took place in Boston, Massachusetts, from December 9 to 11, 2002, and the table of contents for the *Proceedings* shows that *Sapuntzakis* was included in the Migration Section scheduled for Wednesday, December 11. EX1005, 1; EX1020, ¶¶34-35. Additional evidence of its status as a publication more than a year before the claimed priority date of the '844 Patent

- ***Birse* (EX1006)** – U.S. Patent No. 7,089,300 B1, 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.
- ***Holzmann* (EX1007)** – 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 (matured into U.S. Patent No. 8,601,100, issued December 3, 2013 to Holzmann).
- ***Federwisch* (EX1008)** - 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 (matured into U.S. Patent No. 6,993,539, issued January 31, 2006 to Federwisch, et al.).

Sapuntzakis, published in December 2002, is prior art to the '844 patent under 35 U.S.C. § 102(a) and (b) as a printed publication published before (for 102(a)) and more than a year before

---

include:

(1) its availability in several libraries (EX1020, ¶¶37-52; EX1005C-EX1005F attached to Dr. Hsieh-Yee's declaration (showing it available for public access in April 2003 at the University of Wisconsin-Madison Library System and also the University of Saskatchewan Library));

(2) its availability on the Internet (EX1020, ¶¶53-62; EX1005G-EX1005L attached to Dr. Hsieh-Yee's declaration (showing that by January 16, 2003, interested users would be able to discover the webpage of the Technical Sessions of the OSDI '02 Symposium proceedings and use it to access *Sapuntzakis*, that USENIX members would be able to access the full paper by that date, and that non-members could become USENIX members to access the paper or could access the paper for free in December 2003; additionally the full paper was archived once by Internet Archive on September 28, 2004 (*see* MPEP §2128(II)(E))); and

(3) citations to it by others (EX1020, ¶63; EX1005M attached to Dr. Hsieh-Yee's declaration (listing 5 publications between June 2003 and October 2003 that cited to *Sapuntzakis*).

Thus, to the extent more evidence is required beyond its publication in the *Proceedings* in December 2002, there is ample evidence that *Sapuntzakis* qualifies as a printed publication because it was disseminated or otherwise made available to persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence. *See, e.g.,* EX1020, generally; MPEP §§ 2128(I), (II).

(for 102(b)) the earliest claimed priority date of the '844 patent. *See also supra,* n. 9 (regarding *Sapuntzakis'* status a printed publication more than a year before the earliest claimed priority date).

Birse is prior art under 35 U.S.C. § 102(e) because it is a patent issued under §151 naming another inventor and issuing from an application effectively filed on September 18, 1999, before the earliest claimed priority date of the '844 patent.[11]

Holzmann is prior art under 35 U.S.C. § 102(e) because it is the publication under § 122(b) of an application naming another inventor and effectively filed on June 22, 2004, before the earliest claimed priority date of the '844 patent.[12]

Federwisch is prior art under 35 U.S.C. §§ 102(a), (b), and (e) because it is a printed publication printed on September 25, 2003, before (for § 102(a)) and more than a year before (for § 102(b)) the earliest claimed priority date of the '844 patent, and it is an application for a patent, published under § 122(b), filed on March 19, 2002, before the earliest claimed priority date of the '844 patent.

None of the references were in an IDS or applied during the original prosecution of the '844 patent and they are not cumulative of any prior art considered by the USPTO.

As demonstrated below, *Sapuntzakis, Birse, Holzmann,* and *Federwisch* present substantial new questions of patentability.

B. *Sapuntzakis, Birse, Holzmann,* and *Federwisch* Present Substantial New Questions of Patentability

Unified submits that the prior art references *Sapuntzakis, Birse, Holzmann,* and *Federwisch* raise new "substantial question[s] of patentability" because "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." *See* MPEP § 2242.

---

[11] Moreover, as noted in Section III.C, the '844 patent is not entitled to a priority date before its own filing date. Therefore, to the extent it becomes relevant to this Request, *Birse* also qualifies as prior art under 35 U.S.C. § 102(a) based on its issuance as a patent on August 8, 2006, before the February 21, 2007, filing date and correct priority date of the '844 patent.

[12] Additionally, because the '844 patent is not entitled to a priority date before its own filing date (Section III.C), to the extent it becomes relevant to this Request, *Holzman* also qualifies as prior art under §§ 102(a) and (b) because it is a printed publication published on December 22, 2005, before (for § 102(a)) and more than a year before (for § 102(b)) the February 21, 2007, filing date and correct priority date of the '844 patent.

While the original examiner was unable to locate prior art that disclosed the claimed methods and systems, the examiner did not have the primary prior art relied upon in this Request. *See, e.g.,* EX1001, "References Cited."

As presented in this Request, *Sapuntzakis, Birse, Holzmann,* and *Federwisch* present substantial new questions of patentability that were not previously considered by the Office during the original examination. Therefore, the substantial new questions of patentability presented herein warrant reexamination of the Challenged Claims of the '844 patent *See, e.g.,* Section V.

This Request presents several substantial new questions of patentability (SNQs) for resolution. These SNQs are referred to using a reference number for convenience herein (e.g., SNQ1, SNQ2, etc.). The SNQs presented in this Request are the following:

SNQs	References	Basis	Claims
SNQ1	*Sapuntzakis* in view of *Holzmann*	§ 103	1-13
SNQ2	*Sapuntzakis* in view of *Federwisch*	§ 103	14-27
SNQ3	*Birse* in view of *Holzmann*	§ 103	1-13
SNQ4	*Birse* in view of *Federwisch*	§ 103	14-27

As demonstrated below, the prior art references render the Challenged Claims of the '844 patent unpatentable.

This Request is supported by the Declaration of Dr. Erez Zadok, which describes the scope and content of the prior art at the time of the alleged invention of the '844 patent. Decl., *generally.*

V. DETAILED EXPLANATION OF THE PERTINENCE AND MANNER OF APPLYING THE PRIOR ART REFERENCES TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED

The present request presents proposed rejections based on the references *Sapuntzakis, Birse, Holzmann,* and *Federwisch.*

Unified notes that the proposed rejections based on SNQs 1-4 below are based on obviousness under 35 U.S.C. § 103(a). Unified notes that, even if there are minor differences between cited disclosures of the prior art and the claimed concepts, the claims are still obvious, as a POSA would not have appreciated "some new synergy" or unexpected results from the

limitations given the disclosures of the prior art and the general knowledge of a POSA. *KSR Intern. Co. v. Teleflex Inc.*, 550 US 398, 417 (2007). When compared with the prior art described below and in light of the knowledge of a POSA, any "differences between the claimed invention and the prior art are such that the claimed invention as a whole would have been obvious before the effective filing date of the claimed invention to a person having ordinary skill in the art to which the claimed invention pertains." 35 U.S.C. § 103.

The proposed rejections (SNQs) are listed above and are described below in further detail.

A. SNQ1: *Sapuntzakis* in view of *Holzmann* Presents a Significant New Question of Patentability for Challenged Claims 1-13

1. Overview of SNQ1

Claims 1-13 are obvious as explained herein, accompanied by the claim chart provided in Section V.A.5 and the Zadok Declaration (Decl. ¶309-412), are rendered obvious by *Sapuntzakis* in view of *Holzmann*. Notably, *Sapuntzakis* discloses the claimed root/leaf architecture, including where leaf images only contain additional data blocks not previously contained in the root image and changes made by respective compute nodes to the blocks of the root image. *Holzmann* discloses any additional caching limitations to the extent, if any, not already taught by *Sapuntzakis* and/or obvious to a POSA.

2. Overview of *Sapuntzakis*

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. EX1005, 2. This is achieved by reducing the amount of data sent over the network through use of several mechanisms. Of interest here is the use of "copy-on-write" ("COW") disk images that track just the updates to the disk-storage component of capsules. EX1005, 2.

Computers and storage act as caches of capsules. As users travel (commute), the system can move their capsules to computers close to them, giving them a consistent environment. EX1005, 3. 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. EX1005, 6.

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. EX1005, 6. This approach shares similarities with disk imaging, but the disclosed system allows multiple capsules to exist on the same machine. EX1005, 5.

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. EX1005, 6.

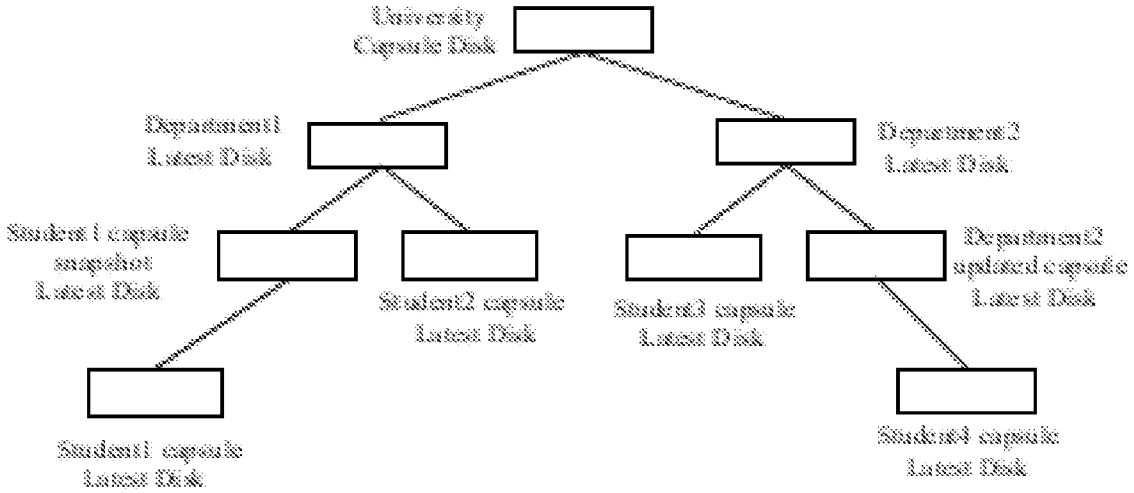


Figure 1: An example capsule hierarchy.

***Sapuntzakis*, Figure 1**

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. EX1005, 6. Because the root COW disk contains a copy of all blocks, the search is guaranteed to terminate. EX1005, 6-7.

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. EX1005, 7.

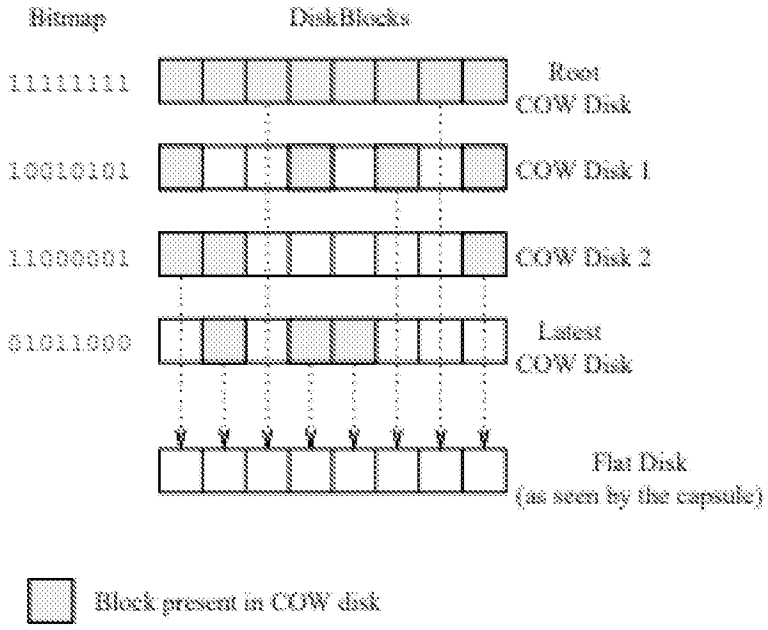


Figure 2: An example capsule disk.

***Sapuntzakis*, Figure 2**

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. EX1005, 7.

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. EX1005, 7; *see also*, Decl. ¶¶224-42.

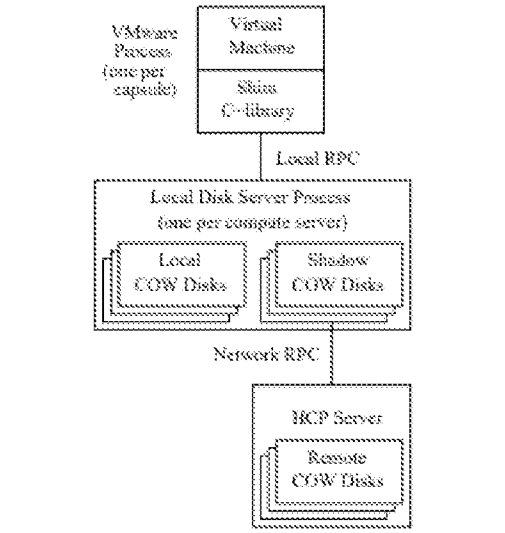


Figure 3: Implementation of capsule disks and demand paging.

Sapuntzakis, Figure 3

Sapuntzakis is analogous to the '844 patent. *Sapuntzakis* is in the same field of endeavor as the '844 patent, i.e., the management of whole OS images being shared by users on a network. *Compare, e.g.,* EX1001, 1:31-2:51 ("Clustered computing involves multiple computer nodes…that work together to achieve a common task"; "A common example of this is Operating System (OS) software image management, where the compute nodes are activated with the distributed application environment by either diskless booting protocols or remote software installation to local storage"; "Thus, embodiments of the present disclosure provide an operating system-independent system and method for distributing an application environment to a computer node…allow[ing] creation of boot images on the fly without significantly diminishing bring-up time";) *with* EX1005, 1 ("This paper shows how to quickly move the state of a running computer across a network, including the state in its disks, memory, CPU registers, and I/O devices…[which] includes the entire operating system"); 7 (describing how to access the needed disk blocks quickly from the entire stored capsule disk on the network).

Sapuntzakis is also reasonably pertinent to a particular problem with which the inventors were concerned, such as efficient storage/transfer of OS images, with both *Sapuntzakis* and the '844 patent disclosing caching blocks of the root/boot disk images to improve efficiency. *Compare* EX1001, Title, 2:54-67 (describing issues with known systems

either "a large bring-up time," "wasted disk space," and a "cumbersome" process), 2:36-45 (addressing the known issues with its disclosed "technology for, among other things, root image caching and indexing for block-level distributed application management…"); *with* EX1005, 1 (noting the inefficiencies of known systems tying a user to a particular computer, the added difficulties of maintaining separated systems that end up drifting apart, and acknowledging concerns with "reduc[ing] the amount of data sent over the network" when a computer's capsule may contain "gigabytes of disk data"); 7 (describing demand paging of the entire capsule disk, explaining how "[e]ach remote COW disk has a corresponding local *shadow COW* disk which contains all the locally cached blocks of the remote COW disk).

3. Overview of *Holzmann* (EX1007)

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. EX1007, [0017]. In Figure 1, a solid state disk system 101 is disclosed having an internal architecture of component devices combined to achieve it purposes. EX1007, [0017-20].

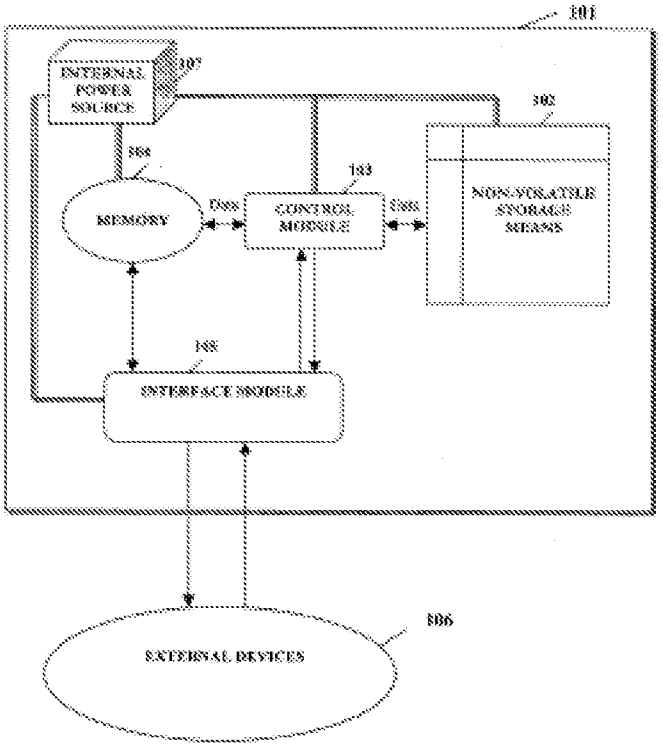


Fig. 1

***Holzmann*, Figure 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 access to the most commonly used data. EX1007, Abstract, [0021], [0024]; *see also*, Decl. ¶¶243-56.

Holzmann is analogous to the '844 patent. *Holzmann* is in the same field of endeavor as the '844 patent, i.e., the management of whole OS images being shared by users on a network. *Compare, e.g.,* EX1001, 1:31-2:51 ("Clustered computing involves multiple computer nodes…that work together to achieve a common task"; "A common example of this is Operating System (OS) software image management, where the compute nodes are activated with the distributed application environment by either diskless booting protocols or remote software installation to local storage"; "Thus, embodiments of the present disclosure provide an operating system-independent system and method for distributing an application environment to a computer node…allow[ing] creation of boot images on the fly without significantly diminishing bring-up time";) *with* EX1007, Abstract, [0010] ("The invention is directed to a system and method for booting multiple servers or other network resources using a single operating system image"; presenting *Holzmann's* disclosure to address the "need in the art for a system and method for booting multiple servers from a storage area networking using a single operating system image").

Holzmann is also reasonably pertinent to a particular problem with which the inventors were concerned, such as efficient storage/transfer of OS images, with both *Holzman* and the '844 patent disclosing caching blocks of the root/boot disk images to improve efficiency. *Compare* EX1001, Title, 2:54-67 (describing issues with known systems either "a large bring-up time," "wasted disk space," and a "cumbersome" process), 2:36-45 (addressing the known issues with its disclosed "technology for, among other things, root image caching and indexing for block-level distributed application management…"); *with* EX1007, [0009] (describing issues with known systems including "separate operating system image must be stored for each server in the network" which is "wasteful and expensive"), [0010] (*Holzman's* system addressing the "need in the art for a system and method for booting multiple servers from a

storage area networking using a single operating system image").

4. Combination of *Sapuntzakis* and *Holzmann*

The proposed combination of *Sapuntzakis* with *Holzmann* is implementing *Holzmann's* details for how to cache data for each OS into the virtual machines of *Sapuntzakis.*

The Supreme Court in *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398 (2007) adopted a flexible approach to evaluating whether a patent claiming a combination of elements of prior art is obvious. Under *KSR* and as detailed in the Zadok Declaration (EX1003), a POSA would have been motivated to combine *Sapuntzakis* and *Holzmann*.

A POSA would have been motivated to modify *Sapuntzakis* with *Holzmann*'s caching techniques, because, while *Sapuntzakis* discloses caching, it does not provide many details of how to do so. A POSA 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). *See* Decl. ¶331-50. That is, so motivated by *Sapuntzakis'* disclosure of caching, to the extent a POSA required implementation details, they would have turned to *Holzman*. *See* Decl. ¶¶331-50; MPEP § 2143(I)(Rationale (G)). Moreover, application of one of the several disclosed caching techniques described in *Holzmann* to *Sapuntzakis* represents the use of *Holzmann's* known techniques for caching OS to improve the similar virtual machines of *Sapuntzakis* in the same way. *See* Decl. ¶¶331-50; MPEP § 2143(I)(Rationale (C)).

A POSA would have had a reasonable expectation of success in the combination given the significant overlap in technologies. Both are concerned with the management of whole OS images being shared by users on a network, particularly efficient storage/transfer of such images and both disclose caching blocks of the root/boot disk images to improve efficiency. *See, e.g.,* EX1005, 1, 7; EX1007, Abstract, [0010]. Modifying *Sapuntzakis* with *Holzmann's* caching teachings would have been simple to a POSA, and would not have required extensive experimentation or imposed undue burden – especially when both disclose caching. *See* Decl. ¶¶331-50. 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. For the same reasons, the combination would have yielded predictable results (as caching is well-understood to improve performance and resource utilization). *See* Decl.

¶¶331-50. A POSA would also have a high expectation of success, as *Holzmann* merely describes caching in greater detail than *Sapuntzakis*. *See* Decl. ¶¶331-50.

5. Claim Chart Providing Detailed Analysis[13]

Claim 7[14]	SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, *Sapuntzakis* discloses *a method for providing data* (capsules) to *a plurality of compute nodes* (user computers). ***See, e.g., Sapuntzakis:*** "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." *Sapuntzakis,* 3.
[7.1] storing blocks of a root image of said compute nodes 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). ***See, e.g., Sapuntzakis:*** "**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." *Sapuntzakis,* 3. "At the root of the hierarchy is a ***root disk***, which is **a complete capsule disk**." *Sapuntzakis,* 6.

[13] This claim chart mapping is further supported with an analysis in prose form provided by Dr. Zadok in his declaration. Decl. at Section XII.A.

[14] Claims 1 and 7 largely mirror each other, with claim 7 disclosing methods that the system of claim 1 is configured to perform. Thus, claim 7 is reviewed first to focus on the prior art disclosures for the steps performed, and claim 1 and its dependents are provided after to identify the components configured to perform each step discussed in the mapping of claim 7.

"Figure 1 shows an example of a capsule hierarchy illustrating how it may be used in a university. The root capsule contains all the software available to all students." *Sapuntzakis,* 6.

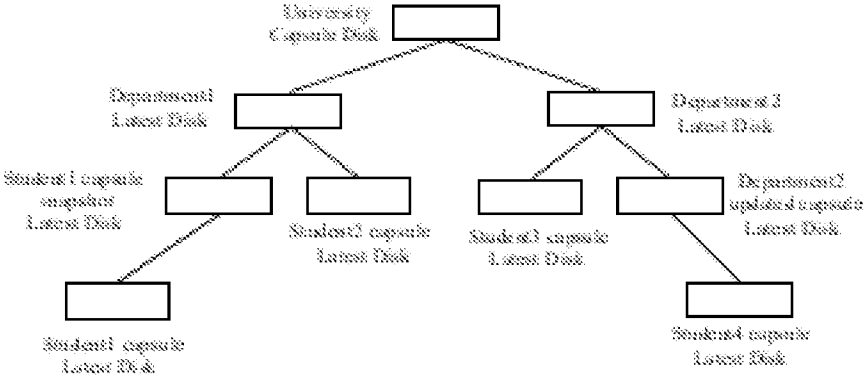


Figure 1: An example capsule hierarchy.

"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." *Sapuntzakis,* 7.

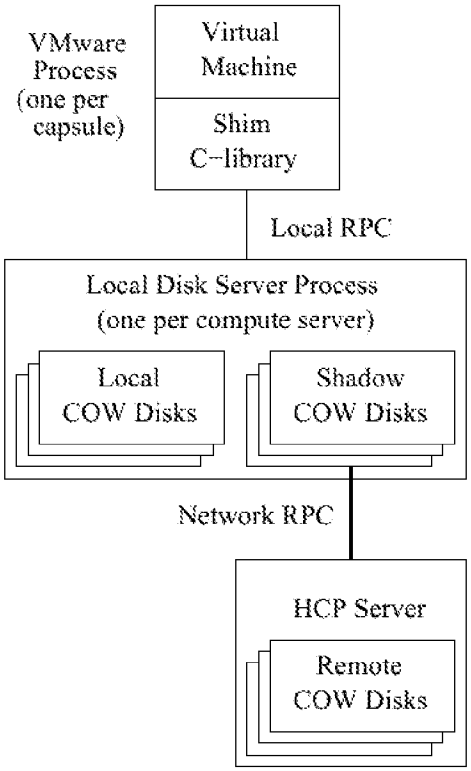
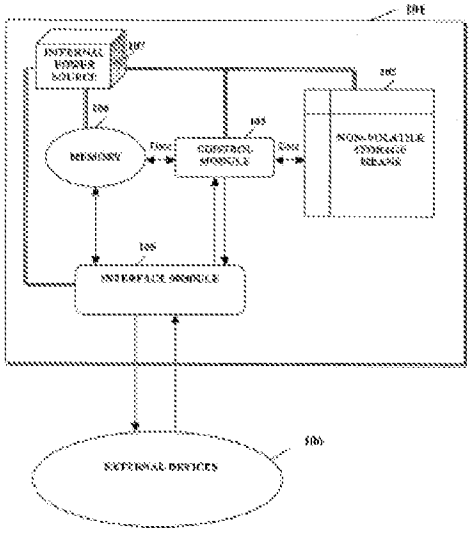


Figure 3: Implementation of capsule disks and demand paging.
Sapuntzakis, Fig. 3 (showing Remote Cow Disks at HCP Server)

	A POSA would have understood that the remote COW Disk images would have been stored on a first storage unit of the server, i.e., on a first storage unit of the HCP server because it is the central server hosting these images. Decl. ¶312.
[7.2] storing leaf images for respective compute nodes 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 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; and	*Sapuntzakis* discloses *storing leaf images* (the local COW disk images of Figure 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). ***<u>See, e.g., Sapuntzakis</u>:*** "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." *Sapuntzakis,* 6. The local COW disk images holding the *leaf images* are *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." *Sapuntzakis,* 6-7. "Since **the latest COW disk is always local**, **all writes are local**." *Sapuntzakis,* 7. A POSA would have understood that, because writes to the local (latest) COW disk occur only in these circumstances, the local COW disk will include only the additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, but the local COW disk will not include blocks of the root image that are unchanged by respective compute nodes. Decl. ¶316. This is the standard operation of CoW snapshots like *Sapuntzakis'* local COW. *See, e.g.,* Decl. at Section VII.E (describing how CoWs are used in caching technologies).

[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	*Sapuntzakis* and *Holzmann* disclose *caching blocks of said root image* (*Sapuntzakis'* remote COW disks) *that have been accessed by at least one of said compute nodes in a cache memory* (*Sapuntzakis'* shadow COW disks in a memory such as *Holtzmann's* memory module 104). ***<u>See, e.g., Sapuntzakis</u>:*** "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." *Sapuntzakis,* 7. "Reads, 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." *Sapuntzakis,* 7. "First, **the memory in most systems caches disk blocks**." *Sapuntzakis,* 8. ***<u>See, e.g., Holzmann</u>*:** "...[T]he invention [of 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**." *Holzmann,* [0002]. A POSA would have understood the servers of *Holzmann* correspond to the compute nodes of the '844 patent. Decl. ¶322.  "FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101

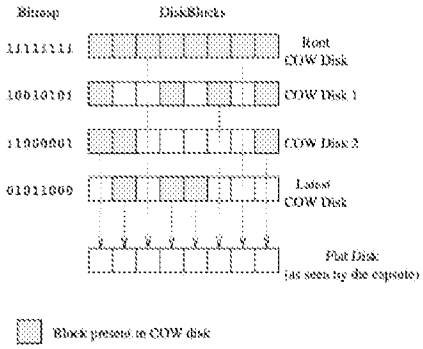
	having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." *Holzmann,* [0017]. "The **memory module 104** comprises **at least one direct-access memory module for holding data currently 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**." *Holzmann*, [0020]. A POSA would have recognized this description as describing a cache memory at least because RAM is often used as fast volatile cache and SSD storage devices are used as relatively fast non-volatile caches. Decl. ¶325; *see also* Decl. at Section VII.D (providing an overview of caching technologies). "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..." *Holzmann,* [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…" *Holzmann*, [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." *Holzmann*, [0033].

	A POSA would have understood that, when the cache of *Holzmann* is used to cache blocks of a read-only shared OS image, there is no need to create separate caches for each client because the read-only image is not subject to corruption. Decl. ¶329. That is, read-only data can easily be shared with multiple users and they will all see the same consistent data. Decl. ¶329. "If 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." *Holzmann,* [0034-35]. A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Sapuntzakis* for the reasons presented in Section V.A.4.
Claim 8	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[8.0] The method as recited in claim 7 further comprising:	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 7 for the reasons presented above for claim 7.
[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; and	*Sapuntzakis* and *Holzmann* disclose *receiving a read request* (*Sapuntzakis'* disclosure of disk server requests and *Holzmann's* teaching of using control module 103 monitoring requests for data) *from at least one of said compute nodes*, *wherein a first portion of the data requested is currently stored in said cache memory* (*Sapuntzakis'* disclosure of shadow COW disks with *Holzmann's* disclosure of how to treat requests when the data is available in the cache): ***See, e.g., Sapuntzakis:*** "Reads, 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." *Sapuntzakis,* 7. ***See, e.g., Holzmann*:** "[T]**he control module 103 monitors the request for operating system data**, and determines whether the…data block requested…has previously been loaded into the operating system cache. **If the data block has been loaded into the cache, and…has not been overwritten….[it] is provided from the cache**…" *Holzmann* [0024]. *See also* Decl. ¶352 (noting that "this limitation describes nothing more than common caching behavior (finding if something is cached in response to a read/write request)"); Decl. at Section VII.D (providing an overview of caching technologies).

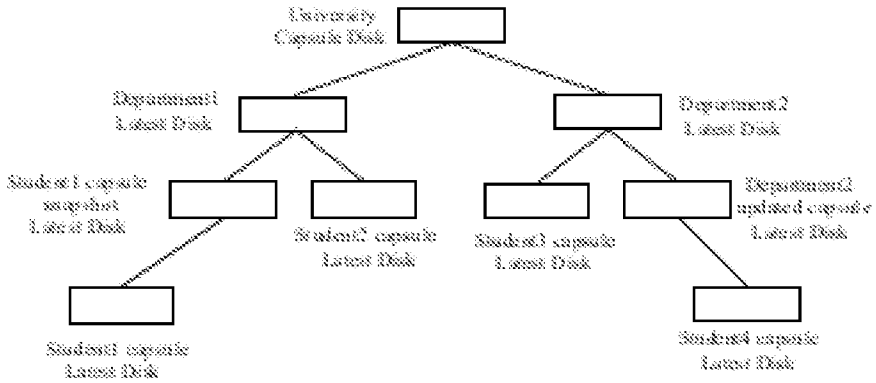
	A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Sapuntzakis* for the reasons presented in Section V.A.4.
[8.2] providing said first portion of said data to said at least one of said compute nodes from said cache memory.	*Sapuntzakis* and *Holzmann* disclose *providing said first portion of said data to said at least one of said compute nodes from said cache memory* (*Sapuntzakis'* disclosure of the shadow COW disks with *Holtzman's* details of providing data from the cache): ***See, e.g., Sapuntzakis:*** "Reads, 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**." *Sapuntzakis,* 7. A POSA would have understood from this passage that if the requested blocks *are* cached, they are served from the cache. Decl. ¶358. ***See, e.g., Holzmann*:** "[T]he control module 103 monitors the request for operating system data, and determines whether the…data block requested…has previously been loaded into the operating system cache. **If the data block has been loaded into the cache, and…has not been overwritten….[it] is provided from the cache**…" *Holzmann* [0024]. *See also* Decl. ¶355 (noting that "this limitation describes nothing more than common caching behavior (returning a cached item if found)"); Decl. at Section VII.D (providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Sapuntzakis* for the reasons presented in Section V.A.4.
Claim 9	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[9.0] The method as recited in claim 8 further comprising: updating said cache memory based on said read request.	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 8 for the reasons presented above for claim 8. *Sapuntzakis* and *Holzmann* further discloses *updating said cache memory* (*Sapuntzakis'* disclosure of updating the shadow COW disks with *Holzmann's* details on the cache memory) *based on said read request*: ***See, e.g., Sapuntzakis:*** "Reads, on the other hand, could either be local or remote. In

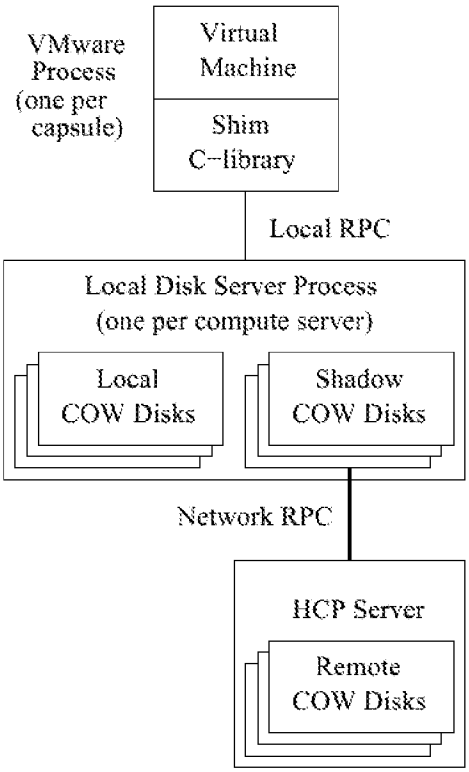
	the case of a remote read….**If the block is not cached locally, it is fetched remotely and added to the shadow COW**." *Sapuntzakis,* 7. ***See, e.g., Holzmann*:** "**If the requested data has not been loaded into the cache 501, the control module 103 copies the data from… memory portion 400 to the cache 501**." *Holzmann*, [0035]. *See also,* Decl. ¶360-63 (noting that "this limitation describes nothing more than common caching behavior (updating a cache based on reading a new item, called a "cache admission" policy")"); Decl. at Section VII.D (providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Sapuntzakis* for the reasons presented in Section V.A.4.
Claim 10	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[10.0] The method as recited in claim 9	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 9 for the reasons presented above for claim 9.
[10.1] 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; and	*Sapuntzakis* and *Holzmann* disclose 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* (*Sapuntzakis'* disclosure of updating the shadow COW with *Holzmann's* details for updating the cache). ***See, e.g., Sapuntzakis*:** "Reads, on the other hand, could either be local or remote. In the case of a remote read….**If the block is not cached locally, it is fetched remotely and added to the shadow COW**." *Sapuntzakis,* 7. ***See, e.g., Holzmann*:** "[T]he control module 103 monitors the request for operating system data, and determines whether the…data block requested…has previously been loaded into the operating system cache. If the data block has been loaded into the cache, and…has not been overwritten….[it] is provided from the cache…" *Holzmann* [0024]. "**If the requested data has not been loaded into the cache 501, the control module 103 copies the data from… memory portion 400 to the cache 501**." *Holzmann*, [0035]. *See also,* Decl. ¶368 (noting that "this limitation describes nothing more than common caching behavior (updating a cache based on reading a new item, called a "cache admission" policy")"); Decl. at Section VII.D

	(providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Sapuntzakis* for the reasons presented in Section V.A.4.
[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.	*Sapuntzakis* and *Holzmann* disclose *removing the least recently accessed data from said cache memory if the amount of data in said cache memory is above a threshold value* (*Sapuntzakis'* disclosure of replacing the data ("entry") with the lowest use count with, and to the extent more is required, *Holzmann's* acknowledgement that those skilled in the art would be aware of many cache configurations). ***See, e.g., Sapuntzakis:*** "On a miss, the cache adds the entry to the bucket, possibly **evicting an existing entry**. … 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** …." *Sapuntzakis,* 8. ***See, e.g., Holzmann:*** "It will be understood by those skilled in the art that still more cache configurations can be used in accordance with the invented system. *Holzmann* [0038]. *See also,* Decl. ¶372 (noting that "this limitation describes nothing more than common caching behavior (freeing some items from a cache to make room for new items")"); Decl. at Section VII.D (providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Sapuntzakis* for the reasons presented in Section V.A.4.
Claim 11	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 7 for the reasons presented above for claim 7. *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* (a consistent environment). That is, *Sapuntzakis'* COW disk hierarchy is invisible to the client, which perceives, instead, a single cohesive storage image corresponding to the '844 patent's merged root and leaf blocks. ***See, e.g., Sapuntzakis:*** "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." *Sapuntzakis,* 3. "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." *Sapuntzakis,* 7. "Reads involve searching the latest COW disk [the leaf image] and its ancestor disks in turn until the required block is found." *Sapuntzakis,* 6-7.  Figure 2: An example capsule disk. "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**." *Sapuntzakis,* 7. A POSA would have understood this to mean that the COW disk hierarchy is also invisible to the client, which perceives, instead, a single cohesive storage image corresponding to the '844 patent's merged root and leaf blocks. Decl. ¶382.
Claim 12	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[12.0] The method as recited in claim 11 wherein said merging occurs at an operational level between file systems of the respective compute nodes and said first storage unit, respective second storage units,	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 11 for the reasons presented above for claim 11. *Sapuntzakis* discloses that *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). ***See, e.g., Sapuntzakis:*** "The COW disk implementation interfaces with GSX Server

and said cache memory.	through a shim library….The shim library intercepts GSX Server's I/O requests to disk image files…and redirects them to a local disk server." *Sapuntzakis,* 7. "The local disk server translates these requests to COW disk requests, and executes the I/O operations against the COW disks." *Sapuntzakis,* 7. "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." *Sapuntzakis,* 7. A POSA would have understood the references to the COW disks in these quotes refer to all three of the root COW disks, the local (leaf) COW disks, and the shadow (cache) COW disks. Decl. ¶388. Indeed, consistent with how caching technologies worked at the time, a POSA would have understood that even without the presence of a COW snapshot, operating systems provide a cohesive view of data that logically merges what is in the cache and what is not cached. Decl. ¶388; Decl. at Section VII.D (providing an overview of caching technologies).
Claim 13	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[13.0] The method as recited in claim 7 wherein said first storage unit is read-only.	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 7 for the reasons presented above for claim 7. *Sapuntzakis* further discloses *the first storage unit* (storing root disk images and possibly certain intermediate COW disks) *is read-only* (immutable). ***See, e.g., Sapuntzakis:*** "**At the root of the hierarchy is a root disk, which is a complete capsule disk**. All other nodes represent a COW disk. Each path of COW disks originating from the root in the capsule hierarchy represents a capsule disk; the COW disk at the end of the path for a capsule disk is its latest disk. We cannot directly run a capsule whose latest disk is not a leaf of the hierarchy. We must first derive a new child capsule by adding a new child disk to the latest disk and all updates are made to the new disk. Thus, **once capsules have children, they become immutable**; this property simplifies the caching of capsules." *Sapuntzakis,* 7 (describing Figure 1). A POSA would have understood that the capsules with children are immutable and cannot be run directly means that they are read-only. Decl. ¶392.

Claim 1	SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)
[1.0] A system for providing data to a plurality of compute nodes, the system comprising:	To the extent the preamble is limiting, *Sapuntzakis discloses a system* (Collective system with computers and storage) *for providing data* (capsules) *to a plurality of computer nodes* (user computers). *See also, e.g.,* element [7.0]. ***See, e.g., Sapuntzakis:*** "**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**." *Sapuntzakis,* 3.
[1.1] a first storage unit configured to store blocks of a root image of said compute nodes;	*Sapuntzakis* discloses *a first storage unit* (a storage at the HCP server) *configured to store blocks of a root image* (root disc) *of said compute nodes* (computers). *See also, e.g.,* element [7.1]. ***See, e.g., Sapuntzakis:*** "At the root of the hierarchy is a ***root disk***, which is **a complete capsule disk**." *Sapuntzakis,* 6. "Figure 1 shows an example of a capsule hierarchy illustrating how it may be used in a university. The root capsule contains all the software available to all students." *Sapuntzakis,* 6.  Figure 1: An example capsule hierarchy. "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." *Sapuntzakis,* 7.

	 Figure 3: Implementation of capsule disks and demand paging. *Sapuntzakis,* Fig. 3 (showing Remote Cow Disks at HCP Server) A POSA would have understood that the remote COW Disk images would have been stored on a first storage unit of the server, i.e., on a first storage unit of the HCP server because it is the central server hosting these images. Decl. ¶312.
[1.2] a plurality of second storage units configured to store leaf images of respective compute nodes, 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 said root image, wherein said leaf images of respective compute nodes do not include	*Sapuntzakis* discloses *a plurality of second storage units* (units attached either to the HCP server or the client computer) *configured to store leaf images* (the local COW disk images of Figure 3), as described below. *Sapuntzakis* discloses that *said leaf images include only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of said root image and that said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes* for the reasons presented in element [7.2], presented above. *See also* Decl. at Section VII.E and ¶316. ***<u>See, e.g., Sapuntzakis</u>:*** "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." *Sapuntzakis,* 6.

blocks of said root image that are unchanged by respective compute nodes; and	The local COW disk images holding *the leaf images* are *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." *Sapuntzakis,* 6-7. "Since the latest COW disk is always local, all writes are local." *Sapuntzakis,* 7. A POSA would have understood that, because writes to the local (latest) COW disk occur only in these circumstances, the local COW disk will include only the additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, but the local COW disk will not include blocks of the root image that are unchanged by respective compute nodes. Decl. ¶316. This is the standard operation of CoW snapshots like *Sapuntzakis'* local COW. *See, e.g.,* Decl. at Section VII.E (describing how CoWs are used in caching technologies).
[1.3] a cache configured to cache blocks of said root image previously accessed by at least one of said compute nodes.	*Sapuntzakis* and *Holzmann* discloses *a cache* (*Sapuntzakis'* remote COW disks stored in a memory such as *Holtzman's* memory module 104) *configured to cache blocks of said root image previously accessed by at least one of said compute nodes* (*Sapuntzakis'* shadow COW disks). *See also, e.g.,* element [7.3]. ***See, e.g., Sapuntzakis:*** "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." *Sapuntzakis,* 6. "Reads, 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." *Sapuntzakis,* 7. "First, **the memory in most systems caches disk blocks**." *Sapuntzakis,* 8. ***See, e.g., Holzmann*:** "...[T]he invention [of 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."** ***Holzmann,*** [0002]. A POSA would have understood the servers of *Holzmann* correspond

to the compute nodes of the '844 patent. Decl. ¶322.

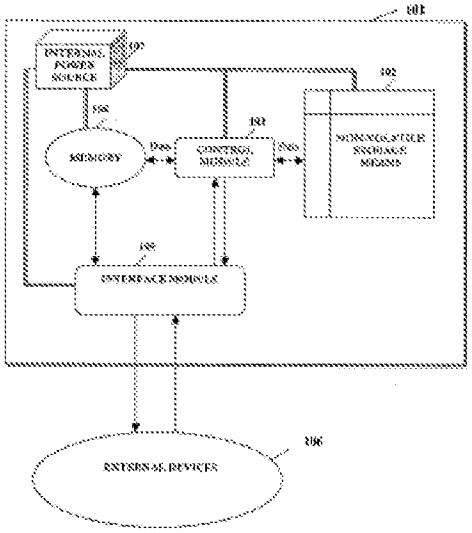


Fig. 1

"FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." *Holzmann,* [0017].

"**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**." *Holzmann*, [0020].

A POSA would have recognized this description as describing a cache memory at least because RAM is often used as fast volatile cache and SSD storage devices are used as relatively fast non-volatile caches. Decl. ¶325; *see also* Decl. at Section VII.D (providing an overview of caching technologies).

"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..."

	Holzmann, [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…" *Holzmann*, [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." *Holzmann*, [0033]. A POSA would have understood that, when the cache of *Holzmann* is used to cache blocks of a read-only shared OS image, there is no need to create separate caches for each client because the read-only image is not subject to corruption. Decl. ¶329. That is, read-only data can easily be shared with multiple users and they will all see the same consistent data. Decl. ¶329. "If 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." *Holzmann,* [0034-35]. A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the cache of *Sapuntzakis* for the reasons presented in Section V.A.4.
Claim 2	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[2.0] The system as recited in claim 1 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* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Sapuntzakis* and *Holzmann* further discloses *said cache being configured to store X most recently accessed blocks of said root image, and wherein X represents a cache threshold value* (*Sapuntzakis'* disclosure of replacing the data ("entry"), e.g., evicting an existing entry, and to the extent more is required, *Holzmann's* acknowledgement that those skilled in the art would be aware of many

	cache configurations). A POSA would have understood this limitation describes nothing more than common caching behavior, i.e., standard cache management policies, where caches naturally have limited room and must use thresholds to decide what to keep/evict. Decl. ¶403, Decl. at Section VII.D (particularly the known LRU and LFU cache-management policies). ***See, e.g., Sapuntzakis:*** "On 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." *Sapuntzakis,* 8. ***See, e.g., Holzmann*:** "If the requested data has not been loaded into the cache 501, the control module 103 copies the data from… memory portion 400 to the cache 501." *Holzmann*, [0035]. "It will be understood by those skilled in the art that still more cache configurations can be used in accordance with the invented system. *Holzmann* [0038]. *See also,* Decl. ¶399-403 (noting that "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)"); Decl. at Section VII.D (providing an overview of caching technologies, particularly LRU and LFU cache-management policies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the cache of *Sapuntzakis* for the reasons presented in Section V.A.4.
Claim 3	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[3.0] The system as recited in claim 1 wherein said first storage unit, said second storage units, and said cache are contained within a single storage appliance.	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Sapuntzakis* further discloses, or at least suggest, *said first storage unit, said second storage units, and said cache being contained within a single storage appliance*. A POSA would have understood that, as to *Sapuntzakis,* in settings where communication speed was not the main factor (*i.e.* for students and faculty working on campus on the fast internal network), the "local" COW disks (the leaf images) could be stored on the same device

	as the root images. Decl. ¶405; MPEP § 2143(I)(Rationale (G)) (known advantages of such a configuration as applied to the example *Sapuntzakis* discloses provides the motivation). 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]." *Sapuntzakis,* 8. Thus, POSA would know that caching these on a centralized appliance would be advantageous. Decl. ¶406; MPEP § 2143(I)(Rationale (G)) (known advantages of caching centralized applications providing motivation). 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*). Decl. ¶407-08. It would have also been obvious to try such a configuration, choosing from a finite set of predictable solutions (1. All three storage components within the same appliance, 2. Two of the three storage components within the same appliance, and 3. All three storage components within different appliances) with a reasonable expectation of success given the state of such technologies at the time. *See, e.g.,* Decl. ¶407-08 and Decl. at Section VII.E. *See also* Decl. at Section VII.E (providing an overview of copy-on-write and snapshot technologies).
Claim 4	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[4.0] The system as recited in claim 1 further comprising: 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	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Sapuntzakis,* particularly in view of POSA knowledge of well-known and off the shelf products for providing devices configured to interface as recited in this claim, disclose or at least render obvious this claim. As a preliminary matter, the term Union Block Device is informally defined in the '844 patent by way of a set of tasks it performs. *See, e.g.,* Section III.A.2. That is, any device or component that operates to perform the recited elements would constitute a UBD because a UBD is essentially described in terms of its capabilities. <u>First, *Sapuntzakis* discloses</u> that its system is *configured to* provide an *interface between respective compute nodes* (user computers) *and said first storage unit* (a first storage unit of the HCP server), *respective second storage units* (units attached either to the HCP server or the client computer), *and said cache* (remote COW disks) *to distribute application environments* (updated and/or complete versions of capsules) *to the compute nodes* (user computers). *See, e.g.,* Section V.A.5, Claim 1, providing mappings for the components of

said root image with the blocks of respective leaf images.	*Sapuntzakis'* system interfacing (communicating) with each other to provide complete data (capsules or the entire root image) and/or needed new blocks of data (the ancestors between the root COW disk and the local COW disk) to a plurality of computer nodes (user computers). *Sapuntzakis* also discloses that its system is *configured to create said application environments by merging the blocks of said root image* (the root COW disk) *with the blocks of respective leaf images* (the local COW disk and the ancestors between it and the root). *See, e.g.,* Section V.A.5., Claims 11 and 12, providing mappings for the components of *Sapuntzakis'* system *configured to create a cohesive application environment by merging the blocks of the root image with blocks of the leaf image using data* (claim 11) *from the compute nodes, the first storage unit, the second storage units and the cache memory* (claim 12). Thus, *Sapuntzakis'* disclosures of devices capable of performing the elements of this claim disclose this claim in the same manner as the '844 patent discloses it, i.e., the devices that perform the recited functions are collectively known as the union block devices. Decl. ¶410. *See Sapuntzakis,* 3, 6-7, and Figs. 1 and 3 as mapped for claims 1, 11, and 12 above. <u>Second, to the extent more is required</u>, the '844 patent discloses an example that "In one embodiment, UBDs 230a-n 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. … Because UBDs 230a-n operate below the file system, they are concerned merely with the blocks of data themselves, rather than files they form." EX1001, 5:60-6:1. This is consistent with the basic understanding of how to implement copy-on-write and snapshot technologies, such as the engine/driver used in such off-the-shelf appliances as Netapp. *See, e.g.,* Decl. ¶108 and Decl. at Sections VII.C. and VII.E. A POSA would have understood that *Sapuntzakis* system would be configured with the necessary components to provide the features and functions that *Sapuntzakis* discloses. That is, the basic and well-known information as to how to implement copy-on-write and snapshot technologies is not explicitly disclosed in *Sapuntzakis* because such well-known POSA knowledge would not need to be explicitly stated. Decl. ¶410; *see also* Decl. at Sections VIII.C (describing how the '844 patent discloses UBDs) and VII.E (providing an overview of copy-on-write and snapshot technologies that employed devices that constitute the claimed UBDs). A POSA would have been well aware of storage devices capable of performing the disclosed functions, such as the devices created by

	Netapp, which are single storage appliances, each supporting multiple volumes, multiple base images, multiple snapshots, and multiple caches (both volatile and non-volatile) and employing a CoW engine/driver that resides between the file system layer and the actual storage devices. *See, e.g.,* Decl. ¶410 and Decl. at Section VII.E. That is, the claimed UBDs are merely off-the-shelf devices that any POSA would employ when configuring *Sapuntzakis'* system. A POSA would have understood each volume in a Netapp or similar appliance to be employed by the system of *Sapuntzakis* to provide the functions disclosed by *Sapuntzakis*, thereby providing the claimed UBDs. *See, e.g.,* Decl. ¶410 and Decl. at Section VII.E. In other words, the "low level drivers" disclosed in the '844 patent as UBDs are merely describing well-known and off-the-shelf products such as Netapp's CoW engine/driver that resides between the file system layer and the actual storage devices. *See, e.g.,* Decl. ¶108. Accordingly, the devices in communication with each other and performing the functions of this claim as detailed in the mappings for *Sapuntzakis* for claims 1, 11, and 12 disclose the claimed UBDs of this claim and/or the POSA knowledge to employ known devices for performing the functions *Sapuntzakis* discloses (e.g., a Netapp or similar appliance) disclose this element. Decl. ¶410.
Claim 5	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[5.0] The system as recited in claim 4 wherein said union block devices comprise low-level drivers for interfacing between the file systems of respective compute nodes and said first storage unit, respective second storage units, and said cache.	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 4 for the reasons presented above for claim 4. As explained for claim 4, the term Union Block Device is informally defined in the '844 patent by way of a set of tasks it performs. *See, e.g.,* Section III.A.2. Thus, the system components as necessary to provide such features, taken together, embody the Union Block Device of this claim. Decl. ¶410. Reading *Sapuntzakis,* a POSA would have understood that *Sapuntzakis* would have been implemented with storage systems capable of performing the functions in those references, such as the storage systems created by Netapp or similar appliances. *See, e.g.,* mapping of claim 4 above, Decl. at Sections VII.C and VII.E. These systems provide components—e.g., the claimed *low-level drivers—for interfacing between* the components to accomplish the disclosed functions. *See, e.g.,* Decl. ¶410-411, Decl. ¶108 (noting that Netapp's CoW driver resides between the file system layer and the actual storage devices), and Decl. at Section VII.E. Accordingly, the devices in communication with each other and performing the functions of this claim provided in the mappings of *Sapuntzakis* to claims 1, 11, and 12 above, as understood to be

	implemented by a POSA (*see, e.g.,* mapping of *Sapuntzakis* to claim 4), disclose the UBDs recited in this claim. Decl. ¶410-11, Decl. at Sections VII.C and VII.E.
Claim 6	**SNQ1: *Sapuntzakis* (EX1005) and *Holzmann* (EX1007)**
[6.0] The system as recited in claim 1 wherein said first storage unit is read-only.	*Sapuntzakis* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Sapuntzakis* further discloses *the first storage unit* (storing root disk images and possibly certain intermediate COW disks) *is read-only* (immutable). ***See, e.g., Sapuntzakis:*** "**At the root of the hierarchy is a root disk, which is a complete capsule disk**. All other nodes represent a COW disk. Each path of COW disks originating from the root in the capsule hierarchy represents a capsule disk; the COW disk at the end of the path for a capsule disk is its latest disk. We cannot directly run a capsule whose latest disk is not a leaf of the hierarchy. We must first derive a new child capsule by adding a new child disk to the latest disk and all updates are made to the new disk. Thus, **once capsules have children, they become immutable**; this property simplifies the caching of capsules." *Sapuntzakis,* 6 (describing Figure 1). A POSA would have understood that the capsules with children are immutable and cannot be run directly means that they are read-only. Decl. ¶392.

B. SNQ2: *Sapuntzakis* in view of *Federwisch* Presents a Significant New Question of Patentability for Challenged Claims 14-27

1. Overview of SNQ2

Claims 14-27 are obvious as explained herein, accompanied by the claim chart provided in Section V.B.5 and the Zadok Declaration (Decl. ¶413-90), are rendered obvious by *Sapuntzakis* in view of *Federwisch*. Notably, *Sapuntzakis* discloses the claimed root/leaf architecture, including where leaf images only contain additional data blocks not previously contained in the root image and changes made by respective compute nodes to the blocks of the root image. *Federwisch* discloses any additional indexing limitations to the extent, if any, not already taught by *Sapuntzakis* and/or obvious to a POSA.

2. Overview of *Sapuntzakis*

An overview of *Sapuntzakis* is provided in Section V.B.2.

3. Overview of *Federwisch*

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." EX1008, Abstract. That is, 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…" EX1008, Abstract, [0071-72].

Each snapshot (the "leaf image") has its own index of file blocks. EX1008, [0109]. The system performs re-indexing when updating destination snapshots' index maps. EX1008, [0134, 0137]. It uses NVRAM and buffer caches for both snapshots and active file systems. EX1008, [0048, 0053, 0058, 0070]. Data is shared ("exported" over a network), thus "allowing source and destination systems of different vendors to share the information." EX1008, [0015, 0086, 0120]; *see also*, Decl. ¶¶257-73.

Federwisch is analogous to the '844 patent. *Federwisch* is in the same field of endeavor as the '844 patent, i.e., the management of whole OS images being shared by users on a network. *Compare, e.g.,* EX1001, 1:31-2:51 ("Clustered computing involves multiple computer nodes…that work together to achieve a common task"; "A common example of this is Operating System (OS) software image management, where the compute nodes are activated with the distributed application environment by either diskless booting protocols or remote software installation to local storage"; "Thus, embodiments of the present disclosure provide an operating system-independent system and method for distributing an application environment to a computer node…allow[ing] creation of boot images on the fly without significantly diminishing bring-up time";) *with* EX1008, Abstract ("A system and method for remote asynchronous replication or mirroring of changes in a source file system snapshot in a destination replica file system using a scan (via a scanner) of the blocks…which identifies changed blocks in the respective snapshot files based"), [0002] ("a file server or filer includes a storage operating system that implements a file system to logically organize the

information").

Federwisch is also reasonably pertinent to a particular problem with which the inventors were concerned, such as efficient storage/transfer of OS images, with both *Federwisch* and the '844 patent disclosing indexing to improve efficiency. *Compare* EX1001, Title, 2:54-67 (describing issues with known systems either "a large bring-up time," "wasted disk space," and a "cumbersome" process), 2:36-45 (addressing the known issues with its disclosed "technology for, among other things, root image caching and indexing for block-level distributed application management…"); *with* EX1008, Abstract, [0071-72] (describing how to efficiently store/transfer OS snapshots with a "scanner [that] 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").

4. Combination of *Sapuntzakis* and *Federwisch*

The proposed combination of *Sapuntzakis* with *Federwisch* is implementing *Federwisch's* details of indexing into *Sapuntzakis'* storage systems, files, and caches thereof.

The Supreme Court in *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398 (2007) adopted a flexible approach to evaluating whether a patent claiming a combination of elements of prior art is obvious. Under *KSR* and as detailed in the Zadok Declaration (EX1003), a POSA would have been motivated to combine *Sapuntzakis* and *Federwisch*.

A POSA would have been motivated to modify *Sapuntzakis* with *Federwisch's* indexing techniques 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. Decl. ¶429-51. Thus, a POSA 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*). Decl. ¶429-51. That is, so motivated by *Sapuntzakis'* disclosure of indexing, to the extent a POSA required implementation details, they would have turned to *Federwisch*. *See* Decl. ¶429-51; MPEP § 2143(I)(Rationale (G)). Moreover, application of the disclosed indexing techniques described in *Federwisch* to *Sapuntzakis* represents the use of *Federwisch's* known techniques for indexing to improve the storage systems of *Sapuntzakis* in the same way. *See* Decl. ¶429-51; MPEP § 2143(I)(Rationale (C)).

Moreover, a POSA would have known that indexing of any data is an age-old technique long utilized in computer systems. *See, e.g.,* Decl. ¶449, Decl. at Section VII.H. Just as a library catalog provides an efficient way of locating books on shelves, so do computers create, manage, and update indexes of many data types (files, disk blocks, images, virtual machines, users, networks, and many more). *See, e.g.,* Decl. ¶449, Decl. at Section VII.H.

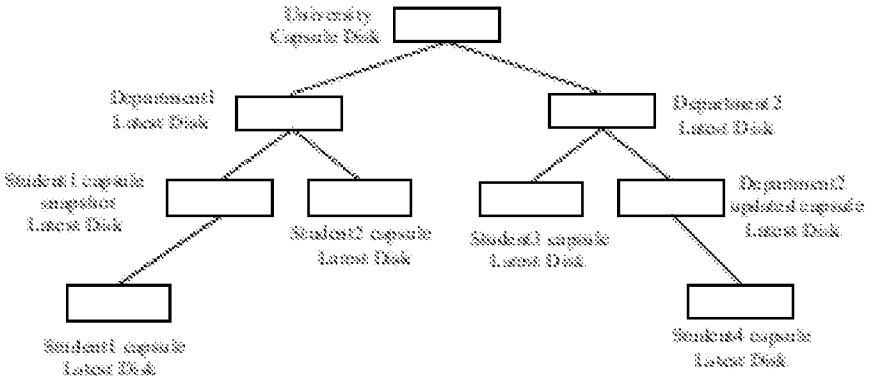
Thus, a POSA would have had a reasonable expectation of success because indexing of data is an age-old technique long utilized in computer systems. *See* Decl. ¶429-51. Moreover, there was significant overlap in *Sapuntzakis'* and *Federwisch's* disclosures—both operate at and recognize disk blocks at the block layer (while also recognizing that file systems interoperate and work atop the block layer), disclose/use CoW snapshots (a POSA would have known CoW techniques enable snapshots), and disclose caching blocks of the root/boot disk images. Thus, a POSA would have had a reasonable expectation that the indexing techniques that were disclosed in *Federwisch* would operate in *Sapuntzakis'* similar system.

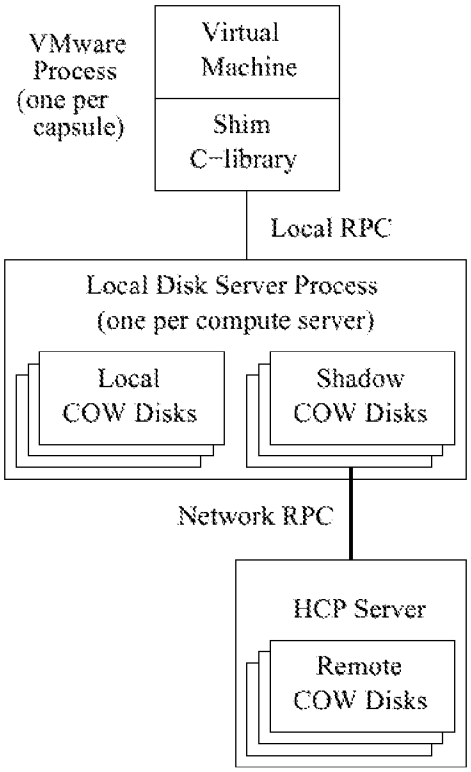
Modifying *Sapuntzakis* to include *Federwisch's* indexing would have been simple to a POSA, and would not have required extensive experimentation or imposed undue burden – including because *Sapuntzakis* already indexed data objects. For the same reasons, the combination would have yielded predictable results, as indexes are well understood to structure and locate data objects efficiently, and a POSA would have had a high expectation of success, because indexing techniques were well understood for decades beforehand. *See* Decl. ¶429-51.

5. Claim Chart Providing Detailed Analysis[15]

Claim 14	SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)
[14.0] A system for indexing file systems for a plurality of compute nodes, the system comprising:	To the extent the preamble is limiting, *Sapuntzakis discloses a system for indexing file systems* (Collective system with the file systems of respective computers and storage) *for a plurality of computer nodes* (user computers). *See also, e.g.,* Section V.A.5, element [7.0]. ***See, e.g., Sapuntzakis:*** "**Computers and storage in the Collective system** act as

[15] This claim chart mapping is further supported with an analysis in prose form provided by Dr. Zadok in his declaration. Decl. at Section XII.B.

	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." *Sapuntzakis,* 3. "The 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." *Sapuntzakis,* 8. *See also* elements below, particularly element [14.3.d], to the extent more is required.
[14.1] a first storage unit configured to store blocks of a root image of said compute nodes;	*Sapuntzakis* discloses *a first storage unit* (remote COW disks on HCP server of Figure 3) *configured to store blocks of a root image (root disk) of said compute nodes* (computers). *See also, e.g.,* Section V.A.5, element [7.1]. ***<u>See, e.g., Sapuntzakis</u>:*** "**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." *Sapuntzakis,* 3. "At the root of the hierarchy is a ***root disk***, which is **a complete capsule disk**." *Sapuntzakis,* 6. "Figure 1 shows an example of a capsule hierarchy illustrating how it may be used in a university. The root capsule contains all the software available to all students." *Sapuntzakis,* 6.  Figure 1: An example capsule hierarchy.

	"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." *Sapuntzakis,* 7.  Figure 3: Implementation of capsule disks and demand paging. *Sapuntzakis,* Fig. 3 (showing Remote Cow Disks at HCP Server) A POSA would have understood that the remote COW Disk images would have been stored on a first storage unit of the server, i.e., on a first storage unit of the HCP server because it is the central server hosting these images. Decl. ¶312.
[14.2] a plurality of second storage units configured to store leaf images of respective compute nodes, said leaf images comprising only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the	*Sapuntzakis* discloses *a plurality of second storage units* (attached either to the HCP server or the client computer) *configured to store leaf images* (the local COW disk images of Figure 3) *of respective compute nodes, said leaf images comprising 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). *See also* Section V.A.5, element [7.2]. ***<u>See, e.g., Sapuntzakis</u>:*** "We can store the disks in these capsules efficiently by

<table>
<tr>
<td>blocks of said 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; and</td>
<td>
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." Sapuntzakis, 6.

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."
Sapuntzakis, 6-7.

"Since the latest COW disk is always local, all writes are local." Sapuntzakis, 7.

A POSA would have understood that, because writes to the local (latest) COW disk occur only in these circumstances, the local COW disk will include only the additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, but the local COW disk will not include blocks of the root image that are unchanged by respective compute nodes. Decl. ¶316. This is the standard operation of CoW snapshots like Sapuntzakis' local COW. See, e.g., Decl. at Section VII.E (describing how CoWs are used in caching technologies).
</td>
</tr>
<tr>
<td>[14.3.a] 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</td>
<td>
Element [14.3.a] is substantially similar to recitations in element [4.0]. See comparison below.
<table>
<tr>
<td>[14.3.a] 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</td>
<td>[4.0] 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]]</td>
</tr>
</table>

The UBDs that correspond to said computer nodes recited in element [14.3.a] are consistent with the recitation of the UBDs for respective computer nodes recited in element [4.0].

Accordingly, Sapuntzakis discloses element [14.3.a] for the reasons presented in the mapping of Sapuntzakis to element [4.0] above in Section V.A.5.
</td>
</tr>
</table>

<table>
<tr>
<td>[14.3.b] [UBDs configured] to distribute said file systems to said compute nodes,</td>
<td>
Element [14.3.b] is substantially similar to recitations in element [4.0]. See comparison below.
<table>
<tr><td>[14.3.b] to distribute said file systems to said compute nodes,</td><td>[4.0] to distribute application environments to the compute nodes,</td></tr>
</table>
File systems and application environment are substantially synonymous in this context, e.g., an application environment is largely defined by the file systems employed at the computer nodes, and by distributing the file systems, the application environment is created/distributed.

Moreover, regardless of the particular definition Patent Owner puts forth, Sapuntzakis' disclosure of a collective system that can distribute capsules or new COW data to give "users a consistent environment" (see, e.g., Sapuntzakis, 3) discloses the claimed UBD to distribute file systems for similar reasons as explained for claim [4.0] and the distribution of an application environment.

Accordingly, Sapuntzakis discloses element [14.3.b] for the reasons presented in the mapping of Sapuntzakis to element [4.0] above in Section V.A.5.
</td>
</tr>
<tr>
<td>[14.3.c] wherein said union block devices are configured to create said file systems by merging the blocks of said root image stored on the first storage unit with the blocks of respective leaf images stored on respective second storage units, and</td>
<td>
Element [14.3.c] is substantially similar to recitations in elements [4.0], [1.1] and [1.2]. See comparison below.
<table>
<tr><td>[14.3.c] wherein said union block devices are configured to create said file systems by merging the blocks of said <u>root image</u> <u>stored on the first storage unit</u> with the blocks of respective <u>leaf images</u> <u>stored on respective second storage units</u>,</td><td>[4.0] 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.

[1.1] <u>a first storage unit configured to store blocks of a root image of said compute nodes</u>

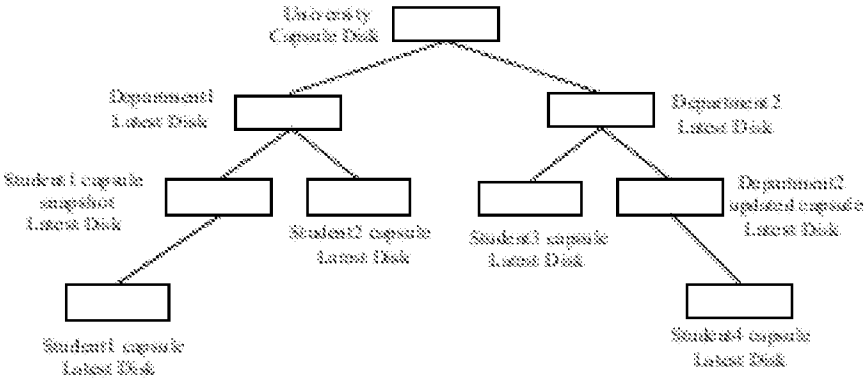
[1.2] <u>a plurality of second storage units configured to store leaf images of respective compute nodes</u></td></tr>
</table>
As explained for element [14.3.b], file systems (element [14.3.b-c) and application environments (element [4.0]) are substantially similar in this context.

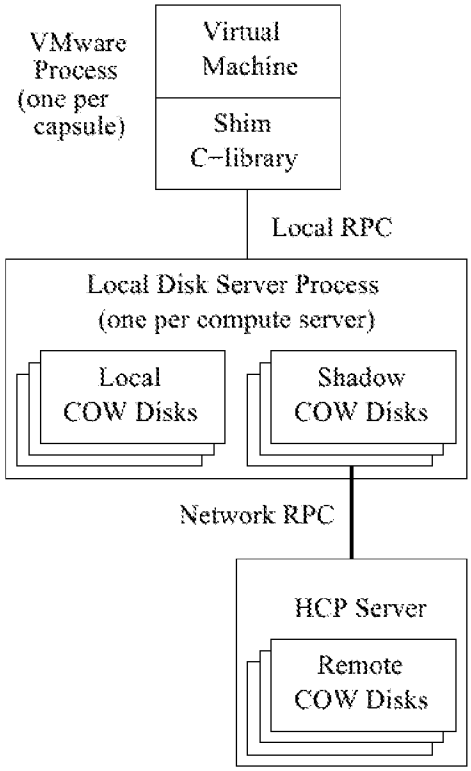
Accordingly, Sapuntzakis discloses element [14.3.c] for the reasons presented in the mapping of Sapuntzakis to elements [4.0], [1.1], and [1.2] above in Section V.A.5.
</td>
</tr>
</table>

[14.3.d] 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.	*Sapuntzakis* and *Federwisch* further disclose *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*. In particular, *Sapuntzakis* discloses that data is *indexed*, but it does not provide many details for the well-known and common technique. Decl. ¶422, 424, To the extent a POSA would require implementation details, Federwisch discloses a "scanner" which serves to *index* an image. Decl. ¶417-20, 426-28, Federwisch Abstract, [0053], [0071], [0072]. ***See, e.g., Sapuntzakis:*** "The 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." *Sapuntzakis,* 8. "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." *Sapuntzakis,* 8. ***See, e.g., Federwisch*:** "A system and method for remote asynchronous replication or mirroring of changes in a source file system snapshot in a destination replica file system 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**." *Federwisch,* Abstract. "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." *Federwisch,* [0053]. "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**." *Federwisch,* [0071]. "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." *Federwisch,* [0072]. Thus, in building the system of *Sapuntzakis,* a POSA would have implemented the indexing system of *Federwisch* on any compute node of *Sapuntzakis* to index the root image and provide the index results to other compute nodes. Decl. ¶419.

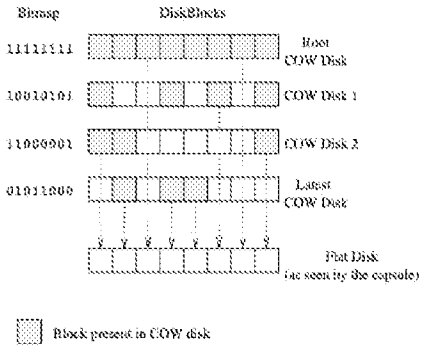
	See also Decl. ¶416-28, ¶429-51, Decl. at Sections VIII.B (discussing the '844 patent's disclosure of indexing) and VII.F (providing an overview of indexing technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Federwisch's* indexing to the indexing of *Sapuntzakis* for the reasons presented in Section V.B.4.
Claim 15	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[15.0] The system as recited in claim 14 wherein said first storage unit and said second storage units are contained within a single storage appliance.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 14 for the reasons presented above for claim 14. *Sapuntzakis* further discloses the remainder of the recitations of claim 15 for the reasons presented for the mapping of *Sapuntzakis* to element [3.0] in Section V.A.5.
Claim 16	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[16.0] The system as recited in claim 14 further comprising: 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.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 14 for the reasons presented above for claim 14. *Sapuntzakis* further discloses the remainder of the recitations of claim 16 for the reasons presented for the mapping of *Sapuntzakis* to element [4.0] in Section V.A.5.

Claim 17	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[17.0] The system as recited in claim 16 wherein said union block devices comprise low-level drivers for interfacing between the file systems of respective compute nodes and said first storage unit, respective second storage units, and said cache.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 16 for the reasons presented above for claim 16. *Sapuntzakis* further discloses the remainder of the recitations of claim 17 for the reasons presented for the mapping of *Sapuntzakis* to element [5.0] in Section V.A.5.
Claim 18	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[18.0] The system as recited in claim 14 wherein said first storage unit is read-only.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 14 for the reasons presented above for claim 14. *Sapuntzakis* further discloses the remainder of the recitations of claim 18 for the reasons presented for the mapping of *Sapuntzakis* to element [13.0] in Section V.A.5.
Claim 19	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[19.0] A method for indexing file systems for a plurality of compute nodes, comprising:	To the extent the preamble is limiting, *Sapuntzakis* discloses a method for indexing file systems (methods employed by its collective system with file systems of respective computers and storage) for a plurality of computer nodes (use computers). *See also, e.g.,* Section V.A.5, element [7.0]. ***See, e.g., Sapuntzakis:*** "**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." *Sapuntzakis,* 3. "The 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** the table." *Sapuntzakis,* 8. *See also* elements below, particularly element [19.4]-[19.5], to the extent more is required.

[19.1] storing blocks of a root image of said compute nodes 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). *See also, e.g.,* Section V.A.5, element [7.1]. ***See, e.g., Sapuntzakis:*** "**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." *Sapuntzakis,* 3. "At the root of the hierarchy is a ***root disk***, which is **a complete capsule disk**." *Sapuntzakis,* 6. "Figure 1 shows an example of a capsule hierarchy illustrating how it may be used in a university. The root capsule contains all the software available to all students." *Sapuntzakis,* 6.  Figure 1: An example capsule hierarchy. "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." *Sapuntzakis,* 7.

	 Figure 3: Implementation of capsule disks and demand paging. *Sapuntzakis,* Fig. 3 (showing Remote Cow Disks at HCP Server) A POSA would have understood that the remote COW Disk images would have been stored on a first storage unit of the server, i.e., on a first storage unit of the HCP server because it is the central server hosting these images. Decl. ¶312.
[19.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images comprising 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, wherein said leaf images for respective compute nodes do not include	*Sapuntzakis* discloses *storing leaf images for respective compute nodes* (the local COW disk images of Figure 3) *on respective second storage units* (attached either to the HCP server or the client computer), *said leaf images comprising 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). *See also* Section V.A.5, element [7.2]. ***<u>See, e.g., Sapuntzakis</u>:*** "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." *Sapuntzakis,* 6.

blocks of said root image that are unchanged by respective compute nodes;	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." *Sapuntzakis,* 6-7. "Since **the latest COW disk is always local, all writes are local**." *Sapuntzakis,* 7. A POSA would have understood that, because writes to the local (latest) COW disk occur only in these circumstances, the local COW disk will include only the additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, but the local COW disk will not include blocks of the root image that are unchanged by respective compute nodes. Decl. ¶316. This is the standard operation of CoW snapshots like *Sapuntzakis'* local COW. *See, e.g.,* Decl. at Section VII.E (describing how CoWs are used in caching technologies).
[19.3] merging the blocks of said root image with the blocks of respective leaf images stored on respective second storage units to create respective file systems for respective compute nodes;	*Sapuntzakis* 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) *stored on respective second storage units* (units attached either to the HCP server or the client computer) *to create respective file systems for respective compute nodes* (user computers). In this way, cohesive respective application environments are created. *See also* Section V.A.5, elements [11.0] and [1.2]. That is, *Sapuntzakis'* COW disk hierarchy is invisible to the client, which perceives, instead, a single cohesive storage image corresponding to the '844 patent's merged root and leaf blocks. ***See, e.g., Sapuntzakis:*** "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." *Sapuntzakis,* 3. "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." *Sapuntzakis,* 7.

	"Reads involve searching the latest COW disk [the leaf image] and its ancestor disks in turn until the required block is found." *Sapuntzakis,* 6-7.  Figure 2: An example capsule disk. "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**." *Sapuntzakis,* 7. A POSA would have understood this to mean that the COW disk hierarchy is also invisible to the client, which perceives, instead, a single cohesive storage image corresponding to the '844 patent's merged root and leaf blocks. Decl. ¶382. The local COW disk images holding *the leaf images* are *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." *Sapuntzakis,* 6-7. "Since **the latest COW disk is always local, all writes are local**." *Sapuntzakis,* 7.
[19.4] receiving indexing results pertaining to said root image from one of said compute nodes; and	As explained for element [14.3.d], *Sapuntzakis* and *Federwisch* further disclose *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*. It logically follows that, since *Sapuntzakis* and *Federwisch* discloses that "at least one of the compute nodes" is configured to "provide the indexing results" to "another of said computer nodes," then the "another one of said compute nodes" is configured to "receiv[e] indexing results" from the "one of said compute nodes." Therefore, *Sapuntzakis* and *Federwisch* disclose this element for the reasons presented for element [14.3.d].

[19.5] providing said indexing results to the others of said compute nodes.	*Sapuntzakis* and *Federwisch* disclose this element for the reasons presented for element [14.3.d].
Claim 20	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[20.0] The method as recited in claim 19 further comprising: storing said indexing results on a shared storage unit.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 19 for the reasons presented above for claim 19. *Sapuntzakis* and *Federwisch* further disclose *storing said indexing results* on "at least one of said compute nodes" that is configured to index said root image and provide the indexing results to another of said compute nodes for the reasons presented for element [14.3.d]. *Federwisch* further discloses *sharing* indexes in a system-agnostic format and the ability to share indexes among devices as "exports" and discloses a "first snapshot [that] may be an exported snapshot of the replicated snapshot on the destination." Thus, a POSA would have found it obvious that the "at least one of said compute nodes" would be *a shared storage unit*. That is, a POSA would have been motivated by the known advantages of storing indexing results on a shared storage unit such as making the index conveniently accessible over network storage volumes. *See, e.g.,* Decl. ¶463, Decl. at Section VII.C (providing an overview of networked and distributed storage system technologies); MPEP § 2143(I)(Rationale G). A POSA would have had a reasonable expectation of success in using a shared storage unit at least because using shared storage was common and well known. Decl. ¶463, Decl. at Section VII.C. ***See, e.g., Federwisch*:** "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." *Federwisch,* ¶[0086] "The **destination file system is exported** read-only the user." *Federwisch,* ¶[0015]. "Reference is made to FIG. 16, which describes a generalized rollback procedure 1600 according to an illustrative embodiment. As a matter of ongoing operation, in step 1605, a "first" snapshot is created. **This first snapshot may be an exported snapshot of the replicated snapshot on the destination**."

Claim 21	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[21.0] The method as recited in claim 19 wherein said merging occurs at an operational level between respective file systems of the compute nodes and said first storage unit and respective second storage units.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 19 for the reasons presented above for claim 19. *Sapuntzakis* further discloses the remainder of the recitations of claim 21 for the reasons presented for the mapping of *Sapuntzakis* to element [12.0] in Section V.A.5.
Claim 22	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[22.0] The method as recited in claim 19 wherein said first storage unit is read-only.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 19 for the reasons presented above for claim 19. *Sapuntzakis* further discloses the remainder of the recitations of claim 22 for the reasons presented for the mapping of *Sapuntzakis* to element [12.0] in Section V.A.5.
Claim 23	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[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:	To the extent the preamble is limiting, *Sapuntzakis* discloses *a computer-readable storage medium having instructions stored thereon that* (software on one or more of the computers and storage in the collective system), *in response to execution by at least one computing device, cause the at least one computing device to* perform steps. *See also* mappings of claim 19 above. ***See, e.g., Sapuntzakis:*** "**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." *Sapuntzakis,* 3. "We have chosen to move x86 computer states because **x86 computers** are common, cheap, **run the software we use**, and have tools for migration." *Sapuntzakis,* 2.
[23.1] receive data blocks of a file system, said data blocks comprising a root image portion and leaf	*Sapuntzakis* discloses *receiving data blocks of a file system* (duplication, distribution, and updating of capsules to many different machines including, for example, receiving such data blocks by one machine). *Sapuntzakis* discloses that *said data blocks comprise a root image*

image portion, said leaf image portion comprising only additional data blocks not previously contained in said root image portion and changes made by said first compute node[16] to the blocks of said root image, wherein said leaf image portion does not include blocks of said root image that are unchanged by said first compute node, wherein said file system is the result of merging said root image portion and said leaf image portion together at the block-level;	*portion* (root image) *and a leaf image portion* (the local COW disk images of Figure 3), *said leaf image portion comprising only* additional data blocks not previously contained in said root image portion (write requests for new data) *and changes made by said first compute node* (one of the computer machines) *to the blocks of the root image* (write request for changes to operating system data), *wherein said leaf image portion does not include blocks of said root image that are unchanged by said first compute node* (only new data and changes are written). *See also* Section V.A.5, element [7.2], and V.B.5, element [19.2]. *Sapuntzakis* discloses that *said file system is the result of merging said root image portion* (the root COW disk) *and said leaf image portion* (the local COW disk and ancestors between it and the root) *together at the block level*. *See also* Section V.A.5., element [11.0], and V.B.5, element [19.3]. ***See, e.g., Sapuntzakis:*** "**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." *Sapuntzakis,* 3. "At the root of the hierarchy is a ***root disk***, which is **a complete capsule disk**." *Sapuntzakis,* 6. "Figure 1 shows an example of a capsule hierarchy illustrating how it may be used in a university. The root capsule contains all the software available to all students." *Sapuntzakis,* 6.

[16] Requester notes that there is no antecedent basis for "said first compute node." For purposes of this Request, the broadest reasonable interpretation of this element includes the interpretation as "a first computer node." Notably, the "first *storage unit*" as recited claim 1 is not equivalent or related to the "first *compute node*" of claim 23. In claim 1, the leaf images of "respective compute nodes" (e.g., of the "first compute node" of this claim 23) are stored in a plurality of "*second* storage units." Indeed, claim 23 does not recite either of the first or second storage units that are recited in claim 1.

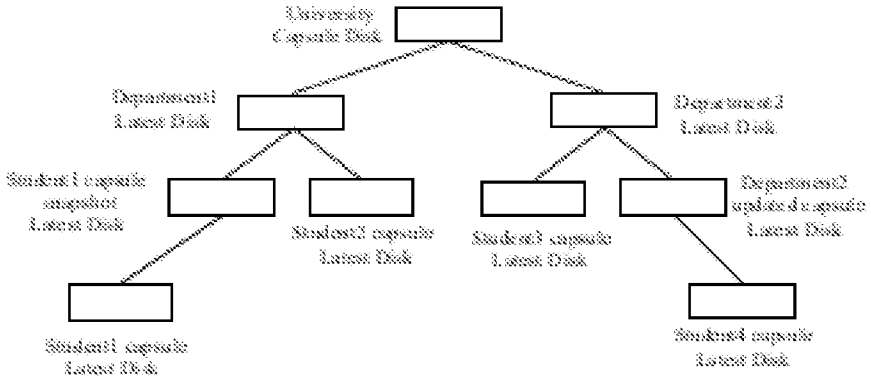


Figure 1: An example capsule hierarchy.

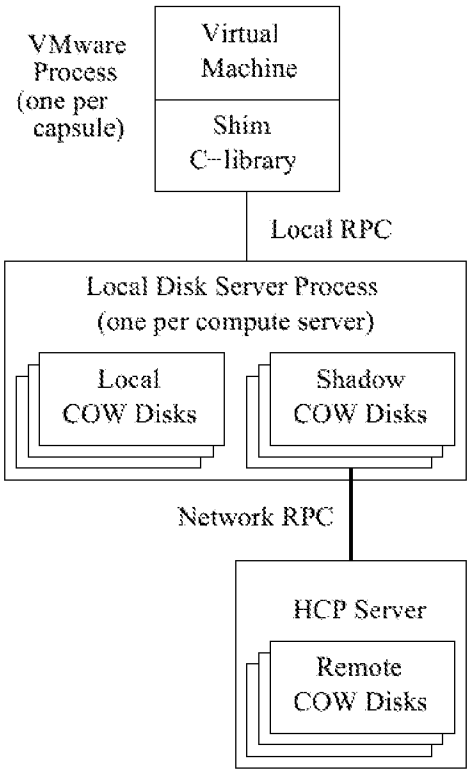
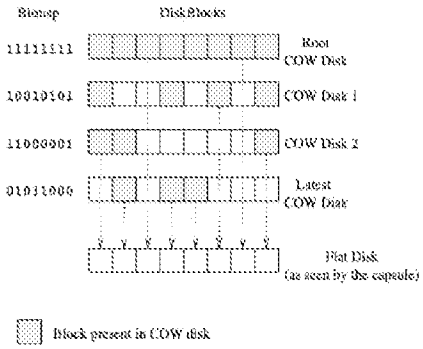


Figure 3: Implementation of capsule disks and demand paging.

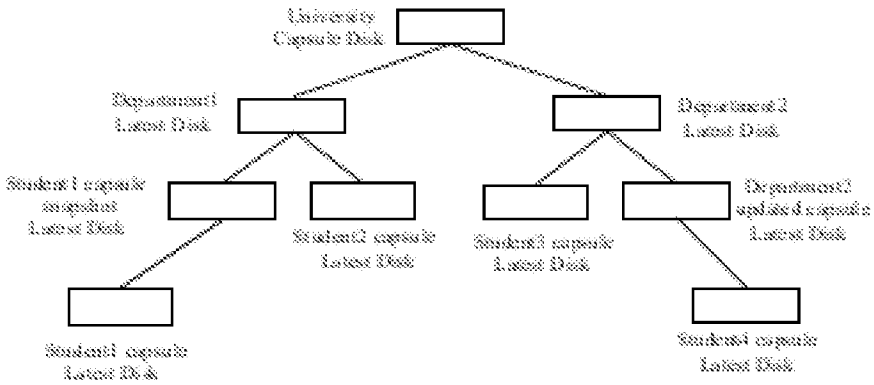
"We can store the disks in these capsules efficiently by creating a hierarchy, where **each child capsule could be viewed as inheriting from the parent capsule with the differences in disk state between parent and child captured in a separate copy-on-write (COW)** virtual disk." *Sapuntzakis,* 6.

"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." *Sapuntzakis,* 6-7. "Since **the latest COW disk is always local, all writes are local**." *Sapuntzakis,* 7. A POSA would have understood that, because writes to the local (latest) COW disk occur only in these circumstances, the local COW disk will include only the additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, but the local COW disk will not include blocks of the root image that are unchanged by respective compute nodes. Decl. ¶316. This is the standard operation of CoW snapshots like *Sapuntzakis'* local COW. *See, e.g.,* Decl. at Section VII.E (describing how CoWs are used in caching technologies). "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." *Sapuntzakis,* 7. "Reads involve searching the latest COW disk [the leaf image] and its ancestor disks in turn until the required block is found." *Sapuntzakis,* 6-7.  Figure 2: An example capsule disk. "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**." *Sapuntzakis,* 7. A POSA would have understood this to mean that the COW disk hierarchy is also invisible to the client, which perceives, instead, a single cohesive storage image corresponding to the '844 patent's merged root and leaf blocks. Decl. ¶382.
[23.2] index said root image portion; and	As explained for element [14.3.d], *Sapuntzakis* and *Federwisch* further disclose *indexing said root image* and providing the indexing results to another of said compute nodes. This encompasses, and thus discloses, the *indexing of said root image portion* recited in this element. Therefore, *Sapuntzakis* and *Federwisch* disclose this element for the reasons presented for element [14.3.d].

[23.3] provide the results of said indexing to a second compute node,	As explained for element [14.3.d], *Sapuntzakis* and *Federwisch* further disclose indexing said root image and *providing the indexing results to another of said compute nodes*. This encompasses, and thus discloses, *providing the results of said indexing to* another, e.g., *a second, compute node*. Therefore, *Sapuntzakis* and *Federwisch* disclose this providing the results of said indexing to a second compute node for the reasons presented for element [14.3.d].
[23.4] wherein said logic encoded in the one or more tangible media[17] comprise computer executable instructions executed by the first compute node.	*Sapuntzakis* discloses *logic encoded in one or more tangible media* (software on one or more of the computers and storage in its collective system, such as its x86 computers that run the software it uses). *Sapuntzakis* discloses that *the logic comprises computer executable instructions* at least because it uses computers like x86 computers (e.g., *the first compute node*) to *execute* the *instructions* (e.g., run the software). *Sapuntzakis* discloses that *the instructions are executed by the first computer node* (e.g., the writes and reads to local COWs are governed by instructions being executed by that local computer). "**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." *Sapuntzakis,* 3. "We have chosen to move x86 computer states because **x86 computers** are common, cheap, **run the software we use**, and have tools for migration." *Sapuntzakis,* 2. "At the root of the hierarchy is a *root disk*, which is a complete capsule disk." *Sapuntzakis,* 6.

[17] Requester notes that there is no antecedent basis for "said logic" or "the one or more tangible media." For purposes of this Request, the broadest reasonable interpretation of this element includes the interpretation that this element is intended to confirm that the "instructions stored" on "A computer-readable storage medium" as recited in preamble is in the form of logic or instructions that have been encoded in on or more tangible media forms, and—as the rest of this element recites—that form comprises "computer executable instructions executed by the first computer node."

	"Figure 1 shows an example of a capsule hierarchy illustrating how it may be used in a university. The root capsule contains all the software available to all students." *Sapuntzakis,* 6.  Figure 1: An example capsule hierarchy. "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." *Sapuntzakis,* 6-7. "Since the latest COW disk is always local, all writes are local." *Sapuntzakis,* 7.
Claim 24	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[24.0] The computer-readable storage medium of claim 23, wherein the instructions further cause the at least one computing device to store said results of said indexing on a shared storage unit accessible by said second compute node.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 23 for the reasons presented above for claim 23. *Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, *the instructions further cause the at least one computing device to store said results of said indexing on a shared storage unit* for the reasons presented for claim [20.0]. *Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, that *the shared storage unit is accessible by said second computer node* (e.g., "to another of said compute nodes") for the reasons presented for element [14.3.d].

Claim 25	SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)
[25.0] The computer-readable storage medium of claim 23, wherein the instructions further cause the at least one computing device to index said leaf image portion.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 23 for the reasons presented above for claim 23. *Sapuntzakis* and *Federwisch* further disclose *the instructions further causing the at least one computing device to index said leaf image portion.* *Sapuntzakis* discloses root and leaf images represented at the block level, wherein the leaf images store changed blocks of the root image as described in the mapping of *Sapuntzakis* to elements [7.1] and [7.2] of SNQ1. *See* Section V.A.5, elements [7.2]-[7.3]; *see also* Section V.B.5, element [14.2]. *Federwisch* discloses an indexing system operating at the block level capable of identifying and indexing changed blocks. Decl. ¶480-82, *Federwisch* [0071], [0072]. That is, each snapshot (corresponding to the '844 patent's "leaf image") has its own index of file blocks. Decl. ¶483; *Federwisch,* [0109]. Thus, the combination of these disclosures results in *the at least one computing device indexing of leaf image portions* with *Federwisch's* indexing system that identifies and indexes changed blocks, i.e., the leaf images that store the changed blocks of the root image as *Sapuntzakis* discloses. *See also* Decl. ¶483. A POSA would have been motivated to combine *Sapuntzakis* with *Federwisch* for the reasons presented in Section V.B.4. Additionally, a POSA would have known that indexing is an old and common technique. Decl. ¶476. A POSA would have understood that it is desirable to index only changed portions of target rather than re-indexing the entire target each time. Decl. ¶476. That is, a POSA would be motivated by the known benefit of a more efficient indexing of only the changed portions to implement *Federwisch's* indexing to the storage of *Sapuntzakis'* changed blocks. MPEP § 2143(I)(Rationale G). A POSA would have had a reasonable expectation of success at least because indexing was a well-known technique and storage and file systems include timestamps that denote when the files/data were last modified (e.g., the system was already set up in such a way to make indexing an easy process). Decl. ¶476; Decl. at Sections VII.B and VII.F. ***<u>See, e.g., Sapuntzakis</u>:*** "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." *Sapuntzakis,* 6. 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." *Sapuntzakis,* 6-7. "Since the latest COW disk is always local, all writes are local." *Sapuntzakis,* 7. A POSA would have understood that, because writes to the local (latest) COW disk occur only in these circumstances, the local COW disk will include only the additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of the root image, but the local COW disk will not include blocks of the root image that are unchanged by respective compute nodes. Decl. ¶316. This is the standard operation of CoW snapshots like *Sapuntzakis'* local COW. *See, e.g.,* Decl. at Section VII.E (describing how CoWs are used in caching technologies). "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." *Sapuntzakis,* 7. "Reads, 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." *Sapuntzakis,* 7. "First, the memory in most systems caches disk blocks." *Sapuntzakis,* 8. ***<u>See, e.g., Federwisch:</u>*** "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.**" *Federwisch,* [0071]. "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." *Federwisch,* [0072].

	"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)." *Federwisch*, [0109].
Claim 26	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[26.0] The computer-readable storage medium of claim 25, 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.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 25 for the reasons presented above for claim 25. *Federwisch* further disclose *the instructions further causing the at least one computing device to re-index said file system by re-indexing said leaf image portion* (*Federwisch's* re-indexing of incremental snapshots) *and merging the results of said re-indexing of said leaf image portion with said results of said indexing of said root image portion* (*Federwisch's* updating of the destination snapshot's index maps). Thus, the combination of these disclosures results in the *re-indexing of leaf image portions and merging of the results of the re-indexing with the results of indexing of said root portion* as *Federwisch* discloses in the system that *Sapuntzakis* discloses. *See also* Decl. ¶¶484-90. A POSA would have been motivated to combine *Sapuntzakis* with *Federwisch* for the reasons presented in Section V.B.4. Additionally, a POSA would have known that indexing is an old and common technique. Decl. ¶485. A POSA would have understood that when some data changes on a disk of 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. Decl. ¶485. That is, a POSA would be motivated by the known benefit of a more efficient indexing of only the changed portions to implement *Federwisch's* indexing to the storage of *Sapuntzakis'* changed blocks. MPEP § 2143(I)(Rationale G). Moreover, when some data changes, for example in any form of a simple database (such as an index), a POSA would have understood that the new index data would be easily merged with older data to, for instances, purge the now-stale index entries. Decl. ¶485; *see also* Decl. at Sections VII.B (review of data storage and file system technologies) and VII.F (review of indexing technologies). That is, a POSA would be motivated by the known benefit of purging old and now-incorrect index entries to merge the results of the re-indexing of the leaf image portion with the indexing of the root portion. A POSA would have had a reasonable expectation of success at least because indexing was a well-known technique and it was well within a

	POSA's ability to re-index and then merge the new index data with the older data. Decl. ¶485; Decl. at Sections VII.B (review of data storage and file system technologies) and VII.F (review of indexing technologies). ***Federwisch:*** "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)."** *Federwisch*, [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. Decl. ¶490.
Claim 27	**SNQ2: *Sapuntzakis* (EX1005) and *Federwisch* (EX1008)**
[27.0] The computer-readable storage medium of claim 23 wherein said root image portion is read-only.	*Sapuntzakis* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 23 for the reasons presented above for claim 23. *Sapuntzakis* further discloses the remainder of the recitations of claim 21 for the reasons presented for the mapping of *Sapuntzakis* to element [13.0] in Section V.A.5.

C. SNQ 3: *Birse* in view of *Holzmann* Presents a Significant New Question of Patentability for Challenged Claims 1-13

1. Overview of SNQ1

Claims 1-13 are obvious as explained herein, accompanied by the claim chart provided in Section V.C.5 and the Zadok Declaration (Decl. ¶492-565), are rendered obvious by *Birse* in view of *Holzmann*. Notably, *Birse* discloses the claimed root/leaf architecture, including where leaf images only contain additional data blocks not previously contained in the root image and changes made by respective compute nodes to the blocks of the root image. *Holzmann* discloses any additional caching limitations to the extent, if any, not already taught by *Birse* and/or obvious to a POSA.

2. Overview of Birse

Birse relates to the provision of an OS in a net-booted environment. EX1006, 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. EX1006, 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. EX1006, 2:65-3:3.

The NC system includes an NC client 150 (*e.g.*, personal computer) coupled to an NC server 170 via data communication line 140. EX1006, 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). EX1006, 5:9-29, 6:43- 48.

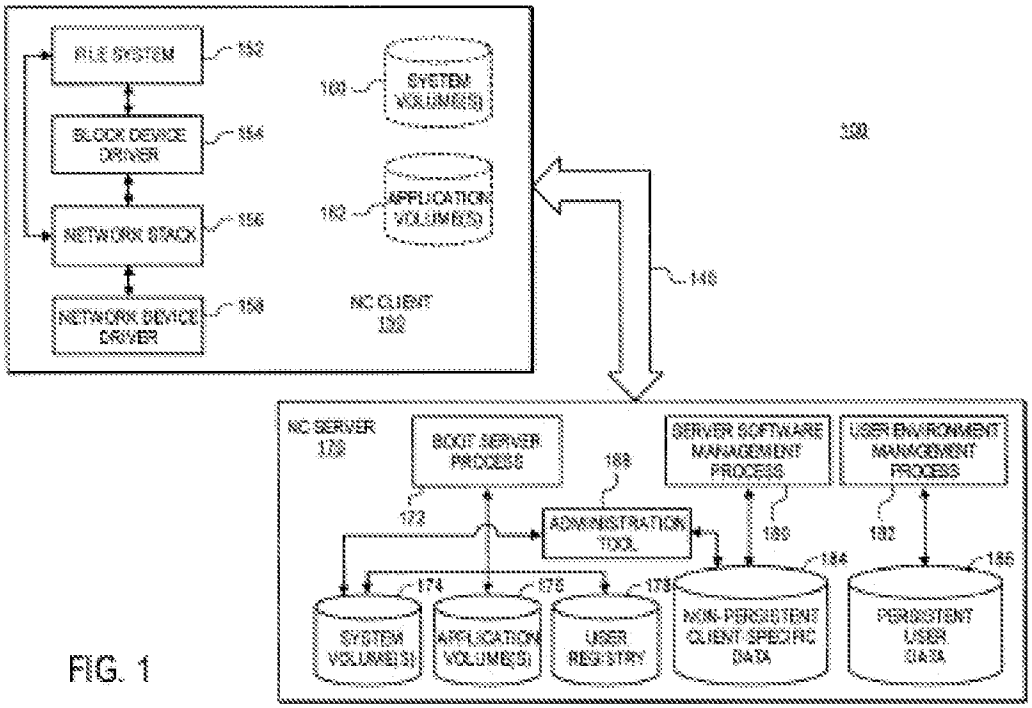


FIG. 1

***Birse*, Figure 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). EX1006, 5:46-6:34.

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. EX1006, 7:6-60.

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. Id. 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. EX1006, 7:41-46. *See also*, Decl. ¶274-305.

Birse is analogous to the '844 patent. *Birse* is in the same field of endeavor as the '844 patent, i.e., the management of whole OS images being shared by users on a network. *Compare, e.g.,* EX1001, 1:31-2:51 ("Clustered computing involves multiple computer nodes…that work together to achieve a common task"; "A common example of this is Operating System (OS) software image management, where the compute nodes are activated with the distributed application environment by either diskless booting protocols or remote software installation to local storage"; "Thus, embodiments of the present disclosure provide an operating system-independent system and method for distributing an application environment to a computer node…allow[ing] creation of boot images on the fly without significantly diminishing bring-up time";) *with* EX1006, 12:42-47 ("The NC shared data folder 640 includes a shared operating system image 660, a boot image 670, and a clients folder 680. 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."), 12:42-47 (explaining *Birse* is concerned with "providing a reliable and maintainable operating system in a net-booted environment").

Birse is also reasonably pertinent to a particular problem with which the inventors were concerned, such as efficient storage/transfer of OS images. *Compare* EX1001, Title, 2:54-67

(describing issues with known systems either "a large bring-up time," "wasted disk space," and a "cumbersome" process), 2:36-45 (addressing the known issues with its disclosed "technology for, among other things, root image caching and indexing for block-level distributed application management…"); *with* EX1006, 15:41-45 (noting the "[t]he main motivation for banding is to reduce disk space requirements of the NC client shadow system images. According to one embodiment, the reduction is achieved by only storing those bands which have actually been written to" to improve OS management efficiency).

3. Overview of Holzmann

An overview of *Holzmann* is provided in Section V.B.2.

4. Combination of *Birse* and *Holzmann*

The proposed combination of *Birse* with *Holzmann* is implementing *Holzmann's* details for how to cache data for the operating systems of *Birse*.

Under *KSR* and as detailed in the Zadok Declaration (EX1003), a POSA would have been motivated to combine *Birse* and *Holzmann*.

While *Birse* discloses the presence of cache as a feature on some embodiments of the system's server and client computers, it lacks detail how caching may be applied to its boot images. That is, *Birse* mentions common caching features that any operating system would use. Decl. ¶518, Decl. at Section VII.D; EX1006, 8:17-20 ("The processor may also be coupled to other types of storage areas/memories (e.g., cache, Flash memory, disk, etc.), which could be considered as part of the memory 254 or separate from the memory 254."), 8:39-41 ("Macintosh and Intel systems often have multiple busses, such as a peripheral bus, a dedicated cache bus, etc."). But *Birse* does not provide implementation details on these well-known caching features. Decl. ¶519. A POSA would have understood that *Birse's* clients and servers are regular computers that use typical caching. Decl. ¶519, Decl. at Section VII.D. To the extent a POSA required implementation details, a POSA would have turned to *Holtzmann*, which provides details of caches for each system/server being booted, as well as different cache configurations. Decl. ¶520; MPEP § 2143(I)(Rationale G) (so motivated by *Birse's* disclosure of caching, a POSA would have turned to *Holzmann* for implementation details). *Holzmann* suggests various forms of caches (EX1007, [0034], [0038]) and provides specific examples where each server or operating system has its own

dedicated cache (EX1007, [0021], Fig. 5). Decl. ¶¶ 521-523. Thus, a POSA would have been motivated to modify *Birse* with *Holzmann*'s caching techniques including because, while *Birse* discloses typical caching, *Holzmann* offers implementation details of how to cache data for the OS distribution purpose (*e.g.*, *Birse*'s boot images). Decl. ¶¶525-26.

A POSA would have considered the combination of *Birse* with *Holzmann* in particular because of the overlap in their disclosures. Both are concerned with efficient management of whole OS images and booting systems over a network. *See, e.g.,* Decl. ¶¶ 511-516; EX1006, 12:42-47, 1:55-57, 15:41-45; EX1007, Abstract, [0004], [0008]. Moreover, while a POSA would have known how to implement many different caching techniques, each with their own pros and cons (Decl. ¶526, Decl. at Section VII.D), Holzmann offers details of how to cache data for each operating system (e.g., *Birse's* boot images), and *Holzmann* goes further and discloses several forms of caches: unique caches per server, caches that are statically or dynamically allocated, non-associative caches, set-associative caches, and fully associative ones. Decl. ¶526.

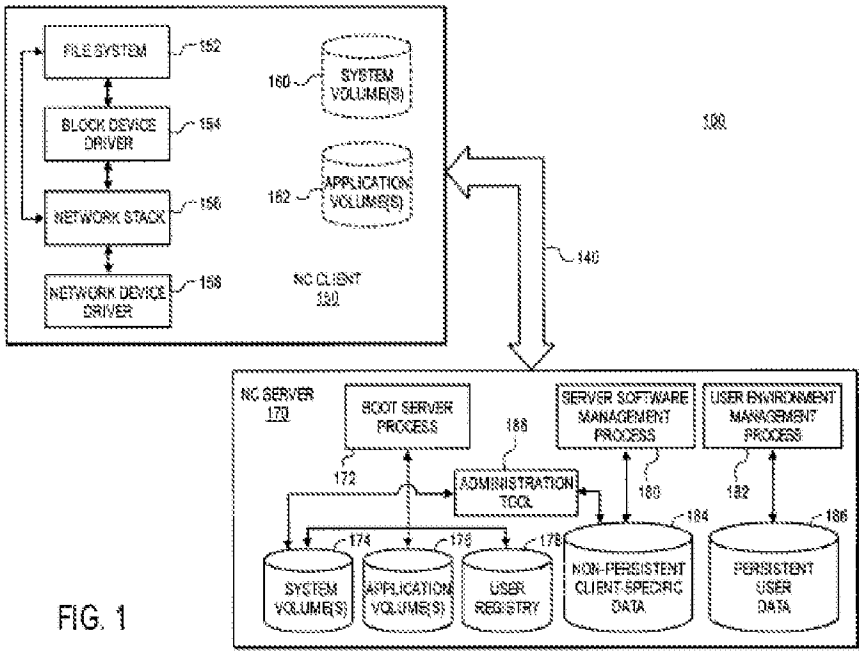
Indeed, *Holzmann* itself provides a motivation for the particular implementation of using separate dedicated caches for each server or operating system—"[b]y allocating separate cache space for each server or other network resource, the inventive system prevents corruption of data, overwriting of data, and other harmful actions that would lead to resource malfunction." EX1007, [0033]; MPEP § 2143(I)(Rationale G). Thus, using the known technique of *Holzmann's* separate dedicated caches to improve the similar system of *Birse* in the same way (e.g., to prevent corruption, overwriting, and other resource malfunctions). MPEP § 2143(I)(Rationale C).

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. Decl. ¶527, Decl. at Sections VII.D and VII.A. Modifying *Birse* with *Holzmann's* caching teachings would have been simple to a POSA, and would not have required extensive experimentation or imposed undue burden – especially when both disclose caching. Decl. ¶528. The combination is merely the application of a known techniques of *Holtzmann's* caching technique to the known system of *Birse* that was ready for such improvements as demonstrated by its broad disclosure of using caching, yielding the predictable result of *Birse's* system employing *Holtzmann's* particular caching technique. MPEP § 2143(I)(D).

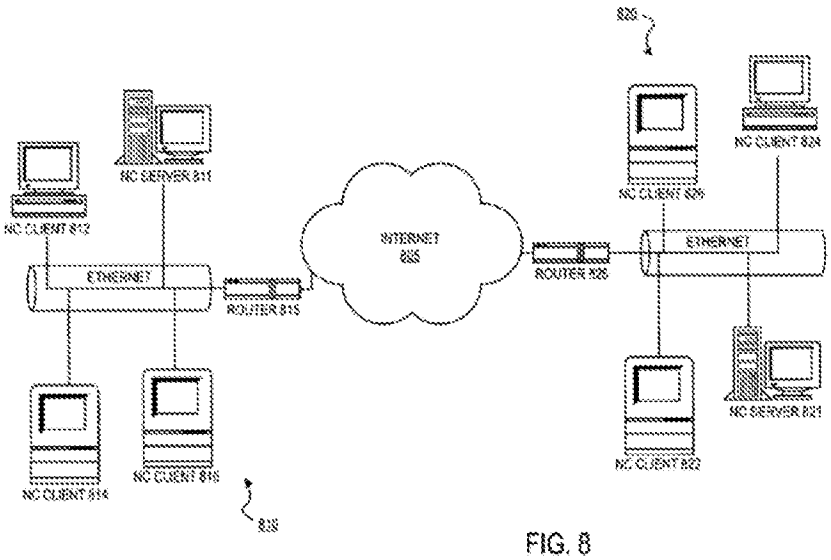
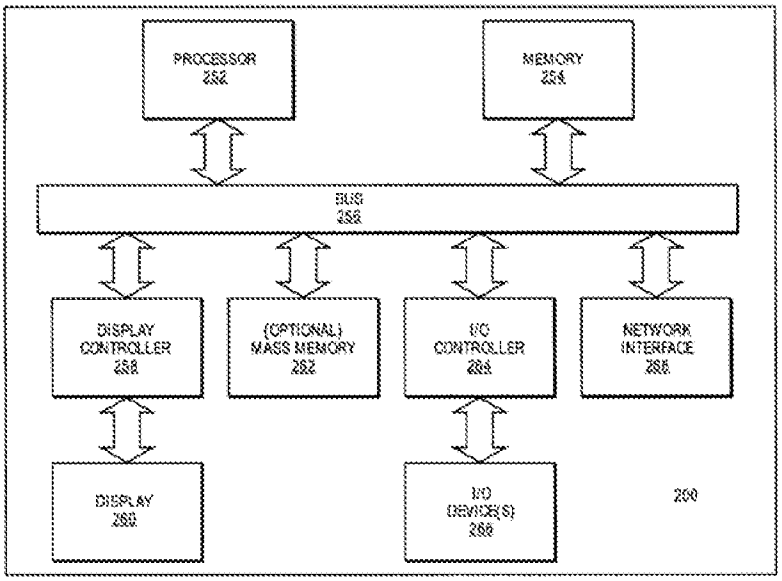
For the same reasons, the combination would have yielded predictable results (as caching is well-understood to improve performance and resource utilization). A POSA would also have a

high expectation of success, as *Holzmann* merely describes caching in greater detail than *Birse*. Decl. ¶510-28.

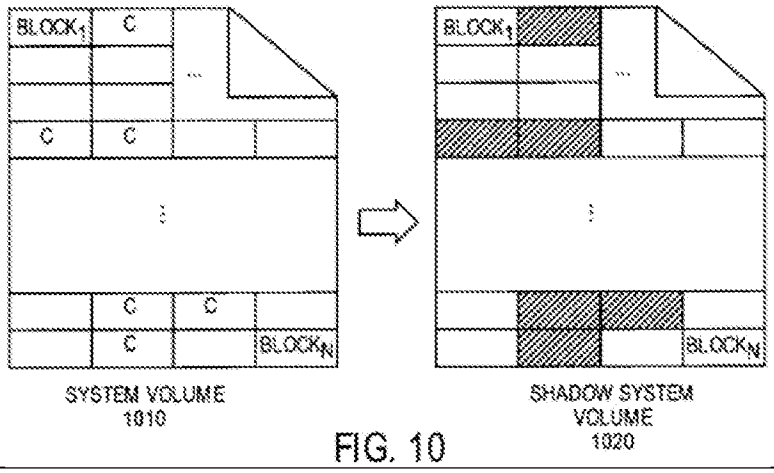
5. Claim Chart Providing Detailed Analysis[18]

Claim 7	SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)
[7.0] A method for providing data to a plurality of compute nodes, comprising:	To the extent the preamble is limiting, *Birse* discloses *a method for providing data* (from NC server 170) *to a plurality of compute nodes* (NC clients 150, 812, 814, 816, 822, 824, 826): ***See, e.g., Birse:*** "...**[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." EX1006, 4:56-60. *See also* 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications).  FIG. 1 "FIG. 8 is a simplified block diagram illustrating two NC systems 810 and 820 coupled via the Internet 805." *Birse,* 14:8-

[18] This claim chart mapping is further supported with an analysis in prose form provided by Dr. Zadok in his declaration. Decl. at Section XII.C.

	9.  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. *See* 4:27-57.  FIG. 2
[7.1] storing blocks of a root image of said compute nodes on a first storage unit;	*Birse* discloses *storing blocks of a root image* (system volumes 174, 660, 922, 924) *of said compute nodes* (multiple NC clients 150*) on a first storage unit* (one or more hard drives 610 of NC server 170, logically separate in NC folders/directories 630, 640): ***<u>See, e.g., Birse</u>:*** "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." *Birse,* 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." *Birse,* 14:27-34, 41-44. "[T]he system volumes 174...may be independently distributed across multiple server computer systems." *Birse,* 6:59-63. "FIG. 6 [below] 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**…. The NC shared data folder 640 **includes a shared operating system image 660**, a boot image 670, and a clients folder 680." *Birse,* 12:30-44. *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." *Birse,* 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:

	 FIG. 10
[7.2] storing leaf images for respective compute nodes 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 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; and	*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): ***<u>See, e.g., Birse</u>:*** "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." *Birse,* 12:54–59 (discussing FIG. 6, in [7.1] above). "**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." *Birse,* 7:35-53; see *also* 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." *Birse,* 10:63-65, 11:6-17.

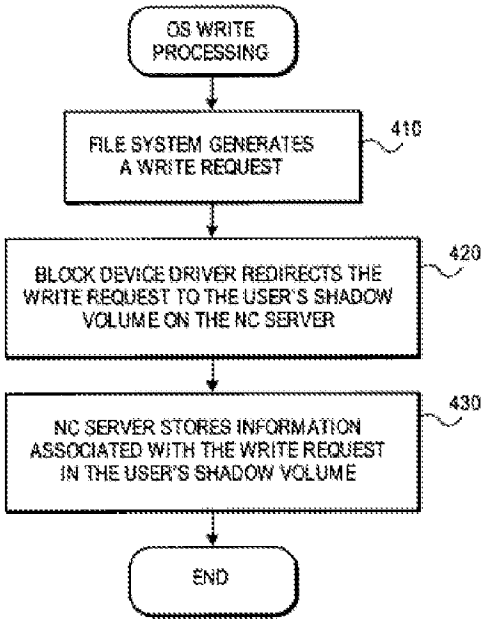
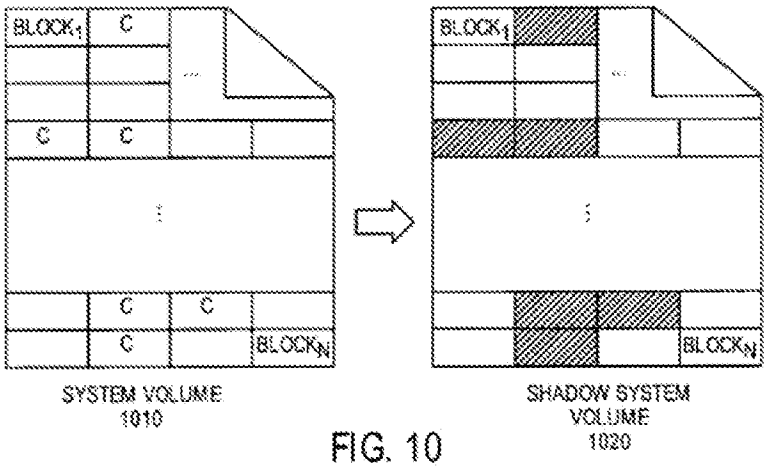


FIG. 4

"[In one embodiment,] **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." *Birse,* 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." *Birse,* 14:49-53. "...[T]he 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**." *Birse,* 14:56-61.
[7.3] caching blocks of said root image that have been accessed by at least one of said compute nodes in a cache memory.	*Birse* and *Holzmann* disclose *caching blocks of said root image* (*Birse's* system volumes 174, 660, 922, 924) *that have been accessed by at least one of said compute nodes* (*Birse's* NC clients 150, 812, 814, 816, 822, 824, 826) *in a cache memory* (*Holzmann's* memory module 104): ***<u>See, e.g., Birse</u>:*** *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." *Birse,* 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:  FIG. 10 *See also* mapping of *Birse* to element [7.2] above. ***<u>See, e.g., Holzmann</u>*:** "...[T]he invention [of 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." *Holzmann,* [0002].

A POSA would have understood the servers of *Holzmann* correspond to the compute nodes of the '844 patent. Decl. ¶322.

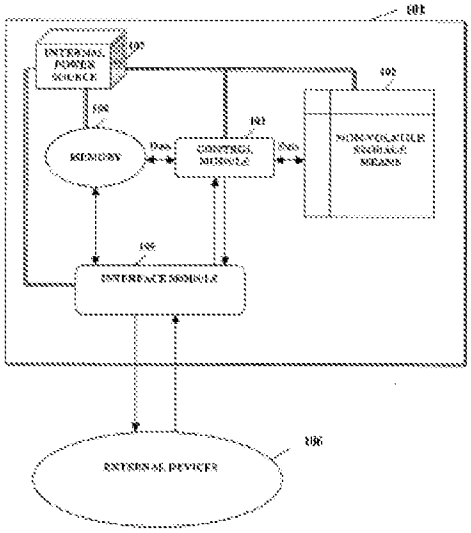


Fig. 1

"FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." *Holzmann,* [0017].

"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**." *Holzmann,* [0020].

A POSA would have recognized this description as describing a cache memory at least because RAM is often used as fast volatile cache and SSD storage devices are used as relatively fast non-volatile caches. Decl. ¶325; *see also* Decl. at Section VII.D (providing an overview of caching technologies).

"**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..." *Holzmann,* [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**…" *Holzmann*, [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." *Holzmann*., [0033]. A POSA would have understood that, when the cache of *Holzmann* is used to cache blocks of a read-only shared OS image, there is no need to create separate caches for each client because the read-only image is not subject to corruption. Decl. ¶329. That is, read-only data can easily be shared with multiple users and they will all see the same consistent data. Decl. ¶329. "If 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." *Holzmann,* [0034-35]. A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Birse* for the reasons presented in Section V.C.4. *See also* Decl. ¶¶507-09; Decl. at Section XII.C.2.
Claim 8	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[8.0] The method as recited in claim 7 further comprising:	*Birse* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 7 for the reasons presented above for claim 7.

[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; and	*Holzmann* discloses *receiving a read request* (request for operating system data received by control module 103) *from at least one of said compute nodes, wherein a first portion of the data requested is currently stored in said cache memory* (if the data block has been loaded into the cache…[it] is provided from the cache): ***See, e.g., Holzmann*:** "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..." *Holzmann,* [0021]. "[T]he **control module 103** monitors the request for operating system data, and determines whether the…data block requested…has previously been loaded into the operating system cache. **If the data block has been loaded into the cache, and…has not been overwritten….[it] is provided from the cache…**" *Holzmann* [0024]. *See also* Decl. ¶352 (noting that "this limitation describes nothing more than common caching behavior (finding if something is cached in response to a read/write request)"); Decl. at Section VII.D (providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Birse* for the reasons presented in Section V.C.4. *See also* Decl. ¶¶529-32, Decl. at Section XII.C.2.
[8.2] providing said first portion of said data to said at least one of said compute nodes from said cache memory.	*Holzmann* discloses *providing said first portion of said data* (data block of the operating system data that has been loaded into the cache) *to said at least one of said compute nodes from said cache memory*. ***See, e.g., Holzmann*:** "[T]he control module 103 monitors the request for operating system data, and determines whether the…data block requested…has previously been loaded into the operating system cache. **If the data block has been loaded into the cache**, and…has not been overwritten**….[it] is provided from the cache**…" *Holzmann* [0024]. *See also* Decl. ¶355 (noting that "this limitation describes nothing more than common caching behavior (returning a cached item if found)"); Decl. at Section VII.D (providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation

	in applying the implementation details of *Holzmann* to the caching of *Birse* for the reasons presented in Section V.C.4. *See also* Decl. ¶¶533-35, Decl. at Section XII.C.2.
Claim 9	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[9.0] The method as recited in claim 8 further comprising: updating said cache memory based on said read request.	*Birse* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 8 for the reasons presented above for claim 8. *Holzmann* further discloses *updating said cache memory based on said read request* (requested data is loaded into the cache if not already there). ***See, e.g., Holzmann*:** "**If the requested data has not been loaded into the cache 501, the control module 103 copies the data from… memory portion 400 to the cache 501**." *Holzmann*, [0035]. *See also,* Decl. ¶360-63 (noting that "this limitation describes nothing more than common caching behavior (updating a cache based on reading a new item, called a "cache admission" policy")"); Decl. at Section VII.D (providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Birse* for the reasons presented in Section V.C.4. *See also* Decl. ¶536, Decl. at Section XII.C.2.
Claim 10	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[10.0] The method as recited in claim 9	*Birse* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 9 for the reasons presented above for claim 9.
[10.1] 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; and	*Holzmann* 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* (requested data is loaded into the cache if not already there) ***See, e.g., Holzmann*:** "[T]he **control module 103** monitors the request for operating system data, and **determines whether the…data block requested…has previously been loaded into the** operating system **cache**. If the data block has been loaded into the cache, and…has not been overwritten….[it] is provided from the cache…" *Holzmann* [0024]. "**If the requested data has not been loaded into the cache 501, the control module 103 copies the data from… memory portion 400 to the cache 501**." *Holzmann*, [0035]. *See also,* Decl. ¶368 (noting that "this limitation describes nothing more than common caching behavior (updating a cache based on reading a new item, called a "cache admission" policy")"); Decl. at Section VII.D

	(providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Birse* for the reasons presented in Section V.C.4. *See also* Decl. ¶538, Decl. at Section XII.C.2.
[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.	*Holzmann* discloses, or at least suggests, *removing the least recently accessed data from said cache memory if the amount of data in said cache memory is above a threshold value* with its recognition that those skilled in the art would be aware of various cache configurations. ***See, e.g., Holzmann*:** "It will be understood by those skilled in the art that still more cache configurations can be used in accordance with the invented system. *Holzmann* [0038]. *See also,* Decl. ¶372 (noting that "this limitation describes nothing more than common caching behavior (freeing some items from a cache to make room for new items")"); Decl. at Section VII.D (providing an overview of caching technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Birse* for the reasons presented in Section V.C.4. *See also* Decl. ¶539, Decl. at Section XII.C.2.
Claim 11	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[11.0] The method as recited in claim 7 further comprising: merging the blocks of said root image with the blocks of respective leaf images to create cohesive respective application environments.	*Birse* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 7 for the reasons presented above for claim 7. *Birse* further 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): ***See, e.g., Birse*:** "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." *Birse,* 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].**

	Advantageously, 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." *Birse,* 9:2-10. "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. **In 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**." *Birse,* 11:39-41, 11:55-12:10.

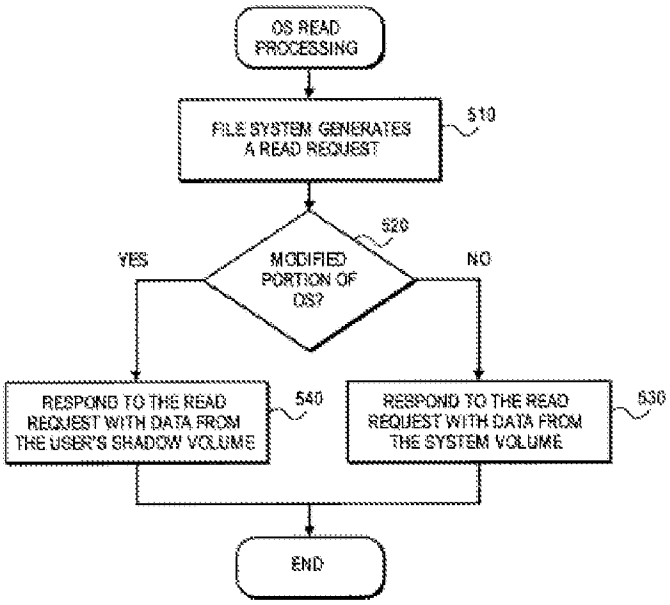


FIG. 5

Claim 12	SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)
[12.0] The method as recited in claim 11 wherein said merging occurs at an operational level between file systems of the respective compute nodes and said first storage unit, respective second storage units, and said cache memory.	*Birse* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 11 for the reasons presented above for claim 11, and, in particular, that the block device drivers 154 (which serve as union block devices) *work* below the client file system, and, hence, *at an operational level.* Thus, the disclosures provided for claim 11 show that *the merging occurs at an operational level* (operate below the client file system) *of the respective compute nodes* (multiple NC clients 150) *and said first storage unit* (NC server 170, e.g., with one or more hard drives 610), *and respective second storage units* (NC client folders 690 #1...n). Because the system of *Birse, e.g.,* including block drivers (154), 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 POSA to combine *Birse* with *Holzmann's* teachings regarding caching methods to extend the function of *Birse's* system, implementing the algorithm of Figure 5, to incorporate blocks stored in the *cache memory*, as well. Decl. ¶545. ***See, e.g., Birse*:** "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." *Birse,* 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]. Advantageously, 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." *Birse,* 9:2-10. "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.** In 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." *Birse,* 11:39-41, 11:55-12:10.

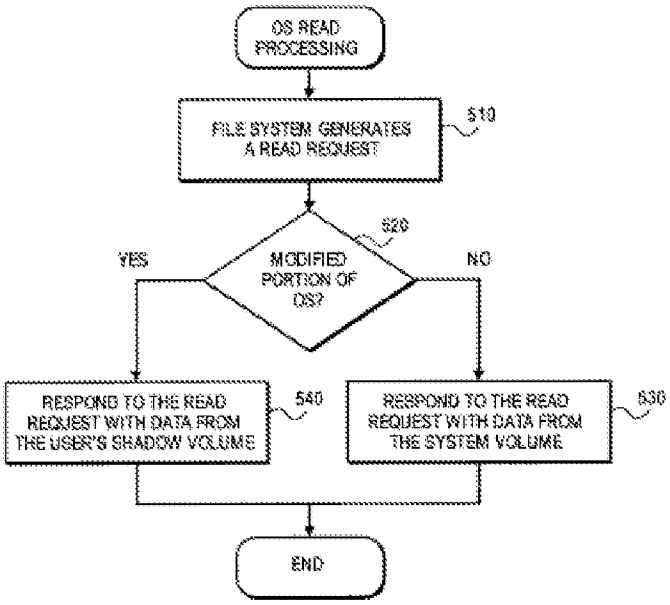


FIG. 5

***See, e.g., Holzmann*:**

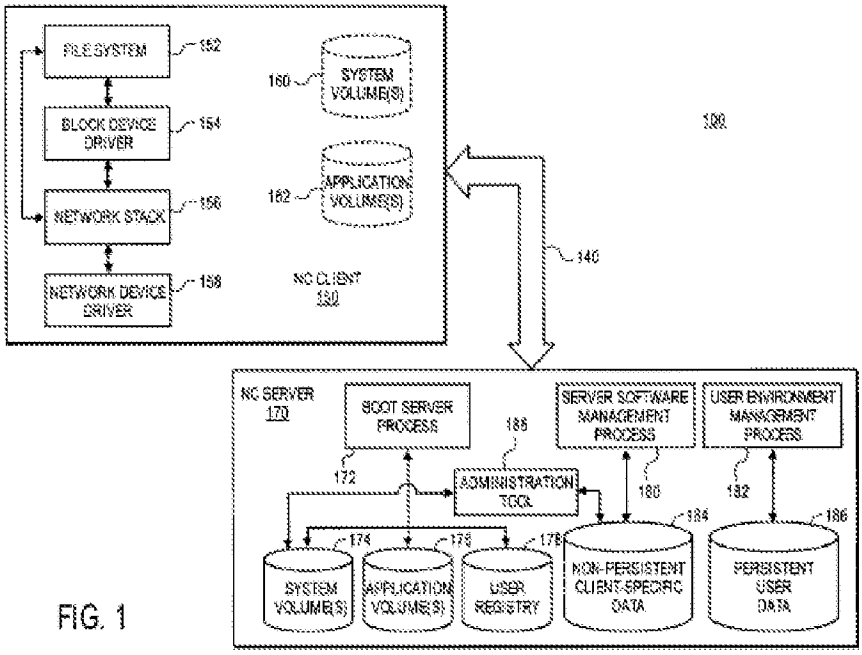
> "The **memory module 104 comprises at least one direct-access memory module for holding data currently or recently used by external devices 106**. The memory module 104 is more quickly accessible, and **performs read and write processing functions more quickly**, than non-volatile or disk memory, such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

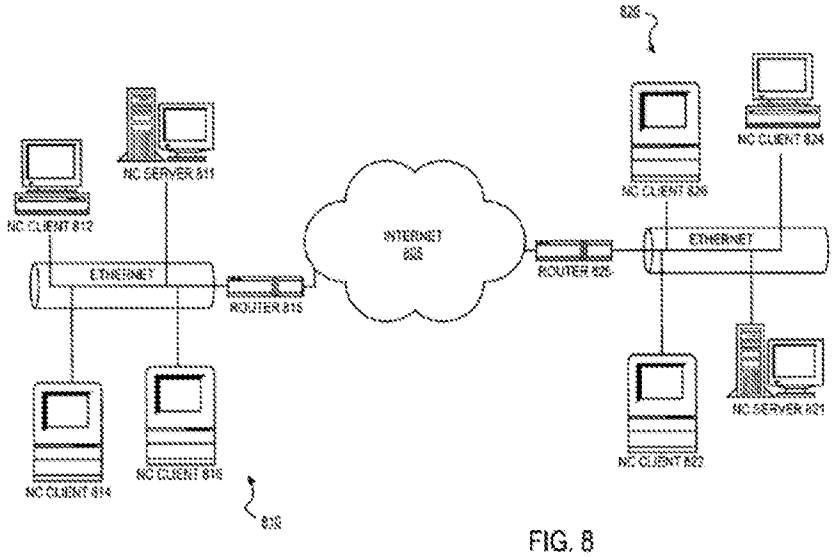
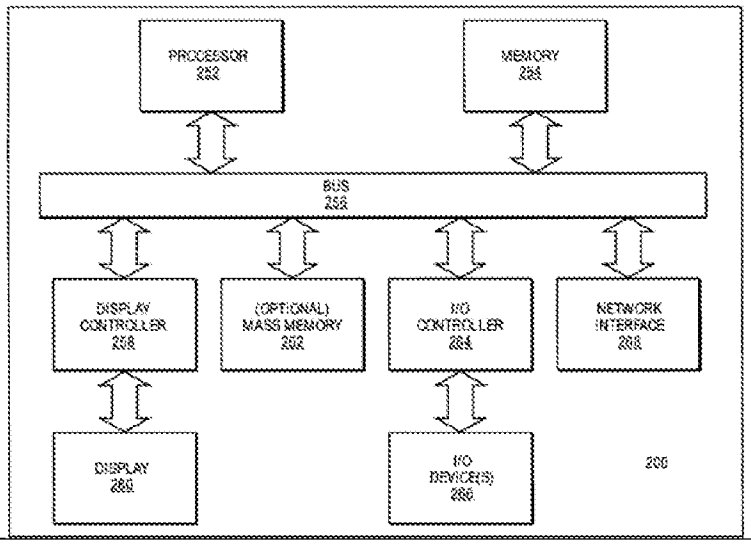
A POSA would have recognized this description as describing a cache memory at least because RAM is often used as fast volatile cache and SSD storage devices are used as relatively fast non-volatile caches. Decl. ¶325; *see also* Decl. at Section VII.D (providing an overview of caching technologies).

> "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..." *Holzmann,* [0021].

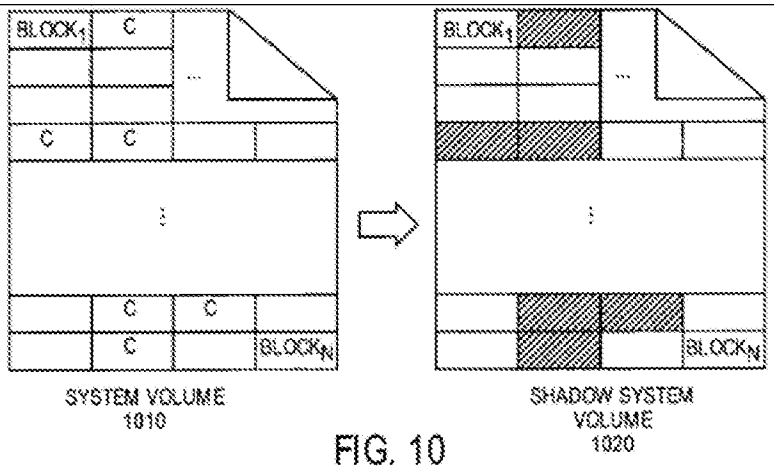
> "The external device 106 requests operating system data

	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…" *Holzmann*, [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." *Holzmann*, [0033]. A POSA would have understood that, when the cache of *Holzmann* is used to cache blocks of a read-only shared OS image, there is no need to create separate caches for each client because the read-only image is not subject to corruption. Decl. ¶329. That is, read-only data can easily be shared with multiple users and they will all see the same consistent data. Decl. ¶329. "If 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." *Holzmann,* [0034-35]. A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Birse*—and thus, the merging at the operational level between file systems of the respective compute nodes and said cache memory--for the reasons presented in Section V.C.4. *See also* Decl. ¶¶544-45; Decl. at Section XII.C.2.
Claim 13	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[13.0] The method as recited in claim 7 wherein said first storage unit is read-only.	*Birse* in view of *Holzmann* disclose, or at least suggest, the method as recited in claim 7 for the reasons presented above for claim 7. *Birse* further discloses *the first storage unit* (e.g., root image system volume 174 stored on hard drive 610 of NC server 170) *is read-only*. "The system volumes 174 include a protected, **read-only**, **master copy of the operating system software**." *Birse,* 5:13-14.

Claim 1	SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)
[1.0] A system for providing data to a plurality of compute nodes, the system comprising:	To the extent the preamble is limiting, *Birse* discloses *a system* (NC server 170 and NC clients 150 operating over Internet 805) *for providing data* (e.g., a stored copy of operating system or an application) *to a plurality of computer nodes* (NC clients 150, 812, 814, 816, 822, 824, 826). *See also, e.g.,* element [7.0]. ***See, e.g., Birse:*** **"...[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." EX1006, 4:56-60. *See also* 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications).  FIG. 1 "FIG. 8 is a simplified block diagram illustrating two NC systems 810 and 820 coupled via the Internet 805." *Birse,* 14:8-9.

	 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. *See* 4:27-57.  FIG. 2
[1.1] a first storage unit configured to store blocks of a root image of said compute nodes;	*Birse* discloses *a first storage unit* (one or more hard drives 610 of NC server 170, logically separate in NC folders/directories 630, 640) *configured to store blocks of a root image* (system volumes 174, 660, 922, 924) *of said compute nodes* (multiple NC clients 150). *See also, e.g.,* element [7.1]. ***<u>See, e.g., Birse</u>:*** "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**." *Birse,* 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." *Birse,* 14:27-34, 41-44. "[T]he system volumes 174...may be independently distributed across multiple server computer systems." *Birse,* 6:59-63. "FIG. 6 [below] 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**…. The NC shared data folder 640 **includes a shared operating system image 660**, a boot image 670, and a clients folder 680." *Birse,* 12:30-44. *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." *Birse,* 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:

	 FIG. 10
[1.2] a plurality of second storage units configured to store leaf images of respective compute nodes, 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 said 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; and	*Birse* discloses *a plurality of second storage units* (NC client folders 690 #1...n) *configured to store leaf images* (shadow volumes/images 930, 695) *of respective compute nodes* (NC clients 150, 812, 814, 816, 822, 824, 826), as described below. *Birse* discloses that *said leaf images include 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) for the reasons presented in element [7.2], presented above. *See also* Decl. at Section VII.E and ¶316. ***See, e.g., Birse:*** "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." *Birse*, 12:54–59 (discussing FIG. 6, in [7.1] above). "**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." *Birse,* 7:35-53; see *also* 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." *Birse,* 10:63-65, 11:6-17.

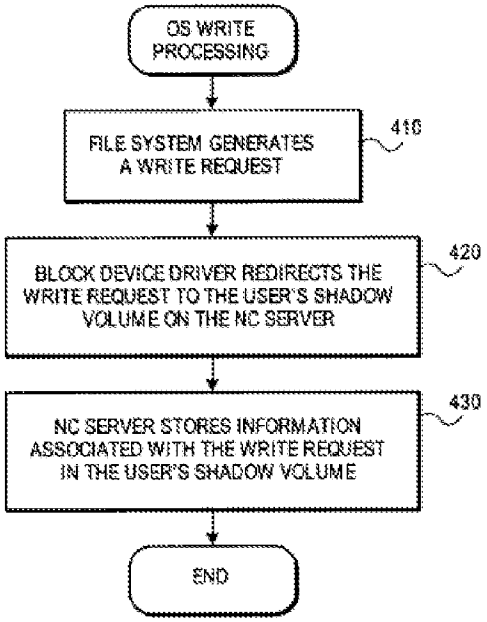
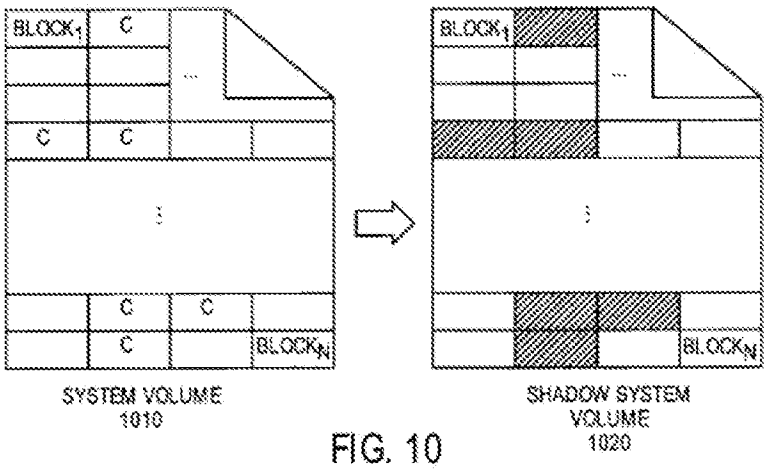


FIG. 4

"[In one embodiment,] **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." *Birse,* 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." *Birse,* 14:49-53. "...[T]he 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**." *Birse,* 14:56-61.
[1.3] a cache configured to cache blocks of said root image previously accessed by at least one of said compute nodes.	*Birse* and *Holzmann* discloses *a cache* (e.g., the cache operated in *Holzmann's* memory module 104) *configured to cache blocks of said root image* (*Birse's* system volumes 174, 660, 922, 924) previously accessed by at least one of said compute nodes (*Birse's* NC clients 150, 812, 814, 816, 822, 824, 826). *See also, e.g.,* element [7.3]. ***See, e.g., Birse*:** *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." *Birse,* 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:  FIG. 10

See also mapping of *Birse* to element [7.2] above.

***See, e.g., Holzmann*:**

> "...[T]he invention [of 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**." *Holzmann,* [0002].

A POSA would have understood the servers of *Holzmann* correspond to the compute nodes of the '844 patent. Decl. ¶322.

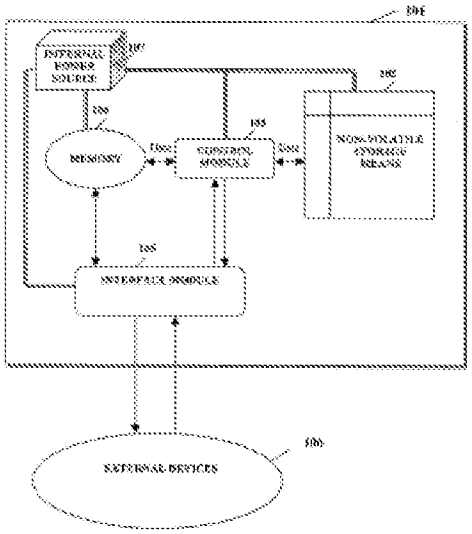


Fig. 1

> "FIG. 1 is a block diagram illustrating the component parts of the invented system and the functions of each component part. The invented system comprises a solid state disk system 101 having a storage means 102, a control module 103, a memory module 104, an interface module 105 that communicates with external devices 106, and an internal power source 107." *Holzmann,* [0017].

> "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**." *Holzmann*, [0020].

A POSA would have recognized this description as describing a cache memory at least because RAM is often used as fast volatile cache and SSD storage devices are used as relatively fast non-volatile caches.

	Decl. ¶325; *see also* Decl. at Section VII.D (providing an overview of caching technologies). "**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..." *Holzmann,* [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**…" *Holzmann,* [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." *Holzmann*, [0033]. A POSA would have understood that, when the cache of *Holzmann* is used to cache blocks of a read-only shared OS image, there is no need to create separate caches for each client because the read-only image is not subject to corruption. Decl. ¶329. That is, read-only data can easily be shared with multiple users and they will all see the same consistent data. Decl. ¶329. "If 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." *Holzmann,* [0034-35]. A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the caching of *Birse* for the reasons presented in Section V.C.4. *See also* Decl. ¶¶507-09; Decl. at Section XII.C.2.

Claim 2	SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)
[2.0] The system as recited in claim 1 wherein said cache is configured to store X most recently accessed blocks of said root image, and wherein X represents a cache threshold value.	*Birse* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Holzmann* further discloses, or at least suggests, *said cache being configured to store X most recently accessed blocks of said root image, and wherein X represents a cache threshold value*. A POSA would have understood this limitation describes nothing more than common caching behavior, i.e., standard cache management policies, where caches naturally have limited room and must use thresholds to decide what to keep/evict. Decl. ¶556, Decl. at Section VII.D (particularly the known LRU and LFU cache-management policies). ***See, e.g., Holzmann*:** "If the requested data has not been loaded into the cache 501, the control module 103 **copies the data from… memory portion 400 to the cache 501**." *Holzmann*, [0035]. "It will be understood by those skilled in the art that still more cache configurations can be used in accordance with the invented system. *Holzmann* [0038]. *See also,* Decl. ¶552-56 (noting that "this limitation describes nothing more than common caching behavior (freeing some items from a cache whose size is by definitions. Limited, to make room for new items)"); Decl. at Section VII.D (providing an overview of caching technologies, particularly LRU and LFU cache-management policies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the cache of *Birse* for the reasons presented in Section V.A.4.
Claim 3	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[3.0] The system as recited in claim 1 wherein said first storage unit, said second storage units, and said cache are contained within a single storage appliance.	*Birse* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Birse* and *Holzman* disclose the remainder of the elements of claim 3. *Birse* further discloses *said first storage unit* (holding the core system volumes and the user system volumes) *and said second storage units* (holding the shadow system volumes) *contained within a single storage appliance*. *Holzmann* discloses the use of various caches, including *a cache* mechanism for server drives. A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Holzmann* to the cache of

	Birse for the reasons presented in Section V.A.4. A POSA would have found it obvious to implement the cache system of *Holzmann* on the storage device holding the first and second units at least because it would have improved 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*). Decl. ¶561; MPEP § 2143(I)(Rationale (G)) (known advantages of such a configuration as applied to the example *Holzmann* provides the motivation). Moreover, a POSA would have known that the storage systems created by Netapp are all single storage appliances; each such appliance supports multiple volumes, multiple base images, multiple snapshots, and multiple caches (both volatile and non-volatile). Decl. ¶561. This would have provided a POSA a reasonable expectation of success in employing the cache of *Holzmann* as part of the single storage appliance employed for *Birse's* first and second storage units. It would have also been obvious to try such a configuration, choosing from a finite set of predictable solutions (1. All three storage components within the same appliance, 2. Two of the three storage components within the same appliance, and 3. All three storage components within different appliances) with a reasonable expectation of success given the state of such technologies at the time. *See, e.g.,* Decl. ¶407-08 and Decl. at Section VII.E. *See also* Decl. at Section VII.E (providing an overview of copy-on-write and snapshot technologies). ***See, e.g., Birse:*** "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. 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." *Birse,* 12:30–37

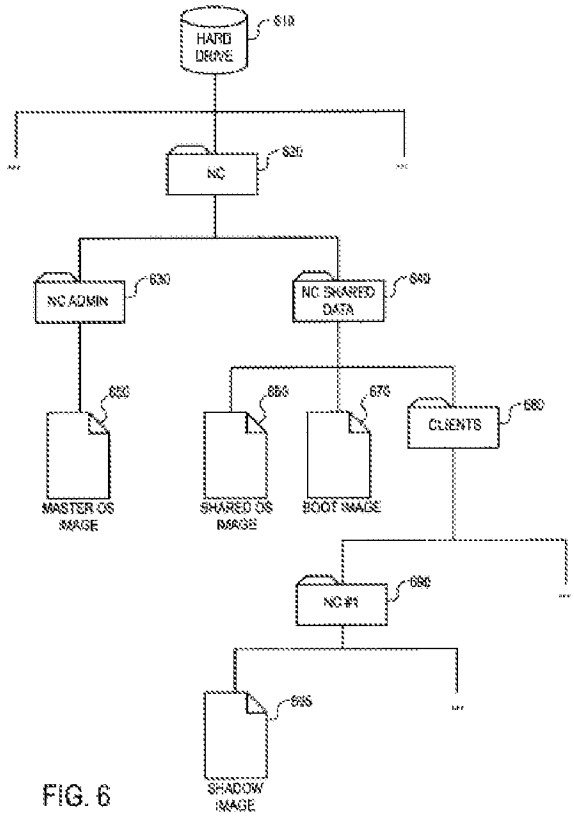


FIG. 6

See, e.g., Holzmann:

"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." EX1007, [0021].

Claim 4	SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)
[4.0] The system as recited in claim 1 further comprising: a plurality of union block devices configured to interface between respective compute nodes and said first	*Birse* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Birse* and *Holzmann,* particularly in view of POSA knowledge of well-known and off the shelf products for providing devices configured to interface as recited in this claim, disclose or at least render obvious this claim. As a preliminary matter, the term Union Block Device is informally

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.	defined in the '844 patent by way of a set of tasks it performs. *See, e.g.,* Section III.A.2. That is, any device or component that operates to perform the recited elements would constitute a UBD because a UBD is essentially described in terms of its capabilities. <u>First, *Birse* in view of *Holzmann* discloses</u> that its system *is configured to provide an interface between respective compute nodes* (*Birse's* NC clients 150, 812, 814, 816, 822, 824, 826) *and said first storage unit* (*Birse's* one or more hard drives 610 of NC server 170, logically separate in NC folders/directories 630, 640), *respective second storage units* (*Birse's* NC client folders 690 #1...n), *and said cache* (as taught in *Holzmann*) *to distribute application environments* (e.g., system volumes 174, 660, 922, 924) *to the compute nodes* (*Birse's* NC clients). *See, e.g.,* Section V.C.5, Claim 1, providing mappings for the components of *Birse's* system *interfacing* (communicating) with each other to provide complete data (system volumes) and/or needed new blocks of data (e.g., write requests for new data or changes to operating system data) to a plurality of computer nodes (*Birse's* NC clients). *Birse* also discloses that its system is *configured to create said application environments by 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). *See, e.g.,* Section V.B.5, Claims 11 and 12, providing mappings for the components of *Birse's* system *configured to create a cohesive application environment by merging the blocks of the root image with blocks of the leaf image using data* (claim 11) *from the compute nodes, the first storage unit, the second storage units and the cache memory* (claim 12). Thus, *Birse's* (with *Holzmann's* implementation details for caching) disclosures of devices capable of performing the elements of this claim disclose this claim in the same manner as the '844 patent discloses it, i.e., the devices that perform the recited functions are collectively known as the *union block devices*. Decl. ¶563. *See Birse* as mapped for claims 1, 11, and 12 above; *Holzmann* as mapped for element [1.3] above. <u>Second, to the extent more is required</u>, the '844 patent discloses an example that "In one embodiment, UBDs 230a-n 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. … Because UBDs 230a-n operate below the file system, they are concerned merely with the blocks of data themselves, rather than files they form." EX1001, 5:60-6:1. This is consistent with the basic understanding of how to implement copy-on-write and snapshot technologies, such as the engine/driver used in such off-the-shelf appliances as Netapp. *See, e.g.,* Decl. ¶108, ¶563, and Decl. at Sections

	VII.C. and VII.E. A POSA would have understood that *Birse's* system (in view of *Holzmann's* caching disclosures) would be configured with the necessary components to provide the features and functions that *Birse* discloses. That is, the basic and well-known information as to how to implement copy-on-write and snapshot technologies is not explicitly disclosed in *Birse* because such well-known POSA knowledge would not need to be explicitly stated. Decl. ¶410; *see also* Decl. at Sections VIII.C (describing how the '844 patent discloses UBDs) and VII.E (providing an overview of copy-on-write and snapshot technologies that employed devices that constitute the claimed UBDs). A POSA would have been well aware of storage devices capable of performing the disclosed functions, such as the devices created by Netapp, which are single storage appliances, each supporting multiple volumes, multiple base images, multiple snapshots, and multiple caches (both volatile and non-volatile) and employing a CoW engine/driver that resides between the file system layer and the actual storage devices. *See, e.g.,* Decl. ¶410 and Decl. at Section VII.E. That is, the claimed UBDs are merely off-the-shelf devices that any POSA would employ when configuring *Birse's* system (in view of *Holzmann's* caching disclosures). A POSA would have understood each volume in a Netapp or similar appliance to be employed by the system of *Birse's* to provide the functions disclosed by *Birse*, thereby providing the claimed UBDs. *See, e.g.,* Decl. ¶410, ¶563, and Decl. at Section VII.E. In other words, the "low level drivers" disclosed in the '844 patent as UBDs are merely describing well-known and off-the-shelf products such as Netapp's CoW engine/driver that resides between the file system layer and the actual storage devices. *See, e.g.,* Decl. ¶108, ¶563. Accordingly, the devices in communication with each other and performing the functions of this claim as detailed in the mappings for *Birse* (in view of *Holzmann's* caching disclosures) for claims 1, 11, and 12 disclose the claimed UBDs of this claim and/or the POSA knowledge to employ known devices for performing the functions *Birse* (e.g., a Netapp or similar appliance) disclose this element. Decl. ¶410, ¶563.
Claim 5	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[5.0] The system as recited in claim 4 wherein said union block devices comprise low-level drivers for interfacing between the	*Birse* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 4 for the reasons presented above for claim 4. *Birse* and *Holzmann,* particularly in view of POSA knowledge of well-known and off the shelf products for providing devices configured to interface as recited in this claim, disclose or at least render obvious this claim.

file systems of respective compute nodes and said first storage unit, respective second storage units, and said cache.	As explained for claim 4, the term Union Block Device is informally defined in the '844 patent by way of a set of tasks it performs. *See, e.g.,* Section III.A.2. Thus, the system components as necessary to provide such features, taken together, embody the Union Block Device of this claim. Decl. ¶410. Reading *Birse* (in view of *Holzmann's* caching disclosures), a POSA would have understood that *Birse* would have been implemented with storage systems capable of performing the functions in those references, such as the storage systems created by Netapp or similar appliances. *See, e.g.,* mapping of claim 4 above, Decl. at Sections VII.C and VII.E. These systems provide components—e.g., the claimed *low-level drivers—for interfacing between* the components to accomplish the disclosed functions. *See, e.g.,* Decl. ¶410-411, Decl. ¶108 (noting that Netapp's CoW driver resides between the file system layer and the actual storage devices), ¶563, and Decl. at Section VII.E. Accordingly, the devices in communication with each other and performing the functions of this claim provided in the mappings of *Birse* (in view of *Holzmann's* caching disclosures) to claims 1, 11, and 12 above, as understood to be implemented by a POSA (*see, e.g.,* mapping of *Birse* and *Holzmann* to claim 4), disclose the UBDs recited in this claim. Decl. ¶410-11, ¶563, Decl. at Sections VII.C and VII.E. *See Birse* as mapped for claims 1, 11, and 12 above; *Holzmann* as mapped for element [1.3] above.
Claim 6	**SNQ3: *Birse* (EX1006) and *Holzmann* (EX1007)**
[6.0] The system as recited in claim 1 wherein said first storage unit is read-only.	*Birse* in view of *Holzmann* disclose, or at least suggest, the system as recited in claim 1 for the reasons presented above for claim 1. *Birse* further discloses *the first storage unit* (e.g., root image system volume 174 stored on hard drive 610 of NC server 170) *is read-only*. ***See, e.g., Birse:*** "The system volumes 174 include a protected, **read-only**, **master copy of the operating system software**." *Birse,* 5:13-14.

D. SNQ4: *Birse* in view of *Federwisch* Presents a Significant New Question of Patentability for Challenged Claims 14-27

1. Overview of SNQ4

Claims 14-27 are obvious as explained herein, accompanied by the claim chart provided in Section V.D.5 and the Zadok Declaration (Decl. ¶566-646), are rendered obvious by *Birse* in view

of *Federwisch*. Notably, *Birse* discloses the claimed root/leaf architecture, including where leaf images only contain additional data blocks not previously contained in the root image and changes made by respective compute nodes to the blocks of the root image. *Federwisch* discloses any additional indexing limitations to the extent, if any, not already taught by *Birse* and/or obvious to a POSA.

2. Overview of *Birse*

An overview of *Birse* is provided in Section V.C.2.

3. Overview of *Federwisch*

An overview of *Federwisch* is provided in Section V.B.3.

4. Combination of *Birse* and *Federwisch*

The proposed combination of *Birse* with *Federwisch* is implementing *Federwisch's* details of indexing into *Birse's* storage systems and files.

The Supreme Court in *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398 (2007) adopted a flexible approach to evaluating whether a patent claiming a combination of elements of prior art is obvious. Under *KSR* and as detailed in the Zadok Declaration (EX1003), a POSA would have been motivated to combine *Birse* and *Federwisch*.

A POSA would have been motivated to modify *Birse* with *Federwisch's* indexing techniques because, while it suggests that the boot images are indexed and organized through bootfile selection and a registry (EX1006, 10:5-8, 10:16-18; Decl. ¶¶576-77), it does not provide many details. Thus, a POSA building *Birse's* 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 *Birse*). Decl. ¶¶572, 576-77. That is, so motivated by *Birse's* disclosure of indexing, to the extent a POSA required implementation details, they would have turned to *Federwisch*. *See* Decl. ¶577; MPEP § 2143(I)(Rationale (G)). Moreover, application of the disclosed indexing techniques described in *Federwisch* to *Birse* represents the use of *Federwisch's* known techniques for indexing to improve the storage systems of *Sapuntzakis* in the same way. *See* Decl. ¶¶572, 577; MPEP § 2143(I)(Rationale (C)).

Moreover, a POSA would have known that indexing of any data is an age-old technique long utilized in computer systems. *See, e.g.,* Decl. ¶¶572, 577, Decl. at Section VII.H. Thus, a

POSA would have had a reasonable expectation of success because indexing of data is an age-old technique long utilized in computer systems. *See* Decl. ¶¶603-605.

Additionally, there was significant overlap in *Birse's* and *Federwisch's* disclosures—both 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 POSA would have known CoW techniques enable snapshots), and disclose caching blocks of the root/boot disk images. Decl. ¶¶ 583-94. Thus, a POSA would have had a reasonable expectation that the indexing techniques that were disclosed in *Federwisch* would operate in *Birse's* similar system.

Modifying *Birse* to include *Federwisch's* indexing would have been simple to a POSA, and would not have required extensive experimentation or imposed undue burden – including because *Birse* suggests that the boot images are indexed and organized through bootfile selection and a registry. Decl. ¶596 (citing EX1006, 10:5-8, 10:16-18). 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 POSA would have had a high expectation of success, because indexing techniques were well understood for decades beforehand. *See* Decl. ¶¶603-05.

5. Claim Chart Providing Detailed Analysis[19]

Claim 14	SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)
[14.0] A system for indexing file systems for a plurality of compute nodes, the system comprising:	To the extent the preamble is limiting, *Birse* discloses *a system for indexing file systems* (NC systems connected via the internet) *for a plurality of computer nodes* (NC clients). *See also, e.g.,* Section V.C.5, element [7.0]. ***See, e.g., Birse:*** **"...[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." EX1006, 4:56-60. *See also* 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications).

[19] This claim chart mapping is further supported with an analysis in prose form provided by Dr. Zadok in his declaration. Decl. at Section XII.D.

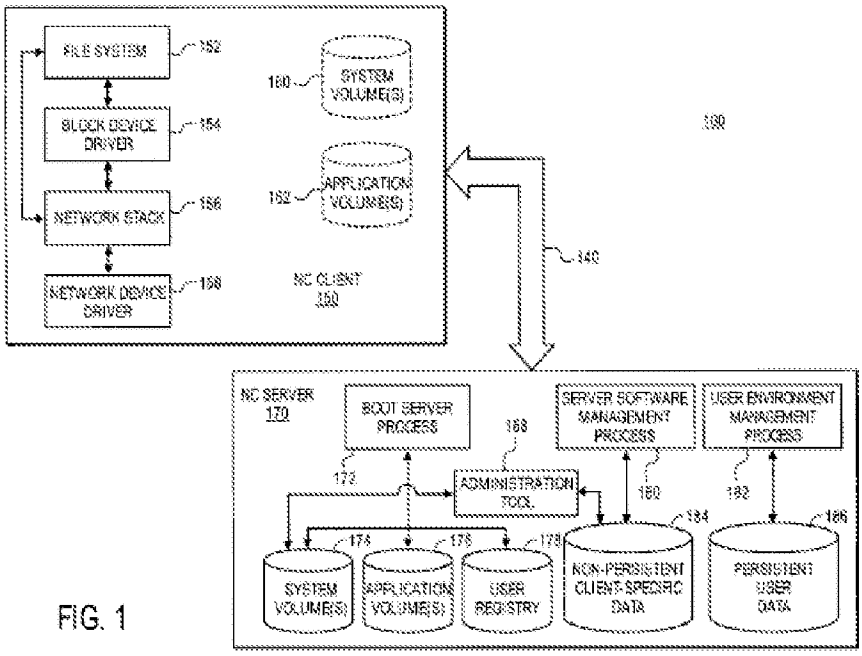


FIG. 1

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

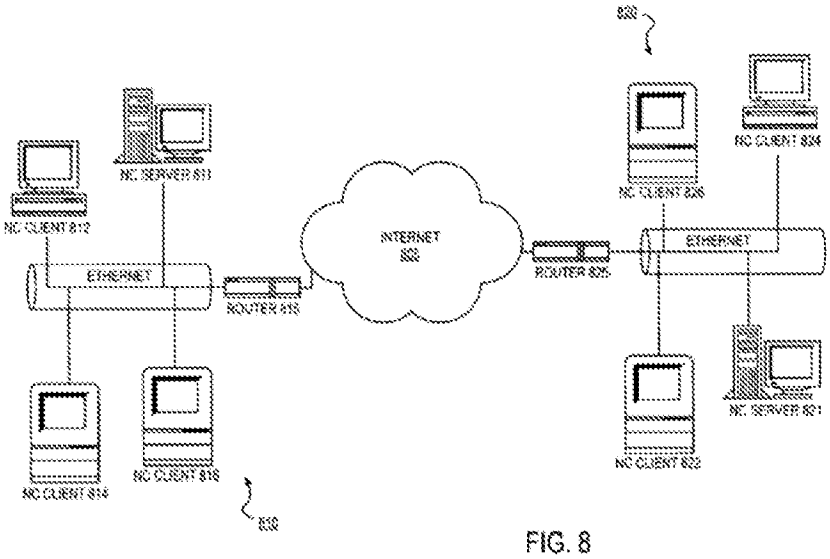
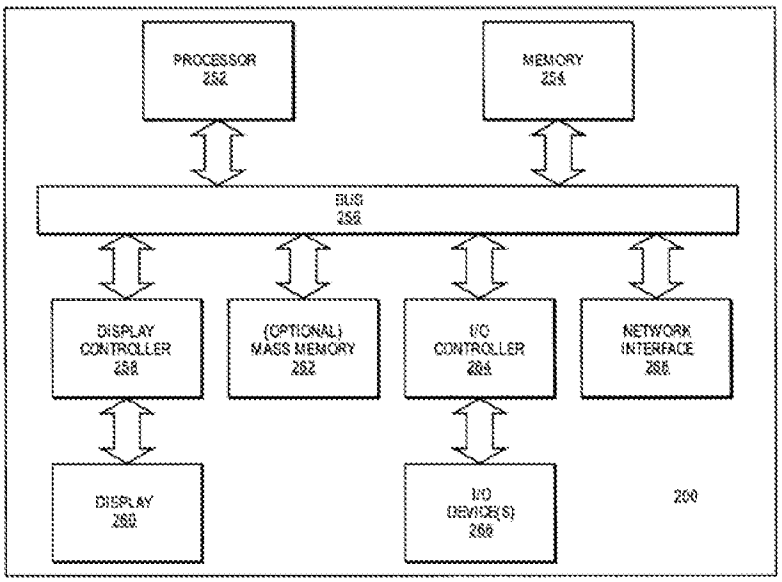
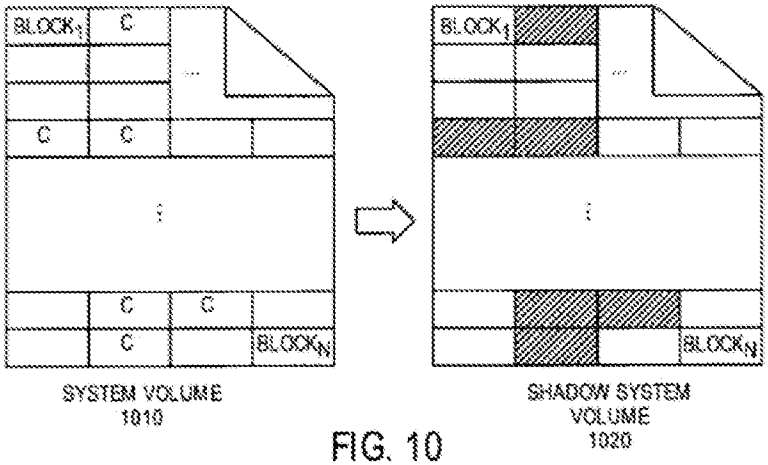


FIG. 8

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. *See* 4:27-57.

	 FIG. 2 *See also* elements below, particularly element [14.3.d], to the extent more is required.
[14.1] a first storage unit configured to store blocks of a root image of said compute nodes;	*Birse* discloses *a first storage unit* (one or more hard drives 610 of NC server 170, logically separate in NC folders/directories 630, 640) *configured to store blocks of a root image* (system volumes 174, 660, 922, 924) *of said compute nodes* (multiple NC clients 150). *See also, e.g.,* Section V.C.5, element [7.1]. ***<u>See, e.g., Birse</u>:*** "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**." *Birse,* 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." *Birse,* 14:27-34, 41-44. "[T]he system volumes 174...may be independently distributed across multiple server computer systems." *Birse,* 6:59-63.

	"FIG. 6 [below] 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**.... The NC shared data folder 640 **includes a shared operating system image 660**, a boot image 670, and a clients folder 680." *Birse,* 12:30-44. *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." *Birse,* 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:  FIG. 10
[14.2] a plurality of second storage units configured to store leaf images of respective compute nodes, said leaf images comprising only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the blocks of said root	*Birse* discloses *a plurality of second storage units* (NC client folders 690 #1...n) *configured to store leaf images* (shadow volumes/images 930, 695) **of respective compute nodes, said leaf images comprising 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). *See also* Section V.C.5, element [7.2]. ***<u>See, e.g., Birse</u>:*** "In this example, **the NC clients are numbered from 1 to N**

image, wherein said leaf images of respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes; and	**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." *Birse,* 12:54–59 (discussing FIG. 6, in [7.1] above). "**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." *Birse,* 7:35-53; see *also* 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." *Birse,* 10:63-65, 11:6-17.

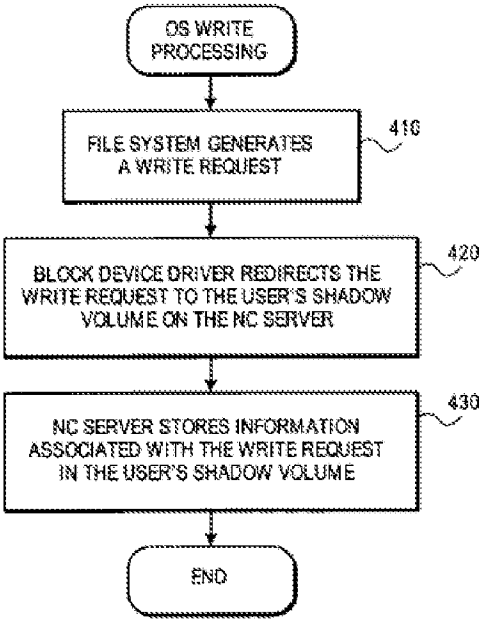


FIG. 4

"[In one embodiment,] **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." *Birse,* 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." *Birse,* 14:49-53.
>
> "...[T]he 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." *Birse,* 14:56-61.

<table>
<tr>
<td>[14.3.a] 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</td>
<td>Element [14.3.a] is substantially similar to recitations in element [4.0]. See comparison below.
<table>
<tr><td>[14.3.a] 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</td><td>[4.0] 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]]</td></tr>
</table>
The UBDs that correspond to said computer nodes recited in element [14.3.a] are consistent with the recitation of the UBDs for respective computer nodes recited in element [4.0].

Accordingly, Birse discloses element [14.3.a] for the reasons presented in the mapping of Birse to element [4.0] above in Section V.C.5.</td>
</tr>
<tr>
<td>[14.3.b] [UBDs configured] to distribute said file systems to said compute nodes,</td>
<td>Element [14.3.b] is substantially similar to recitations in element [4.0]. See comparison below.
<table>
<tr><td>[14.3.b] to distribute said file systems to said compute nodes,</td><td>[4.0] to distribute application environments to the compute nodes,</td></tr>
</table>
File systems and application environment are substantially synonymous in this context, e.g., an application environment is largely defined by the file systems employed at the computer nodes, and by distributing the file systems, the application environment is created/distributed.

Moreover, regardless of the particular definition Patent Owner puts forth, Birse's disclosure of a system that can distribute system volumes and/or needed new blocks of data discloses the claimed UBD to distribute file systems for similar reasons as explained for claim [4.0] and the distribution of an application environment.

Accordingly, Birse discloses element [14.3.b] for the reasons presented in the mapping of Birse to element [4.0] above in Section V.C.5.</td>
</tr>
<tr>
<td>[14.3.c] wherein said union block devices are configured to create said file systems by merging the blocks of said root image stored on the first storage unit with the blocks of respective leaf images stored on respective second storage units, and</td>
<td>Element [14.3.c] is substantially similar to recitations in elements [4.0], [1.1] and [1.2]. See comparison below.
<table>
<tr><td>[14.3.c] wherein said union block devices are configured to create said file systems by merging the blocks of said <u>root image</u> <u>stored on the first storage unit</u> with the blocks of respective <u>leaf images</u> <u>stored on respective second storage units,</u></td><td>[4.0] 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.

[1.1] <u>a first storage unit configured to store blocks of a</u></td></tr>
</table></td>
</tr>
</table>

			root image of said compute nodes [1.2] *a plurality of second storage units configured to store leaf images of respective compute nodes* | As explained for element [14.3.b], *file systems* (element [14.3.b-c) and *application environments* (element [4.0]) are substantially similar in this context. Accordingly, *Birse* discloses element [14.3.c] for the reasons presented in the mapping of *Birse* to elements [4.0], [1.1], and [1.2] above in Section V.C.5.
[14.3.d] 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.	*Birse* and *Federwisch* further disclose *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*. In particular, *Birse* suggests that its data is *indexed*, but it does not provide many details for the well-known and common technique. Decl. ¶¶572, 576-77, To the extent a POSA would require implementation details, *Federwisch* discloses a "scanner" which serves to *index* an image. Decl. ¶¶573-74, 578-81, *Federwisch,* Abstract, [0053], [0071], [0072]. ***See, e.g., Birse*:** "In order to allow the NC server 320 to recognize the NC client 310 next time it boots, **information regarding the NC client 310 may be stored in the user registry 178**." *Birse,* 10:5-8. "If the NC client 310 is known to the NC server 320, then at step 365 **information regarding the NC client 310**, such as its IP address **may be retrieved from the user registry 178**." *Birse,* 10:16-18. ***See, e.g., Federwisch*:** "A system and method for remote asynchronous replication or mirroring of changes in a source file system snapshot in a destination replica file system 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**." *Federwisch,* Abstract. "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."

	Federwisch, [0053]. "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**." *Federwisch,* [0071]. "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." *Federwisch,* [0072]. Thus, in building the system of *Birse,* a POSA would have implemented the indexing system of *Federwisch* on any compute node of *Birse* to index the root image and provide the index results to other compute nodes. *See also* Decl. ¶¶569-81, 582-605, Decl. at Sections VIII.B (discussing the '844 patent's disclosure of indexing) and VII.F (providing an overview of indexing technologies). A POSA would have been motivated and had a reasonable expectation in applying the implementation details of *Federwisch's* indexing to the indexing of *Birse* for the reasons presented in Section V.D.4.
Claim 15	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[15.0] The system as recited in claim 14 wherein said first storage unit and said second storage units are contained within a single storage appliance.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 14 for the reasons presented above for claim 14. *Birse* further discloses the remainder of the recitations of claim 15 for the reasons presented for the mapping of *Birse* to element [3.0] in Section V.C.5.
Claim 16	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[16.0] The system as recited in claim 14 further comprising: a plurality of union block devices configured to interface between respective compute nodes and said first storage unit and respective second storage units, said	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 14 for the reasons presented above for claim 14. *Birse* further discloses the remainder of the recitations of claim 16 for the reasons presented for the mapping of *Birse* to element [4.0] in Section V.C.5.

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.	
Claim 17	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[17.0] The system as recited in claim 16 wherein said union block devices comprise low-level drivers for interfacing between the file systems of respective compute nodes and said first storage unit, respective second storage units, and said cache.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 16 for the reasons presented above for claim 16. *Birse* further discloses the remainder of the recitations of claim 17 for the reasons presented for the mapping of *Birse* to element [5.0] in Section V.C.5.
Claim 18	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[18.0] The system as recited in claim 14 wherein said first storage unit is read-only.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 14 for the reasons presented above for claim 14. *Birse* further discloses the remainder of the recitations of claim 18 for the reasons presented for the mapping of *Birse* to element [13.0] in Section V.C.5.
Claim 19	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[19.0] A method for indexing file systems for a plurality of compute nodes, comprising:	To the extent the preamble is limiting, *Birse* discloses *a method for indexing file systems* (methods employed by its NC systems connected via the internet) *for a plurality of computer nodes* (NC clients). *See also, e.g.,* Section V.C.5, element [7.0]. ***See, e.g., Birse*:** "...**[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." EX1006, 4:56-60. *See also* 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications).

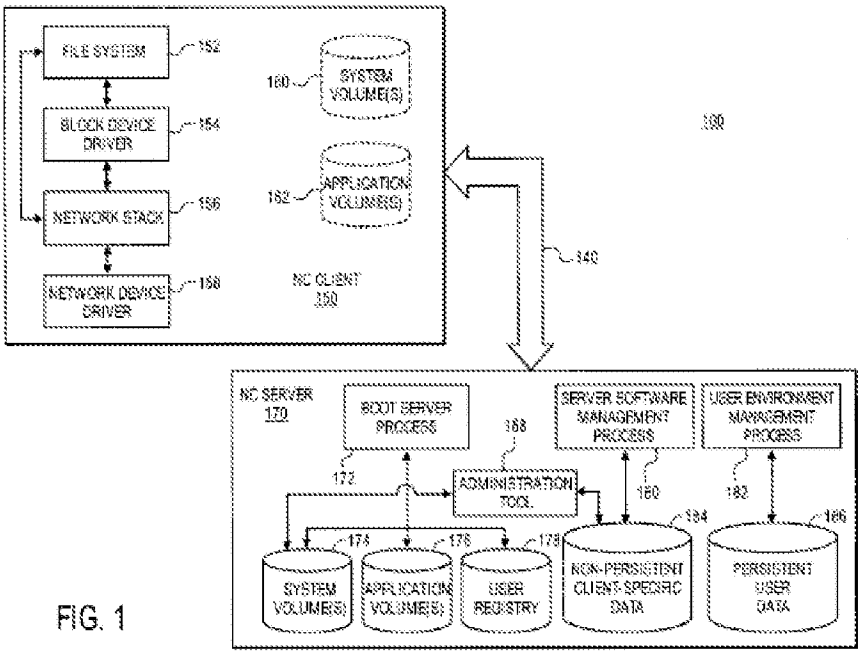


FIG. 1

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

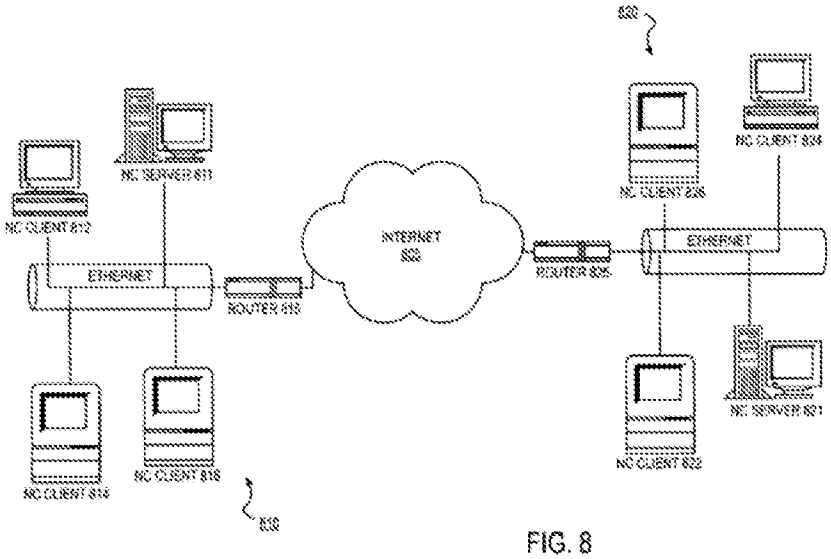
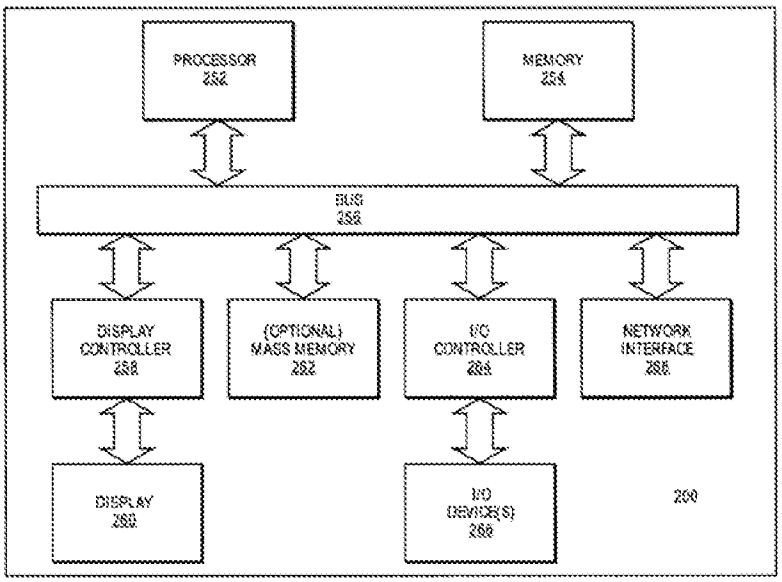
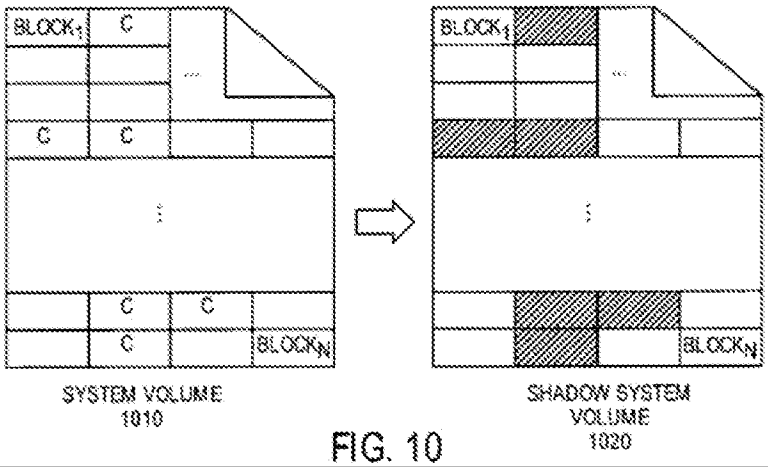


FIG. 8

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. *See* 4:27-57.

	 FIG. 2 *See also* elements below, particularly element [19.4]-[19.5], to the extent more is required.
[19.1] storing blocks of a root image of said compute nodes on a first storage unit;	*Birse* discloses *storing blocks of a root image* (system volumes 174, 660, 922, 924) *of said compute nodes* (multiple NC clients 150) *on a first storage unit* (one or more hard drives 610 of NC server 170, logically separate in NC folders/directories 630, 640) *configured to store blocks of a root image* (system volumes 174, 660, 922, 924). *See also, e.g.,* Section V.C.5, element [7.1]. ***See, e.g., Birse:*** "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**." *Birse,* 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." *Birse,* 14:27-34, 41-44. "[T]he system volumes 174...may be independently distributed across multiple server computer systems." *Birse,*

	6:59-63. "FIG. 6 [below] 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**…. The NC shared data folder 640 **includes a shared operating system image 660**, a boot image 670, and a clients folder 680." *Birse,* 12:30-44. *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." *Birse,* 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:  FIG. 10
[19.2] storing leaf images for respective compute nodes on respective second storage units, said leaf images comprising only additional data blocks not previously contained in said root image and changes made by respective compute nodes to the	*Birse* discloses *storing leaf images* (shadow volumes/images 930, 695) *for respective compute nodes on respective second storage units* (NC client folders 690 #1...n), *said leaf images comprising 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). *See also* Section V.C.5, element [7.2]. ***See, e.g., Birse:***

blocks of the root image, wherein said leaf images for respective compute nodes do not include blocks of said root image that are unchanged by respective compute nodes;	"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." *Birse,* 12:54–59 (discussing FIG. 6, in [7.1] above). "**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." *Birse,* 7:35-53; see *also* 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." *Birse,* 10:63-65, 11:6-17.

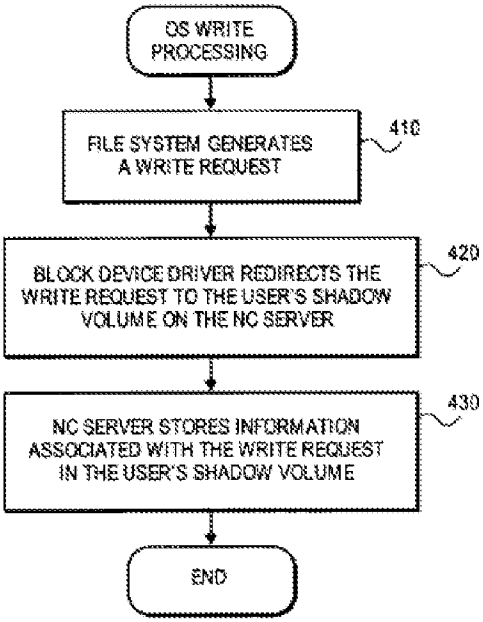


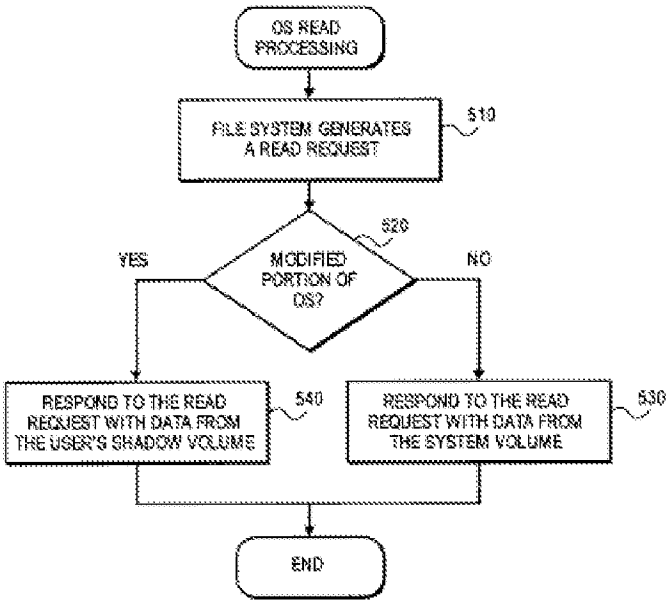
FIG. 4

"[In one embodiment,] **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." *Birse,* 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." *Birse,* 14:49-53.

"...[T]he 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**." *Birse,* 14:56-61.

[19.3] merging the blocks of said root image with the blocks of respective leaf images stored on respective second storage units to create respective file systems for respective compute nodes;	*Birse* discloses merging the blocks of said root image (system volumes 174, 660, 922, 924) with the blocks of respective leaf images (shadow volumes/images 930, 695) stored on respective second storage units (NC client folders 690 #1...n) to create respective file systems for respective compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826). In this way, cohesive respective application environments (dynamically merged system environments) are created. *See also* Section V.C.5, elements [11.0] and [1.2]. ***See, e.g., Birse:*** "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." *Birse,* 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].** Advantageously, 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." *Birse,* 9:2-10. "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. **In 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**." *Birse,* 11:39-41, 11:55-12:10.  FIG. 5
[19.4] receiving indexing results pertaining to said root image from one of said compute nodes; and	As explained for element [14.3.d], *Birse* and *Federwisch* further disclose *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*. It logically follows that, since *Birse* and *Federwisch* discloses that "at least one of the compute nodes" is configured to "provide the indexing results" to "another of said computer nodes," then the "another one of said compute nodes" is configured to "receiv[e] indexing results" from the "one of said compute nodes." Therefore, *Birse* and *Federwisch* disclose this element for the reasons presented for element [14.3.d].
[19.5] providing said indexing results to the others of said compute nodes.	*Birse* and *Federwisch* disclose this element for the reasons presented for element [14.3.d].

Claim 20	SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)
[20.0] The method as recited in claim 19 further comprising: storing said indexing results on a shared storage unit.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 19 for the reasons presented above for claim 19. *Birse* and *Federwisch* further disclose *storing said indexing results on* "at least one of said compute nodes" that is configured to index said root image and provide the indexing results to another of said compute nodes for the reasons presented for element [14.3.d]. *Federwisch* further discloses sharing indexes in a system-agnostic format and the ability to share indexes among devices as "exports" and discloses a "first snapshot [that] may be an exported snapshot of the replicated snapshot on the destination." Thus, a POSA would have found it obvious that the "at least one of said compute nodes" would be *a shared storage unit*. That is, a POSA would have been motivated by the known advantages of storing indexing results on a shared storage unit such as making the index conveniently accessible over network storage volumes. *See, e.g.,* Decl. ¶617, Decl. at Section VII.C (providing an overview of networked and distributed storage system technologies); MPEP § 2143(I)(Rationale G). A POSA would have had a reasonable expectation of success in using a shared storage unit at least because using shared storage was common and well known. Decl. ¶617, Decl. at Section VII.C. ***See, e.g., Federwisch*:** "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." *Federwisch,* ¶[0086] "The **destination file system is exported** read-only the user." *Federwisch,* ¶[0015]. "Reference is made to FIG. 16, which describes a generalized rollback procedure 1600 according to an illustrative embodiment. As a matter of ongoing operation, in step 1605, a "first" snapshot is created. **This first snapshot may be an exported snapshot of the replicated snapshot on the destination**."

Claim 21	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[21.0] The method as recited in claim 19 wherein said merging occurs at an operational level between respective file systems of the compute nodes and said first storage unit and respective second storage units.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 19 for the reasons presented above for claim 19. *Birse* further discloses the remainder of the recitations of claim 21 for the reasons presented for the mapping of *Birse* to element [12.0] in Section V.C.5.
Claim 22	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[22.0] The method as recited in claim 19 wherein said first storage unit is read-only.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 19 for the reasons presented above for claim 19. *Birse* further discloses the remainder of the recitations of claim 22 for the reasons presented for the mapping of *Birse* to element [12.0] in Section V.C.5.
Claim 23	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[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:	To the extent the preamble is limiting, *Birse* discloses *a computer-readable storage medium having instructions stored thereon that* (software on one or more of the computers and storage in the NC system), *in response to execution by at least one computing device, cause the at least one computing device to perform steps*. *See also* mappings of claim 19 above. ***See, e.g., Birse*:** "...[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**." EX1006, 4:56-60. *See also* 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications).

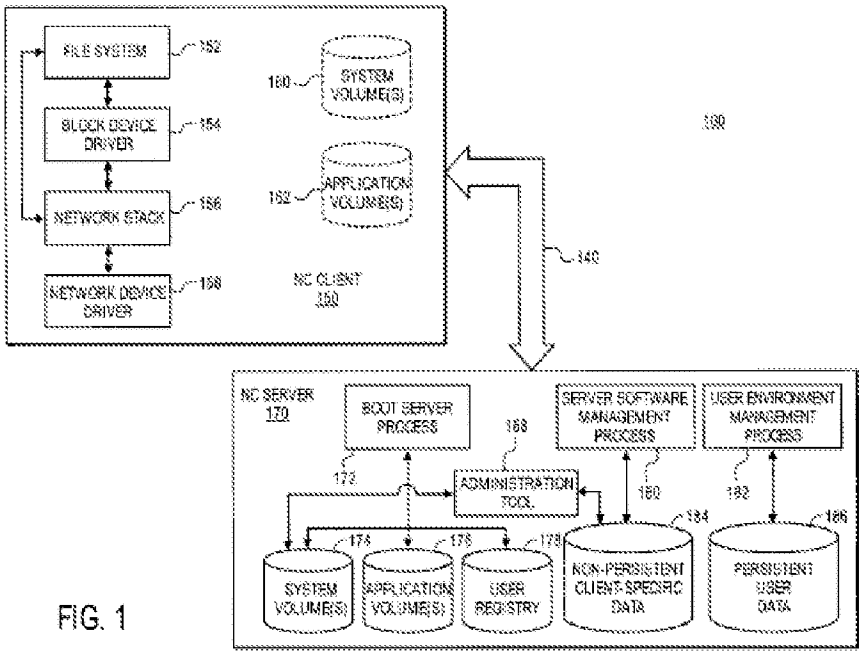


FIG. 1

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

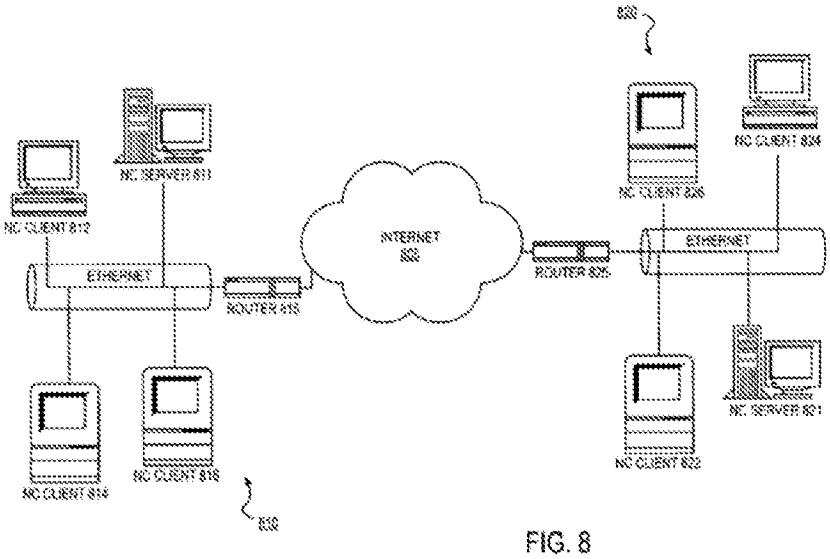
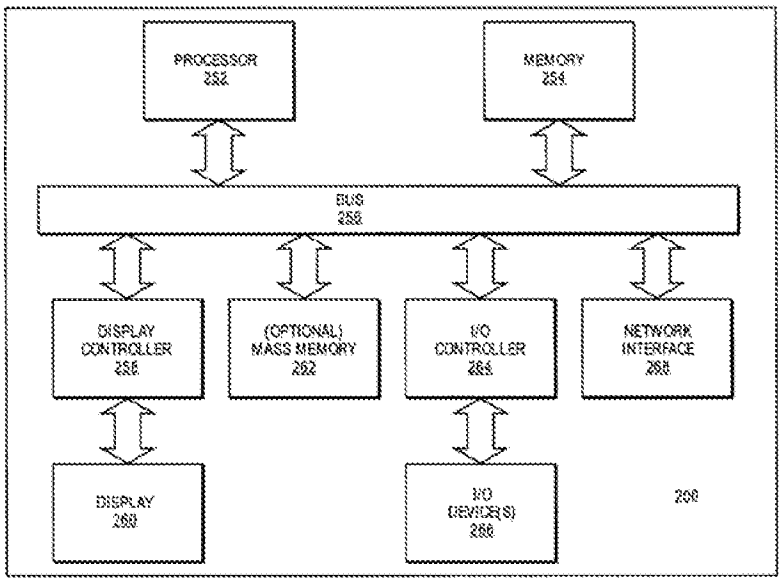
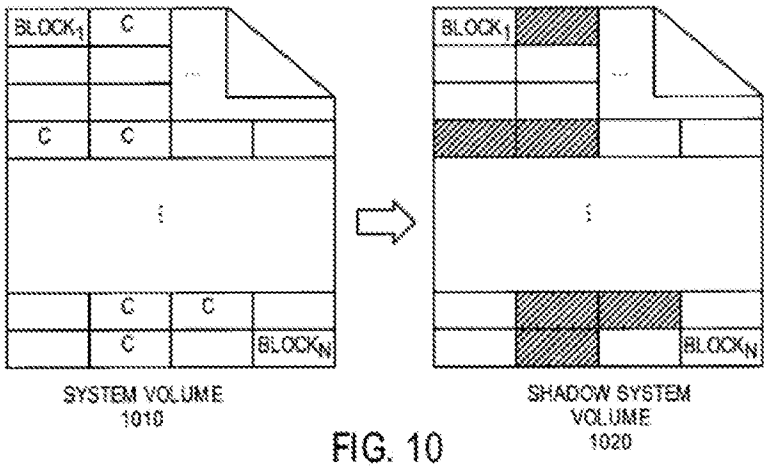


FIG. 8

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. *See* 4:27-57.

	 FIG. 2
[23.1] receive data blocks of a file system, said data blocks comprising a root image portion and leaf image portion, said leaf image portion comprising only additional data blocks not previously contained in said root image portion and changes made by said first compute node[20] to the blocks of said root image, wherein said leaf image portion does not include blocks of said root image that are	*Birse* discloses *receiving data blocks of a file system* (e.g., system volume 174, 922, 924, 1010 stored as blocks as shown in Fig. 10). *Birse* discloses that *said data blocks comprise a root image portion* (system volumes 174, 660, 922, 924) *and a leaf image portion* (shadow volumes/images 930, 695), *said leaf image portion comprising only additional data blocks not previously contained in said root image portion* (write requests for new data) *and changes made by said first compute node to the blocks of the root image* (write request for changes to operating system data), *wherein said leaf image portion does not include blocks of said root image that are unchanged by said first compute node* (only new data and changes are written). *See also* Section V.C.5, element [7.2], and V.D.5, element [19.2]. *Birse* discloses that said file system is the result *of merging said root image portion* (system volumes 174, 660, 922, 924) *and said leaf image portion* (shadow volumes/images 930, 695) together at the block level. *See also* Section V.C.5., element [11.0], and V.D.5, element [19.3]. ***<u>See, e.g., Birse</u>:***

[20] Requester notes that there is no antecedent basis for "said first compute node." For purposes of this Request, the broadest reasonable interpretation of this element includes the interpretation as "a first computer node." Notably, the "first *storage unit*" as recited claim 1 is <u>not</u> equivalent or related to the "first *compute node*" of claim 23. In claim 1, the leaf images of "respective compute nodes" (e.g., of the "first compute node" of this claim 23) are stored in a plurality of "*second* storage units." Indeed, claim 23 does not recite either of the first or second storage units that are recited in claim 1.

unchanged by said first compute node, wherein said file system is the result of merging said root image portion and said leaf image portion together at the block-level;	 FIG. 10 *Birse,* Fig. 10 (showing the shared system volume 660 (a.k.a. system volume 174, 922, 924, 1010) and shadow system volume stored as blocks). "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." *Birse,* 12:54–59 (discussing FIG. 6). "**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." *Birse,* 7:35-53; see *also* 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." *Birse,* 10:63-65, 11:6-17.

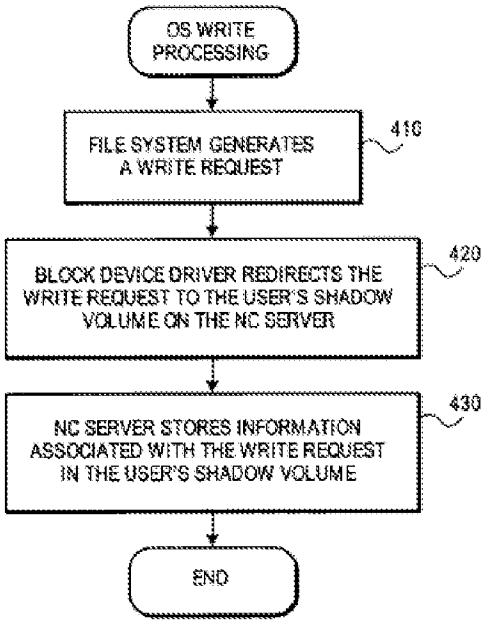


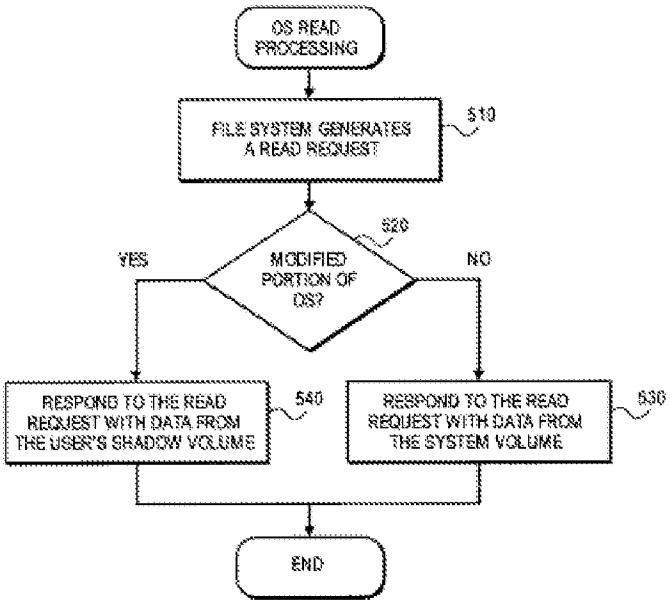
FIG. 4

"[In one embodiment,] **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." *Birse,* 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." *Birse,* 14:49-53.

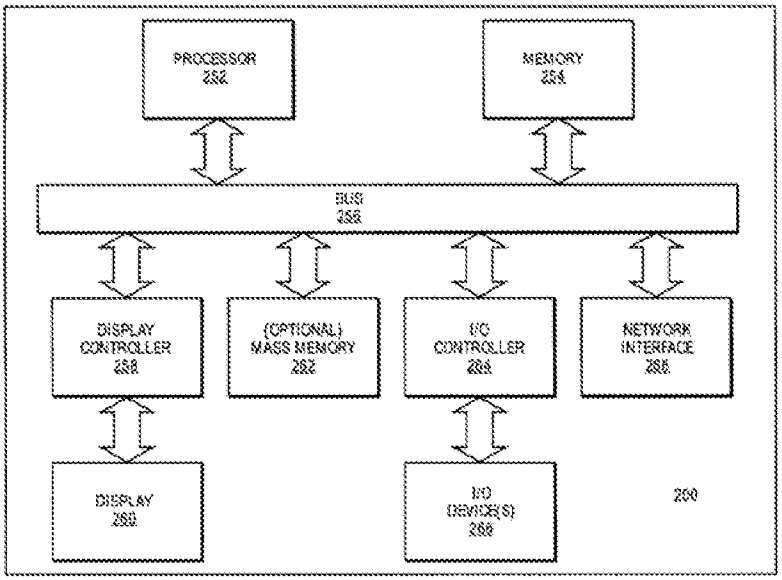
"...[T]he shadow system volume 930 is used by the block

	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**." *Birse,* 14:56-61. "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." *Birse,* 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]**. Advantageously, 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." *Birse,* 9:2-10. "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. **In 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.**" *Birse,* 11:39-41, 11:55-12:10.

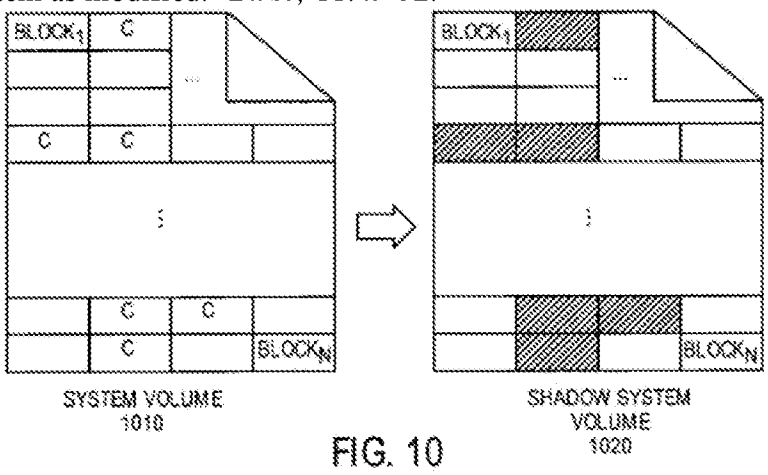
	 FIG. 5
[23.2] index said root image portion; and	As explained for element [14.3.d], *Birse* and *Federwisch* further disclose indexing said root image and providing the indexing results to another of said compute nodes. This encompasses, and thus discloses, the indexing of said root image portion recited in this element. Therefore, *Birse* and *Federwisch* disclose this element for the reasons presented for element [14.3.d].
[23.3] provide the results of said indexing to a second compute node,	As explained for element [14.3.d], *Birse* and *Federwisch* further disclose *indexing said root image and providing the indexing results to another of said compute nodes*. This encompasses, and thus discloses, the *providing the results of said indexing to* another, e.g., *a second, compute node*. Therefore, *Birse* and *Federwisch* disclose *providing the results of said indexing to a second compute node* for the reasons presented for element [14.3.d].

[23.4] wherein said logic encoded in the one or more tangible media[21] comprise computer executable instructions executed by the first compute node.	*Birse* discloses *logic encoded in one or more tangible media* (software on one or more of the computers and storage in its NC clients and NC servers). *Birse* discloses that *the logic comprises computer executable instructions at least because it uses computers* (e.g., NC servers and NC clients) *to execute the instructions* (e.g., run the application). *Birse* discloses that *the instructions are executed by the first computer node* (e.g., the system image mounted to NC client 150, NC clients executing write requests, etc.). ***See, e.g., Birse*:** "...[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**." EX1006, 4:56-60. *See also* 2:51-3:14 (alternative embodiments providing data from NC server to NC clients to upgrade operating system, upgrade applications). "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." *Birse,* 12:44–47. "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

[21] Requester notes that there is no antecedent basis for "said logic" or "the one or more tangible media." For purposes of this Request, the broadest reasonable interpretation of this element includes the interpretation that this element is intended to confirm that the "instructions stored" on "A computer-readable storage medium" as recited in preamble is in the form of logic or instructions that have been encoded in on or more tangible media forms, and—as the rest of this element recites—that form comprises "computer executable instructions executed by the first computer node."

	recorded instead in a client- specific shadow volume residing on the NC server 170." *Birse,* 10:63-65, 11:6-17. 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. *See* 4:27-57.  FIG. 2
Claim 24	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[24.0] The computer-readable storage medium of claim 23, wherein the instructions further cause the at least one computing device to store said results of said indexing on a shared storage unit accessible by said second compute node.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 23 for the reasons presented above for claim 23. *Birse* in view of *Federwisch* disclose, or at least suggest, the *instructions further cause the at least one computing device to store said results of said indexing on a shared storage unit* for the reasons presented for claim [20.0]. *Birse* in view of *Federwisch* disclose, or at least suggest, that *the shared storage unit is accessible by said second computer node* (e.g., "to another of said compute nodes") for the reasons presented for element [14.3.d].
Claim 25	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[25.0] The computer-readable storage medium of claim 23, wherein the instructions further cause the at least one computing device to	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 23 for the reasons presented above for claim 23. *Birse* and *Federwisch* further disclose *the instructions further causing the at least one computing device to index said leaf image portion.* *Birse* discloses *root and leaf images represented at the block level, wherein the leaf images store changed blocks of the root image* as

index said leaf image portion.	described in the mapping of *Birse* to elements [7.1] and [7.2] of SNQ3. *See* Section V.C.5, elements [7.2]-[7.3]; *see also* Section V.D.5, element [14.2]. *Federwisch* discloses an indexing system operating at the block level capable of identifying and indexing changed blocks. Decl. ¶¶635-37, *Federwisch* [0071], [0072]. That is, each snapshot (corresponding to the '844 patent's "leaf image") has its own index of file blocks. Decl. ¶638; *Federwisch,* [0109]. Thus, the combination of these disclosures results in *the indexing of leaf image portions* with *Federwisch's* indexing system that identifies and indexes changed blocks, i.e., the leaf images that store the changed blocks of the root image as *Birse* discloses. *See also* Decl. ¶¶634-38. A POSA would have been motivated to combine *Birse* with *Federwisch* for the reasons presented in Section V.D.4. Additionally, a POSA would have known that indexing is an old and common technique. Decl. ¶631. A POSA would have understood that it is desirable to index only changed portions of target rather than re-indexing the entire target each time. Decl. ¶631. That is, a POSA would be motivated by the known benefit of a more efficient indexing of only the changed portions to implement *Federwisch's* indexing to the storage of *Birse's* changed blocks. MPEP § 2143(I)(Rationale G). A POSA would have had a reasonable expectation of success at least because indexing was a well-known technique and storage and file systems include timestamps that denote when the files/data were last modified (e.g., the system was already set up in such a way to make indexing an easy process). Decl. ¶631; Decl. at Sections VII.B and VII.F. ***See, e.g., Birse*:** "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." *Birse,* 7:35-53; see *also* 11:30-

	32. "[In one embodiment,] **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." *Birse,* 11:49-52.  FIG. 10 *Birse,* Fig. 10 (showing the shared system volume **660** (a.k.a. system volume **174**, **922**, **924**, **1010**) and shadow system volume stored as blocks). ***See, e.g., Federwisch:*** "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.**" *Federwisch,* [0071]. "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." *Federwisch,* [0072]. "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)." *Federwisch*, [0109].
Claim 26	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[26.0] The computer-readable storage medium of claim 25, wherein the instructions further cause the at least one	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 25 for the reasons presented above for claim 25. *Federwisch* further discloses *the instructions further causing the at least one computing device to re-index said file system by re-indexing said leaf image portion* (*Federwisch's* re-indexing of incremental

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.	snapshots) *and merging the results of said re-indexing of said leaf image portion with said results of said indexing of said root image portion* (*Federwisch's* updating of the destination snapshot's index maps). Thus, the combination of these disclosures results in *the re-indexing of leaf image portions and merging of the results of the re-indexing with the results of indexing of said root portion* as *Federwisch* discloses in the system that *Birse* discloses. *See also* Decl. ¶¶640, 644. A POSA would have been motivated to combine *Birse* with *Federwisch* for the reasons presented in Section V.D.4. Additionally, a POSA would have known that indexing is an old and common technique. Decl. ¶640. A POSA would have understood that when some data changes on a disk of 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. Decl. ¶640. That is, a POSA would be motivated by the known benefit of a more efficient indexing of only the changed portions to implement *Federwisch's* indexing to the storage of *Birse's* changed blocks. MPEP § 2143(I)(Rationale G). Moreover, when some data changes, for example in any form of a simple database (such as an index), a POSA would have understood that the new index data would be easily merged with older data to, for instances, purge the now-stale index entries. Decl. ¶640; *see also* Decl. at Sections VII.B (review of data storage and file system technologies) and VII.F (review of indexing technologies). That is, a POSA would be motivated by the known benefit of purging old and now-incorrect index entries to merge the results of the re-indexing of the leaf image portion with the indexing of the root portion. Decl. ¶640. A POSA would have had a reasonable expectation of success at least because indexing was a well-known technique and it was well within a POSA's ability to re-index and merge the new index data with the older data. Decl. ¶640; Decl. at Sections VII.B (review of data storage and file system technologies) and VII.F (review of indexing technologies). ***<u>See, e.g., Federwisch:</u>*** "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).” *Federwisch*, [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. Decl. ¶645.
Claim 27	**SNQ4: *Birse* (EX1006) and *Federwisch* (EX1008)**
[27.0] The computer-readable storage medium of claim 23 wherein said root image portion is read-only.	*Birse* in view of *Federwisch* disclose, or at least suggest, the system as recited in claim 23 for the reasons presented above for claim 23. *Birse* further discloses the remainder of the recitations of claim 21 for the reasons presented for the mapping of *Birse* to element [13.0] in Section V.C.5.

VI. SECONDARY CONSIDERATIONS

This Request shows the Challenged Claims of the ’844 patent are unpatentable in view of the prior art references. The Applicant did not identify any evidence of secondary considerations during prosecution. Further, the clear teachings in the prior art outweigh any supposed “secondary considerations.” *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1, 36 (1966).

VII. DISCLOSURE OF CONCURRENT LITIGATION, REEXAMINATION, AND RELATED PROCEEDINGS

Based on available information, the ’844 patent is or has been involved in these matters:

Case Caption	Status
Intellectual Ventures I LLC et al v. The Bank of New York Mellon Corp., 3-25-cv-00631, (NDTX, Mar. 15, 2025)	Pending
Intellectual Ventures I LLC et al v. Nationwide Mutual Insurance Co., 3-25-cv-00632, (NDTX, Mar. 15, 2025)	Pending
Intellectual Ventures I LLC et al v. Am. Airlines, Inc., 4-24-cv-00980, (EDTX, Nov. 2, 2024)	Pending
Intellectual Ventures I LLC et al v. Southwest Airlines Co., 7-24-cv-00277 (WDTX, Nov. 2, 2024)	Pending
Assurant, Inc. v. Intellectual Ventures I LLC et al., 1-24-cv-00344, (D. Del, Mar. 15, 2024) (*Declaratory Judgement Action*)	Pending
Intellectual Ventures I LLC et al v. JP Morgan Chase Bank, Nat’l Assoc., 2-23-cv-00523, (EDTX, Nov. 15, 2023)	Terminated Jan. 17, 2025
Intellectual Ventures I LLC et al v. Comerica Inc., 2-23-cv-00524 (EDTX, Nov. 15, 2023)	Terminated Jan. 17, 2025
Intellectual Ventures I LLC et al v. Liberty Mutual Holding Co. et al., 2-23-cv-00525 (EDTX Nov. 15, 2023)	Terminated Jan. 17, 2025

The '844 patent was also involved in one *Inter Partes* Review: *Liberty Mutual Tech. Group, Inc. et al v. Intellectual Ventures II LLC*, IPR2025-00202 (PTAB, Nov. 2024), Terminated April 24, 2025. As explained above and further addressed in Section VIII below, this Request is based on IPR2025-00202, filed after that IPR was terminated due to settlement of the parties.

VIII. DISCRETIONARY DENIAL IS NOT WARRANTED

Requester respectfully submits that discretionary denial under 35 U.S.C. § 325(d) is not warranted. This Request seeks cancellation of the Challenged Claims over the same prior art relied upon in IPR2025-00202—an IPR dismissed due to settlement prior to an institution decision. Applying the facts of this case to the *Becton, Dickenson* factors, the asserted art and arguments are admittedly similar to those in IPR2025-00202 (factors (a), (b), and (d)), but the art and arguments were never evaluated because IPR2025-00202 was dismissed before institution (factor (c)). *Advanced Bionics, LLC v. Med-El Electromedizinische Gerate GmbH*, IPR2019-01469, Paper 6, 7-11, n. 10 (citing *Becton, Dickenson* factors) (*precedential*). Accordingly, the Board erred in its evaluation of the art and arguments in IPR2025-00202 by the documented lack of consideration of these arguments (factor (e) IPR2025-00202, Paper 8 (Joint Motion to Terminate), Paper 9 (Decision (granting motion to terminate IPR2025-00202)). The evidence and facts presented herein show the clear unpatentability of Challenged Claims 1-27 of the '844 patent, which warrants their reconsideration in this Request (factor (f)).

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

While 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. Indeed, despite being "pertinent" to Applicants' disclosure, it was never mentioned again (even after Nguyen was discarded). This, at the same time a different Examiner, assigned the "grandparent" '622 Application, repeatedly and successfully cited Haun to support rejection of all similar claims, resulting in abandonment of that application.

Accordingly, the Board should not exercise discretion under § 325(d).

IX. CONCLUSION

Reexamination and cancellation of claims 1-27 of the '844 patent is respectfully requested. The Commissioner is hereby authorized to charge Deposit Account 50-6990 under Docket No. IVI844 the *Ex Parte* Reexamination fee of $13,545 under 37 C.F.R. § 1.20(c)(1). Unified believes no other fee is due with this submission, however, the Commissioner is hereby authorized to charge any fee deficiency or credit any over-payment to Deposit Account 50-6990.

Please direct all correspondence in this matter to the undersigned.

Respectfully submitted,

Dated: April 30, 2025

By: /*Jessica L.A. Marks*/

Jessica L.A. Marks (Reg. No. 67,451)
Unified Patents, LLC
4445 Willard Avenue, Suite 600
Chevy Chase, MD 20815
202.847.5260
jessica@unifiedpatents.com

Jason Wejnert (Reg. No.55,722)
Unified Patents, LLC
4445 Willard Avenue, Suite 600
Chevy Chase, MD 20815
jwejnert@unifiedpatents.com

Counsel for Requester Unified Patents, LLC