

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

THE PETITIONER
Cisco Systems, Inc.,

v.

THE PATENT OWNER
UMBRA Technologies Ltd.

U.S. Patent No. 11,146,632
Issue Date: October 12, 2021
Title: DATA BEACON PULSER(S) POWERED BY INFORMATION
SLINGSHOT

Inter Partes Review No.: IPR2024-00344

**PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO.
11,146,632 UNDER 35 U.S.C. §§ 311-319 and 37 C.F.R. §§ 42.1-.80, 42.100-
.107**

Mail Stop “PATENT BOARD”
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. OVERVIEW	1
III. GROUND FOR STANDING (37 C.F.R. § 42.104(A))	5
IV. REASONS FOR THE REQUESTED RELIEF	5
A. Summary of the '632 Patent.....	5
1. Overview of the '632 Patent	Error! Bookmark not defined.
B. Prosecution History	9
C. Claim Construction.....	11
D. Priority Date of the Challenged Claims	11
E. Person of Ordinary Skill in the Art	11
F. State of the Art	12
1. Parallel File Systems.....	12
2. Write and Read Queues	14
V. IDENTIFICATION OF CHALLENGES	17
A. Challenged Claims	17
B. Statutory Grounds for Challenges	17
VI. IDENTIFICATION OF HOW THE CHALLENGED CLAIMS ARE UNPATENTABLE	18
A. Challenge 1: Claims 1-7	18
1. The Agarwala Reference.....	18
2. Detailed Application of Agarwala to the Challenged Claims ..	28

B.	The Board Should Not Deny Institution Under 35 U.S.C. § 314(a).....	60
VIII.	Mandatory Notices Under 37 C.F.R. §42.8.....	63
B.	Real Party-in-Interest (37 C.F.R. § 42.8(b)(1)).....	63
C.	Related Matters (37 C.F.R. § 42.8(b)(2)).....	63
1.	Judicial Matters	63
2.	Related Patents.....	63
D.	Lead/Back-up Counsel (37 C.F.R. § 42.8(b)(3)):	64
E.	Notice of Service Information (37 C.F.R. § 42.8(b)(4)):.....	64
F.	GROUND FOR STANDING (37 C.F.R. § 42.204(A))	64
IX.	CONCLUSION	65

TABLE OF AUTHORITIES

PAGE(S)

CASES

<i>Sotera Wireless, Inc. v. Masimo Corp.</i> , IPR2020-01019.....	61
<i>HP Inc. v. Slingshot Printing LLC</i> , IPR2020-01084.....	62

Petitioner’s Exhibit List

<i>Exhibit #</i>	<i>Description</i>
1001	U.S. Patent No. 11,146,632 (“the ’632 Patent”)
1002	Prosecution history of the ’632 Patent (“File History”)
1003	Declaration of Petitioner’s Expert Dr. A.L. Narasimha Reddy (“Reddy”)
1004	U.S. Provisional Patent Application No. 61/739,685
1005	U.S. Patent No. 9,582,421 to Agarwala et al.
1006	UMBRA Technologies Ltd.’s First Infringement Analysis of Claims 1-4 & 7 of U.S. Patent No. 11,146,632, <i>UMBRA Technologies Ltd. (UK) et al v. Cisco Sys., Inc.</i> , E.D. Tex. Case No. 1-23-cv-00903, Dkt. 1-8 (Aug. 2, 2023)
1007	U.S. Patent Application Publication No. 2016/0100027 to Haghighi (“’027 Publication”)
1008	U.S. Patent Application Publication No. 2011/0145367 to Ananthanarayanan et al. (“’367 Publication”)
1009	Kassick et al., “Dynamic I/O Reconfiguration for NFS-Based Parallel File System.” In 2011 19 th International Euromicro International Conference on Parallel, Distributed and Network-Based Processing (IEEE, 2011)
1010	Li et al., “Small Files Problem in Parallel File System.” In 2011 International Conference on Network Computing and Information Security (NCIS) (IEEE, 2011)
1011	Marc Farley, <i>Storage Networking Fundamentals: An Introduction to Storage Devices, Subsystems, Applications, Management, and File Systems</i> (Cisco Press 2005)
1012	James O’Reilly, <i>Network Storage: Tools and Technologies for Storing Your Company’s Data</i> (Elsevier Inc. 2017)

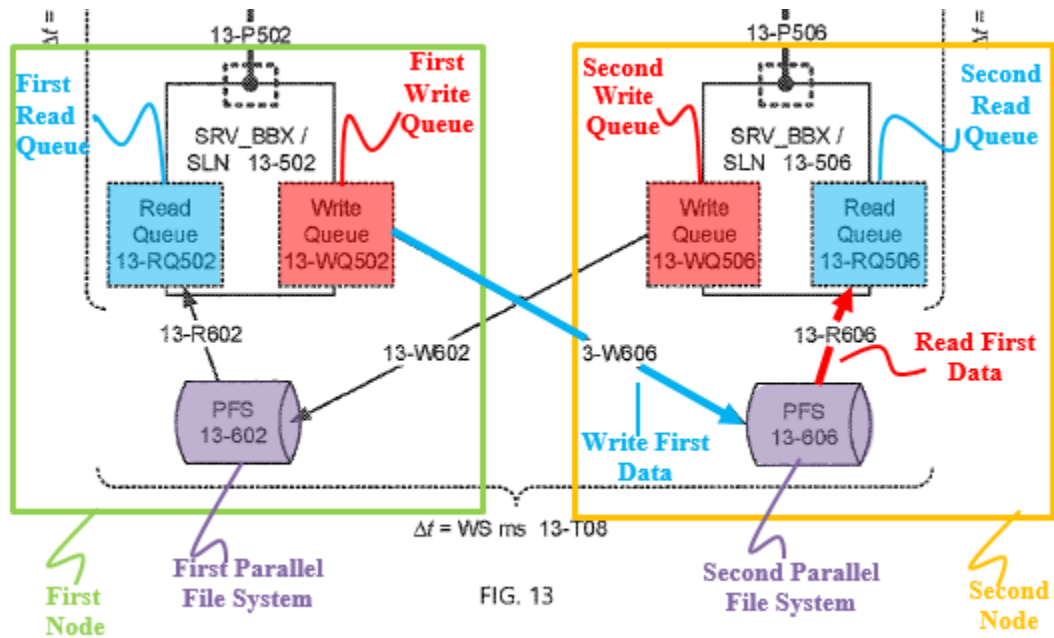
I. INTRODUCTION

Pursuant to 35 U.S.C. §§ 311 *et seq.* and 37 C.F.R. §§ 42.1 *et seq.*, Cisco Systems, Inc. (“Petitioner”) hereby petitions for an *inter partes* review of U.S. Patent No. 11,146,632 (“’632 Patent”). Petitioner respectfully submits that Claims 1-7 (the “Challenged Claims”) of the ’632 Patent are unpatentable under 35 U.S.C. §103 in view of the prior art references discussed herein. This Petition demonstrates by a preponderance of the evidence that there is a reasonable likelihood that Petitioner will prevail with respect to at least one of these claims. Accordingly, it is respectfully requested that the Board institute an *inter partes* review of the ’632 Patent pursuant to 37 C.F.R. § 42.108.

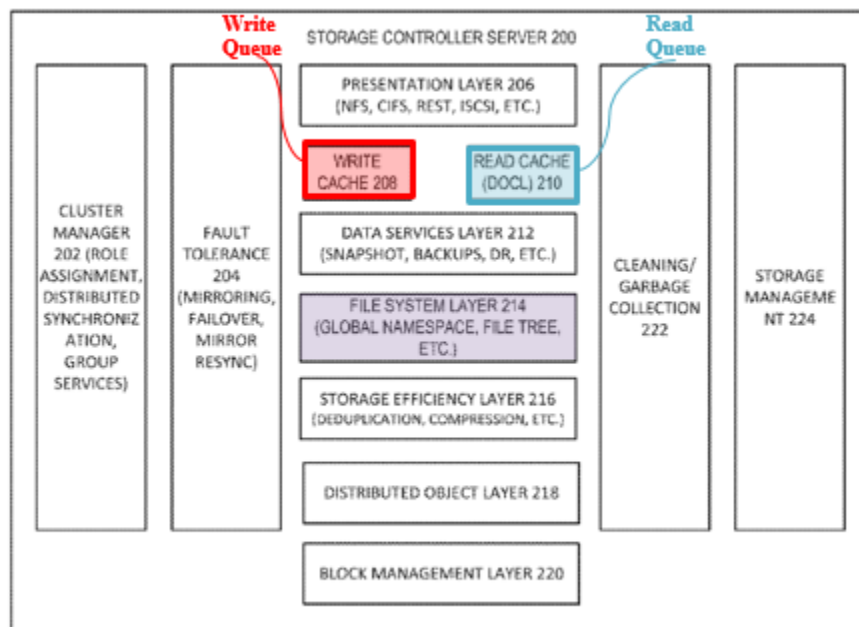
II. OVERVIEW

The Challenged Claims are unpatentable as obvious over the prior art. The claims are directed to a distributed storage system, with each node having a read/write queue and a parallel file system. The read/write queue of a first node writes data from the write queue to a parallel file system of a second node, and the second node reads the data from the parallel file system of the second node and places it in the read queue of the second node. This claimed functionality already existed in the prior art. For example, the prior art Agarwala reference includes a parallel disclosure to that of the ’632 Patent in that it is directed to the same problem and provided the same solution as the ’632 Patent, the only difference

being that Agarwala uses different words to described the identical concepts described and claim in the '632 Patent. As illustrated below, the architecture shown in FIG. 13 of the '632 Patent is substantially identical to the architecture shown in FIG. 2 of Agarwala with common elements being colored with the same color. The read queue (blue) and write queue (red) in the '632 Patent (FIG. 13) is called the read cache (blue) and write cache (red) in Agarwala (FIG. 2). As described in more detail below, a POSITA understood that the write and read caches of Agarwala are the equivalent of and perform the same functions as the read and write queues of the '632 Patent. Agarwala further describes that the read cache 210 (blue) and the write cache 209 (red) of FIG. 2 are provided in the high speed storage (red/blue) of FIG. 13. *EX-1005* ¶0081 (e.g., “[t]he synchronous replicate module 306A receives the write 304A and stores the contents of the write in the high-speed storage 312A”). The parallel file system (PFS) (purple) in FIG 13 of the '632 Patent is called the file system layer (purple) in FIG. 2 and persistent storage (also purple) in FIG. 3 of Agarwala (FIG. 13):



EX-1001 FIG 13.



EX-1005 FIG. 2; EX-1004 FIG. 2.

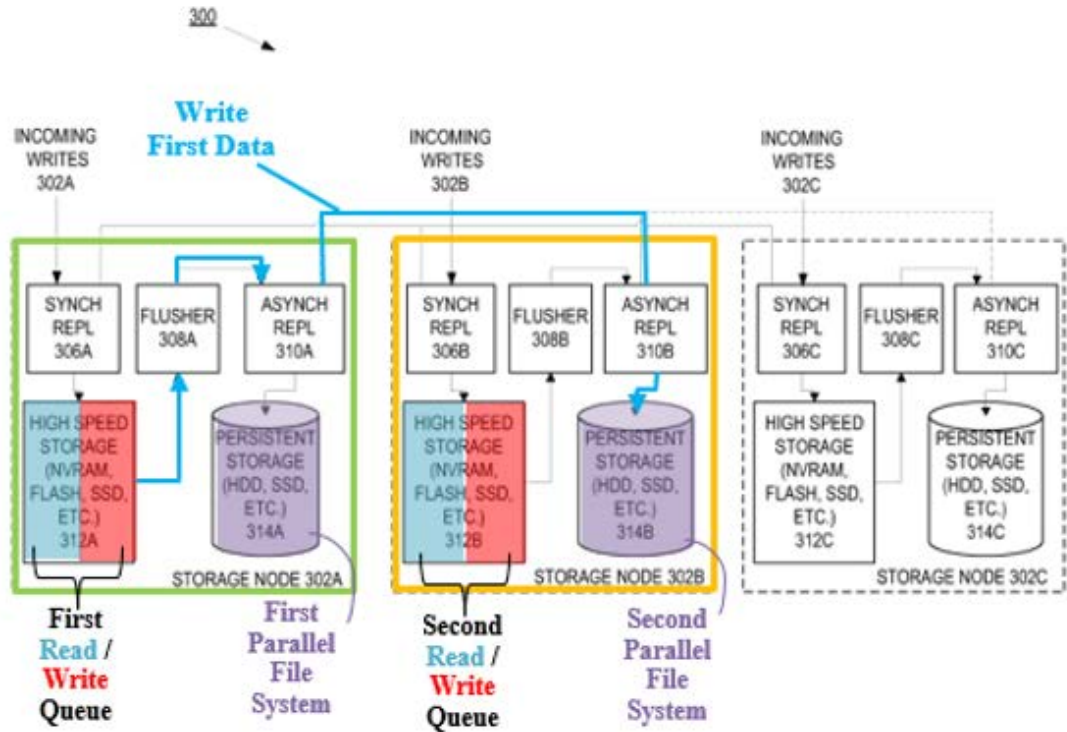


FIGURE 3

EX-1004 FIG. 3.

Just like recited in Claim 1 of the '632 Patent, the read/write cache (*read/write queue*) of the high speed storage of a first node in Agarwala writes data from the write cache (*write queue*) to a persistent storage (*parallel file system*) of a second node, and the second node reads the data from the persistent storage (*parallel file system*) of the second node and places it in read cache (*read queue*) in high speed storage of the second node. Because there are no material differences between the architectures disclosed in the '632 Patent and Agarwala, the Challenged Claims are unpatentable over Agarwala.

III. GROUNDS FOR STANDING (37 C.F.R. § 42.104(A))

Petitioner certifies that the '632 Patent is available for *inter partes* review and that Petitioner is not barred or estopped from requesting *inter partes* review of the Challenged Claims on the grounds identified herein. 37 C.F.R. § 42.104(a). This Petition is filed pursuant to 37 C.F.R. § 42.106(a).

IV. REASONS FOR THE REQUESTED RELIEF

As explained in §§ II and VI-VIII of this Petition and in the attached Declaration of Petitioner's Expert, Dr. A.L. Narasimha Reddy ("Reddy"), *EX-1003*, the network system for providing data beacons, as described and claimed in the '632 Patent, was obvious over the prior art to a person of ordinary skill in the art ("POSITA") at the time of the invention.

A. Summary of the '632 Patent

The '632 Patent is titled "Data Beacon Pulser(s) Powered by Information Slingshot" and its disclosure "relates to networks, and more particularly, to the topology, configuration and operation of a data beacon pulser (DBP)":

The present disclosure relates generally to networks, and more particularly, to the topology, configuration and operation of a data beacon pulser (DBP). A DBP offers fast, efficient, and dependable one-way casting/multi-casting of information globally. A DBP can be utilized for transmission of financial data, news feeds, seismic data, and many other applications where dependable and

accurate near wire speed dissemination of rapidly changing information is time critical.

EX. 1001 1:29-37.

However, as discussed below in §§IV.A, B and V.A, the claims of the '632 Patent are not directed to a “data beacon pulser”-specific invention, but instead claim a generic distributed network storage node topology that was well known in the art before the filing of the '632 Patent. FIG. 13 of the '632 Patent “illustrates one example of the Data Beacon Pulser (DBP) mechanism framework and flow ... and demonstrates how DBP can utilize ‘Slingshot’ to make information from a source region 13-310 available to a client 13-100 in another region in as timely a fashion as possible”, but the topology of the Challenged Claims is related only to what is illustrated and described in connection with the annotated bottom portion of FIG. 13 (below). *EX-1001* 16:8-13; *also* 13:21-26 (describing FIG. 7); *EX-1003*

¶ 28

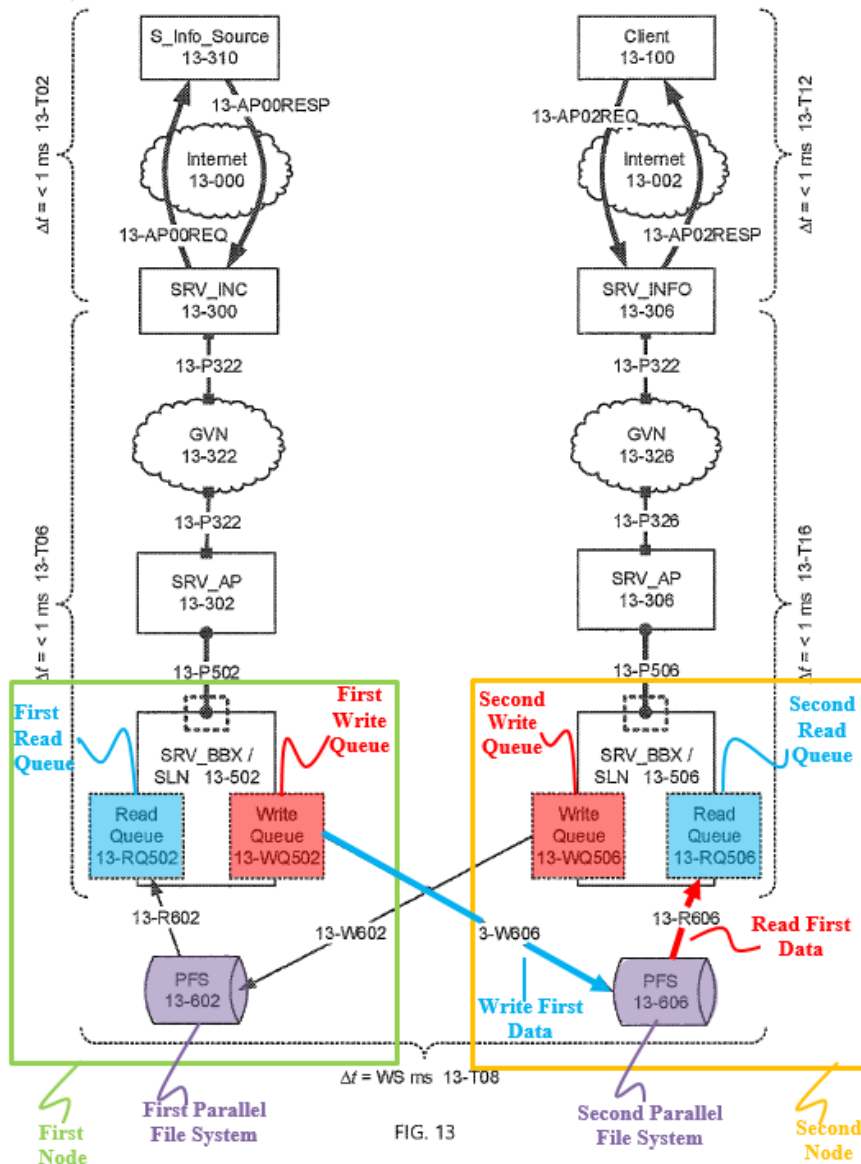


FIG. 13

FIG. 13 shows the components that are recited in the Challenged Claims, e.g., claim 1. A first node includes a first Read Queue, a first Write Queue and a first parallel file system (PFS) device, and a second node includes a second Read Queue, a second Write Queue and a second parallel file system (PFS) device. *EX-1001* 16:23-34; 13:27-43; 16:4-5. Data is written from the first Write Queue of the

first node to the second PFS of the second node, and that data is read from the second PFS of the second node to the Read Queue of the second node. *Id*; *EX-1003* ¶ 28.

The second PFS (PFS 606) is described in the specification as “a parallel file system device in the remote region 606”. *EX-1001* 13:27-46. Additionally, all the parallel file systems are referred to as “PFS devices.” *EX-1001* FIG. 12, 16:4-5 (“In practical use, all connected nodes can concurrently connect with PFS devices in all other regions and locations”). *EX-1003* ¶ 29.

The ’632 Patent provides no details regarding the parallel file system used in its system and does not purport to offer any improvements to parallel file systems, which were known in the art well before the ’632 Patent as described below in §IV.F. *EX-1003* ¶ 30.

Likewise, the ’632 Patent provides no details regarding the Write Queues and Read Queues used in its system—except that the queues hold data—and does not purport to offer any improvements to write and read queues, which were also known in the art well before the ’632 Patent as described below in §IV.F. *EX-1003* ¶ 31.

The Challenged Claims claim only a generic distributed storage topology including first and second nodes that was well-known in the art and was commonly used long before the '632 Patent. Claim 1 is the only independent claim and recites:

A network system for providing data beacons,
comprising:

a first node comprising a first read queue, a first write queue, and a first parallel file system;

a second node comprising a second read queue, a second write queue, and a second parallel file system;

wherein the first node writes first data from the first write queue to the second parallel file system; and

wherein the second node reads the first data from the second parallel file system and places the first data in the second read queue.

B. Prosecution History

The '632 Patent issued from U.S. Application No. 16/095,908 (the '908 Application), filed on October 23, 2018 as a U.S. national stage application of International Application No. PCT/IB2017/000580, filed April 26, 2017. The '908 Application included claims 1-7 at filing, including one independent claim. *EX-1002* p. 2. The application was allowed in a first action allowance, with the

Examiner acknowledging the claimed “non-complex subject matter” but nonetheless focusing on the unclaimed—but disclosed in the specification—concept of “data beacon *pulser*” as lacking in the art considered by the Examiner:

The application filed on October 23, 2018 is eligible for a First Action Allowance due to the novelty of the claimed subject matter... Although independent *Claim 1 appears at first to be somewhat non-complex subject matter*, it was found to be novel in the art...

In addition, a search for the term “data beacon pulser,” which is *disclosed in the specification* as the technology upon which the invention is based, was not found in any prior art other than that of the inventor. A broader search for “data beacon” in the same paragraph as variations of the term “pulse” yielded the reference, Crinon et al. (US 20120196646 A1) and related referenced by the same inventors. However, those references deal with mobile networks, and do not disclose any information about read queues, write queues, or parallel file systems. Examiner also reviewed the prior art references provided as being relevant in the PCT International Search Report, but did not find that the references teach the claimed subject matter with the required level of specificity.

EX-1002 pp. 9-10 (emphasis added).

C. Claim Construction

Petitioner proposes that each claim term in the Challenged Claims be given its plain and ordinary meaning in this proceeding, and that no specific construction of any claim term is required because the prior art relied on in this Petition meets each of the claim terms under any reasonable construction.

It is noted that this interpretation is only applicable to the *inter partes* review sought herein and should not be construed as constituting, in whole or in part, the Petitioner's own interpretation of any claims for any other purposes, including any litigation. Accordingly, Petitioner expressly reserves the right to present an interpretation of a claim term in other proceedings, which is different, in whole or in part, from that presented in this Petition.

D. Priority Date of the Challenged Claims

The '632 Patent claims priority to the following provisional applications each filed April 26, 2016: 62/327,907, 62/327,846, and 62/327,911. Patent Owner bears the burden of establishing priority to the provisional applications.

The prior art cited below predates the earliest possible priority date, April 26, 2016, and thus for purposes of this Petition, this issue is moot.

E. Person of Ordinary Skill in the Art

With respect to the '632 Patent, a POSITA in 2016 would have been familiar with network communications and distributed storage systems, including network file systems. *EX-1003* ¶ 37. A POSITA would have a working knowledge of

techniques for accessing data (reading and writing data) in distributed network storage systems, including parallel files systems. A POSITA would have gained knowledge of these concepts through a mixture of training and work experience, such as by having a Bachelor's degree in computer science and four years of experience; or by obtaining a Master's degree in computer science, but having only one to two years of experience; or by having no formal education but experience in storage systems of at least eight years. *EX-1003* ¶ 37.

F. State of the Art

The following section describes the state of the art for '632 Patent as of April 2016. *EX-1003* ¶ 39. The prior art references, and the discussions of what was known to a POSITA, provide the factual support for the general description of the state of the art at the time of the invention, provide additional motivation to modify or combine the reference(s). Accordingly, these references should be considered by the Board.

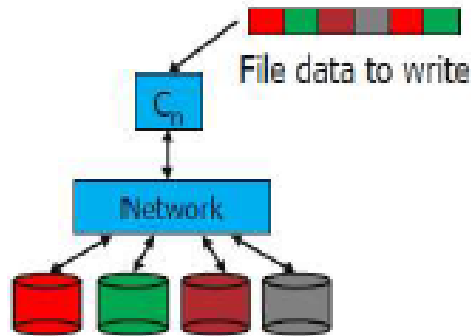
1. Parallel File Systems

File systems have two key roles, including organizing and maintaining the namespace (i.e., directory hierarchy and file names) and storing the contents of files and providing an interface through which stored files can be read and data can be written. Local file systems are used by a client with direct access to the disk,

and networked file systems (NFSs) provide access to clients that might not have direct access to the disk. *EX-1003* ¶ 40; *EX-1010* p. 228; *EX-1011* p. 31.

A parallel file system (PFS) is a file system designed to store data across multiple networked servers and to facilitate high-performance access through simultaneous, coordinated input/output operations between clients and storage nodes. The PFS breaks up a data set into blocks and distributes, or “stripes”, the blocks to multiple storage drives, which can be located in local and/or remote servers. A global namespace is used to facilitate data access. Reads and writes of data to distributed storage devices can be done through multiple I/O paths concurrently, which provides a significant performance benefit. Capacity and bandwidth can be scaled to accommodate large quantities of data. One such well-known parallel file system is Lustre (www.lustre.org), which was created back in 1999. Another well-known parallel file system is the Parallel Virtual File System (PVFS), which was developed in 1993 and changed its name to OrangeFS (OrangeFS.org) as of its 2011 release. *EX-1003* ¶¶ 41-43; *EX-1010* pp. 227-28; *EX-1009* p. 11; *EX-1012* pp. 142-43, 146.

As noted above, striping is the basic mechanism used in parallel files systems to improve performance where a file is split into fixed-size blocks that are written to separate disks in order to facilitate parallel access, as illustrated below.



Principally, this technique allows multiple servers, disks and network links to be leveraged during concurrent I/O operations. This eliminates bottlenecks and can improve performance. *EX-1003* ¶ 44; *EX-1008* ¶ 21 (describing General Parallel Files System (GPFS) where files are striped across disks in the file system, balancing loading on the disks and improving throughput by reading and writing blocks in parallel); *EX-1010* p. 228; *EX-1012* pp. 143.

2. Write and Read Queues

Storage systems have long used write and read queues for holding data being written to and read from file systems. Write queues are used as a staging area for data that is to be written into storage. Read queues are used to hold data read from storage for access by applications and clients, for example, and to allow for speedy access to the read data and for subsequent read requests. *EX-1003* ¶ 45.

U.S. Patent Application Publication No. 2016/0100027 to Haghighi (“’027 Publication”) (*EX-1007*) discloses a storage system that includes local servers 105 and remote servers 110 connected by a network 130. Each server includes a

storage device 185, 190 (e.g., solid state devices or drives (SSDs) or hard disk drives (HDDs)) and memory 165, 170, including queues 175, 180 for accessing (reading/writing) data on the storage devices 185, 190. *EX-1007* ¶¶ 0023, 0028, 0029. Information can be structured, stored, and accessed in the storage system via files, blocks, logical block address (LBA), logical unit number (LUN), key-value pairs, objects, or the like. *EX-1007* ¶ 24, 27.

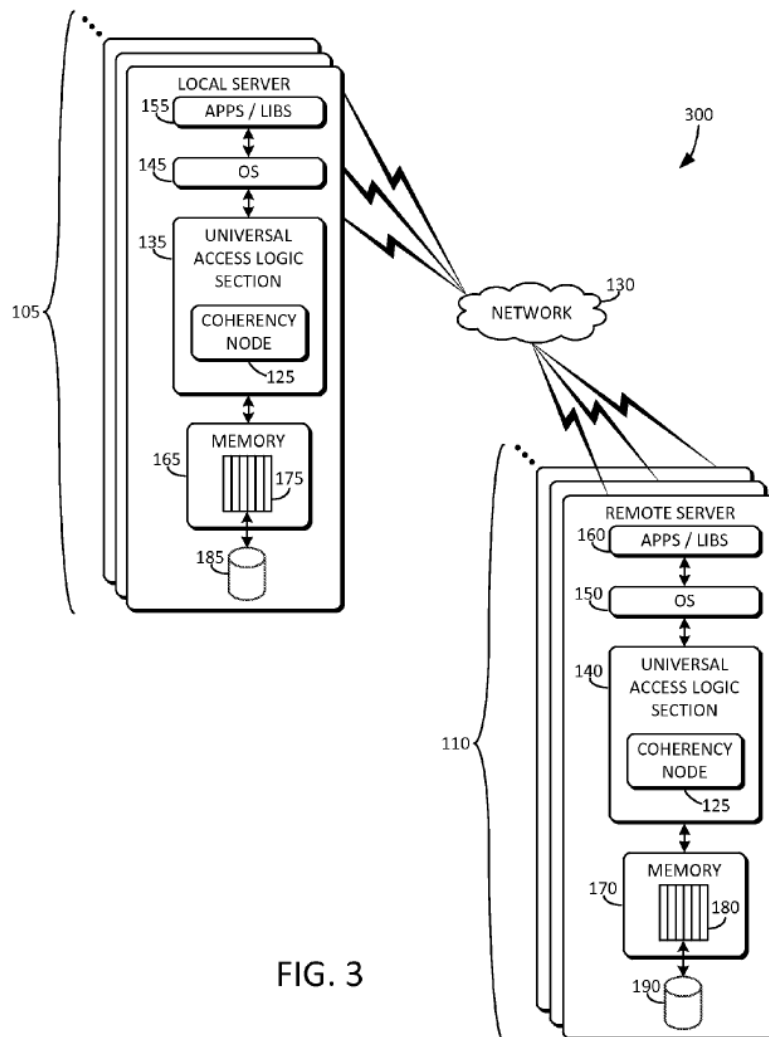


FIG. 3

Each local and remote server in the system can read data stored on any one of the other individual servers, and each server can write data to storage devices of any other server in the system. *EX-1007* ¶ 0036.

Each of the servers has write data queues that queue local write data to be written to the local storage devices and incoming data queues that queue data received from a remote server for writing to the local storage device. *EX-1007* ¶¶ 0040, 00041.

Each of the servers also includes queues for data that is being read from storage, both for local and remote applications. *EX-1007* ¶¶ 0040, 0043.

For example, with reference to local server 105 shown in FIG. 6 (below), the '027 Publication expressly discloses that each local memory array 625 includes multiple storage queues 621, including (i) multiple local write storage queues (e.g., 626, 628, and 622) and (ii) multiple local read storage queues (e.g., 627, 629, and 623). *EX-1007* ¶ 0044, 0049. Network interface card (NIC) 645 of local server 105 also includes outgoing data queues 677 for sending write data to remote servers 110 for writing to remote storage devices 640, and incoming data queues 667 for receiving remote read data from remote servers 110. *EX-1007* ¶ 0050.

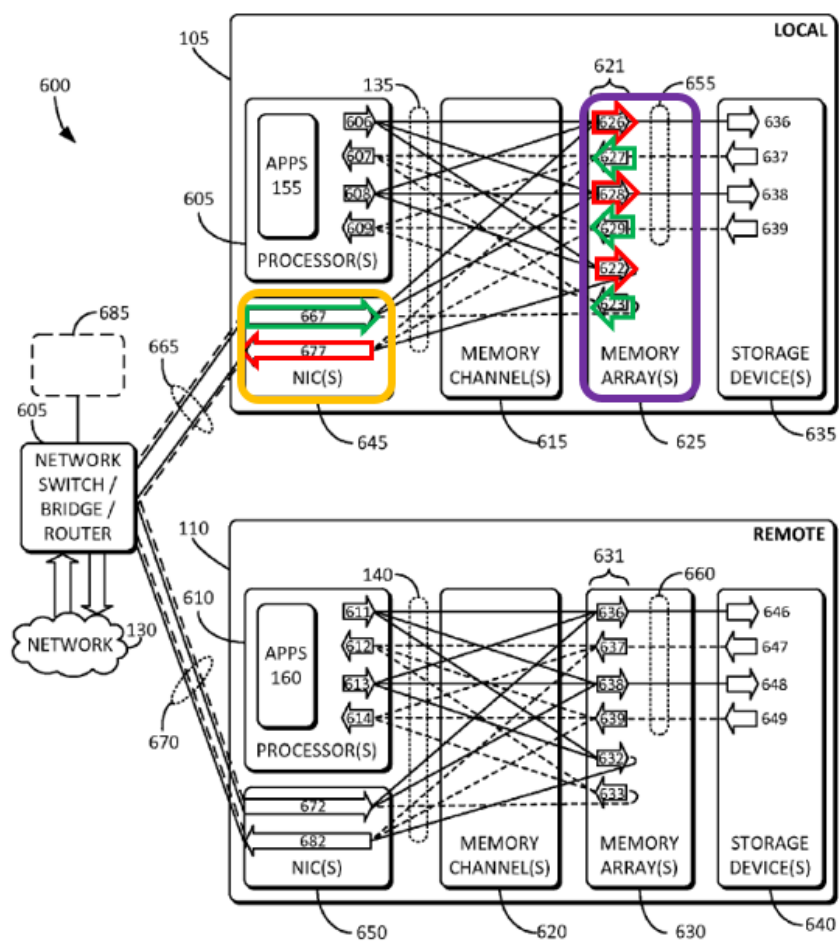


FIG. 6

V. IDENTIFICATION OF CHALLENGES

A. Challenged Claims

Claims 1-7 of the '632 Patent are challenged in this Petition.

B. Statutory Grounds for Challenges

The Challenges are set forth in detail below and summarized as follows:

Ground	Claims	Basis	Reference
1	1-7	§ 103	Agarwala in view of Knowledge of a POSITA

Ground 1: Claims 1-7 of the '632 Patent are invalid as being obvious over Agarwala.

U.S. Patent No. 9,582,421 to Agarwala et al. is titled “Distributed Multi-Level Caching For Storage Appliances” and issued on February 28, 2017 from an application filed on December 19, 2013. *EX-1005*. The '421 Patent claims priority to and incorporates by reference U.S. Provisional Application 61/739,685 (*EX-1004*) filed December 19, 2012. *EX-1005* 1:6-8. The '421 Patent is prior art to the '632 Patent under §102(a)(2)(AIA). Together the '421 Patent and the incorporated '685 Provisional are referred to herein as “Agarwala”. Agarwala was not cited or applied by the examiner during prosecution of the '632 Patent.

VI. IDENTIFICATION OF HOW THE CHALLENGED CLAIMS ARE UNPATENTABLE

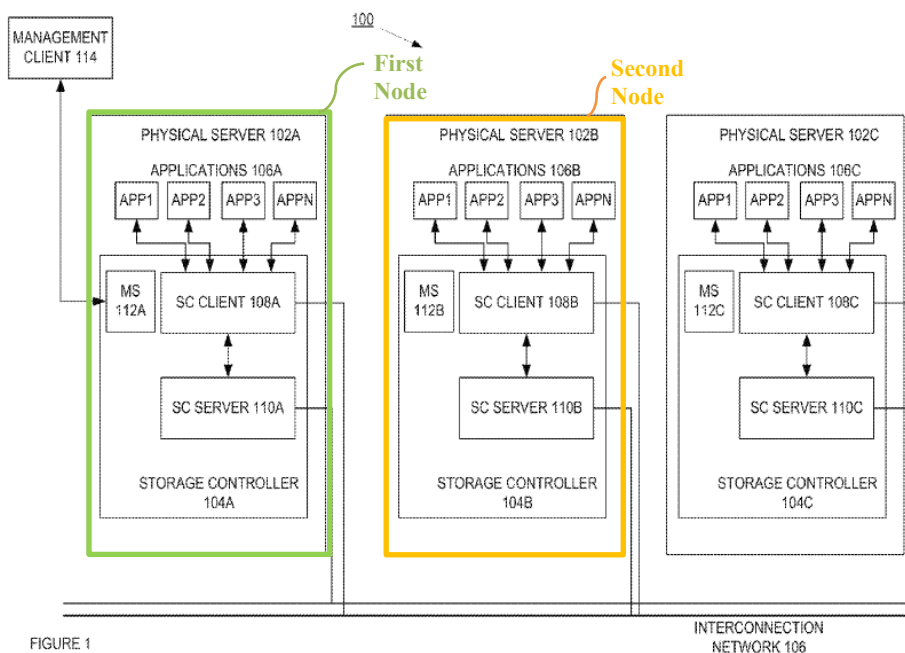
A. Challenge 1: Claims 1-7

1. The Agarwala Reference

Agarwala's Storage Nodes

Agarwala's disclosure relates to “distributed multi-layer caching in a distributed storage system.” *EX-1005* 1:11-13; *EX-1004* ¶0001 (“This invention relates generally to a storage system and more particularly to software defined distributed storage system.”). Agarwala refers to the storage system as “StorFS,” and FIG. 1 illustrates the StorFS system including physical servers 102A-C (which

Agarwala also calls “storage nodes 102A-C”) interconnected by network 106. *EX-1005* 3:59-62; *EX-1004* ¶¶ 0013, 0047.



EX-1005 FIG. 1; *EX-1004* FIG. 1. A first storage node (e.g., storage node 102A) and a second storage node (e.g., storage node 102B) map to the *first node* and *second node* of the Challenged Claims.

Each storage node 102 includes a storage controller server 110 that manages the underlying storage (e.g., hard disk drive (HDD), solid state devices (SSD), PCIe flash, etc.), where the storage controllers collectively provide storage functionality to storage controller (SC) clients 108. *EX-1005* 3:67-4:6; *EX-1004* ¶¶ 0047. Each SC client 108 processes I/O requests from applications that are local

to its physical storage node and routes them to the appropriate local or remote SC server 110 for processing. *EX-1005* 4:7-12; *EX-1004* ¶0047.

Agarwala's Write and Read Queues

Figure 2 is a more detailed illustration of a storage controller/storage controller server 200. *EX-1005* 5:58-59; *EX-1004* ¶0052.

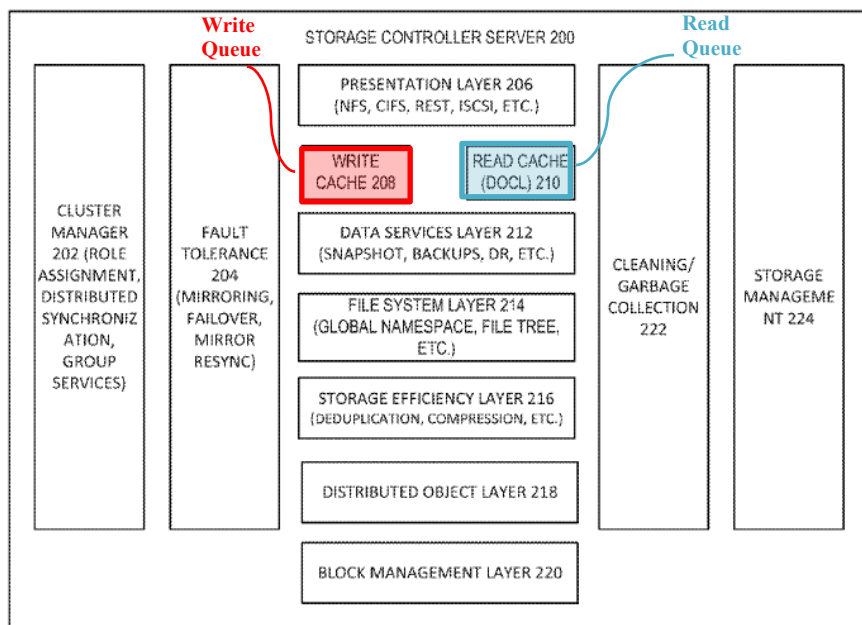


FIGURE 2

EX-1005 FIG. 2; *EX-1004* FIG. 2. Each storage controller includes a write cache 208 and a read cache (DOCL) 210, which map to the write and read queues, respectively, of the Challenged Claims.

[T]he Write Cache 208 and the Read Cache 210 Layers provide acceleration for write and read I/O respectively using fast storage devices. In one embodiment, the Write Cache Layer 208 includes the write log as described below. In one embodiment, the Read Cache

Layer 210 includes a deduplicated object cache layer (DOCL) that is used to cache recently added and accessed objects for the StorFS system.

EX-1005 6:10-17; *EX-1004* ¶0052.

A POSITA understood that Agarwala's read cache is a *read queue*. *EX-1003* ¶¶ 56, 71. First, objects are initially read from persistent storage (e.g., SSD 412 or the HDD 410 of FIG. 4) and added to the read cache where they are held for reading. *EX-1005* 8:27-28, 8:27-55; *EX-1004* ¶¶0061-0062. Second, in this read cache, items can be moved through and out of the read cache, as in a queue. For example, cached items can be promoted to the Most Recently Used (MRU) list based on an initial read of an object and the Most Frequently Used (MFU) list depending on frequency of access, as well as be evicted to create room for new contents:

FIG. 3B is a block diagram of one embodiment of a DOCL layer 1 (L1) cache 350. In one embodiment, the L1 cache 350 includes two types of list: Recency-based eviction lists 352; and frequency-based eviction lists 354... The recency-based eviction lists 352 are used to track objects being added to the cache and include an eviction list for data objects 360 and metadata objects 358... As the objects are read, they are initially promoted to the recency list (e.g., a Most Recently Used (MRU) list). In one embodiment, the frequency-based

eviction lists are used to track accesses to objects in the cache and include an eviction list for data objects 364 and metadata objects 362. If this object is accessed within a certain time period again, the object is promoted to the frequency list (e.g., a Most Frequently Used (MFU) list.)... In one embodiment, a hotness/coldness of an object measures how recently the object was added or accessed by the StorFS system. In this embodiment, an object that has not been recently added or accessed can be a candidate for eviction from the L1 cache. In one embodiment, an object that has not been recently added or accessed can be considered “cold” or “stale” and is an object that is one that can be evicted from the L1 cache. In one embodiment, eviction from and updates to the L1 cache is at the object granularity.

EX-1005 7:5-47; *EX-1004* ¶0058.

A POSITA would likewise understand that Agarwala’s write cache is a *write queue*. *EX-1003* ¶¶ 58, 78. Agarwala’s write cache includes a write log for holding or queuing data to be written to the persistent storage at a later time. *EX-1005* 6:12-14; *EX-1004* ¶¶ 0004, 00006, claim 16 (“if the active log partition is full after the writing of the data, transferring the data to persistent storage”). When the storage control server receives a write storage request, it first “stores this data in a write log by appending the data at the end of the write log. This can lead to a fast write for the data that allows a quick write storage fulfillment for the requesting

application.” *EX-1004 ¶0041*. When the write log is full, the SC server “flushes the write log to persistent storage.” *EX-1004 ¶0041*.

Data is also written from the write log of Agarwala’s write cache in one storage node to other storage nodes both synchronously and asynchronously to “efficiently replicate data across the cluster”. *EX-1004 ¶¶0041, 0043, 0059*. With reference to FIG. 3 (below):

Figure 3 is an illustration of one embodiment of an efficient distributed replication. In Figure 3, the system 300 uses a combination of synchronous and asynchronous techniques to efficiently replicate data across the cluster. In one embodiment, storage nodes of SC servers 302A-C can each receive an incoming client write 304A-C. In one embodiment, the storage node 302A receives the incoming client write with synchronous replicate module 306A. The synchronous replicate module 306A receives the write 304A and stores the contents of the write in the high-speed storage 312A (e.g., NVRAM, Flash, SSD, etc.). In one embodiment, the synchronous replicate module 306A stores the write contents at the backend of the high-speed storage 312A as described above. A flusher module 308A flushes the contents of the write to the asynchronous replication module 310A...

In one embodiment, the asynchronous replication module 310A stores the content to persistent storage 314A. In this embodiment, the asynchronous replication module 310A can additionally replicate this content to the other storage nodes 302B-C via the asynchronous replication modules 310B-C, respectively. These receiving asynchronous replication modules 310B-C, each store the content in the corresponding local perspective storage 314B-C, respectively.

Ex-1004 ¶¶0081-0082.

FIG. 3, annotated below and as discussed above, illustrates data from the first node (green) written to the second node's (orange) parallel file system (purple):

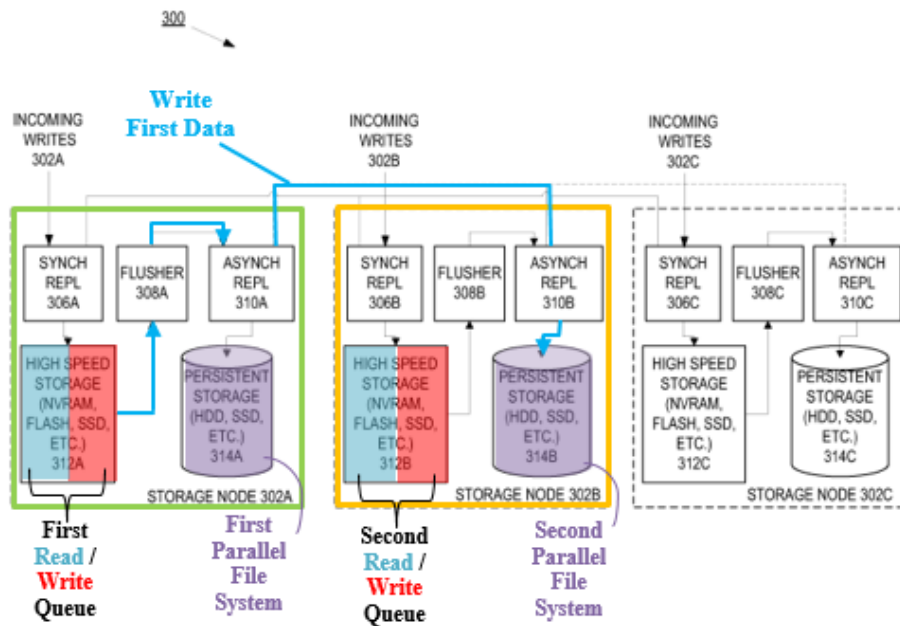


FIGURE 3

EX-1004 FIG. 3.

Agarwala's Parallel File System

A POSITA would also understand that Agarwala's storage system implements a *parallel file system*, including in its file system and persistent storage. EX-1003 ¶60.

First, Agarwala's storage controller (SC) servers "collectively" provide storage for the storage clients across multiple distributed nodes:

The SC servers 110A-C manage their underlying storage (e.g., Hard disk drive (HDD), Solid state drives (SSD), PCIe flash, etc.) and collectively provide reliable and unified storage functionality to the SC clients. The SC

client 108A-C processes input/output (I/O) requests from the applications that are local to its physical storage node and routes them to the appropriate SC servers for processing.

EX-1005 4:3-9; *EX-1004* ¶0047. Across the multiple distributed nodes, the workload is partitioned, distributed, and parallelized. *EX-1005* 5:13-27, 5:39-41; *EX-1004* ¶¶ 0050-0051.

Second, as described in connection with FIG. 2 (above), the storage controllers of Agarwala's StorFS system implement a file system, including a global distributed object namespace and logical global namespace:

In one embodiment, the Distributed Object Layer 218 of the storage control server uses an API of the Block Management Layer 220 to provide a global distributed object namespace that is accessible over the network...

In one embodiment, the File System Layer 214 provides a logical global namespace abstraction to organize and locate data in the cluster... In one embodiment, the Presentation Layer 206 provides an interface to access the StorFS storage using well-known standard protocols like NFS, CIFS, REST, iSCSI, etc.

EX-1005 5:63-6:21; *EX-1004* ¶0052.

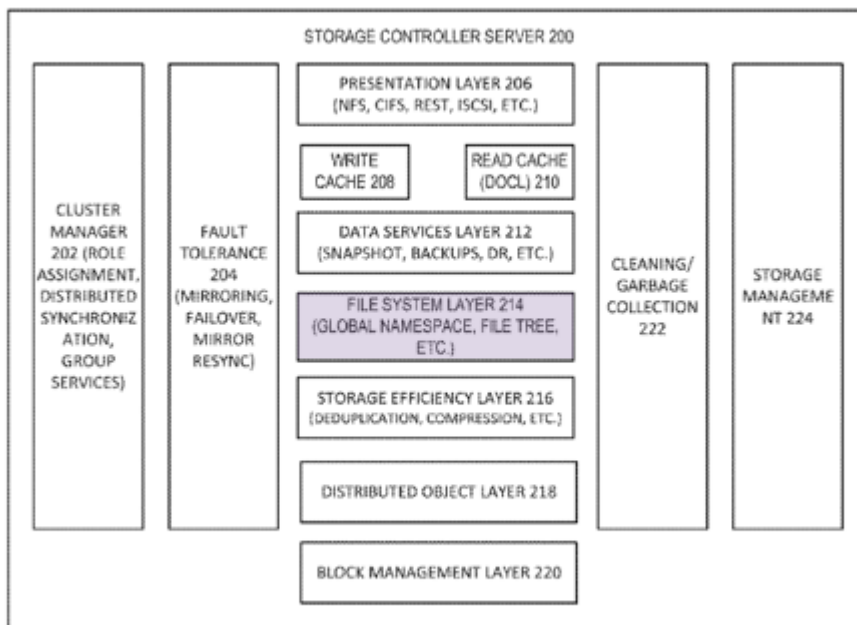


FIGURE 2

“[T]he StorFS system is capable of supporting different policies at different levels of granularity: at the file system level, at the individual file level, etc.” *EX-1004* ¶¶0097, 0098.

Third, as is characteristic of a parallel file system, “the StorFS system stripes file and file system data across multiple storage nodes in our cluster.” *EX-1004* ¶122. “[E]ach logical file (or entity) is partitioned into equal sized ‘stripe units’ [where] [t]he location of a stripe unit is determined based on a mathematical placement function.” *EX-1005* 4:30-33; *EX-1004* ¶¶0040, 0048. Further, the workload is distributed and parallelized across multiple storage nodes, as is also the characteristic of a parallel file system:

In one embodiment, the “Stripe_Unit_Per_Stripe” attribute determines the number of total virtual nodes that an entity can be spread across. This enables distributing and parallelizing the workload across multiple storage nodes (e.g., multiple SC servers 110A-C)...In one embodiment, the StorFS system 100 uses a two-level indexing scheme that maps the logical address (e.g. offset within a file or an object) to a virtual block address (VBA) and from the VBAs to physical block address (PBA). In one embodiment, the VBAs are prefixed by the ID of the vNode in which they are stored. This vNode identifier (ID) is used by the SC client and other StorFS system 100 components to route the I/O to the correct cluster node. The physical location on the disk is determined based on the second index, which is local to a physical node.

EX-1005 5:13-27, 5:39-41; *EX-1004* ¶¶ 0050-0051.

2. Detailed Application of Agarwala to the Challenged Claims

a. Claim 1

[1.0] A network system for providing data beacons, comprising:

To the extent the preamble is limiting, Agarwala discloses this limitation.

EX-1004; EX-1005. Agarwala discloses “the StorFS system is a storage system that includes a plurality of storage nodes coupled together by an interconnection

network” (*a network system*). *EX-1004* ¶0039; *EX-1005* 3:59-62. With respect to *providing data beacons*, the body of claim 1 does not refer to “data beacons” and instead only recites that “first data” is written (e.g., [1.3] *from the first write queue to the second parallel file system*) and read (e.g., [1.4] *from the second parallel file system to the second read queue*). As such, “data” that is written or read across nodes is an example of a “data beacon”, and Agarwala discloses that its StorFS system writes and read data across nodes (*providing data beacons*). For example:

In one embodiment, the input/out request can be a request to read content from the StorFS system, write content to the StorFS system, and/or request status of content stored in the StorFS system.

EX-1004 ¶0053, *EX-1005* 4:50-53. Data is mirrored from the write log in the write cache of one node across the network to other nodes for storage in persistent storage. *EX-1004* ¶0043 (describing asynchronous replication across nodes).

Further, a POSITA understood that data of different types can be stored in Agarwala’s StorFS system based on the applications. For example, the ’027 Publication discloses that “information can be structured, stored, and accessed in the storage system via files, blocks, logical block address (LBA), logical unit number (LUN), key-value pairs, objects, or the like.” *EX-1007* ¶24, 27, *EX-1003* ¶65.

That data that is written or read across nodes is an example of a “data beacon” is also consistent with Plaintiff’s assertions in the Related Litigation. *See EX-1006* p. 3 (“The HX Data Platform controller handles all read and write requests for volumes that the hypervisor accesses and thus intermediates all I/O from the virtual machines and containers....’, i.e., Cisco HyperFlex HX-Series nodes in a Cisco HyperFlex System are a network system for providing data beacons.”), p. 9 (“[Node2] commits the write to the active local write log on the caching SSD [A1] and sends duplicate copies to other nodes, e.g., ‘[A2] on Node 1’, i.e. the first node writes first data from the first write queue to the second parallel file system”).¹

Therefore, a POSITA understood that Agarwala’s StorFS system is *a network system for providing data beacons*.

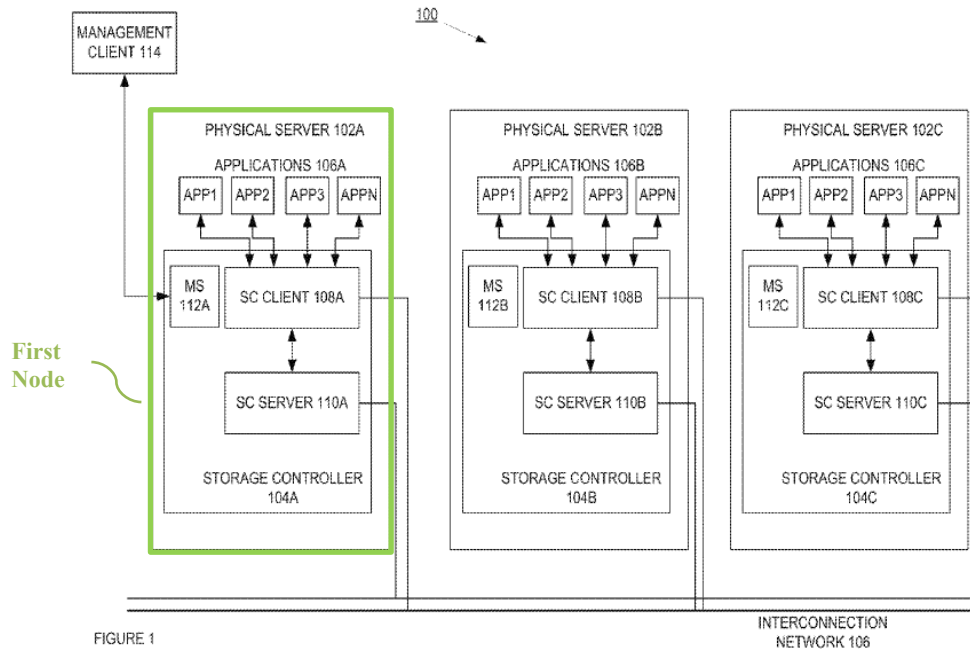
[1.1] a first node comprising a first read queue, a first write queue, and a first parallel file system;

First Node

Agarwala discloses that its storage system—the “StorFS system”—includes storage nodes 102A-C coupled by an interconnection network 116 (a first storage

¹ Citations to *EX-1006* are to the internal page number of the chart attached to the Exhibit. For example, *EX-1006* p. 3 is a citation to *EX-1006* p. 3 of 31 as reflected in the footer of the exhibit.

node of which is a *first node*). *EX-1005* 3:59-62. FIG. 1 illustrates the StorFS system:



EX-1005 3:59-62, FIG. 1; *EX-1004* ¶¶0013, 0047, FIG. 1.

Each storage node contains a storage controller server (110A – C in FIG. 1 above) that is used to manage the storage request of the local and/or remote applications using the storage. *EX-1004* ¶0039.

The SC servers 110A-C manage their underlying storage (e.g., Hard disk drive (HDD), Solid state drives (SSD), PCIe flash, etc.) and collectively provide reliable and unified storage functionality to the SC clients. The SC client 108A-C processes input/output (I/O) requests from the applications that are local to its physical storage node

and routes them to the appropriate SC servers for processing.

EX-1005 4:3-9; EX-1004 ¶0047.

First Write Queue/First Read Queue

As illustrated in FIG. 2 below, each controller server contains a write cache (*write queue* (red)) and a read cache (*read queue* (blue)):

[T]he Write Cache 208 and the Read Cache 210 Layers provide acceleration for write and read I/O respectively using fast storage devices. In one embodiment, the Write Cache Layer 208 includes the write log as described below. In one embodiment, the Read Cache Layer 210 includes a deduplicated object cache layer (DOCL) that is used to cache recently added and accessed objects for the StorFS system.

EX-1005 6:10-17; EX-1004 ¶ 0052.

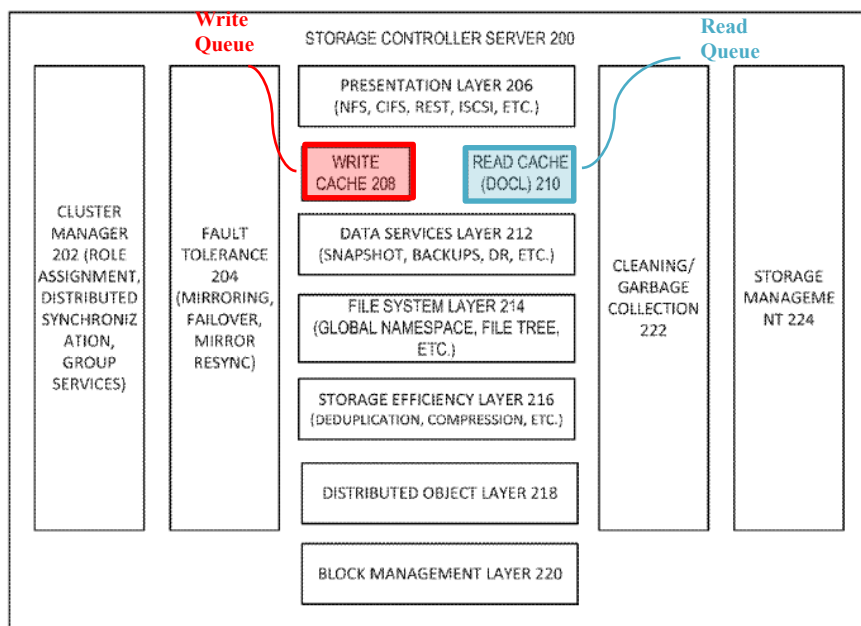


FIGURE 2

EX-1005 FIG. 2; EX-1004 FIG. 2.

A POSITA understood that Agarwala's read cache is a *read queue*. EX-1003 ¶71. First, objects are initially read from persistent storage (e.g., SSD 412 or the HDD 410 of FIG. 4) and added to the read cache where they are held for reading. EX-1005 8:27-28, 8:27-55; EX-1004 ¶¶0061-0062. Second, in this read cache items can be moved through and out of the read cache, as in a queue. For example, cached items can be promoted to the Most Recently Used (MRU) list based on an initial read of an object and the Most Frequently Used (MFU) list depending on frequency of access, as well as be evicted to create room for new contents:

FIG. 3B is a block diagram of one embodiment of a DOCL layer 1 (L1) cache 350. In one embodiment, the

L1 cache 350 includes two types of list: Recency-based eviction lists 352; and frequency-based eviction lists 354... The recency-based eviction lists 352 are used to track objects being added to the cache and include an eviction list for data objects 360 and metadata objects 358... As the objects are read, they are initially promoted to the recency list (e.g., a Most Recently Used (MRU) list). In one embodiment, the frequency-based eviction lists are used to track accesses to objects in the cache and include an eviction list for data objects 364 and metadata objects 362. If this object is accessed within a certain time period again, the object is promoted to the frequency list (e.g., a Most Frequently Used (MFU) list.)... In one embodiment, a hotness/coldness of an object measures how recently the object was added or accessed by the StorFS system. In this embodiment, an object that has not been recently added or accessed can be a candidate for eviction from the L1 cache. In one embodiment, an object that has not been recently added or accessed can be considered “cold” or “stale” and is an object that is one that can be evicted from the L1 cache. In one embodiment, eviction from and updates to the L1 cache is at the object granularity.

EX-1005 7:5-47; EX-1004 ¶0058.

The '632 Patent expressly discloses that the financial industry examples pertaining to time sensitive data it provides are only illustrative and the invention

can be used more broadly “in many other industries and/or applications where information from one location is required to be transmitted to other locations.” *EX-1001* 10:21-25. Other examples of sectors include academic, scientific, military, healthcare, and other areas. *EX-1001* 5:16-20. But to the extent Patent Owner argues that the read queue must be a specific type of queue (e.g., a first in first out (FIFO) queue for time sensitive data such as financial data), it would have been obvious to a POSITA to configure Agarwala’s read cache as a FIFO queue, which was known structure for queues. *EX-1003* ¶¶73-74. First, as noted above, like a FIFO queue, Agarwala discloses that older objects (e.g., objects that are “cold” or “stale”) are evicted from the cache in favor of more recently accessed or added objects, which is consistent with FIFO processing. *EX-1005* 7:5-47; *EX-1004* ¶¶0058; *EX-1003* ¶75. For example, the MRU and MFU operate as FIFO queues for data that are accessed only once. Second, a POSITA would have been motivated to implement the cache as a FIFO queue to the extent the data being read is time sensitive. *EX-1003* ¶76.

That the read cache of Agarwala is a *read queue* is also consistent with Plaintiff’s assertions in the Related Litigation. See *EX-1006* p. 5 (identifying a “RC Read Cache” as a *read queue*), p. 11 (asserting read cache is *second read queue*).

A POSITA likewise understood that Agarwala’s write cache is a *write queue*. *EX-1003* ¶78. Agarwala’s write cache includes a write log for holding or

queuing data to be written to the persistent storage. *EX-1005* 6:12-14; *Ex-1004* ¶¶0004, 00006, claim 16 (“if the active log partition is full after the writing of the data, transferring the data to persistent storage”). When the storage control server receives a write storage request, it first “stores this data in a write log by appending the data at the end of the write log. This can lead to a fast write for the data that allows a quick write storage fulfillment for the requesting application.” *EX-1004* ¶0041. When the write log is full, the SC server “flushes the write log to persistent storage.” *EX-1004* ¶0041.

As noted above, the ’632 Patent expressly discloses that the financial industry examples pertaining to time sensitive data it provides are only illustrative and the invention can be used more broadly “in many other industries and/or applications where information from one location is required to be transmitted to other locations.” *EX-1001* 10:21-25. Other examples of sectors include academic, scientific, military, healthcare, and other areas. *EX-1001* 5:16-20. But to the extent Patent Owner argues that the write queue must be a specific type of queue (e.g., a first in first out (FIFO) queue for time sensitive data such as financial data), it would have been obvious to a POSITA to configure Agarwala’s write cache as a FIFO queue, which was known structure for queues. *EX-1003* ¶80. A POSITA understood that a log maintains writes in the order in which the writes are completed, and to maintain data consistency the log is written in order to the

storage system i.e., the log is used as a FIFO write queue. *EX-1003* ¶80. First, as noted above, like a FIFO queue, Agarwala discloses that data is appended at the end of the write cache, which is consistent with FIFO processing. *EX-1004* ¶0041; *EX-1003* ¶81. Second, flushing the write cache when full results in the older data (i.e., first data stored “in” the write cache) being the first data “out” of the write cache to make room for new data, which is consistent with FIFO processing. *EX-1004* ¶0041; *EX-1003* ¶82. Third, a POSITA would have been motivated to implement the cache as a FIFO queue to the extent the data being written is time sensitive, such that the most recently added data remains accessible in the write cache and older data is stored in the persistent storage upon receipt of new data. *EX-1003* ¶83.

That the write cache of Agarwala is a *write queue* is also consistent with Plaintiff’s assertions in the Related Litigation. See *EX-1006* p. 5 (identifying a “primary active write log” of the “caching layer” as a *write queue*), p. 9 (accusing “active local write log on caching SSD” as *first write queue*).

First Parallel File System

Finally, A POSITA also understood that Agarwala’s storage system implements a *parallel file system*, including in its in its file system and persistent storage. *EX-1003* ¶85.

First, Agarwala's storage controller (SC) servers "collectively" provide storage for the storage clients across multiple distributed nodes:

The SC servers **110A-C** manage their underlying storage (e.g., Hard disk drive (HDD), Solid state drives (SSD), PCIe flash, etc.) and collectively provide reliable and unified storage functionality to the SC clients. The SC client **108A-C** processes input/output (I/O) requests from the applications that are local to its physical storage node and routes them to the appropriate SC servers for processing.

EX-1005 4:3-9; EX-1004 ¶0047. Across the multiple distributed nodes, the workload is partitioned, distributed, and parallelized. *EX-1005 5:13-27, 5:39-41; EX-1004 ¶¶ 0050-0051.*

Second, as described in connection with FIG. 2 (above), the storage controllers of Agarwala's StorFS system implement a file system, including a global distributed object namespace and logical global namespace:

In one embodiment, the Distributed Object Layer 218 of the storage control server uses an API of the Block Management Layer 220 to provide a global distributed object namespace that is accessible over the network... In one embodiment, the File System Layer 214 provides a logical global namespace abstraction to organize and locate data in the cluster... In one embodiment, the Presentation Layer 206 provides an interface to access

the StorFS storage using well-known standard protocols like NFS, CIFS, REST, iSCSI, etc.

EX-1005 5:63-6:21; *EX-1004* ¶¶0052, 0122.

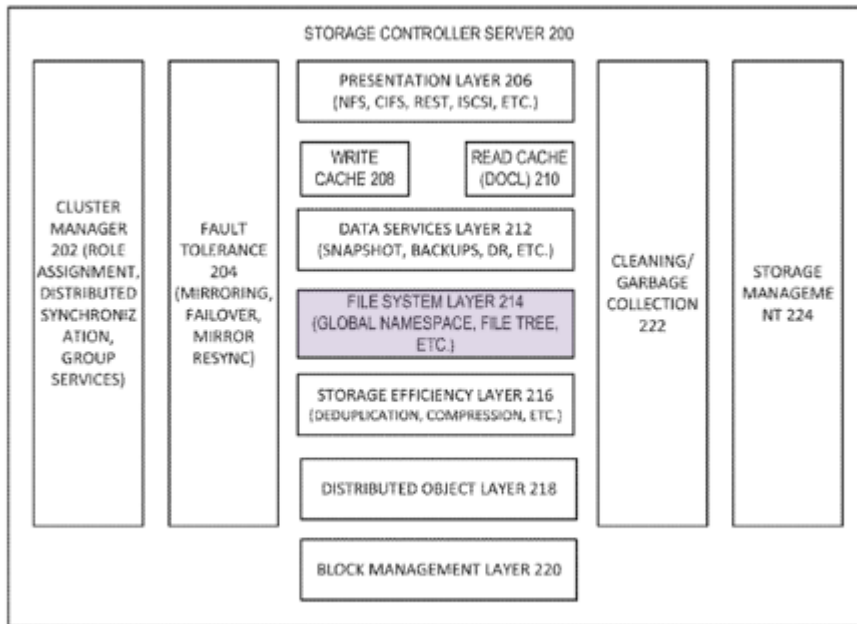


FIGURE 2

EX-1004; *EX-1005*. “[T]he StorFS system is capable of supporting different policies at different levels of granularity: at the file system level, at the individual file level, etc.” *EX-1004* ¶¶0097, 0098.

Third, as is characteristic of a parallel file system, “the StorFS system stripes file and file system data across multiple storage nodes in our cluster.” *EX-1004* ¶122. “[E]ach logical file (or entity) is partitioned into equal sized ‘stripe units’ [where] [t]he location of a stripe unit is determined based on a mathematical placement function.” *EX-1005* 4:30-33; *EX-1004* ¶¶0040, 0048. Further, the

workload is distributed and parallelized across multiple storage nodes, as is also the characteristic of a parallel file system:

In one embodiment, the “Stripe_Unit_Per_Stripe” attribute determines the number of total virtual nodes that an entity can be spread across. This enables distributing and parallelizing the workload across multiple storage nodes (e.g., multiple SC servers 110A-C)...In one embodiment, the StorFS system 100 uses a two-level indexing scheme that maps the logical address (e.g. offset within a file or an object) to a virtual block address (VBA) and from the VBAs to physical block address (PBA). In one embodiment, the VBAs are prefixed by the ID of the vNode in which they are stored. This vNode identifier (ID) is used by the SC client and other StorFS system 100 components to route the I/O to the correct cluster node. The physical location on the disk is determined based on the second index, which is local to a physical node.

EX-1005 5:13-27, 5:39-41; *EX-1004* ¶¶ 0050-0051.

Fourth, Agarwala discloses that its StorFS system manages storage requests of local and remote applications using its storage. *EX-1004* ¶ 39. Agarwala also discloses that storage nodes can be located remote from one another. *EX-1004* ¶¶ 47, 59, 87. That Agarwala’s persistent storage at storage node 302A is a *parallel file system* is consistent with the disclosure in the ’632 Patent that data from a first

node in a first region can be written to a parallel file system device at a node in a remote region. *EX-1001* 13:27-36; 16:4-5 (“In practical use, all connected nodes can concurrently connect with PFS devices in all other regions and locations.”).

Finally, that Agarwala’s persistent storage is a *parallel file system* is consistent with Plaintiff’s assertions in the Related Litigation. *See EX-1006* p. 5 (“[T]he IO Visor intercepts workload traffic and stripes the block across available nodes in the cluster... The data platform implements a log-structured file system...”), p. 11 (reading data from “Cache Drive” or “Capacity Drive” to “read cache” is reading “the first data from the second parallel file system and plac[ing] the first data in the second read queue”).

FIG. 3 illustrates the relationship of the read and write queue with the parallel file system of the first node:

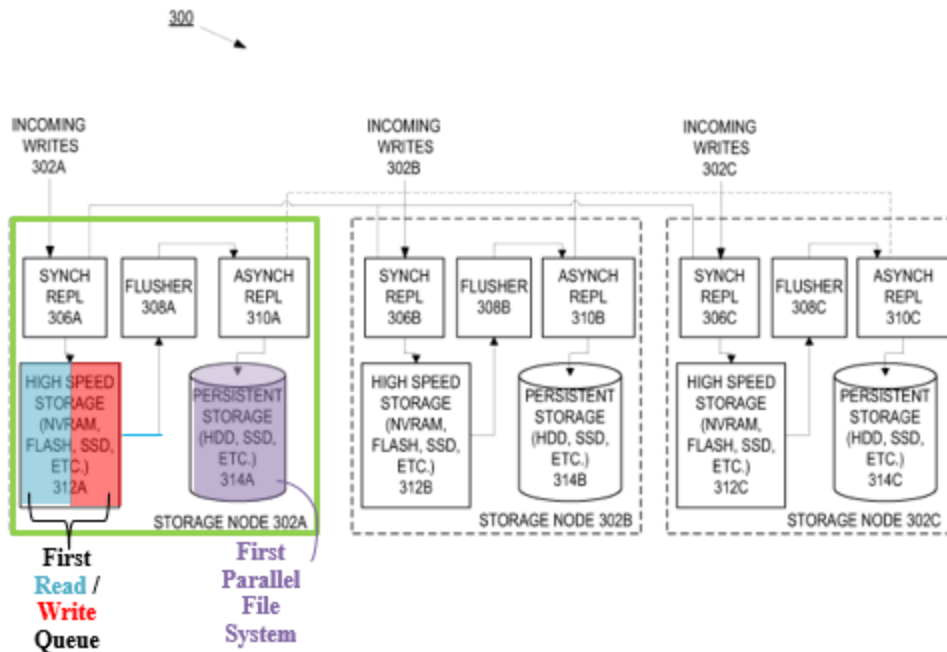


FIGURE 3

Agarwala discloses that it implements the read/write queues using high speed storage, SSDs and multiple levels of caches, to hide the latency of slower HDDs from the users. In one embodiment, the StorFS system writes data sequentially on SSD because random writes incur higher latency and reduce the life of SSD. In addition to taking data blocks' recency and frequency into account, the StorFS system considers SSD's erase block size for eviction of "cold" data that may no longer be required by the application. *EX-1004* ¶0058. For example, the

synchronous replicate module 306A receives the incoming write 304A² and stores the contents of the write in the high-speed storage 312A (e.g., NVRAM, Flash, SSD, etc.). *EX-1004 ¶¶0059, 0081*. With respect to read operations, in one embodiment, an object is initially read from the persistent storage and added to cache which includes a most recently used (MRU) list and a most frequently used (MFU) list. Upon access, the object is added to the MRU list. If the object is not accessed again while in the MRU list, the object will go stale and be evicted from the MRU list. If the object is accessed the object is moved to the MFU list. *EX-1005 8:27-45*.

As discussed, below in more detail, FIG. 3 also illustrates how the read/write queue 312A of a first node communicates with the parallel file system 314B of a second node.

[1.2] a second node comprising a second read queue, a second write queue, and a second parallel file system;

Agarwala discloses this limitation for the reasons discussed in limitation [1.1]. As illustrated in FIG. 3, a *second storage node* (orange) contains a *write queue* (red), *read queue* (blue), and *parallel file system* (purple):

² This incoming write is mislabeled as 302A in FIG. 3.

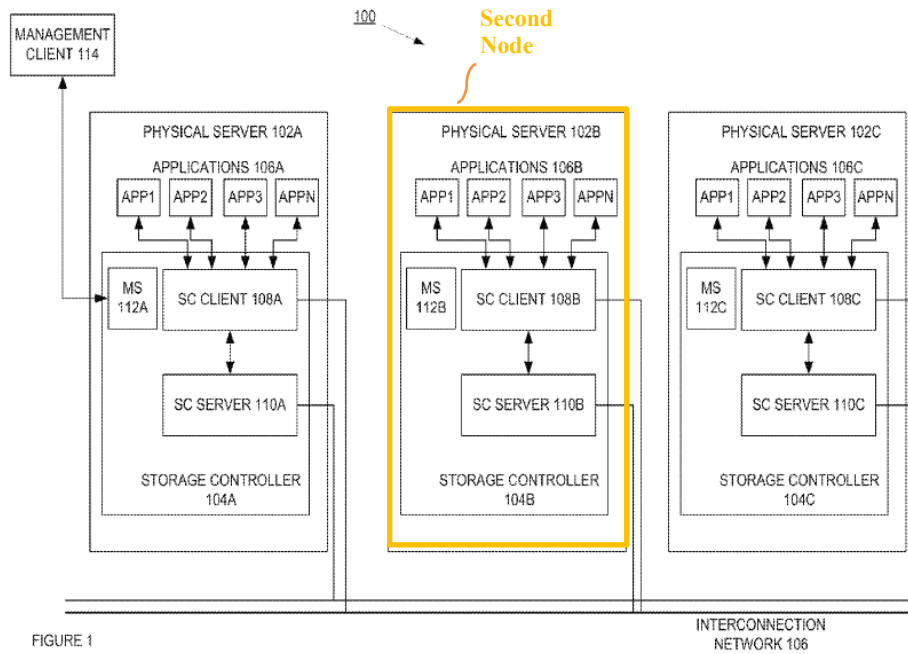
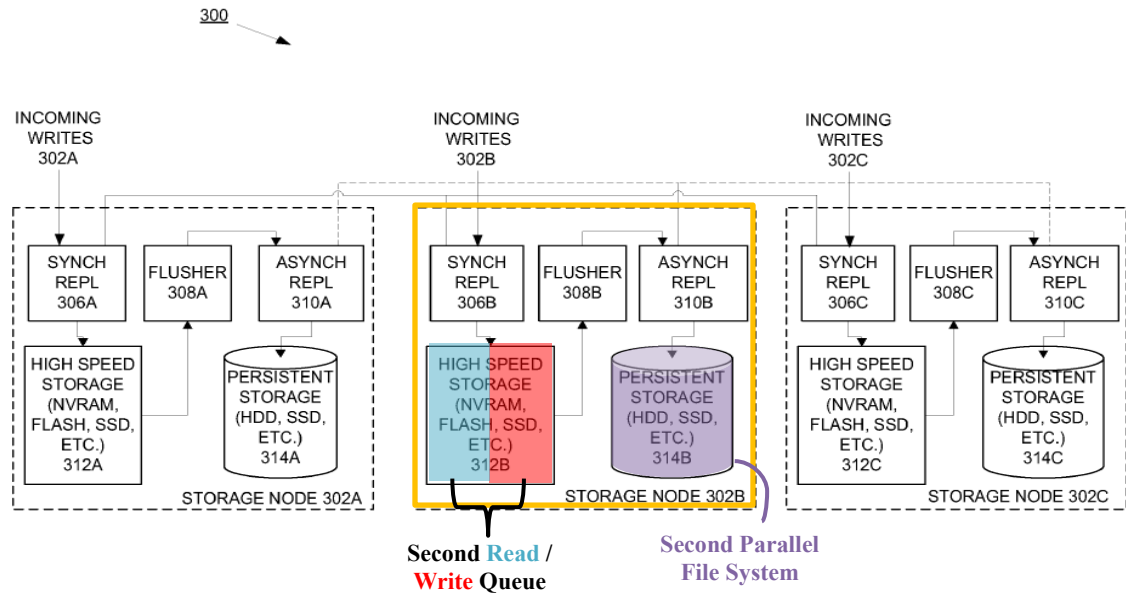


FIGURE 1

EX-1005 FIG. 1; EX-1004 FIG. 1.



EX-1004 FIG. 3. See also EX-1004 FIGS. 1-2; EX-1005 FIGS. 1-2.

[1.3] wherein the first node writes first data from the first write queue to the second parallel file system; and

Agarwala discloses a *first node*, as described above in [1.1], that sends data from the write log (*writes first data from the first write queue*) of the first node to persistent storage in another storage controller server (*to the second parallel file system*).

For example, this write occurs at least as part of Agarwala's asynchronous replication of data to mirror nodes, where data is written from the write log of Agarwala's write cache in one storage node to other storage nodes to "efficiently replicate data across the cluster". *EX-1004 ¶¶0041, 0043, 0059*. With reference to FIG. 3 (below):

Figure 3 is an illustration of one embodiment of an efficient distributed replication. In Figure 3, the system 300 uses a combination of synchronous and asynchronous techniques to efficiently replicate data across the cluster. In one embodiment, storage nodes of SC servers 302A-C can each receive an incoming client write 304A-C. In one embodiment, the storage node 302A receives the incoming client write with synchronous replicate module 306A. The synchronous replicate module 306A receives the write 304A and stores the contents of the write in the high-speed storage 312A (e.g., NVRAM, Flash, SSD, etc.). In one embodiment, the synchronous replicate module 306A stores the write contents at the backend of the high-speed storage 312A as described above. A flusher module 308A

flushes the contents of the write to the asynchronous replication module 310A...

In one embodiment, the asynchronous replication module 310A stores the content to persistent storage 314A. In this embodiment, the asynchronous replication module 310A can additionally replicate this content to the other storage nodes 302B-C via the asynchronous replication modules 310B-C, respectively. These receiving asynchronous replication modules 310B-C, each store the content in the corresponding local perspective storage 314B-C, respectively.

Ex-1004 ¶¶0081-0082.

FIG. 3, annotated below, further illustrates data from the first node (green) written to the second node's parallel file system (purple):

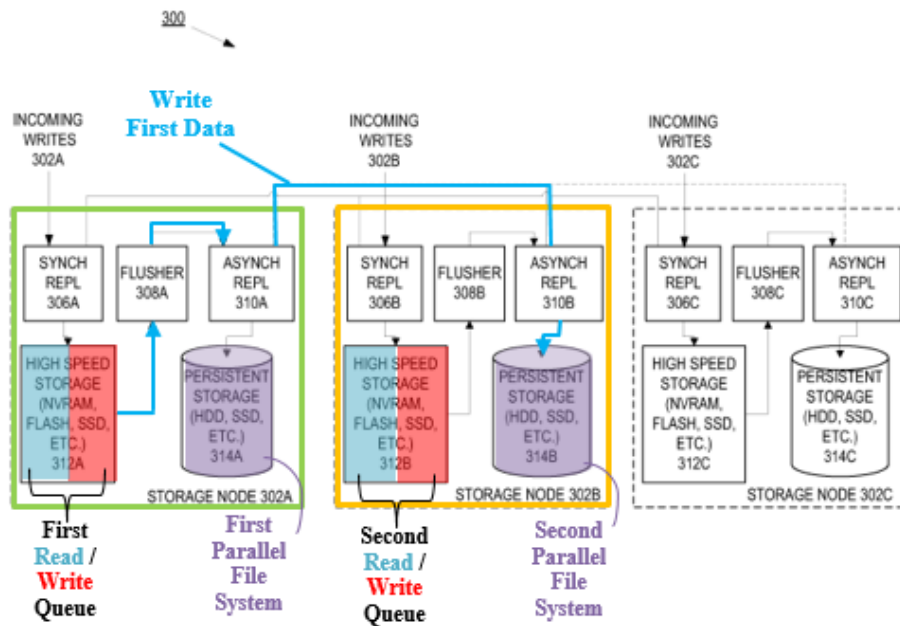


FIGURE 3

EX-1004 FIG. 3.

Agarwala's disclosure of data being written from a write cache in one storage node to another storage node is also consistent with Plaintiff's assertions in the Related Litigation. See EX-1006 p. 9 ("[Node2] commits the write to the active local write log on the caching SSD [A1] and sends duplicate copies to other nodes, e.g., '[A2] on Node 1', i.e. the first node writes first data from the first write queue to the second parallel file system").

[1.4] wherein the second node reads the first data from the second parallel file system and places the first data in the second read queue.

Agarwala discloses that after data is written into the persistent storage of a second storage node, it will be read from persistent storage and added to the read cache (*the second node reads the first data from the second parallel file system and places the first data in the second read queue*). A POSITA understood that this read occurs following replication of the data from a first node to a second mirror node as part of an asynchronous replication. See [1.3]; EX-1003 ¶¶98.

For example, Agarwala discloses that when there is a read cache miss for data object, the data object is read from the persistent storage (*second parallel file system*) into the read cache (*second read queue*):

In one embodiment the StorFS system uses the L2 cache in tandem with the L1 cache for a multi-layer DOCL. In one embodiment, the L2 cache works as follows: On L1 cache miss, the object is looked in the L2 cache. If there is hit, the object is read and promoted to the MFU list in the L1 cache. **If there is a miss the object is read from StorFS persistent store and promoted to the MRU list in the L1 cache.**

EX-1005 10:44-51. The read process is also disclosed in FIG. 9B, where at block 950 content is read from persistent storage upon a cache miss and placed in the cache storage at block 966. EX-1004 ¶¶0061-62; FIG. 9.

Agarwala's reading from persistent storage and placing the read data into a read cache corresponds to the *second node reads the first data from the second*

parallel file system and places the first data in the second read queue is also consistent with Plaintiff's assertions in the Related Litigation. See EX-1006 p. 11 (reading data from "Cache Drive" or "Capacity Drive" to "read cache" is reading the first data from the second parallel file system and placing the first data in the second read queue).

b. Claim 2

[2] The network system of claim 1, wherein the second node writes second data from the second write queue to the first parallel file system; and wherein the first node reads the second data from the first parallel file system and places the second data in the first read queue.

Agarwala discloses or at least renders obvious *wherein the second node writes second data from the second write queue to the first parallel file system; and wherein the first node reads the second data from the first parallel file system and places the second data in the first read queue.*

First, as set forth above (see [1.3] and [1.4]), Agarwala discloses *the first node writes first data from the first write queue to the second parallel file system and the second node reads the first data from the second parallel file system and places the first data in the second read queue.*

Second, Agarwala also discloses that any of the storage nodes can write to a different storage node within the network. EX-1004 ¶¶0007-0008, 0039 (mirroring

across multiple nodes); *EX-1005* 4:7:6-12; FIG. 2 (Fault Tolerance 204 (mirroring...)).

A POSITA understood that just as first data is written from the first write queue of the first node of Agarwala to the second parallel file system of the second node of Agarwala, and that first data is placed in the second read queue (*see* [1.3] and [1.4]), other (second) data can also be written from the second write queue of the second node of Agarwala to the first parallel file system of the first node of Agarwala, and that second data is then placed in the second read queue. *EX-1003* ¶104. Therefore, Agarwala discloses or renders obvious that *the second node writes second data from the second write queue to the first parallel file system; and wherein the first node reads the second data from the first parallel file system and places the second data in the first read queue.*

c. **Claim 3**

[3] The network system of claim 1, wherein the first data is a carrier file comprising a header, a body, and a footer.

Agarwala discloses *wherein the first data is a carrier file comprising a header, a body, and a footer.*

In its infringement allegations, Plaintiff accuses network packets of satisfying this claim:

Claim #3	Accused Instrumentalities
3. The network system of claim 1, wherein the first data is a carrier file comprising a header, a body, and a footer.	Network packets typically include a header, and a payload including a footer, thus a “first data” as “carrier file” can be inferred.

EX-1006 p. 28.

Agarwala’s storage nodes are coupled together by interconnection network. *EX-1004* ¶¶0039, 0116, FIG. 1; *EX-1004* 3:59-62, FIG. 1. A POSITA understood that communications between storage nodes over Agarwala’s interconnection network would use packets. *EX-1003* ¶108. Therefore, as PO does, it can be inferred that Agarwala discloses *wherein the first data is a carrier file comprising a header, a body, and a footer*.

Further, the ’632 Patent discloses that header information can contain information about the host origin, host destination, timestamp, and other information. *EX-1004* 12:64-66. A POSITA understood that it is well known for network packets to include headers with origin and destination information. *EX-1003* ¶109. The ’632 Patent also discloses that the footer can include payload checksum information to validate the integrity of the body data. *EX-1005* 12:64-66, FIG. 6, FIG. 9. A POSITA understood that it is well known for network packets to include footers with checksum information for the packet payload. *EX-1003* ¶110.

To the extent that PO asserts that Agarwala does not expressly use the term carrier file, a POSITA understood that data of different types could be stored in Agarwala's StorFS system based on the applications. Also, a POSITA understood a common way to move data around is using carrier files, and thus it would be obvious to modify the teachings of Agarwala to use carrier files. *EX-1003* ¶111 For example, Haghighi discloses that "information can be structured, stored, and accessed in the storage system via files, blocks, logical block address (LBA), logical unit number (LUN), key-value pairs, objects, or the like." *EX-1007* ¶ 24, 27. *EX-1003* ¶112 A POSITA understood that a common structure of a carrier file included a header, footer and body. *EX-1003* ¶113. A POSITA would be motivated to use carrier files for sending the first data from the cache of the first node to the second node for storage in persistent storage because it would have provided an efficient method of transferring data in bulk. *EX-1003* ¶114. Indeed, Agarwala discloses that in embodiments its systems transfer data as a large bulk transfer so that this data transfer is more efficient. *EX-1004* ¶43.

d. Claim 4

[4] The network system of claim 3, wherein the first nodes subsequently writes additional data to the first parallel file system.

Agarwala discloses *wherein the first nodes³ subsequently writes additional data to the first parallel file system.*

Agarwala discloses that the first node writes first data to the persistent storage of the first node (*first parallel file system*). For example, Agarwala discloses that “when the write log is full, the SC server flushes the write log to persistent storage.” *EX-1004 ¶¶ 0042; EX-1005 11:64-67.* “[A] ‘flusher’ moves the logged content asynchronously from the high-speed storage to its permanent location for persistence.” *EX-1004 ¶¶ 0059, 0069-0070* (describing asynchronous flush of data to persistent storage of target node and mirror nodes); FIGS. 3, 11B.

A POSITA understood that Agarwala discloses that there would be writes of data beyond the first data (i.e., *additional data*) after (*subsequently*) to the write of the first data during normal system operation in Agarwala, because that is the purpose of moving the first data from the write log to persistent storage, i.e., to allow additional data to be stored in the write log and then also be written from the write log to persistent storage. *EX-1003 ¶117.* Therefore, Agarwala discloses *wherein the first nodes subsequently writes additional data to the first parallel file system.*

e. **Claim 5**

³ Per Claim 1 there is only one “first node”.

[5] The network system of claim 4, wherein the additional data is written at a set frequency.

Agarwala discloses *wherein the additional data is written at a set frequency*.

As discussed with respect to Claim 4, the “flusher” of Agarwala moves data from the high speed cache storage to the persistent storage. Agarwala further discloses that the “flusher” operates “periodically” to move (*write*) the data to persistent storage:

New incoming writes are logged and synchronously replicated in SSDs or other high-speed storage medium.

Periodically, a ‘flusher’ 308A-C moves the logged content asynchronously from the high-speed storage to its permanent location for persistence.

EX-1004 ¶0083.

Further, a POSITA understood that Agarwala uses “synchronous” to refer to writes from the client to the write log of fast storage and replication to its mirror nodes, and uses “asynchronous” to refer to the operation where data is moved or “flushed” from high-speed storage to persistent storage of the node and replicated to its mirror nodes. *EX-1003 ¶121; EX-1004 ¶¶0007-0008, 0041, 0043, 0059, 0069, 0083.* These “asynchronous” operations occurs on a scheduled basis, i.e., *at a set frequency*, as noted above. *EX-1004 ¶0083.* Therefore, a POSITA understood

that Agarwala discloses *wherein the additional data is written at a set frequency*.

EX-1003 ¶ 123.

f. **Claim 6**

[6] The network system of claim 4, wherein the additional data only contains information that has changed since the first data was written.

Agarwala discloses *wherein the additional data only contains information that has changed since the first data was written*.

For example, Agarwala discloses a deduplication module located within the flusher described above in connection with Claim 4. *EX-1004* ¶00104; FIG. 6. The deduplication module contains deduplication analyzer that “generates the dedup hint table by analyzing the streams of new objects that are written to the storage”. *EX-1004* ¶00104, FIG. 6. The dedup hint table is used “to determine whether data is being duplicated”. *EX-1004* ¶0045. “[T]he writelog flusher checks the dedup table to avoid storing duplicates to the persistent storage.” *EX-1004* ¶00104.

In one embodiment, during the flushing operation, the StorFS system, determines if an entity is stored elsewhere by using a deduplication hints table. In one embodiment, the hints table stores characteristics of the top-K storage entities in the StorFS system. If there is match with the entity being flushed, the StorFS system updates the metadata for this entity, but does not flush the entity to

the persistent storage. If not, the StorFS system flushes the entity to persistent storage.

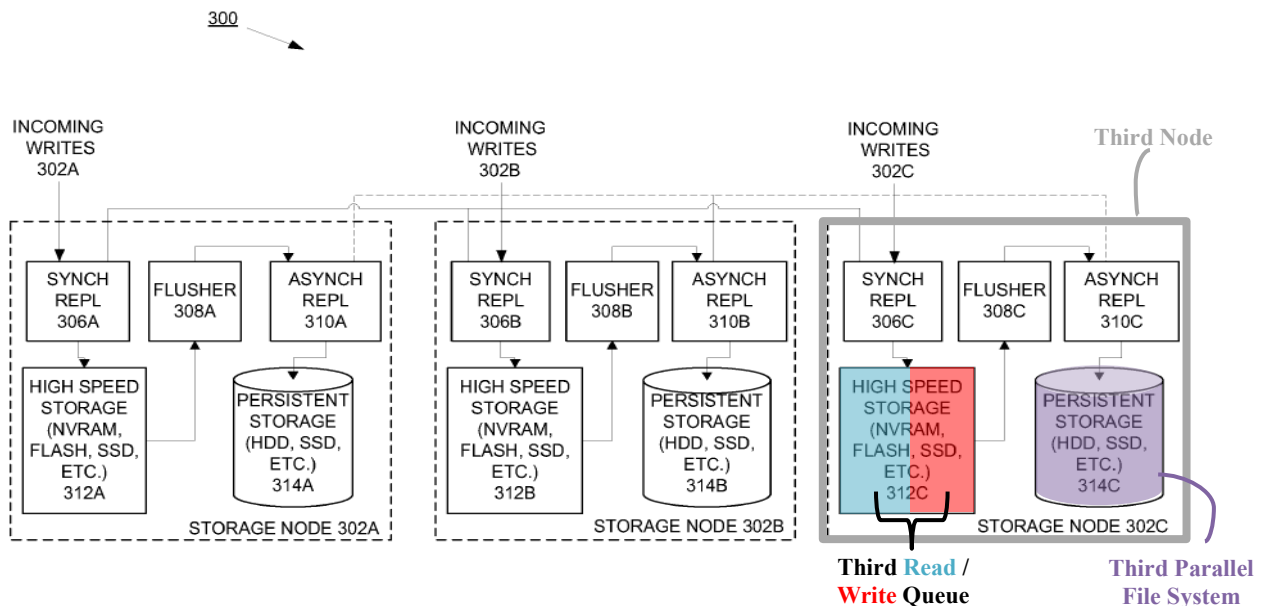
EX-1004 ¶0042.

By not writing duplicate data to persistent storage, a POSITA understood that Agarwala discloses that *additional data only contains information that has changed since the first data was written.* EX-1003 ¶126.

g. **Claim 7**

[7.0] The network system of claim 1, further comprising a third node comprising a third read queue, a third write queue, and a third parallel file system;

Agarwala discloses this limitation for the same reasons as recited with respect to Claim [1.1]. Agarwala FIG. 3 (annotated below) shows a *third storage node* (grey) contains a *write queue* (red), *read queue* (blue), and *parallel file system* (purple):



EX-1004 FIG. 3. See also EX-1004 FIGS. 1-2; EX-1005 FIGS. 1-2.

[7.1] wherein the first node writes first data from the first write queue to the second and third parallel file systems at the same time; and

Agarwala discloses *wherein the first node writes first data from the first write queue to the second and third parallel files system at the same time.*

First, as analyzed with respect to Claim [1.3], Agarwala's first storage node (*first node*) writes data (*first data*) from the write log of the write cache (*first write queue*) to the persistent storage of the second storage node (*second parallel file system*) as part of asynchronous replication. See Claim [1.3]

Second, Agarwala discloses that this asynchronous replication involves replication to multiple storage nodes. For example:

In this embodiment, the asynchronous replication module 310A can additionally replicate this content to the other storage nodes 302B-C via the asynchronous replication modules 310B-C, respectively. These receiving asynchronous replication modules 310B-C, each store the content in the corresponding local perspective storage 314B-C, respectively.

EX-1004 ¶0082. A POSITA understood that one of the multiple mirror nodes would be the second node and the other, the third node. EX-1003 ¶131.

A POSITA understood that Agarwala uses “asynchronous” to refer to the operation where data is moved or “flushed” from high-speed storage to persistent storage of the node and simultaneously replicated to its mirror nodes. *EX-1003* ¶132; *EX-1004* ¶¶0008, 0043, 0059, 0069, 0083. These “asynchronous” operations occurs on a scheduled basis, i.e., *at a set frequency*, as noted above. *EX-1004* ¶0083. Thus, a POSITA understood that periodically, asynchronous replication module 310A of the first node would replicate data to the second and third storage nodes, and that this replication to the second and third storage nodes would occur *at the same time* because it is efficient to make multiple copies at the same time at the first node to send to the second and third nodes. *EX-1004* ¶0070, FIG. 11B (one step 1156 of “Asynchronously Send the Data and Metadata to the Vnode and Mirror Nodes in the AMS”); *EX-1003* ¶134.

To the extent that PO asserts that the Agarwala does not expressly state that the first data is replicated at the second and third nodes *at the same time*, a POSITA, understood that a common way to replicate data to multiple locations was to do so at the same time, and thus it would be obvious to modify the teachings of Agarwala so that asynchronous replication module 310A is programmed to replicate first data at the second and third nodes at the same time because doing would be a more efficient use of resources and was a common and well-known technique that had predictable results. A POSITA understood that

writing to multiple nodes at the same time provided the benefit of keeping replicas consistent across the system. *EX-1003* ¶135.

Therefore, a POSITA understood that Agarwala discloses that a *first node writes first data from the first write queue to the second and third parallel file system at the same time.*

[7.2] wherein the third node reads the first data from the third parallel file system and places the first data in the third read queue.

Agarwala discloses *wherein the third node reads the first data from the third parallel file system and places the first data in the third read queue.*

As discussed above, Agarwala discloses that the second node reads the first data from the second parallel file system and places the first data in the second read queue. *See* [1.4]. Agarwala also discloses that the first data is replicated to the third parallel file system of the third node. *See* [7.1]. For the same reasons as set forth in connection with [1.4], a POSITA would have understand that the first data would also be read from the third parallel file system to the third read queue, e.g., upon a cache miss, and thus Agarwala discloses or renders obvious *wherein the third node reads the first data from the third parallel file system and places the first data in the third read queue.*

VII. THE BOARD SHOULD NOT EXERCISE ITS DISCRETION AND DENY INSTITUTION

A. The Board Should Not Deny Institution Under 35 U.S.C. § 325(d)

Denial under § 325(d) is not warranted because the challenges presented in this Petition are neither cumulative nor redundant to the prosecution of the '632 Patent. None of the asserted prior art references were cited or considered by the Examiner during prosecution of the '632 Patent. The references in this Petition were neither relied upon during prosecution nor are cumulative of art relied upon during prosecution. Further, the '632 Patent has not been previously challenged in an IPR.

B. The Board Should Not Deny Institution Under 35 U.S.C. § 314(a)

The PTAB has discretion to deny institution under 35 U.S.C. § 314(a). Related to the Director's June 2022 interim guidance regarding application of the *Fintiv* factors⁴, (1) the Petition is particularly strong in the underlying merits, and (2) the Petitioner adopts the identical stipulation from *Sotera*, namely, that if the IPR is instituted, Petitioner will not pursue in the Related Litigation any ground

⁴ Interim Procedure For Discretionary Denials In AIA Post Grant Proceedings With Parallel District Court Litigation, PTO Director's Memorandum (June 21, 2022).

raised or that could have been reasonably raised in an IPR for the challenged patent. Thus, under the Interim Guidance, the Board should not deny institution.

Nevertheless, the Board's decision in *Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-01019, Paper 12 (Dec. 1, 2020) instructs that a holistic view of the remaining *Fintiv* factors also weighs in favor of institution.

Factor 1 is neutral because no request for stay has been filed.

Under factor 2, no trial date has been set, nor has a judge been assigned. According to the most recent statistics on median time-to-trial for civil actions (<https://www.uscourts.gov/statistics/table/t-3/federal-judicial-caseload-statistics/2023/03/31>), the median time from filing to trial was 32 months in the Western District of Texas. The Related Litigation was filed on August 2, 2023 and thus would be expected to go to trial April 2026 according to the median time to trial statistics. A Final Written Decision is expected in the present matter in July 2025, 9 months before trial in the Related Litigation. Thus, this factor weighs against discretionary denial.

Factor 3 favors institution. The co-pending litigation is in its early stages, and the investment in it has been minimal. No Scheduling Order has yet been entered. Thus, this factor weighs against discretionary denial.

Under factor 4, in addition to the *Sotera* stipulation that Petitioner issued to PO, PO has only identified in the Complaint a subset of the claims that are being

challenged in the present Petition. Thus, it is unlikely the district court will be addressing the validity of all of the claims challenged by this Petition.

Under factor 5, Petitioner is a defendant in the Related Litigation. Nothing within *Fintiv* suggests that the same party between the proceedings weighs in favor of discretionary denial. See *HP Inc. v. Slingshot Printing LLC*, IPR2020-01084, Paper 13 at 9 (having the “same parties as parallel proceeding” makes factor 5 “neutral”)).

Under factor 6, other circumstances weigh in favor of institution. Here, the merits of the Petition are particularly strong. For example, Agarwala is a parallel disclosure to the ’632 Patent, meaning that it is directed at the same problem and proposes the same solution as the ’632 Patent, and discloses identical architecture and the identical concepts disclosed in the ’632 Patent, but uses some different words, thus demonstrating that the Petition is particularly strong in the underlying merits. The evidence presented, if unrebutted in trial, would plainly lead to a conclusion that one or more claims are unpatentable by a preponderance of the evidence.

When viewed holistically, the timing of the present Petition is reasonable, there has been relatively limited investment in the Related Litigation, and there is minimal overlap between the present IPR and the Related Litigation. Further, coupling these *Fintiv* considerations with compelling evidence of unpatentability

presented in the Petition, the efficiency and integrity of the IPR process is best served by instituting review.

VIII. Mandatory Notices Under 37 C.F.R. §42.8

B. Real Party-in-Interest (37 C.F.R. § 42.8(b)(1))

The real party-in-interest in this Petition is Cisco Systems, Inc.

C. Related Matters (37 C.F.R. § 42.8(b)(2))

1. Judicial Matters

As of the filing date of this Petition and to the best knowledge of Petitioner, the '632 Patent is involved in the following litigations:

Umbra Technologies Ltd. (UK) et al v. Cisco Systems, Inc., U.S.D.C WDTX Case No. 1:23-cv-00903-DII (the "Related Litigation").

Umbra Technologies Ltd. (UK) et al v. VMware, Inc., U.S.D.C WDTX Case No. 1:23-cv-00904-DII.

Administrative Matters:

As of the filing date of this Petition and to the best knowledge of Petitioner, the '632 Patent was not subject to any administrative matters.

2. Related Patents

1. U.S. Patent No. 11,487,717
2. U.S. Patent No. 11,789,910
3. U.S. Patent Application No. 18/461,201

D. Lead/Back-up Counsel (37 C.F.R. § 42.8(b)(3)):

Lead Counsel:

Patrick D. McPherson, USPTO Reg. No. 46,255

DUANE MORRIS LLP, 901 New York Avenue N.W., Suite 700 East,
Washington, D.C. 20001

P: (202) 776-5214; F: (202) 776-7801; PDMcPherson@duanemorris.com

Back-Up Counsel: Patrick Muldoon, USPTO Reg. No. 47,343

DUANE MORRIS LLP, 901 New York Avenue N.W., Suite 700 East,
Washington, D.C. 20001

P: (202) 776-7840; F: (202) 776-7801; PCMuldoon@duanemorris.com

Joseph Powers, USPTO Reg. No. 47,006

DUANE MORRIS LLP, 30 South 17th Street, Philadelphia, PA 19103-4196

P: (215) 979-1842; F: (215) 689-3797; JAPowers@duanemorris.com

E. Notice of Service Information (37 C.F.R. § 42.8(b)(4)):

Please direct all correspondence to lead and back-up counsel at the above addresses. Petitioner consents to electronic service at the email addresses above.

F. GROUNDS FOR STANDING (37 C.F.R. § 42.204(A))

Petitioner certifies that the Patent for which review is sought is available for post grant review and that Petitioner is not barred or estopped from requesting a post grant review of the Challenged Claims on the grounds identified herein. 37 C.F.R. § 42.204(a). This Petition is filed pursuant to 37 C.F.R. § 42.206(a).

IX. CONCLUSION

For the reasons set forth above, Petitioner has established a reasonable likelihood of prevailing with respect to all Challenged Claims of the '632 Patent and requests the Board institute *inter partes* review and then cancel all claims as unpatentable.

Respectfully submitted,

DUANE MORRIS LLP

BY: /Patrick D. McPherson/

Patrick D. McPherson

USPTO Reg. No. 46,255

Duane Morris LLP

901 New York Avenue, N.W., Suite

700 East

Washington, D.C. 20001

Dated: January 29, 2024

CERTIFICATION OF SERVICE ON PATENT OWNER

Pursuant to 37 C.F.R. §§ 42.6(e), 42.8(b)(4) and 42.105, the undersigned certifies that on the 29th of January 2024, a complete and entire copy of this Petition for *Inter Partes* Review of U.S. Patent No. 11,146,632 and all supporting exhibits were served via Federal Express, postage prepaid, to the Patent Owner by serving the correspondence address of record for the '632 Patent:

WILMERHALE/BOSTON
60 STATE STREET
BOSTON, MA 02109
UNITED STATES

/Patrick D. McPherson/
Patrick D. McPherson, Reg. No. 46,255
901 New York Ave., N.W., Suite 700-East
Washington, D.C. 20001
P: (202) 776-7800
F: (202) 776-7801
PDMcPherson@duanemorris.com

ATTORNEY FOR PETITIONER

CERTIFICATE OF COMPLIANCE

Pursuant to 37 C.F.R. § 42.24 *et seq.*, the undersigned certifies that this document complies with the type-volume limitations. This document contains 10,990 words as calculated by the “Word Count” feature of Microsoft Word 2010, the word processing program used to create it.

Dated: January 29, 2024

By: Patrick D. McPherson
Patrick D. McPherson, Reg. No. 46,255
Duane Morris LLP
901 New York Ave., N.W., Suite 700-East
Washington D.C., 20004
P: (202) 776-7800
F: (202) 776-7801
PDMcPherson@duanemorris.com