

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC
Petitioner

v.

LS CLOUD STORAGE TECHNOLOGIES LLC
Patent Owner

Case IPR2023-00777
U.S. Patent No. 6,549,988

**GOOGLE LLC'S PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 6,549,988**

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION	1
II. MANDATORY NOTICES	2
A. Real Party-in-Interest	2
B. Related Matters	2
C. Counsel and Service	4
D. Power of Attorney	4
E. Fees.....	4
III. REQUIREMENTS FOR <i>INTER PARTES</i> REVIEW.....	5
A. Grounds for Standing	5
B. Identification of Challenge and Relief Requested.....	5
IV. THE '988 PATENT	6
V. PROSECUTION HISTORY	9
VI. THE CHALLENGED CLAIMS ARE UNPATENTABLE.....	9
A. State of the Art.....	9
B. Overview of the Prior Art.....	11
1. Heil	11
2. Nakayama	14
3. Srinivasan	16
4. Macklem.....	16

C.	Level of Ordinary Skill in the Art	17
D.	Claim Construction	17
E.	Ground 1: Claims 1-4 and 6 are obvious over Heil and Nakayama under 35 U.S.C. § 103(a)	18
1.	A POSITA Would Have Combined Heil and Nakayama	18
2.	Claim 1	21
i.	Claim 1.pre: “A computer suitable for use in a data storage system comprising a network interconnecting a plurality of such computers”	21
ii.	Claim 1.a: an I/O channel adapter for accepting an incoming I/O request from a host	21
iii.	Claim 1.b: “configuration manager software for enabling said I/O channel adapter to decide whether (i) to route said request to cache, (ii) to route said request to disk, or (iii) to reject said request”	23
iv.	Claim 1.c: “a network adapter for handling network control traffic”	29
v.	Claim 1.d: “a cache memory”	31
vi.	Claim 1.e: “front-end software for handling I/O requests arriving at the I/O channel adapter or the network adapter”	31
vii.	Claim 1.f: “cache manager software, responsive to said front-end software, for handling data stored in said cache memory”	34
viii.	Claim 1.g: “back-end software, responsive to said configuration manager software, for handling reads and writes to disks	

	corresponding to the I/O requests but without communication over the I/O channel adapter, thereby separating disk operations from network and I/O traffic”	36
3.	Claim 2	39
	i. Claim 2.pre: “The system of claim 1 wherein the computers comprise off-the-shelf hardware and operating systems”	39
	ii. Claim 2.a: “an adapter I/O software for accepting incoming I/O requests from a host”	40
	iii. Claim 2.b: “a volume access table employed by the configuration manager to ensure consistency of data stored on the network”	41
4.	Claim 3	44
5.	Claim 4	45
6.	Claim 6	46
	i. Claim 6.pre: “A method of accessing a remote disk over a computer network without incurring network overhead”	46
	ii. Claim 6.a: “a. causing a local host to issue a request over an I/O channel to a local computer”	47
	iii. Claim 6.b: “b. providing a configuration manager on the local computer, the configuration manager routing the request to a remote computer via the computer network”	48
	iv. Claim 6.c: “c. causing the remote computer to check the request against a volume access table”	49

v.	Claim 6.d: “d. causing the remote computer to perform an I/O operation on a disk located on the remote computer and to return data to the local computer”	50
vi.	Claim 6.e: “e. causing the local computer to provide the returned data to the local host via the I/O channel”	51
vii.	Claim 6.f: “f. causing the local computer to check the data against the volume access table to ensure consistency of the data on the local and the remote computers”	52
F.	Ground 2: Claim 5 is obvious over Heil, Nakayama, Srinivasan and Macklem under 35 U.S.C. § 103(a)	53
1.	A POSITA Would Have Combined Heil, Nakayama, Srinivasan, and Macklem	53
2.	Claim 5	56
i.	Claim 5.a: “wherein the configuration manager includes software that checks an access mode in the volume access table”	56
ii.	Claim 5.b: “(i) if the access mode is set to an exclusive mode, causes both reads and writes to be stored in the cache memory, and causes invalidate messages to be sent to remote storage systems”	58
iii.	Claim 5.c: “(ii) if the access mode is set to shared, causes only reads to be stored in the cache memory”	66
iv.	Claim 5.d: “(iii) if the access mode is set to a value other than exclusive or shared, causes reads and writes to be performed directly to a disk without using the cache memory”	68

VII. SECONDARY CONSIDERATIONS	71
VIII. DISCRETION	71
A. The <i>Fintiv</i> Factors Favor Institution	71
B. General Plastic is Inapplicable Here	75
C. 35 U.S.C. §325(d) Favors Institution	75
IX. CONCLUSION	76

TABLE OF AUTHORITIES

Page(s)

CASES

<i>Thryv, Inc. v. Click-To-Call Techs., LP</i> , 140 S. Ct. 1367 (2020).....	78
---	----

AGENCY DECISIONS

<i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, Paper 11 (Mar. 20, 2020)	74
<i>CommScope Technologies, LLC v. Dali Wireless Inc.</i> , IPR2022-01242, Paper No. 23 (PTAB Feb. 27, 2023).....	75, 77
<i>General Plastic Industrial Co., Ltd. v. Canon Kabushiki Kaisha</i> , IPR2016-01357, Paper No. 19 (PTAB Sept. 6, 2017).....	78
<i>Google v. EcoFactor</i> , IPR2022-00473, Paper 7 (PTAB Aug. 3, 2022).....	76
<i>Samsung Elecs. Co. Ltd. v. Dynamics Inc.</i> , IPR2020-00505, Paper 11 (PTAB Aug. 12, 2020).....	78

PETITIONER’S EXHIBIT LIST

<u>Exhibit Number</u>	<u>Description</u>
1001	U.S. Patent No. 6,549,988
1002	File History of U.S. Patent No. 6,549,988 (Application No. 09/236,409)
1003	U.S. Patent No. 5,644,751 (“ <i>Burnett</i> ”)
1004	Declaration of Dr. Paul Franzon, Ph.D.
1005	CV of Dr. Paul Franzon, Ph.D.
1006	U.S. Patent No. 6,173,374 (“ <i>Heil</i> ”)
1007	U.S. Patent No. 5,920,893 (“ <i>Nakayama</i> ”)
1008	Reserved
1009	Macklem, R., <i>Not Quite NFS, Soft Cache Consistency for NFS</i> , Proceedings of the Winter 1994 USENIX Conference, January 1994
1010	United States District Courts – National Judicial Caseload Profile, https://www.uscourts.gov/sites/default/files/fcms_na_distprofile0331.2022.pdf (accessed September 21, 2022)

1011	August 22, 2022 Order Staying Case in <i>LS Cloud Storage Technologies, LLC v. Google LLC</i> , 1:22-cv-00853 (W.D. Tex. 2022)
1012	USPTO Memo: Interim Procedure for Discretionary Denials in AIA Post-Grant Proceedings with Parallel District Court Litigation, June 21, 2022
1013	Order Setting Initial Pretrial Conference for 11/3/2022 in <i>LS Cloud Storage Technologies, LLC v. Google LLC</i> , 1:22-cv-00853 (W.D. Tex. 2022)
1014	D. Patterson, G. Gibson, R. Katz, entitled “A Case for Redundant Arrays of Inexpensive Disks (RAID)”. SIGMOD88: International Conference On Management of Data Chicago Illinois USA June 1 - 3, 1988
1015	R. H. Katz, "Network-attached storage systems," <i>Proceedings Scalable High Performance Computing Conference SHPCC-92.</i> , Williamsburg, VA, USA, 1992, pp. 68-75
1016	Gang Ma and A. L. Narasimha Reddy, "An evaluation of storage systems based on network-attached disks," <i>Proceedings. 1998 International Conference on Parallel Processing (Cat. No.98EX205)</i> , Minneapolis, MN, USA, 1998, pp. 278-285. doi: 10.1109/ICPP.1998.708497

1017	Google LLC's Motion to Stay in <i>LS Cloud Storage Technologies, LLC v. Google LLC</i> , 1:22-cv-00853 (W.D. Tex. 2022)
1018	Srinivasan, V., and Mogul, J., <i>NFS: Experiments with Cache Consistency Protocols</i> , Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, December 1989
1019	November 7, 2022 Scheduling Order in <i>LS Cloud Storage Technologies, LLC v. Google LLC</i> , 1:22-cv-00853 (W.D. Tex. 2022)
1020	Declaration of Dr. James Mullins, Ph.D.
1021	Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, December 1989
1022	Srinivasan, V., and Mogul, J., <i>NFS: Experiments with Cache Consistency Protocols</i> , Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, December 1989
1023	Screen capture of WorldCat showing availability of <i>Operating systems review</i>
1024	Screen capture of Library Catalog showing availability of <i>Operating systems review</i>

1025	Shinjo, Y. and Kiyoki, Y., <i>The Object-Stacking Model for Structuring Object-Based Systems</i> , Proceedings of the Second International Workshop on Object Orientation in Operating Systems, September 1992
1026	Wang, R. and Anderson, T., <i>xFs: A Wide Area Mass Storage File System</i> . Proceedings of IEEE 4th Workshop on Workstation Operating Systems, October 1993
1027	Conference Proceedings - USENIX Winter 1994 Conference, January 1994
1028	Macklem, R., <i>Not Quite NFS, Soft Cache Consistency for NFS</i> , Proceedings of the Winter 1994 USENIX Conference, January 1994
1029	Screen capture of WorldCat showing availability of <i>Proceedings of the Winter 1994 USENIX Conference</i>
1030	Screen capture of Libraries Search showing availability of <i>Proceedings of the Winter 1994 USENIX Conference</i>
1031	Chen, J., Kung, H., and Seltzer, M., <i>MOSS: A Mobile Operating System Substrate</i> , Center for Research in Computing Technology, 1995

1032	Cortes Rossello, Cooperative Caching and Prefetching in Parallel/Distributed File Systems, Universitat Politècnica de Catalunya, 1997
1033	Screen capture of Universitat Politècnica de Catalunya showing availability of <i>Cooperative Caching and Prefetching in Parallel/Distributed File Systems</i>

LISTING OF CHALLENGED CLAIMS

<u>Reference</u>	<u>Claim Limitation</u>
Claim 1	
[1.pre]	A computer suitable for use in a data storage system comprising a network interconnecting a plurality of such computers, the computer comprising:
[1.a]	an I/O channel adapter for accepting an incoming I/O request from a host;
[1.b]	configuration manager software for enabling said I/O channel adapter to decide whether (i) to route said request to cache, (ii) to route said request to disk, or (iii) to reject said request;
[1.c]	a network adapter for handling network control traffic;
[1.d]	a cache memory;
[1.e]	front-end software for handling I/O requests arriving at the I/O channel adapter or the network adapter;

[1.f]	cache manager software, responsive to said front-end software, for handling data stored in said cache memory; and
[1.g]	back-end software, responsive to said configuration manager software, for handling reads and writes to disks corresponding to the I/O requests but without communication over the I/O channel adapter, thereby separating disk operations from network and I/O traffic.

Claim 2	
[2.pre]	The system of claim 1 wherein the computers comprise off-the-shelf hardware and operating systems and further comprise:
[2.a]	an adapter I/O software for accepting incoming I/O requests from a host; and
[2.b]	a volume access table employed by the configuration manager to ensure consistency of data stored on the network.

Claim 3	
[3.pre]	The system of claim 1
[3.a]	wherein the cache memory comprises a portion of a distributed cache memory stored in the computers interconnected by the network.

Claim 4	
[4.pre]	The system of claim 3 further comprising
[4.a]	a volume access table employed by the configuration manager to ensure consistency of data stored in the distributed cache.

Claim 5	
[5.pre]	The system of claim 4
[5.a]	wherein the configuration manager includes software that checks an access mode in the volume access table and
[5.b]	(i) if the access mode is set to an exclusive mode, causes both reads and

	writes to be stored in the cache memory, and causes invalidate messages to be sent to remote storage systems;
[5.c]	(ii) if the access mode is set to shared, causes only reads to be stored in the cache memory; and
[5.d]	(iii) if the access mode is set to a value other than exclusive or shared, causes reads and writes to be performed directly to a disk without using the cache memory.

Claim 6	
[6.pre]	A method of accessing a remote disk over a computer network without incurring network overhead, the method comprising the steps of:
[6.a]	a. causing a local host to issue a request over an I/O channel to a local computer;
[6.b]	b. providing a configuration manager on the local computer, the

	configuration manager routing the request to a remote computer via the computer network;
[6.c]	c. causing the remote computer to check the request against a volume access table;
[6.d]	d. causing the remote computer to perform an I/O operation on a disk located on the remote computer and to return data to the local computer;
[6.e]	e. causing the local computer to provide the returned data to the local host via the I/O channel; and
[6.f]	f. causing the local computer to check the data against the volume access table to ensure consistency of the data on the local and the remote computers.

I. INTRODUCTION

Google LLC (“Google” or “Petitioner”) requests *inter partes* review (IPR) of claims 1-6 (the “Challenged Claims”) of U.S. Patent No. 6,549,988 (the “’988 Patent”) assigned to LS Cloud Storage Technologies LLC (“PO”).

The ’988 Patent relates to cached data storage systems and claims a computer for a data storage system that receives I/O requests from a host via an I/O channel and routes the I/O requests to a cache or disk. The computer may also receive I/O requests from other networked computers via a network adapter, and further includes software for handling reads and writes to cache and disk storage. The ’988 Patent additionally claims a method for remotely accessing the cache and disk of such an apparatus.

As the ’988 Patent admits however, such data storage and caching architectures were already well known. “Numerous prior art systems and methods exist for managing cache memory in a data storage system,” and a “typical data processing system generally involves a cached data storage system that connects to local host computers via I/O channels or remote host computers via network links.” Ex.1001, 1:46-48, 1:19-23. Indeed, U.S. Patent 6,173,374 to Heil discloses a system and method of accessing a remote disk over a computer network by a local host by issuing requests from the local host over an I/O channel to a local host-bus

adapter (“HBA”), and then rerouting the request from the local HBA to remote HBAs over a computer network.

Google respectfully requests review and invalidation of the Challenged Claims based on the grounds as set forth below.

II. MANDATORY NOTICES

A. Real Party-in-Interest

Google LLC is the real party-in-interest for this petition.¹

B. Related Matters

PO asserted the ’988 Patent against Petitioner and other parties in the following cases in the Waco Division of the Western District of Texas:

- *LS Cloud Storage Technologies, LLC v. Google LLC*, 1:22-cv-00853 (W.D. Tex. 2022)
- *LS Cloud Storage Technologies, LLC v. Amazon.com, Inc.*, 6:22-cv-00316 (W.D. Tex. 2022)

¹ Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

- Transferred to the Austin Division as *LS Cloud Storage Technologies, LLC v. Amazon.com, Inc.*, 1:22-cv-01167 (W.D. Tex. 2022), and dismissed with prejudice on March 16, 2023.
- *LS Cloud Storage Technologies, LLC v. Cisco Systems, Inc.*, 6:22-cv-00319 (W.D. Tex. 2022)
 - Dismissed and refiled as *LS Cloud Storage Technologies, LLC v. Cisco Systems, Inc.*, 6:22-cv-00845 (W.D. Tex. 2022)
- *LS Cloud Storage Technologies, LLC v. Microsoft Corporation*, 6:22-cv-00321 (W.D. Tex. 2022)
 - Transferred to the Austin Division as *LS Cloud Storage Technologies, LLC v. Microsoft Corporation*, 1:22-cv-00974 (W.D. Tex. 2022)

The '988 Patent is also the subject of the following *inter partes* review:

- *Microsoft Corporation v. LS Cloud Storage Technologies, LLC*, IPR2023-00622 (PTAB 2023)

The original complaint in the Google litigation did not assert the '988 patent. On December 2, 2022, PO filed an Amended Complaint alleging, for the first time, infringement of the '988 Patent, without obtaining Google's consent or seeking the requisite leave to amend from the Court. PO's amended complaint newly asserting the '988 patent against Google is subject to a motion to dismiss, in part, and strike for violation of the local rules and Federal Rules of Civil Procedure.

C. Counsel and Service

Petitioner appoints the following lead and back-up counsel:

<u>Lead Counsel</u>	<u>Back-up Counsel</u>
James T. Carmichael Reg. No. 45,306 Carmichael IP, PLLC 8607 Westwood Center Drive Suite 270 Tysons, VA 22182 Tel: 703-646-9255 Email: jim@carmichaelip.com	Minghui Yang Reg. No. 71,989 Carmichael IP, PLLC 8607 Westwood Center Drive Suite 270 Tysons, VA 22182 Tel: 703-646-9248 Email: mitch@carmichaelip.com

Petitioner consents to service by electronic mail at the following addresses:

jim@carmichaelip.com;

mitch@carmichaelip.com; and

Google-LS-Cloud-IPR@Carmichaelip.com.

D. Power of Attorney

A power of attorney is filed herewith in accordance with 37 C.F.R.
§42.10(b).

E. Fees

Petitioner concurrently electronically submits the required fees for this
Petition.

III. REQUIREMENTS FOR *INTER PARTES* REVIEW

A. Grounds for Standing

Petitioner certifies that the '988 Patent is available for IPR, and that Petitioner is not barred or estopped from requesting IPR.

B. Identification of Challenge and Relief Requested

Petitioner requests cancellation of the Challenged Claims on the following grounds:

Ground 1: Obviousness of claims 1-4 and 6 based on U.S. Patent 6,173,374 (“*Heil*”) published January 9, 2001 and U.S. Patent 5,920,893 (“*Nakayama*”) published July 6, 1999, under 35 U.S.C. § 103(a); and

Ground 2: Obviousness of claim 5 based on Heil, Nakayama and *Not Quite NFS, Soft Cache Consistency for NFS*, Proceedings of the Winter 1994 USENIX Conference, January 1994 to Macklem (“Macklem”) and *NFS: Experiments with Cache Consistency Protocols*, Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, December 1989 to Srinivasan *et al.* (“Srinivasan”), under 35 U.S.C. § 103(a).

Heil and Nakayama are prior art pursuant to at least (pre-AIA) 35 U.S.C. §102(e). Macklem and Srinivasan are both printed publications publicly available more than one year prior to the priority date of the '988 Patent and thus prior art pursuant to at least (pre-AIA) 35 U.S.C. §102(b). Ex.1020. These grounds are

supported by the Declaration of Dr. Paul Franzon, Ph.D., an expert in the field of computer networking and cache management. Ex.1004. None of Heil, Nakayama, Macklem, or Srinivasan was cited during prosecution.

IV. THE '988 PATENT

The '988 Patent relates to cached data storage systems and describes an apparatus and method for managing a distributed cache across a network using host computers to read and write to cache memories or disk storages. Ex.1001, abstract. Figure 1 shows the preferred embodiment in which hosts 111 and 112 are connected via I/O channels 121 and 122 to PC storage system 131. Ex.1001, 4:67-5:6. PC storage system 131 includes cache memory managed by a cache manager and disks 161 and 162. Ex.1001, 5:15-23, 5:38-50. When the host accesses the cache or storage disk via an I/O channel, the '988 Patent considers the host to be a local host. Ex.1001, 1:18-25.

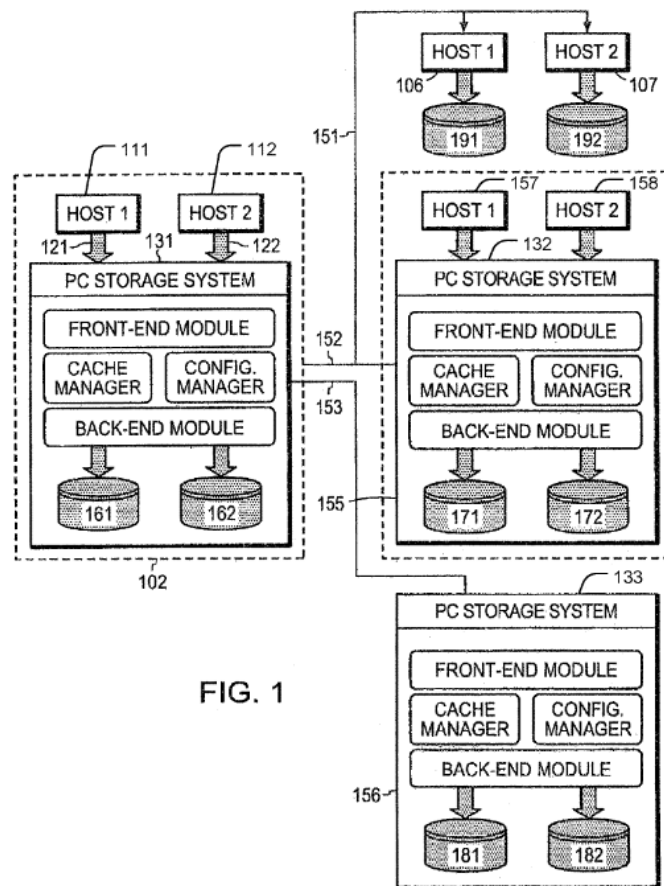


FIG. 1

Ex.1001, Fig. 1.

Alternatively, hosts 111 and 112 may access remote storage system 133 via network connection 153. Ex.1001, 4:66-5:5, 5:15-23. Remote storage system 133 also includes a cache memory managed by a cache manager and disks 181 and 182. Ex.1001, 5:15-23, 5:38-50. When the host accesses the cache or storage disk of a storage system via a network connection, the '988 Patent considers the host to be a remote host. Ex.1001, 1:18-25.

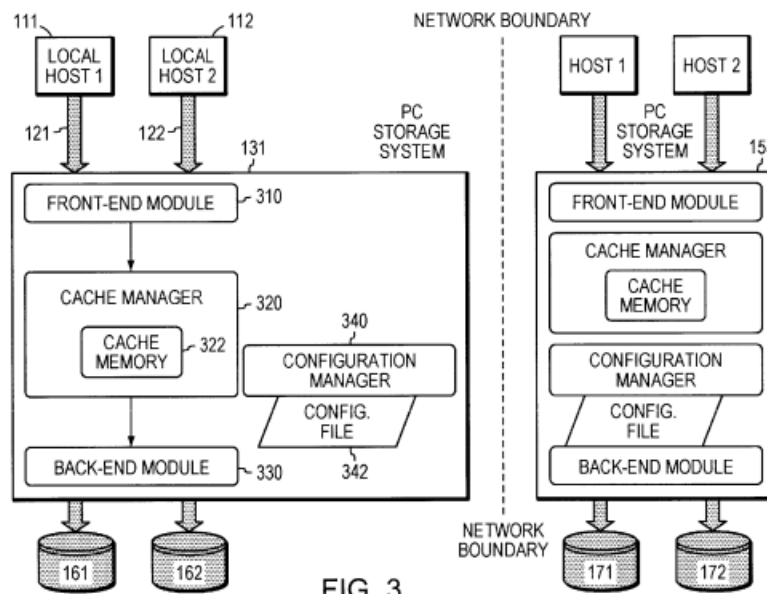


FIG. 3

As Figure 3 shows above, cache manager 320 comprises cache memory 322. Ex.1001, 5:38-50. Cache manager 320 is connected to hosts 111 and 112 by front-end module 310 and is connected to disks 161 and 162 by back-end module 330. Ex.1001, 5:38-50, 5:58-65 (presence of cache memory permits the front-end channels to operate independently of the back-end disk devices). The '988 Patent describes a volume access table for mapping data stored on the network: "Volume access table (450) in FIG. 4 contains a mapping between hosts and volumes specifying an access mode value. If the access mode is set to neither shared nor exclusive configuration manager forwards I/O requests directly to disk. In addition to the access mode said volume access table may contain other values that help to manager and improve performance of said data storage system." Ex.1001, 6:15-21. The volume access table stores values which when modified changes the behavior

of the cache manager: “a user[] actively monitors and changes the behavior of a cache manager by changing values in a volume access table in order to improve performance of said data storage system.” Ex.1001, 6:38-42.

V. PROSECUTION HISTORY

The '988 Patent issued on April 15, 2003, from U.S. Application No. 09/236,409, filed January 22, 1999. The Office issued non-final rejections on November 20, 2000 and March 26, 2002 and a final rejection on June 28, 2002. Ex.1002, 114, 134, 190. In response to the final rejection, applicant argued that “a key feature [of the invention] is the front-end/back-end separation” of the disk operations and the I/O channel, and the Office ultimately allowed the application. *Id.*, 77. The '409 Application issued as the '988 Patent following a Notice of Allowance on December 3, 2002. Ex.1002, 55. Despite applicant's representations concerning the claimed front-end/back-end separation, systems having separate front and back-ends were known in the art and, indeed, none of the prior art references used in this Petition, which as explained below disclose this and other limitations, was discussed or of record during prosecution of the '409 Application.

VI. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. State of the Art

As the '988 Patent's prosecution history and specification confirm, data storage and caching architectures like those disclosed by the '988 Patent were well

known in the art. “Numerous prior art systems and methods exist for managing cache memory in a data storage system,” and a “typical data processing system generally involves a cached data storage system that connects to local host computers via I/O channels or remote host computers via network links.” Ex.1001, 1:46-49, 1:19-24. For example, even in the early 90s, storage systems in which hosts are connected to peripheral devices were well-known. Ex.1004, ¶¶36-50. During this time, storage systems transitioned from node-based storage architectures to network-based architectures that connected hosts to multiple storage servers over a network. *Id.* As a result of this paradigm shift, hosts and storage devices were frequently connected together through a network, another claimed feature of the ’988 Patent. *Id.* Similarly, caching in network-based storage was well-known. Ex.1004, ¶¶47-50.

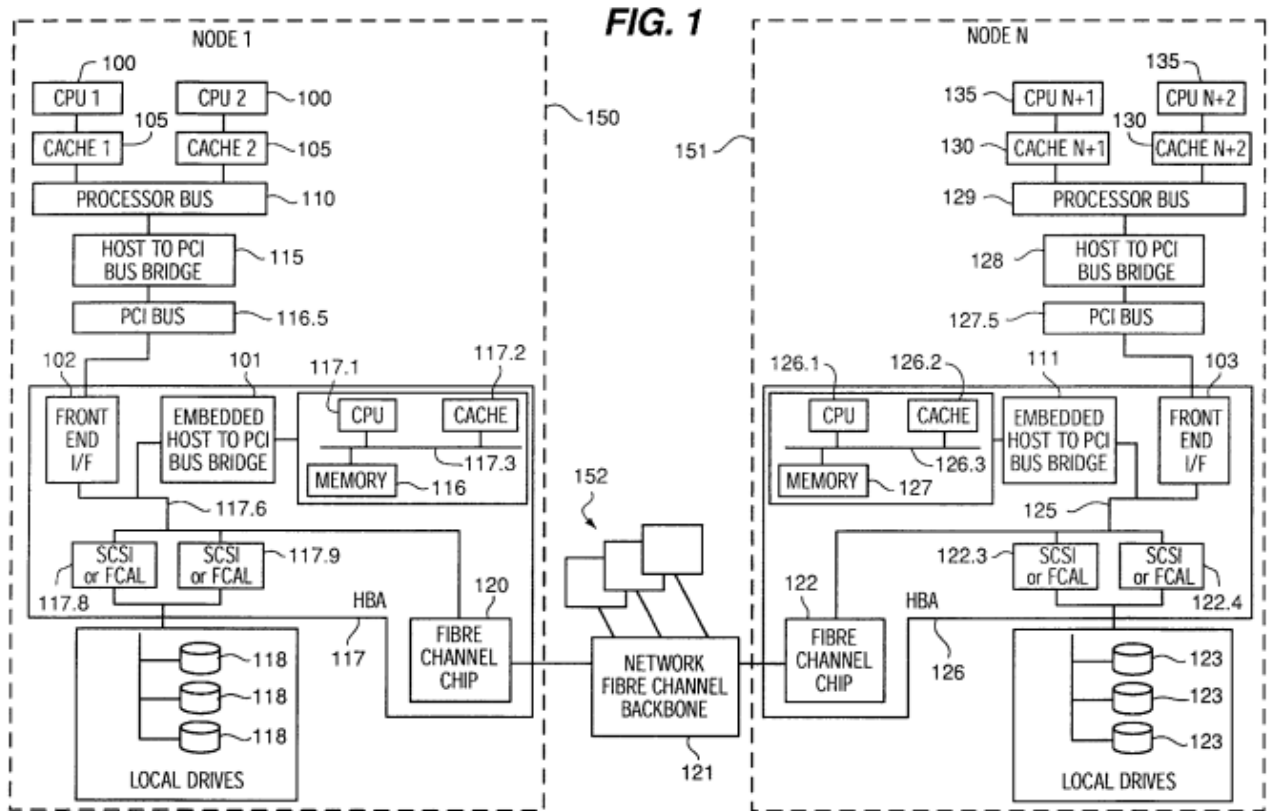
The art at the time of invention also recognized the importance of maintaining data consistency in distributed cache architectures. U.S. Patent 5,644,751 to Burnett et al. discussed by the ’988 Patent as prior art at 2:59-60 describes the importance of maintaining file integrity: “if one of the several nodes is also writing to the file, the client node reading the file may not be accessing the latest updated file that has just been written. As such, the file integrity is lost.” Ex.1003, 1:19-23. Burnett describes DFS as a known solution for distributed caches. *Id.*, 1:23-42. Both Macklem (Ex.1009) and Srinivasan (Ex.1018) published

several years before the '988 Patent similarly suggest that permitting multiple computers to access the same storage may degrade system performance and reduce accuracy. Ex.1004, ¶51. Thus, it was well known in the art at the time of invention to control access to cache and storage such as by “locking” a file, certain volume(s), or certain block(s) and limiting the number of computers which may access the locked file, volume(s), or block(s) at a time. *Id.*

B. Overview of the Prior Art

1. Heil

U.S. Patent 6,173,374 to Heil et al. was filed February 11, 1998 and issued January 9, 2001. Like the '988 Patent, Heil is directed to data storage systems and describes an apparatus and method in which host computers store files in cache or disk storage. Ex.1006, abstract, 1:9-14, 6:34-47; Ex.1004, ¶¶58-60. Heil's Figure 1 shows a configuration remarkably like that shown by Figure 1 of the '988 Patent. In Heil's Figure 1, a host system having a CPU 100 is connected via a PCI bus 116.5 to a host bus adapter (HBA) 117. Ex.1006, 6:34-47; Ex.1004, ¶59. Host bus adapter 117 includes cache memory 117.2 and local memory 116 and is further connected to local drives 118 via interfaces 117.8 and 117.9. Ex.1006, 7:9-15, 7:51-56. The host may process I/O requests locally by accessing the host bus adapter 117 via PCI bus 116.5. Ex.1006, 4:51-57, 6:34-47; Ex.1004, ¶59.



Ex.1006, Figure 1.

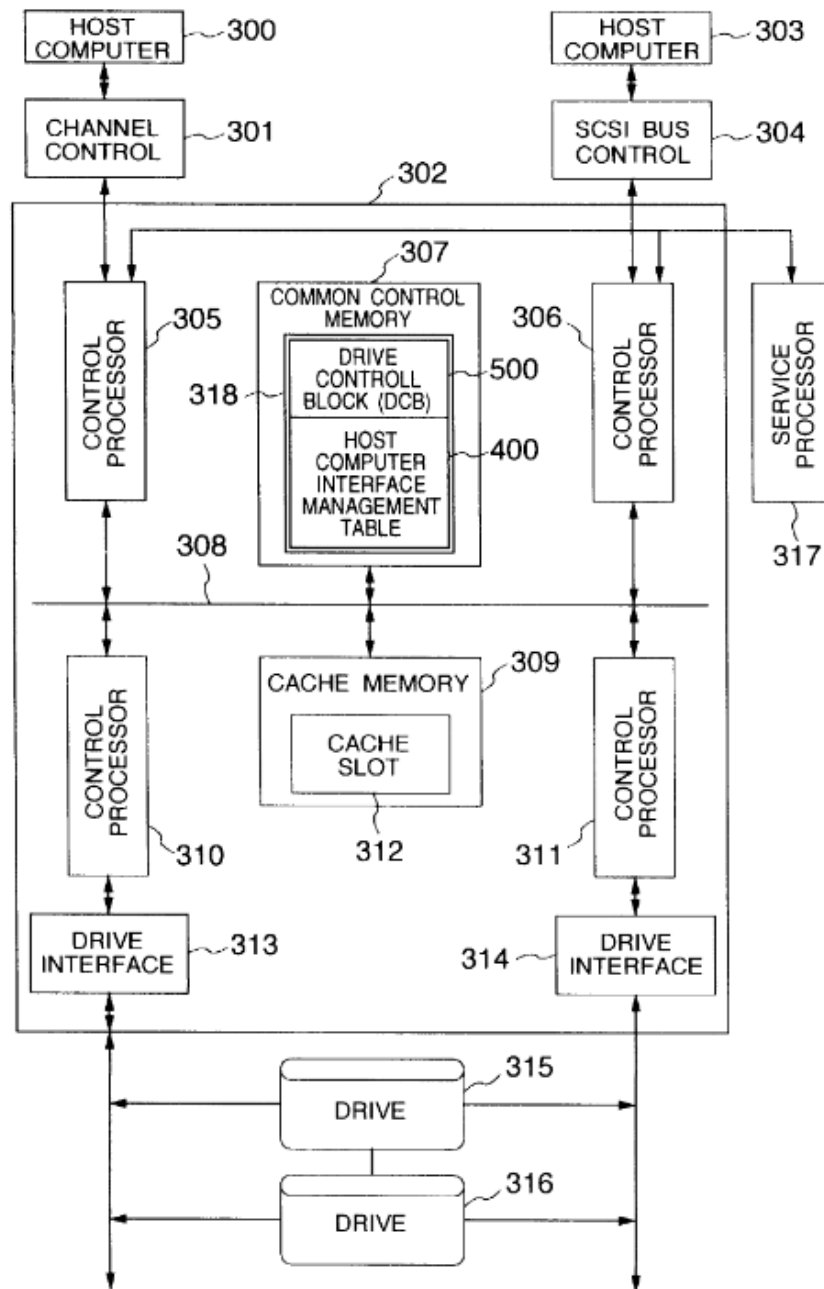
Alternatively, the host (shown in node 1) may access another, remote storage system (shown in node n) via network connection 121. Ex.1006, 8:8-22; Ex.1004, ¶60. The remote storage system of node n also includes cache memory 126.2 and memory 127. Ex.1006, 8:61-9:10. The host bus adapter 117 has a CPU 117.1 which may store data either to cache memory 117.2 or disk drives 118. Ex.1006, 7:21-34. The host bus adapter CPU 117.1 is connected to host CPU 100 via the PCI bus 116.5. Ex.1006, 9:34-47.

Heil recognizes that problems arise when multiple hosts are permitted uncontrolled access to the same storage: “Problems arise where multiple computers, operating in parallel, share data and storage devices. Clearly, a level of coordination is required to assure that each of the computers is aware of updates in the storage devices made by others of the computers in the cluster.” Ex.1006, 1:44-53; Ex.1004, ¶70. Thus, Heil explicitly contemplates architectures to coordinate host access to shared storage. *Id.* To resolve this, Heil further discloses that the consistency of the database may be maintained by a “Oracle Parallel Database Server” including a “distributed lock manager (DLM)” by regulating access to disk blocks: “The Oracle Parallel Database Server maintains the consistency of the database by utilizing a proprietary protocol, the distributed lock manager (DLM), to allow nodes to access the shared storage concurrently. Utilizing DLM software in a cluster of physically shared disks allows all computer nodes to have access to all disks directly through their own I/O subsystem so that each disk appears to be physically local. Each computer node can cache and/or lock shared disk-based structures utilizing the DLM software. For example, if one node wants blocks X, Y, and Z within disks A and B, it must first ask the DLM software for permission. The DLM will grant permission only after it has endured that blocks X, Y, and Z are current.” Ex.1006, 2:18-32.

2. Nakayama

U.S. Patent 5,920,893 to Nakayama et al. was filed June 2, 1997 and issued July 6, 1999. Like the '988 Patent, Nakayama is directed to cached data storage systems and describes an apparatus and method in which host computers provide input to cache memories or storage drives. Ex.1007, abstract, 4:52-65; Ex.1004, ¶¶61-63. Nakayama's Figure 3 shows host computer 300 connected via channel 301 to disk control 302 having cache memory 309. Ex.1007, 4:40-5:9.

FIG. 3



Disk control 302 is further connected to drives 315 and 316 via drive interface 313.

Ex.1007, 4:52-65.

As part of Nakayama's drive access system, Nakayama describes storing data indicating that a host has reserved exclusive access to a drive: "The interhost exclusive information 403 is used when the host computer 300 or 303 exclusively controls the drive access from the other host computer and sets the information 403 'on' when the processor reserves access of the specified drive number and sets it 'off' when the host computer releases the access right." Ex.1007, 5:42-47.

3. Srinivasan

NFS: Experiments with Cache Consistency Protocols is a conference paper presented at the Proceedings of the Twelfth ACM Symposium on Operating Systems Principles in December 1989 by Srinivasan et al. Srinivasan describes protocols for improving cache consistency and performance including open and close file states that limit write operations on a file to a specific client. Ex.1018, 43, 46-47; Ex.1004, ¶64. Srinivasan describes a series of flags that permits a file in cache to be open for reads or open for reads and writes. Ex.1018, 50-51. Srinivasan also discloses a "write-shared" state in which the file is not cached but rather directly written to disk. *Id.*

4. Macklem

Not Quite NFS, Soft Cache Consistency for NFS is a conference paper presented at the Proceedings of the Winter 1994 USENIX Conference in January, 1994 by Macklem. Macklem also describes protocols for improving cache

performance. Ex.1009, 261; Ex.1004, ¶¶65. Macklem describes a read-caching lease that permits a client to read a cached file, and a write-caching lease that permits a client to read and write to a cached file. Ex.1009, 264. Additionally, Macklem describes a third type of procedure—non-caching—that permits the client to access the stored file without caching. Ex.1009, 264, 267.

C. Level of Ordinary Skill in the Art

A person of ordinary skill in the art (“POSITA”) as of the earliest potential priority date of the ’988 Patent would have had a bachelor’s degree in Computer Science, Computer Engineering, or Electrical Engineering, or an equivalent course of study, as well as 2 or more years of academic or industry experience in the field of computer networking, computer architecture and storage management. Ex.1004, ¶¶33-35. More education can supplement practical experience, and vice versa. *Id.* A POSITA would also have had knowledge of the prior art admitted by the ’988 Patent. *Id.*

D. Claim Construction

Claim terms are construed in accordance with their ordinary and customary meaning as understood by a POSITA and the prosecution history pertaining to the patent. 37 C.F.R. §42.100(b). Petitioner is unaware of any “prior claim construction determination” related to the ’988 patent.

For the purposes of this Petition, Petitioner interprets all terms of claims 1-6 in accordance with their ordinary and customary meaning.

E. Ground 1: Claims 1-4 and 6 are obvious over Heil and Nakayama under 35 U.S.C. § 103(a)

1. A POSITA Would Have Combined Heil and Nakayama

It would have been obvious for a POSITA to combine Heil and Nakayama, with a reasonable chance of success in doing so. Ex.1004, ¶¶68-72. Heil and Nakayama are in the same field of invention and disclose similar cache and storage architectures. *Id.*, ¶68 (“Heil ‘relates to distribution of storage data within computer nodes’ [and] Nakayama relates to a ‘storage control system’ ‘for storing input/output data to/from host computers.’”). Both Heil and Nakayama recognize the importance of cache storage in addition to disk storage, and both disclose methods of reading and writing to cache and disk storage from a processor. *Id.* A POSITA seeking to improve storage access for a host computer in a storage system having both cache and disk storage would be motivated to incorporate methods of managing storage systems, such as Nakayama’s, when implementing or improving the storage functionality described by Heil. *Id.*, ¶¶69-72.

Additionally, Heil recognizes that problems arise when multiple hosts are permitted uncontrolled access to the same storage: “Problems arise where multiple computers, operating in parallel, share data and storage devices. Clearly, a level of coordination is required to assure that each of the computers is aware of updates in

the storage devices made by others of the computers in the cluster.” Ex.1006, 1:44-53. To resolve this, Heil discloses that database consistency may be maintained by a “Oracle Parallel Database Server” including a “distributed lock manager (DLM)” by regulating access to disk blocks: “The Oracle Parallel Database Server maintains the consistency of the database by utilizing a proprietary protocol, the distributed lock manager (DLM), to allow nodes to access the shared storage concurrently. Utilizing DLM software in a cluster of physically shared disks allows all computer nodes to have access to all disks directly through their own I/O subsystem so that each disk appears to be physically local. Each computer node can cache and/or lock shared disk-based structures utilizing the DLM software. For example, if one node wants blocks X, Y, and Z within disks A and B, it must first ask the DLM software for permission. The DLM will grant permission only after it has endured that blocks X, Y, and Z are current.” Ex.1006, 2:18-32. As Heil recognizes that unrestricted access to disk drives degrades the consistency of disk storage and explicitly proposes locking drives against access by multiple hosts, a POSITA would recognize that Nakayama’s methods of controlling host access to storage are applicable to Heil and may similarly be used to control host access to improve consistency. Ex.1004, ¶69-72. Indeed, the POSITA would understand that Nakayama provides alternatives and improvements to managing host access to cache and disk storage that are applicable or improvements to Heil, such as by

using Nakayama's read and write operations to access cache and disk storage as in Heil. *Id.*, ¶72 ("Heil describes how 'a large number of processors may be added to a computing environment all of which share access to common storage devices.' Furthermore, Heil discusses 'file level access' and 'block level access.' Nakayama discloses storing data indicating whether a host has exclusive access to a drive (i.e., the host reserves the drive, thereby controlling access to it and excluding other host computers from accessing the drive): 'The interhost exclusive information 403 is used when the host computer 300 or 303 exclusively controls the drive access from the other host computer and sets the information 403 'on' when the processor reserves access of the specified drive number and sets it 'off' when the host computer releases the access right.'").

Thus, a POSITA would implement Heil in view of Nakayama by implementing the details of the disk storage management system of the latter on the former. For instance, a POSITA would incorporate the rejection of requests as taught by Nakayama to Heil's I/O routing routines because Nakayama provides the motivation to do so, "including to prevent simultaneous use of the same storage medium or to prevent a drive from being predominantly used by a single host." Ex.1004, ¶86.

2. Claim 1

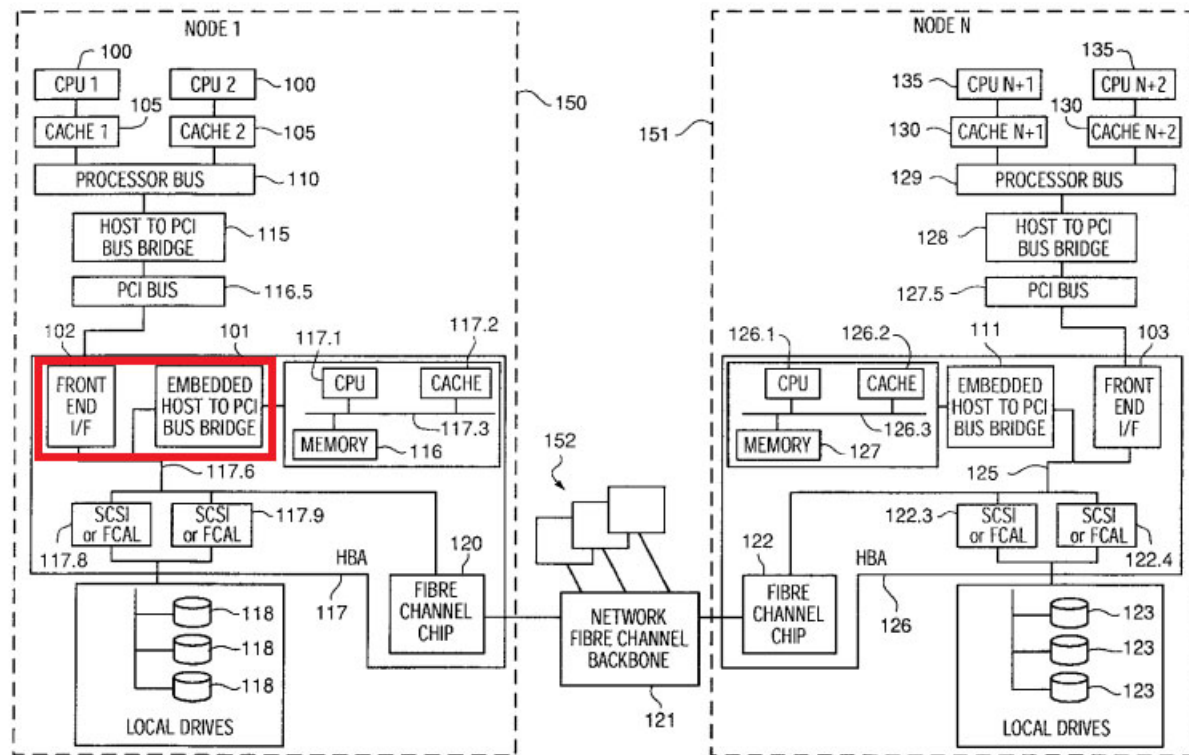
The claimed computer is obvious in view of Heil and Nakayama. Ex.1004, ¶¶73-114.

i. Claim 1.pre

If the Board determines that the preamble is limiting, Heil discloses the preamble. Ex.1004, ¶¶74-75. Heil is titled a “System and method for peer-to-peer accelerated I/O shipping between host bus adapters in clustered computer network” and discloses a “common application of such a clustered environment is for the sharing of disk storage resources.” Ex.1006, 1:36-37. Heil’s intelligent host bus adapter (“HBA”) is a computer suitable for use in a data storage system because it is “a device that adapts (connects) a host computer system to an I/O device[,] includes a CPU, local memory, and cache memory and has embedded intelligence that allows it to function as an intelligent PCI-to-PCI bridge and as a disk drive controller for local disks.” Ex.1004, ¶75; Ex.1006, 4:2-4, 7:9-15, 6:65-7:9. The HBA is part of a network interconnecting a plurality of such computers because the “HBAs form nodes which are interconnected by a fibre channel, such that an HBA may ship I/O requests among the HBAs of the network to be serviced.” Ex.1004, ¶75; Ex.1006, 4:29-39. Heil therefore discloses the preamble.

ii. Claim 1.a

Heil discloses element 1.a. Heil discloses an embedded Host to PCI bus bridge 101, PCI bus bridge 115, PCI bus 116.5, and a front-end interface 102 that accepts I/O traffic (including I/O requests) from the host via the I/O channel, all of which are I/O channel adapters for accepting an I/O request from a host. Ex.1004, ¶¶76-81. A POSITA would have found it obvious to implement the functionality for an I/O channel adapter within any of the components or to distribute the functionality across all the components. Ex.1004, ¶79. All of Heil's I/O channel adapter elements are shown in Figure 1:

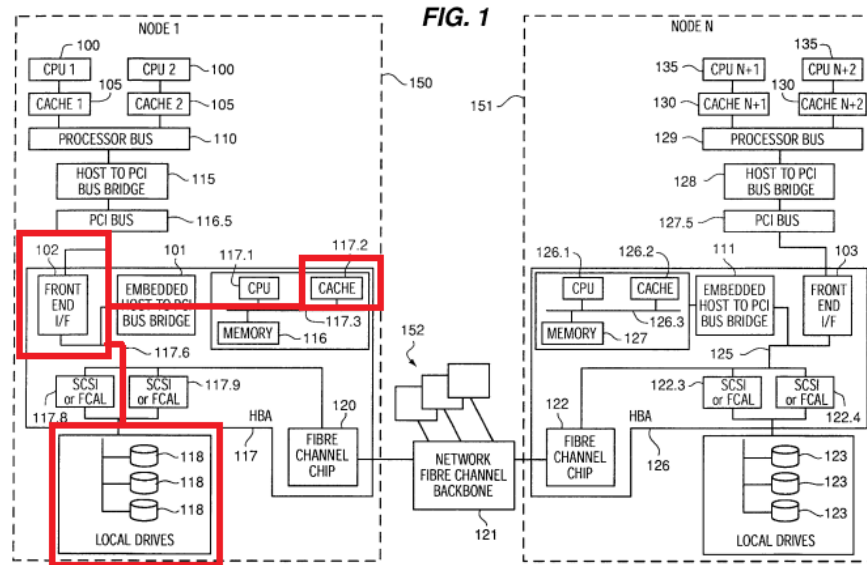


Ex.1006, Fig. 1. Figure 1 shows that HBA 117 receives I/O traffic from the Node 1 host having CPU 100 via an I/O channel between the host and the HBA. Ex.1004,

¶¶78-80; Ex.1006, 6:35-40 (“Node 1 150 includes a host system connected to a HBA 117 via a Host-to-Peripheral Component Interconnect (hereinafter referred to as “PCI”) bus bridge 115 and PCI bus 116.5.”). “[C]ommunications between the Node 1 host and HBA 117 are I/O requests [instructing the HBA] to retrieve files from caches or local storage devices.” Ex.1004, ¶78 (*citing* Ex.1006, 7:36-37, 2:59-65, 4:63-69, 12:4-6); 4:40-41 (“The intelligent HBA(s) of the present invention process I/O requests received from the host system.”); 4:50-53 (“In the preferred embodiment, an HBA receives a block I/O request from the host and Searches through the directory in HBA memory to determine whether the block can be found locally or remotely.”). Thus, Heil discloses element 1.a.

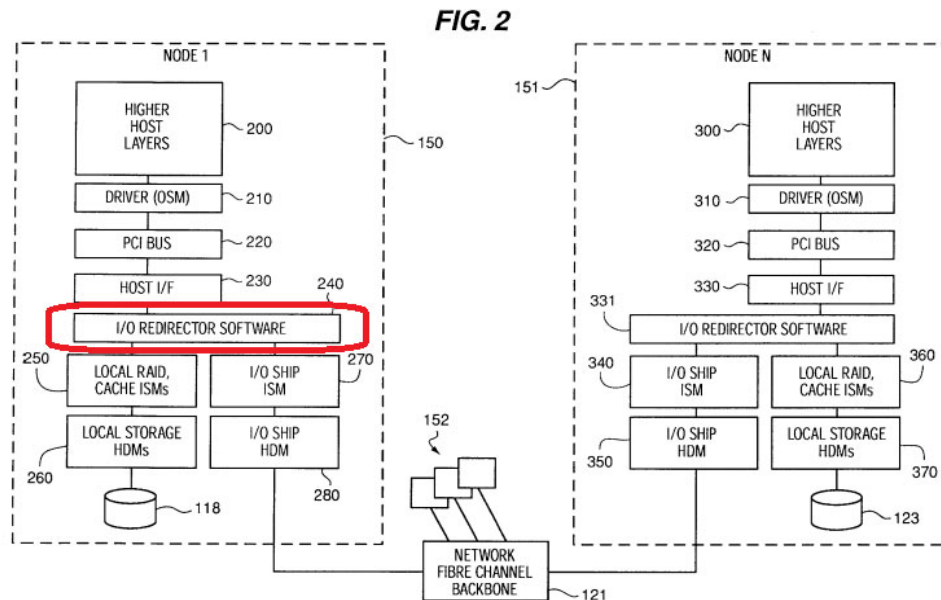
iii. Claim 1.b

Heil in view of Nakayama renders element 1.b obvious. Ex.1004, ¶¶82-88. Heil Figure 1 shows that I/O requests are routed from front-end interface 102 to either cache 117.2 or local drives 118:



Ex.1004, ¶¶83-85 (describing write requests, including data to be written, sent from the host node through Heil’s front-end interface to either cache or local drives); Ex.1006, 7:24-30 (“To improve I/O performance, CPU 117.1 temporarily caches disk data in cache memory 117.2 that is frequently accessed. Later, the CPU 117.1 stores the data permanently in the slower disk drive devices 118.”), 11:66-12:5 (“The remote HBA retrieves the requested blocks of data from its HBA cache, or if the blocks of data are not within cache, then directly from the remote HBA’s local disk.”). Although Heil describes the process of data retrieval from the HBA’s local disk if the data is not found in cache in the context of a remote HBA, a POSITA would understand that, because the architecture of each node is identical as shown in Fig. 1, that the process is equally applicable to a local HBA retrieving data from its cache or local disk. Ex.1004, ¶¶83,n.1. Heil discloses that an HBA uses a “managing means” to coordinate the retrieval of data. Ex.1006, 10:66-11:11

(“Each HBA 117, 126 contains managing means which includes the “Distributed block I/O redirector driver software” 240, 331 (hereinafter referred to as I/O redirector software), the I/O shipping ISM 270, 340 and the I/O shipping HDM 280, 350. The I/O redirector software 240, 331 provides the means to allow the HBA 117, 126 to make the decision whether to satisfy a block I/O request locally or remotely.”). Heil discloses that the “managing means” include I/O redirector software 240, which as shown in Fig. 2 is connected to cache ISMs 250 and local storage HDMs 260:



Ex.1006, Fig. 2, 12:19-34, 5:1-6. Thus, a POSITA would understand that I/O redirector software 240 processes I/O traffic received from the host and services I/O requests by enabling the I/O channel interface (such as the PCI bus bridge 115, PCI bus 116.5, and a front-end interface 102) to decide to route the request to

cache or disk storage based on whether the requested data is found in the cache or not. Ex.1004, ¶¶83-84 (*citing* Ex.1006, 11:66-12:1). To the extent Heil's redirector software determines how to route the I/O request, it would have been a simple design choice for the redirector software to provide the I/O channel interface the location of the requested data for the I/O channel interface to determine how to route the I/O request because Heil's components are explicitly designed to be modular. Ex.1004, ¶85; Ex.1006, 10:43-50. "Because Heil utilizes a directory accessible to the I/O redirector to map storage locations within the network, Heil's I/O redirector software directs the front-end interface to decide whether to route the I/O request to cache or to disk storage." Ex.1004, ¶¶83-85 (*citing* Ex.1006, 12:19-25, 12:34-42, 11:45-54); Ex.1006, 10:66-11:11 ("Each HBA 117, 126 contains managing means which includes the 'Distributed block I/O redirector driver software' 240, 331[...]. The I/O redirector software 240, 331 provides the means to allow the HBA 117, 126 to make the decision whether to satisfy a block I/O request locally or remotely."). Because Heil describes examples of write requests from the host nodes transmitted through the front-end interface of the HBAs to either disk storage or cache, the front-end I/O channel adapter components (such as the embedded Host to PCI bus bridge 101, PCI bus bridge 115, PCI bus 116.5, and front-end interface 102) are capable of routing I/O requests to either cache or disk, and decide where to route the I/O request (to cache

or disk) based on instructions provided by the I/O redirector software 240.

Ex.1004, ¶¶83-84. It would also have been a predictable variation or design choice to organize the functionalities disclosed in Heil in one or more different software modules. *Id.* For example, although Heil Figure 2 shows Local Storage HDMs 260 is connected to I/O redirector software 240 via Cache ISMs 250, a POSITA would understand that Heil's system does not require the I/O redirector software to communicate with the local storage HDMs through the cache ISMs. *Id.*

To the extent Heil's I/O redirector software does not explicitly reject an I/O request in addition to routing an I/O request to cache or disk, incorporating this additional functionality into Heil's I/O redirector software is suggested by Nakayama. Ex.1004, ¶¶85-86. As discussed above, a POSITA would understand that Heil is concerned with maintaining the consistency of stored data and limiting access, such as by rejecting an I/O request, is one manner of maintaining the consistency of stored data. *Id.* Applying Nakayama to Heil, when Nakayama's control processor 305 receives a data access command, the processor attempts to obtain access rights to a drive. Ex.1007, 6:11-14. If the processor fails to obtain access rights to the drive, data access ends and the I/O request is denied. *Id.* Heil itself provides examples of only granting permission (i.e., rejecting permission) when the stored data is ensured to be current: "The DLM will grant permission only after it has en[s]ured that blocks X, Y, and Z are current." Ex.1006, 2:28-31.

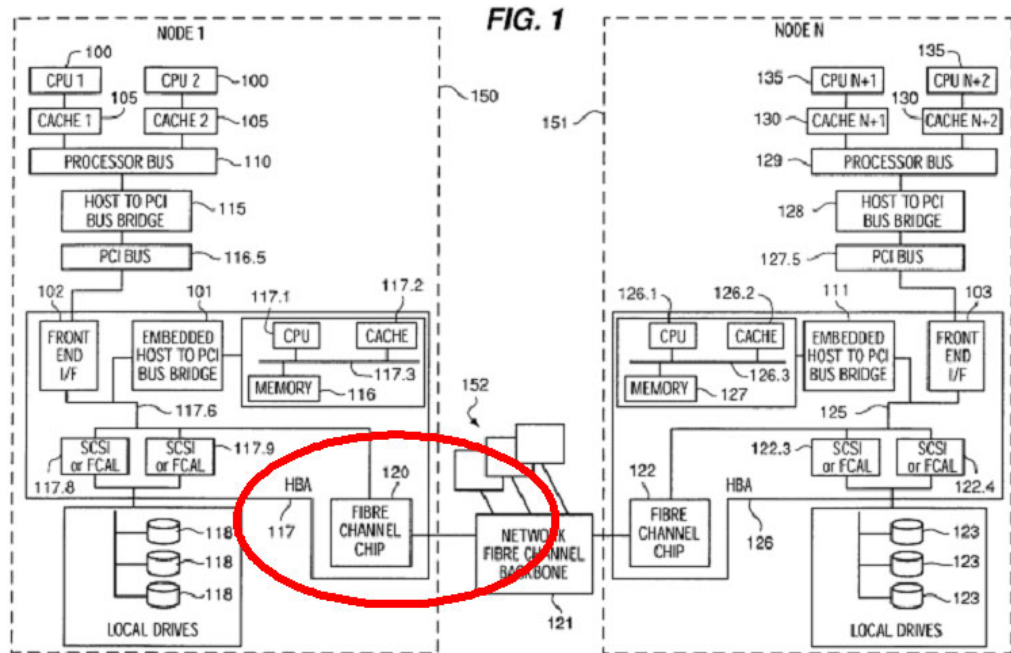
Additionally, Nakayama provides several reasons why limiting access to certain drives and rejecting I/O requests directed to those drives would be advantageous, including to prevent simultaneous use of the same storage medium or to prevent a drive from being predominantly used by a single host. Ex.1004, ¶86.

A POSITA would have been motivated to combine the teachings of Heil and Nakayama to incorporate Nakayama's teaching of rejecting I/O requests to Heil's I/O redirector software. *Id.*, ¶¶86-87. Heil recognizes that multiple hosts are permitted uncontrolled access to the same storage: "Problems arise where multiple computers, operating in parallel, share data and storage devices. Clearly, a level of coordination is required to assure that each of the computers is aware of updates in the storage devices made by others of the computers in the cluster." Ex.1006, 1:44-52. To resolve issues arising from multiple hosts accessing a shared storage, Heil discloses that the consistency of the database may be maintained by a "Oracle Parallel Database Server" including a "distributed lock manager (DLM)" by regulating access to disk blocks. Ex.1006, 2:18-32. As Heil recognizes that unrestricted access to disk drives degrades the consistency of disk storage and explicitly proposes locking drives against access by multiple hosts, a POSITA would recognize that Nakayama's methods of controlling host access to storage are applicable to Heil and may similarly be used to control host access to improve consistency. Ex.1004, ¶¶85-87. Moreover, "because Heil's I/O redirector handles

routing of the I/O request, a POSITA would logically incorporate rejection of requests as taught by Nakayama as part of the same I/O routing routine (i.e., in Heil's I/O redirector software)." *Id.* The POSITA would also have had a reasonable expectation of success because Heil and Nakayama utilize similar storage architectures such that the methods of reading and writing to cache of Nakayama are similarly applicable to Heil. *Id.* Thus, the POSITA would understand that Nakayama's procedure for rejecting an I/O request is equally applicable to Heil should Heil's cache or disk storage be similarly inaccessible as part of an access request. *Id.*

iv. Claim 1.c

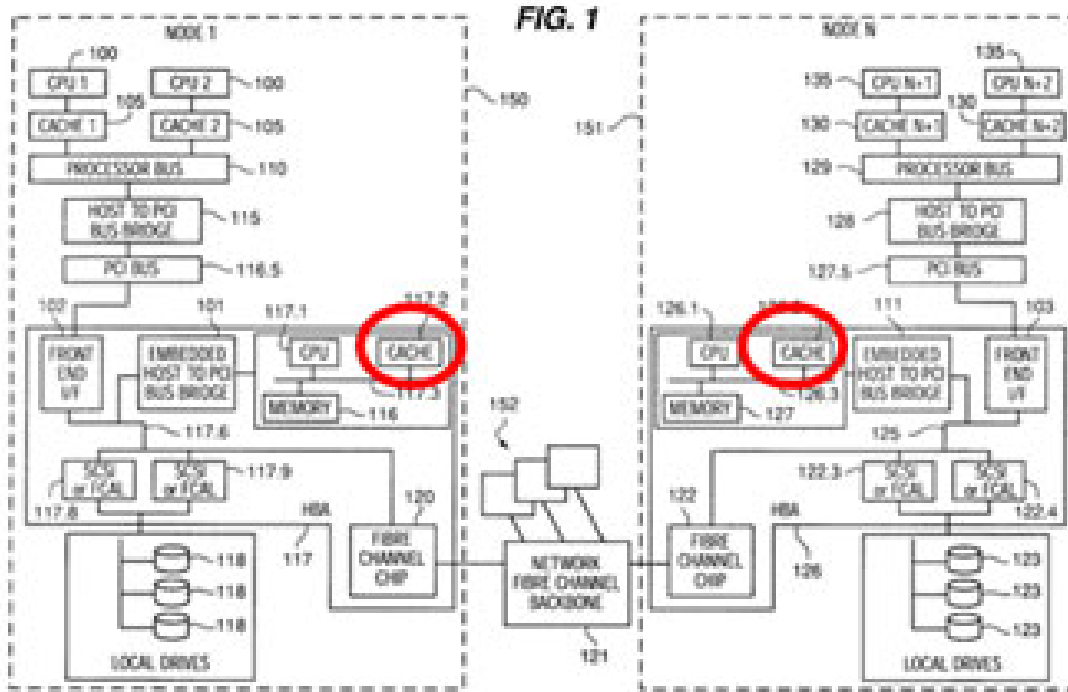
Heil discloses element 1.c. Heil discloses a Fibre channel chip 120, which is a network adapter for handling network control traffic as shown in Figure 1:



Ex.1006, Fig. 1, Ex.1004, ¶¶89-93. Fibre channel chip 120 connects the HBA to fibre channel network 121 comprising numerous other nodes 152 and enables HBA 117 to communicate with other HBAs via the fibre channel network. Ex.1004, ¶¶90-92; Ex.1006, 8:33-38 (“There are three classes of service available through the Fibre Channel backbone: dedicated service, switched service without a dedicated connection but providing guaranteed delivery with an acknowledgment of receipt, and connectionless service without acknowledgment of receipt.”). An individual HBA may transmit I/O requests as switched fibre channel network traffic from its local host to a remote HBA for that remote HBA to service. *Id.* Fibre channel chip 120 is therefore a network adapter of the HBA that handles network control traffic.

v. **Claim 1.d**

Heil discloses element 1.d. Ex.1004, ¶¶94-96. Each HBA (e.g., HBA 117) includes a cache memory (e.g., cache 117.2) as shown in Fig. 1:

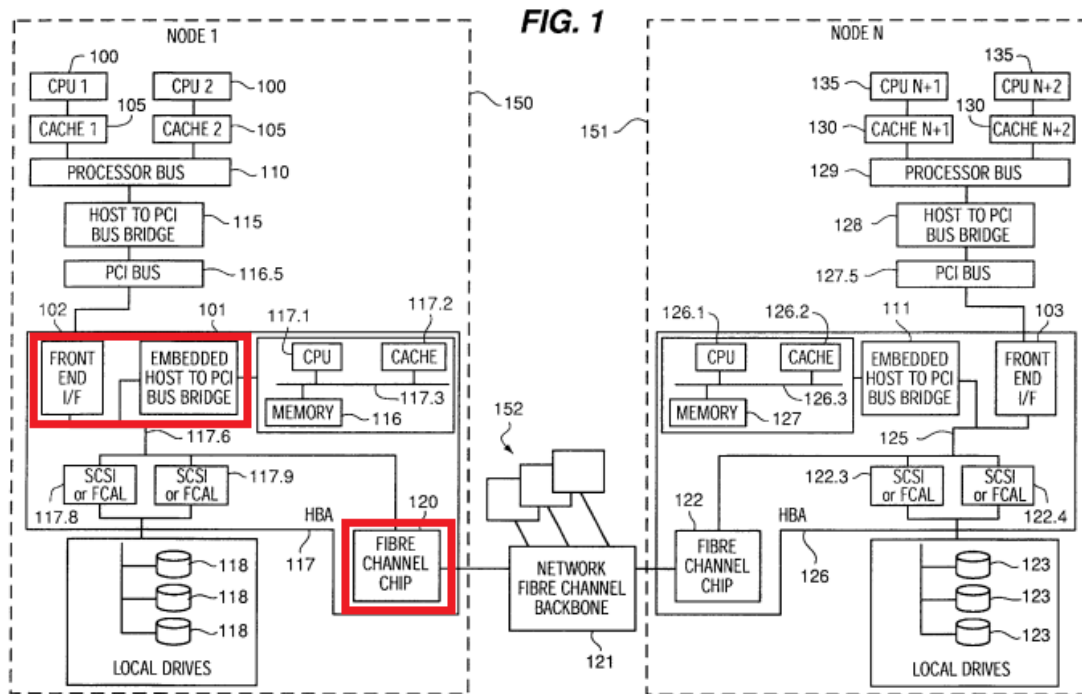


Ex.1006, Fig. 1, 7:9-15 (“HBA 117 also includes a CPU 117.1, local memory 116, and cache memory 117.2...”). Heil therefore discloses a cache memory.

vi. **Claim 1.e**

Heil discloses element 1.e. Ex.1004, ¶¶97-103. Heil’s host interface layer 230 and host interface layer 330 as shown in Fig. 2 are front-end software for handling I/O requests received by the HBA via the I/O channel. *Id.*; Ex.1006, 2:59-62 (“[S]oftware in the host intercepts file or block level I/O requests.”). Heil’s HBAs have both an I/O channel adapter for accepting I/O traffic from a host to the

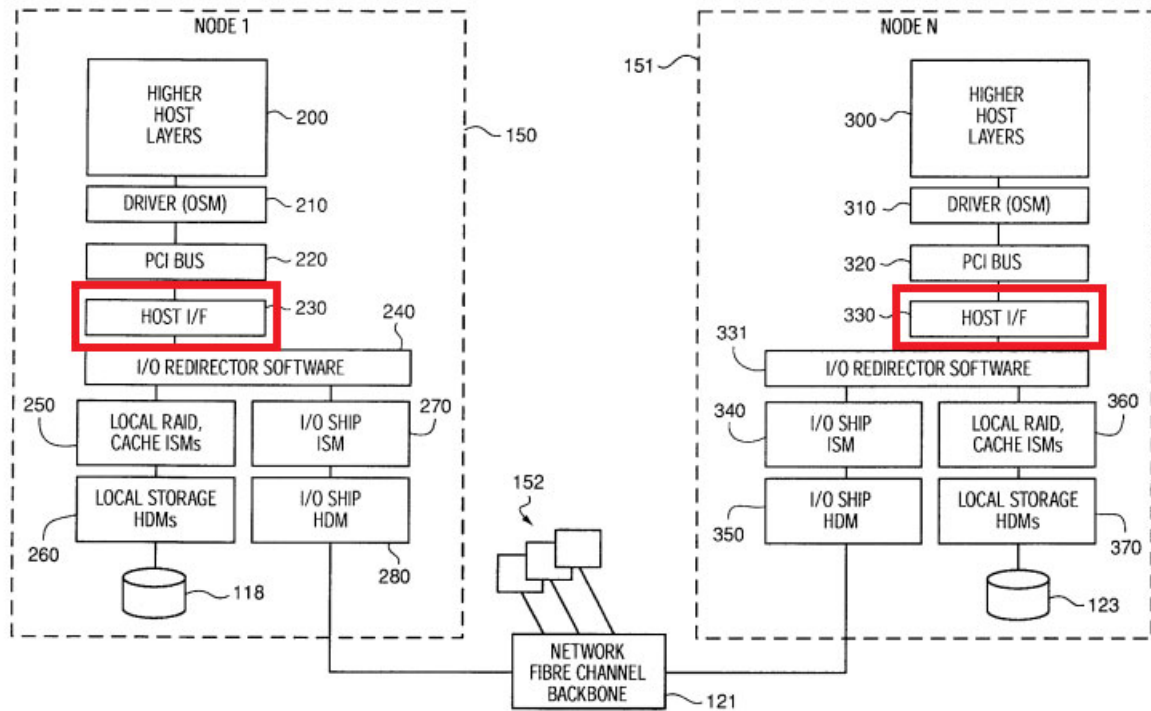
HBA as well as a network adapter for accepting network traffic (which may be I/O traffic) from other HBAs connected via a fibre channel network. Ex.1004, ¶99. As discussed above, the embedded Host to PCI bus bridge 101, PCI bus bridge 115, PCI bus 116.5, and front-end interface 102 are I/O channel adapters and the Fibre Channel Chip 120 is a network adapter, as shown in Figure 1 of Heil:



Ex.1004, ¶99, *see also* Ex.1006, Fig. 1, 6:65-7:7, 8:24-38, 11:54-12:7, 12:30-34.

Heil's Figure 2 shows that interface layers 230 and 330 are between the host and the HBA:

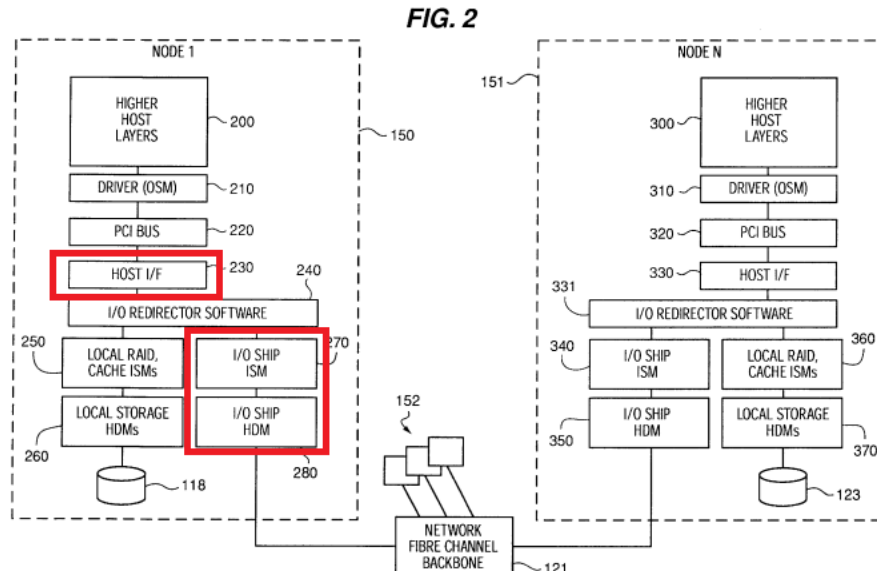
FIG. 2



Ex.1006, Fig. 2, 10:57-58. Heil discloses that these layers manage communications between the host and the HBA's embedded intelligence and would therefore handle requests arriving from the I/O adapter and the network adapter. Ex.1004, ¶¶100-102.

Similarly, Heil describes I/O shipping ISM 270 (ISM refers to intermediate service modules, which are software, *see* Ex.1006, 10:47-49) and I/O shipping HDM 280 (HDM refers to hardware specific code, which are also software, *see* Ex.1006, 10:45-47) as the driver for the fibre channel chip. Ex.1004, ¶101; Ex.1006, 10:43-49. Both the I/O shipping ISM 270 and I/O shipping HDM 280 are front-end software for handling I/O requests received by the HBA via the network

adapter. Ex.1004, ¶¶101-102. The host interface layer 230, host interface layer 330, I/O shipping ISM 270 and I/O shipping HDM 280 together comprise front-end software that handles I/O requests arriving at the I/O channel adapter or the network adapter. It would have been an obvious design choice whether to implement the front-end software as a single software module or two separate software modules. *Id.*, ¶102. Heil Fig. 2 shows that these elements are connected and form a software module:



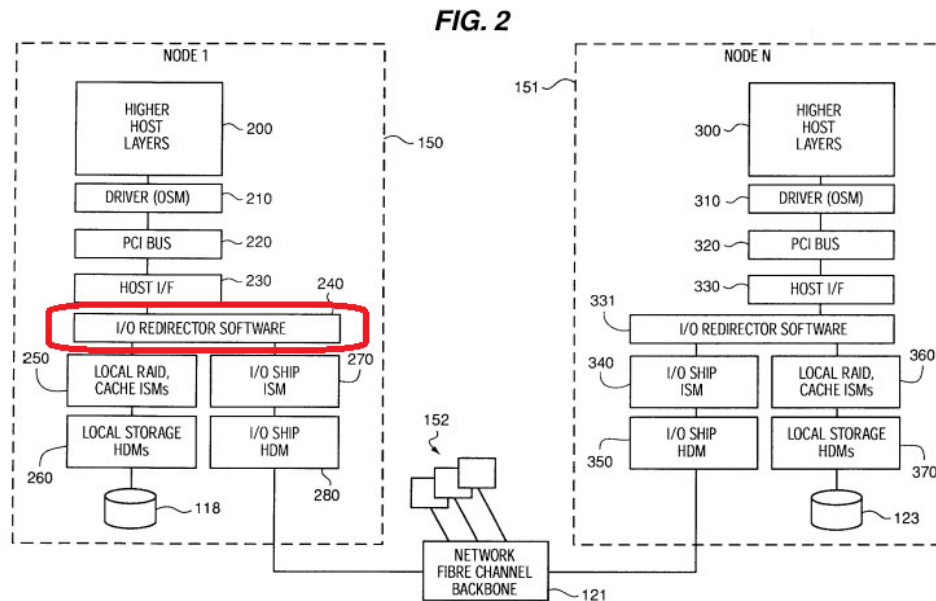
Id. Thus, Heil discloses front-end software for handling I/O requests arriving at the I/O channel adapter or the network adapter.

vii. Claim 1.f

Heil discloses element 1.f. Ex.1004, ¶¶104-108. Heil discloses cache intermediate service modules (“ISMs”), which are cache manager software for handling data stored in the cache memory. *Id.* For example, Heil states that the

HBAs provide caching in response to received I/O requests using “intermediate service modules” (ISMs), which is specific code for performing certain functionality. *Id.*, ¶106. The cache ISMs control I/O operations to cache. *Id.*, ¶¶105-106.

The cache ISMs 250 handle reads and writes to cache. *Id.*, ¶106. Heil’s cache manager is responsive to the Host I/F because the Host I/F handles the I/O request received by the HBA from the host node. *Id.*, ¶107. Figure 2 shows that the cache ISMs 250 are connected to the I/O redirector software 240 and operate by communication from the I/O redirector software:



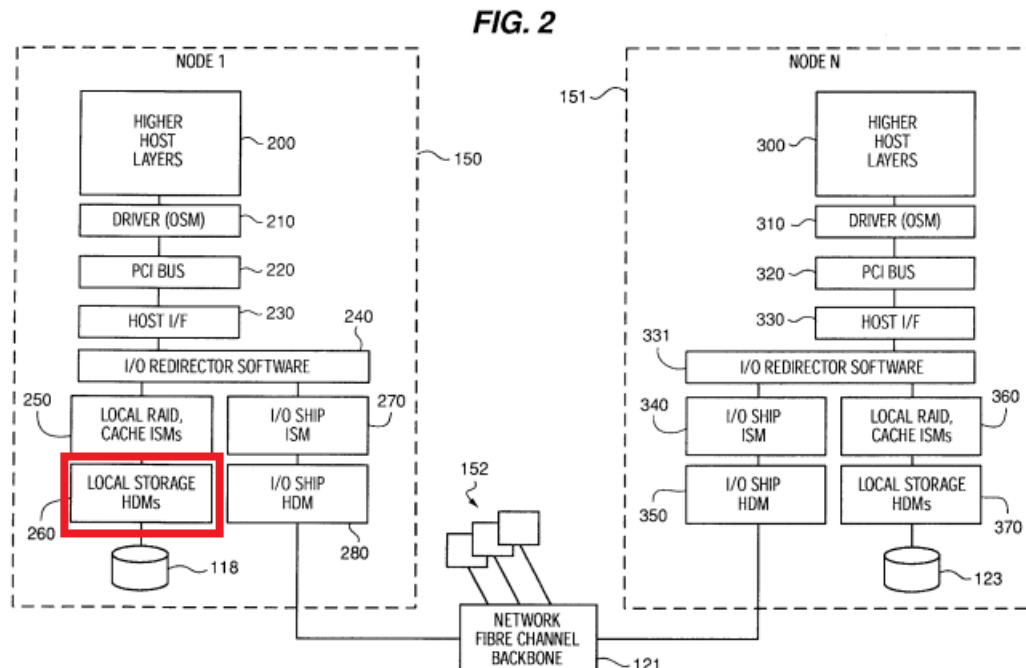
Ex.1006, Fig. 2, 12:19-34, 5:1-6. Figure 2 shows that I/O redirector software 240 may receive I/O requests from either the host via Host I/F 230 or the fibre channel network via I/O ship ISM 270 and I/O ship HDM 280. Ex.1006, 11:3-11 (“The I/O

redirector software 240, 331 provides the means to allow the HBA 117, 126 to make the decision whether to satisfy a block I/O request locally or remotely.”).

“The I/O redirector software 240, 331 provides the means to allow the HBA 117, 126 to make the decision whether to satisfy a block I/O request locally or remotely” and directs the cache ISMs to read and write to disk. Ex.1006, 10:66-11:11. Thus, these cache ISMs are cache manager software that performs I/O operations received by the HBAs to cache.

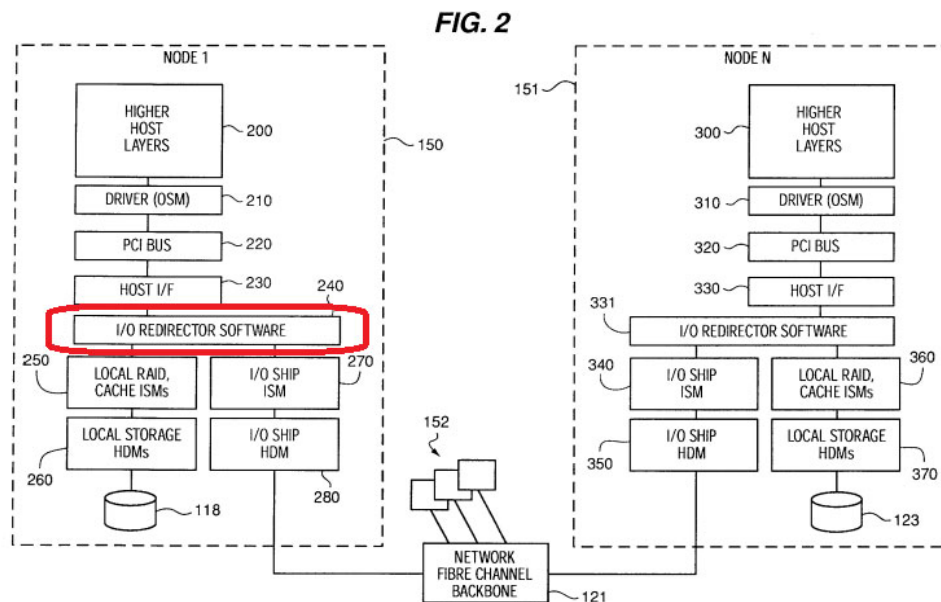
viii. Claim 1.g

Heil discloses element 1.g. Ex.1004, ¶¶109-114. Heil discloses local storage HDMs as shown in Fig. 2 which are back-end software for handling read and write to disk corresponding to the I/O requests:



Ex.1006, Fig. 2 (annotated). Heil describes local storage HDMs as hardware specific code for reading and writing to disk. Ex.1004, ¶110-111. The HBAs read and write to disk via HBA interfaces 117.8 and 117.9 (which may be small computer system interface (SCSI) or Fibre Channel Arbitrated Loop (FCAL) interfaces). *Id.* Drivers for the interfaces, such as the local storage HDMs, direct the local drives 118 to perform I/O operations at the storage device based on the I/O request. *Id.*

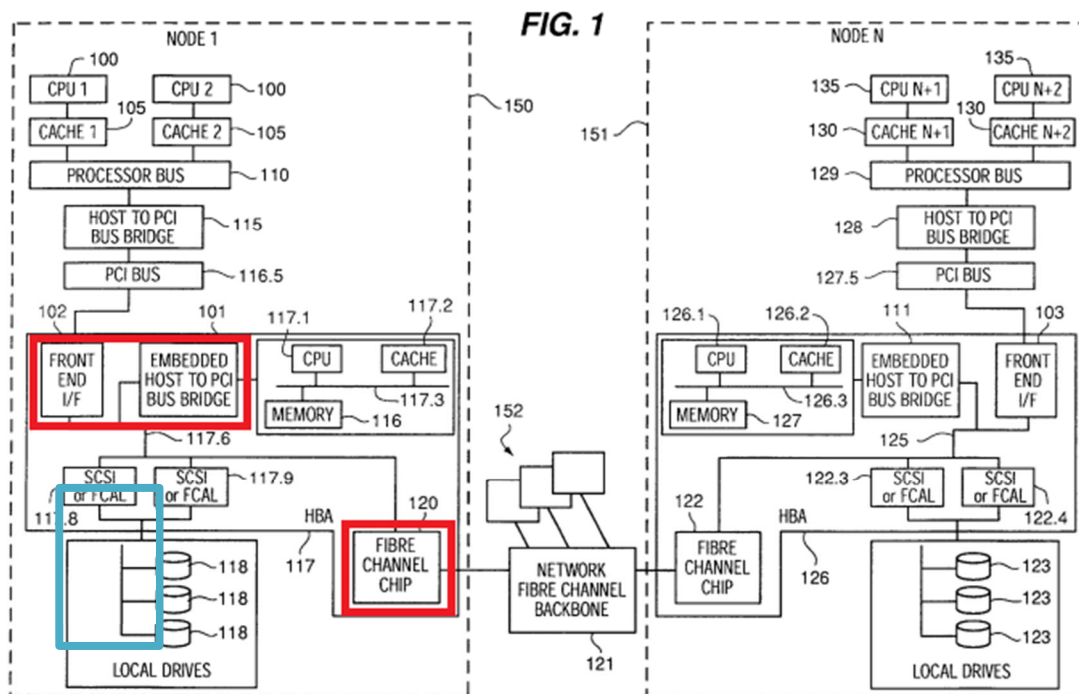
As discussed in Section VI.E.2.iii above, the local storage HDMs 260 handle reads and writes to disk responsive to the I/O redirector software 240, which provides instructions that routes the I/O request to the disk. Figure 2 shows that the local storage HDMs 260 are connected to the I/O redirector software 240 and operate by communication from the I/O redirector software:



Ex.1006, Fig. 2, 12:19-34, 5:1-6. “The I/O redirector software 240, 331 provides the means to allow the HBA 117, 126 to make the decision whether to satisfy a block I/O request locally or remotely” and directs the local storage HDMs to read and write to disk. Ex.1006, 10:66-11:11.

Furthermore, a POSITA would understand that the back-end software handles read and write to disk in Heil without communication over the I/O channel after an I/O request is received. Ex.1004, ¶113. Heil shows by Fig. 2 that the host sends I/O requests to the HBA. *Id.* Because HDM 260 is responsive to the I/O redirector software 240, which is part of the managing means, the “back-end software does not use the I/O channel adapter in the steps required to read and write the disks after responding to the I/O software director,” as shown in Fig. 2. *Id.*, ¶113. For example, “the host I/F layers communicates requests to the HBAs which then service the requests locally.” *Id.* This is also reflected in Figure 1, which shows that “the SCSI unit 117.8 has a direct connection to the drives 118 (outline in blue) that does not go over the PCI bus that carries the host to HBA I/O channel adapter identified in paragraph 90 (in red) [such that] the connection with

the drives are separated from the network traffic:



Ex.1004, ¶113 (citing Ex.1006, Fig. 1). Thus, the back-end software reads and writes to the disks “without communicating over the I/O channel adapter, thereby separating disk operations from network and I/O traffic”.

3. Claim 2

Heil discloses claim 2. Ex.1004, ¶¶115-127.

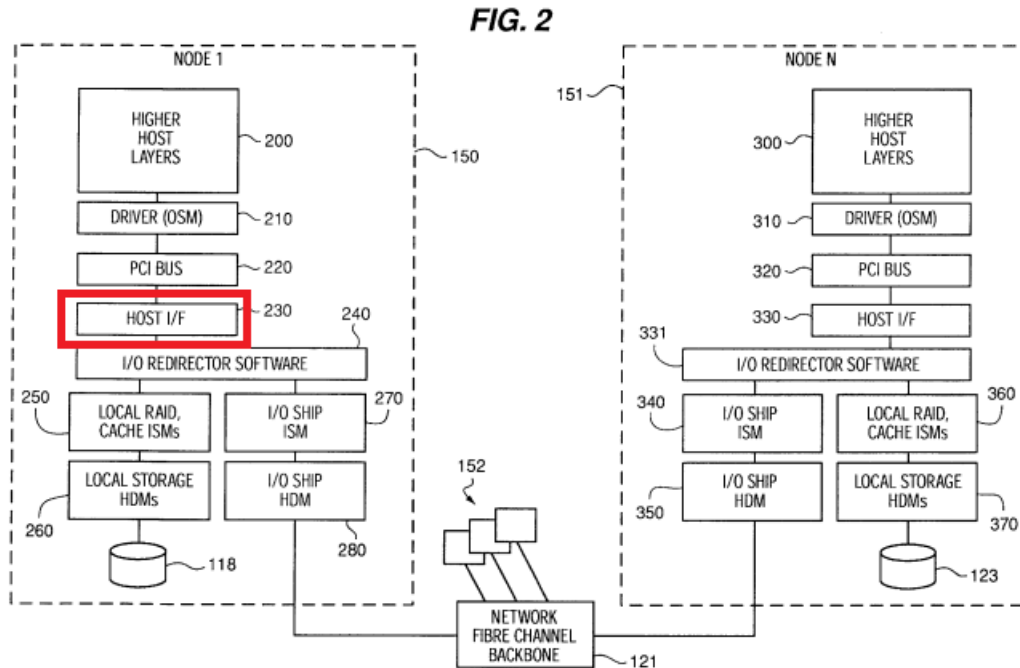
i. Claim 2.pre

If the Board determines that the preamble is limiting, Heil discloses the preamble. Ex.1004, ¶¶116. For example, Heil states that its HBAs comprise “available, commercially popular” components that operate using “presently available, open architecture” software. Ex.1006, 6:48-64 (“An exemplary Host-to-

PCI Bus bridge 115 is the “Saturn II” chip set manufactured as part number 82420 by Intel Corporation. Those skilled in the art will recognize other processor/PCI interface chips well known in the art. The PCI bus 116.5 is but one example of a presently available, commercially popular bus for peripheral device interconnection in host systems.”), 10:28-33 (“the host systems and associated HBA(s) communicate in accordance with the IO standard. The IO Standard is but one example of a presently available, open architecture for developing intelligent device drivers.”). A POSITA would have recognized that the device drivers described by Heil conforming to the I2O standard would operate on commercially available operating systems. Ex.1004, ¶116. Thus, Heil discloses computers comprising off the shelf hardware and operating systems.

ii. Claim 2.a

Heil discloses element 2.a. Ex.1004, ¶¶117-120. Heil discloses an I/O channel adapter for accepting incoming I/O requests from a host. *Id.* In addition, Heil discloses that a host interface layer 230 “manages communications between the host and the HBA's embedded intelligence” as shown in Figure 2:



Ex.1006, Figure 2 (annotated), 10:57-58. Host interface layer 230 is software (Ex.1006, 10:51-59) that accepts I/O requests from the host for redirection by I/O redirector software 240 to either the cache or disk storage of the HBA. Ex.1006, 10:66-11:10, 10:51-52 (“FIG. 2 is a block diagram of the software layers.”). Thus, Heil discloses element 2.a.

iii. Claim 2.b

Heil discloses element 2.b. Ex.1004, ¶¶121-127. Heil discloses that its I/O redirector software 240 (the configuration manager) uses a directory (a volume access table) to maintain consistency of the data stored on the network. *Id.* For example, Heil’s I/O redirector software 240 decides whether to route an I/O request received by the HBA from the host by accessing a directory “mapping the

location of blocks in the entire cluster”—including both local and/or remote locations of blocks—to determine the location of data on the network responsive to the I/O request. *Id.*, ¶122 (Ex.1006, 12:19-25, 12:34-42, 11:45-54). The directory stores “location information regarding blocks of data stored” to determine where to route the I/O request. *Id.* (citing Ex.1006, 4:8-12). The directory is updated as new blocks of data are stored to reduce latency on the network. Ex.1006, 12:15-19, 13:36-48. A POSITA would understand that consistent updates of the directory as blocks of data are created, moved, or deleted ensures the accuracy and consistency of the data stored on the network. *Id.*, ¶¶122-124. Additionally, Heil discloses that the consistency of the database may be maintained by a “Oracle Parallel Database Server” including a “distributed lock manager (DLM)” by regulating access to disk blocks. Ex.1006, 2:18-32. As Heil explicitly describes methods of using the directory to maintain the consistency of data in the storage system, a POSITA would have understood that using the directory with the I/O redirector would similarly maintain the consistency of data in the storage system. Ex.1004, ¶¶123-124. The POSITA would thus understand that Heil’s directory has the same functionality as and discloses the claimed volume access table of claim 2 of the ’988 Patent. *Id.*

Although the term “volume access table” should be construed broadly without reading any particular embodiment of the ’988 Patent into the

construction, to the extent a volume access table contains additional data specifying an access mode, this is rendered obvious by Nakayama. *Id.*, ¶¶125-126.

For example, Nakayama discloses storing data indicating whether a host has exclusive access to a drive (i.e., the host reserves the drive, thereby controlling access to it and excluding other host computers from accessing the drive): “The interhost exclusive information 403 is used when the host computer 300 or 303 exclusively controls the drive access from the other host computer and sets the information 403 ‘on’ when the processor reserves access of the specified drive number and sets it ‘off’ when the host computer releases the access right.”

Ex.1007, 5:43-47, 5:31-34. Nakayama further discloses that it is desirable to prevent any disk storage “from being predominantly used by” a host computer, which would reduce the availability of the disk storage for other hosts that may want to retrieve data from the disk storage or store data to the disk storage.

Ex.1006, 7:54-56. To prevent a particular host from predominantly using a specific disk storage, the POSITA would include this permission information taught by Nakayama in Heil’s directory to control access to each disk storage using the exclusivity indicator as taught by Nakayama. Ex.1007, 5:43-47, 5:31-34.

Moreover, Heil teaches controlling access to blocks of disk storage to maintain consistency (Ex.1006, 2:18-32), and a POSITA would understand that

Nakayama’s access modes would improve control of access to disk drives and

would help achieve Heil's objective of controlling access to disk to maintain consistency of the stored data. Ex.1004, ¶¶125-126.

Thus, Heil's I/O routing software employs a volume access table (i.e., Heil's directory) to route requests to ensure the consistency of the data stored on the network and discloses claim 2.

4. Claim 3

Heil discloses claim 3. Ex.1004, ¶¶128-130. Heil discloses the cache memory of each HBA comprises a portion of a distributed cache memory stored in the HBAs interconnected by the Fibre channel. *Id.*, ¶129. Heil discloses a distributed cache memory because Heil's "node-based cache system is a distributed storage system that stores files in a distributed manner [and f]iles are distributed in the disk storage of multiple HBAs on the same network." *Id.*, ¶129. Heil further discloses the cache memory of each HBA comprises a portion of the distributed cache memory. *Id.*, ¶129. ("[E]ach HBA may receive and store data from other HBAs, including in cache. Heil explicitly teaches saving blocks of data from disk storage in cache such that the blocks of data distributed to disk storage in the networked HBAs are also distributed to cache in the networked HBAs."); Ex.1006, 5:25-28 ("Performance is further enhanced in that each intelligent HBA can provide disk caching of the local disks under its control, and can service hits out of this cache regardless of whether the requesting host is local or remote.").

Because Heil distributes its blocks of data to cache in the networked HBAs, the cache memories of the HBAs form a distributed cache memory. *Id.*, ¶129. Thus, Heil discloses claim 3.

5. Claim 4

Heil discloses claim 4. Ex.1004, ¶¶131-134. Heil's I/O redirector software 240 employs a directory to map and manage data stored in Heil distributed cache. *Id.*, ¶¶132-133. Heil's directory renders the volume access table obvious. *Id.* Heil discloses that its I/O redirector software 240 uses "a directory within memory for storing location information regarding blocks of data stored" to determined where to route the I/O request. Ex.1004, ¶132. The I/O redirector software accesses a directory "mapping the location of blocks in the entire cluster"—including both local and/or remote locations of blocks—to determine the location of data on the network responsive to the I/O request. *Id.* (*citing* Ex.1006, 12:19-25, 12:34-42, 11:45-54). A POSITA would have understood that blocks and volumes are known methods of partitioning a disk or cache; therefore, it would have been an obvious design choice to organize a directory of a disk or cache by volume or by blocks. *Id.*, ¶133. Additionally, the "POSITA would understand that consistent updates of the directory as blocks of data are created, moved, or deleted ensures the accuracy and consistency of the data stored on the network." *Id.*, ¶133.

Furthermore, a POSITA would understand that Heil's directory manages and maps data stored in both disk storage (e.g., local disk storage and remote disk storage) and cache. *Id.* Heil states that the directory is a mapping of all storage locations within the local storage system: the "HBA maps the location of data within its local storage system thereby building a directory containing the location of all locally stored data blocks," Ex.1006, 5:1-5. Heil further states that the same storage system may comprise both local memory and cache memory: "[o]ne skilled in the art will recognize local memory 116 and cache memory 117.2 can exist within the same physical memory." Ex.1006, 7:28-31. Because Heil's storage system is disclosed as including both disk memory and cache memory, the POSITA would understand that Heil's directory would include a mapping of both disk and cache memory. Ex.1004, ¶¶132-133.

Therefore, a "POSITA would understand that Heil's directory has the same functionality as and discloses the volume access table of the '988 Patent." *Id.*

6. Claim 6

The method of accessing a remote disk is obvious in view of Heil and Nakayama. Ex.1004, ¶¶135-157.

i. Claim 6,pre

If the Board determines that the preamble is limiting, Heil discloses the preamble. Ex.1004, ¶¶136-138. Heil discloses a method of accessing a remote disk

of a remote HBA over a computer network. *Id.* Indeed, Heil characterizes its invention as a method of retrieving “data across independent computer nodes of a server cluster” by shipping I/O requests between nodes of the network. Ex.1006, abstract. Each node of the server cluster comprises a host computer and a host bus adapter (HBA). Ex.1006, 6:34-46. A local host may transmit an I/O request to its local HBA via the I/O channel between the local host and local HBA and, if the requested data is not within the cache or disk storage of the HBA, the I/O request is transmitted to other, remote HBAs on the network. Ex.1006, 7:36-50, 11:54-64. The remote HBA then accesses its remote cache or disk storage to retrieve data responsive to the I/O request, and the responsive data is retrieved and transmitted back to the local HBA. Ex.1006, 11:54-12:7. Heil further discloses that its system is structured, like the ’988 Patent, with hosts transmitting I/O requests to local HBAs for service and/or forwarding to remote HBAs to reduce network overhead by streamlining host I/O requests and data retrieval. Ex, 1006, 11:61-65 (“HBA to HBA communication channels are established during initial system configurations so overhead associated with establishing communication is not incurred on every block request over the Fibre Channel backbone.”); *see also id.*, 3:50-63, 5:10-24; Ex.1004, ¶¶136-138. Thus, Heil discloses the method of the preamble of claim 6.

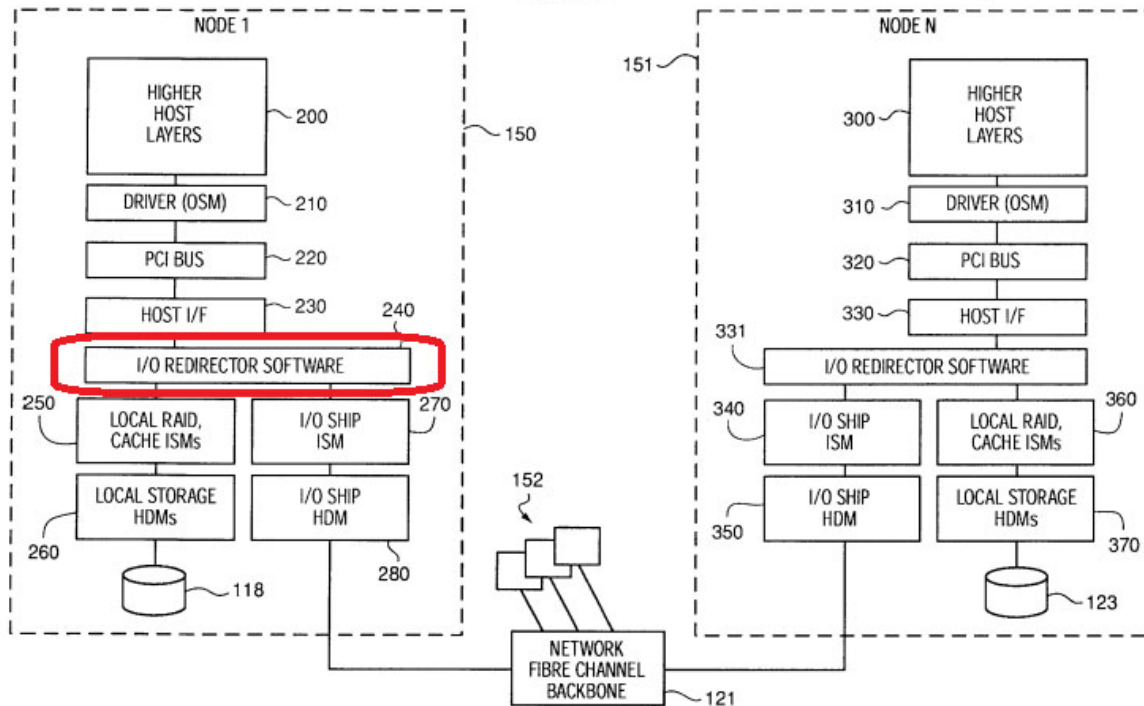
ii. Claim 6.a

Heil discloses element 6.a. Ex.1004, ¶140. Heil discloses a networked storage system of nodes each comprising a host and an associated HBA. Ex.1006, 6:34-46. A particular local host may issue I/O requests over an I/O channel to its local HBA. Ex.1006, 7:36-50. Heil describes the HBA (i.e., HBA 117) as including “a CPU 117.1, local memory 116, and cache memory 117.2, which are connected by internal processor bus 117.3,” which a POSITA would understand to be a computer. Ex.1006, 7:9-15; Ex.1004, ¶140. Heil thus discloses a local host issuing a request over an I/O channel to a local computer.

iii. Claim 6.b

Heil discloses element 6.b. Ex.1004, ¶¶141-144. Heil discloses providing local HBA 117 with I/O redirector software 240 (a configuration manager) to route I/O requests to remote HBAs via the Fibre network. *Id.*; Ex.1006, (“Each [local] HBA 117...contains managing means which includes the “Distributed block I/O redirector driver software” 240...(hereinafter referred to as I/O redirector software)...”). The I/O redirector software processes I/O traffic received by the HBA from the host as shown in Figure 2:

FIG. 2



Heil teaches that, “[w]hen servicing an I/O request, the HBA looks to its cache and disk storage to determine if the requested data may be serviced locally and, if not, the HBA ships the I/O request to a remote HBA having the requested data for the remote HBA to service.” Ex.1004, ¶143. In other words, I/O redirector software 240 may route an I/O request to a remote HBA for servicing if the requested data is not found on the local HBA. *Id.* Thus, Heil discloses element 6.b.

iv. Claim 6.c

Heil discloses element 6.c. Ex.1004, ¶¶145-148. When responding to an I/O request, Heil states that the I/O redirector software of each HBA checks a directory mapping the data storage locations to determine the location of the data responsive

to the I/O request. Ex.1004, ¶146 (*citing* Ex.1006, 4:8-12). For example, Heil states that “a directory within memory for storing location information regarding blocks of data stored” to determine where to route the I/O request and the I/O redirector software accesses a directory “mapping the location of blocks in the entire cluster”—including both local and/or remote locations of blocks—to determine the location of data on the network responsive to the I/O request. Ex.1006, 12:19-25, 12:34-42, 11:45-54. A POSITA would understand that Heil’s directory has the same functionality as and discloses to volume access table of the ’988 Patent. Ex.1004, ¶¶146-148. The POSITA would further understand that, because Fig. 2 shows I/O Ship HDM 280 is connected to the I/O Redirector Software 240, the remote HBA would also check an I/O request received via the I/O Ship HDM 280 from a local HBA against the directory. *Id.* Thus, the remote HBA checks the I/O request against the directory to determine the storage location of data responsive to the request. Ex.1004, ¶¶146-148. Furthermore, a “POSITA would understand that consistent updates of the directory as blocks of data are created, moved, or deleted ensures the accuracy and consistency of the data stored on the network.” Ex.1004, ¶147. Thus, Heil discloses element 6.c.

v. Claim 6.d

Heil discloses element 6.d. Ex.1004, ¶¶149-151. Heil discloses causing the remote computer to perform an I/O operation on a disk located on the remote

computer and to return data to the local computer because, if the data requested by the local host is not within the cache or disk storage of the local HBA, the I/O request is transmitted by the local HBA to other, remote HBAs on the network. Ex.1004, ¶150 (*citing* Ex.1006, 7:36-50, 11:54-64). The remote HBA accesses its remote cache or disk storage to retrieve data responsive to the I/O request, and the responsive data is retrieved and transmitted back to the local HBA. *Id.* Heil therefore discloses element 6.d.

vi. Claim 6.e

Heil discloses element 6.e. Ex.1004, ¶¶152-154. After the local host transmits an I/O request to a local HBA, the local HBA may then transmit the I/O request to a remote HBA for retrieval from the remote disk of the remote HBA. Ex.1004, ¶153 (*citing* Ex.1006, 7:36-50, 11:54-64). The remote HBA retrieves the requested data from its disk and transmits the requested data back to the local HBA, which returns the requested data to the local host via the I/O channel. *Id.* (*citing* Ex.1006, 11:54-12:7, 12:6-7, 6:34-51). Because Heil discloses a single communication path between the requesting host and the HBA, both the I/O request and the data responsive to the I/O request are communicated between the host and the HBA through that communication path. Ex.1006, 7:36-37 (“[O]ne HBA services the I/O requests from CPU 1 and CPU2.”). Thus, this causes the local HBA to provide the returned data to the local host via the I/O channel.

vii. Claim 6.f

Heil discloses element 6.f. Ex.1004, ¶¶155-157. Heil uses a directory mapping data stored within the network to determine where to route the I/O request. *Id.* A local HBA selects a peer HBA having the requested data by searching the directory using the request to determine the location where it may be found, after which the peer HBA retrieves the data responsive to the I/O request and transmits it to the local HBA. Ex.1004, ¶156 (*citing* Ex.1006, 10:66-11:10, 12:11-17). The peer HBA, after retrieving the request data, can store the requested data in cache or disk storage to improve the requested data's accessibility. Ex.1006, 12:1-4. Heil discloses that, when the local HBA receives the requested data from the peer HBA, the local HBA may use that data to update the directory with the information obtained from the peer HBA to reduce network latency: "The HBA demands directory information from peer HBAs using peer-to-peer communications when an I/O requested data block is not found within the local directory 534. In this embodiment, the HBA updates its local directory with the directory information obtained from peer HBAs 536, so subsequent requests for that particular block of data do not have to incur the latency of this directory update." Ex.1006, 12:67-13:3, 12:16-19 ("The initiating HBA will update its directory, via its I/O shipping ISM 270, 340, so subsequent requests for that block of data do not incur the latency of the directory query."). Because Heil discloses

both that the local HBA searches the directory using the request to determine the location of the data and that the local HBA checks the data received from the peer HBA against the directory and updates the directory based on the received data, Heil discloses element 6.f. Ex.1004, ¶156.

F. Ground 2: Claim 5 is obvious over Heil, Nakayama, Srinivasan and Macklem under 35 U.S.C. § 103(a)

1. A POSITA Would Have Combined Heil, Nakayama, Srinivasan, and Macklem

A POSITA would have combined Heil and Nakayama for the reasons discussed in Section VI.E.1. Since Heil already describes a need for and infrastructure that regulates access to stored data, the POSITA would therefore have been motivated to seek out descriptions of specific methodologies for likewise coordinating access control of cache, such as those found in Srinivasan and Macklem, and combine them with the teachings of Heil and Nakayama, and would have had the skill and understanding to combine these known techniques with a reasonable expectation of success. Ex.1004, ¶¶158-168, 167 (“Heil leaves the specific mechanisms to ensure cache consistency up to the POSITA who chooses to practice Heil. The POSITA would look to Srinivasan and Macklem which both describe alternative cache consistency protocols of maintaining cache consistency by managing disk volume access in the context of a distributed file storage system.”).

Specifically, Heil recognizes that problems arise when multiple hosts are permitted uncontrolled access to the same storage: “Problems arise where multiple computers, operating in parallel, share data and storage devices. Clearly, a level of coordination is required to assure that each of the computers is aware of updates in the storage devices made by others of the computers in the cluster.” Ex.1006, 1:44-52. To resolve issues arising from multiple hosts accessing a shared storage, Heil discloses that the consistency of the database may be maintained by a “Oracle Parallel Database Server” including a “distributed lock manager (DLM)” by regulating access to disk blocks: “The Oracle Parallel Database Server maintains the consistency of the database by utilizing a proprietary protocol, the distributed lock manager (DLM), to allow nodes to access the shared storage concurrently. Utilizing DLM software in a cluster of physically shared disks allows all computer nodes to have access to all disks directly through their own I/O subsystem so that each disk appears to be physically local. Each computer node can cache and/or lock shared disk-based structures utilizing the DLM software. For example, if one node wants blocks X, Y, and Z within disks A and B, it must first ask the DLM software for permission. The DLM will grant permission only after it has endured that blocks X, Y, and Z are current.” Ex.1006, 2:18-32. As Heil recognizes that unrestricted access to disk drives degrades the consistency of disk storage and explicitly proposes locking drives against access by multiple hosts, a POSITA

would recognize that Srinivasan's and Macklem's cache management systems that limit cache access would be applicable to the Heil and Nakayama combination and may similarly be used to control host access to improve consistency. Ex.1004, ¶167 ("Heil leaves the specific mechanisms to ensure cache consistency up to the POSITA who chooses to practice Heil. The POSITA would look to Srinivasan and Macklem which both describe alternative cache consistency protocols of maintaining cache consistency by managing disk volume access in the context of a distributed file storage system. Though Srinivasan and Macklem both modify NFS to implement their cache consistency protocols, the techniques could be regularly extended to other distributed file system software such as Heil's."). Srinivasan and Macklem both describe methods of improving cache consistency and performance that a POSITA would apply to Heil's and Nakayama's cache storage systems. *Id.* A POSITA seeking to improve storage access for a host computer in a storage system having both cache and disk storage would be motivated to incorporate methods of improving storage efficiency by limiting access, such as Srinivasan's and Macklem's, to improve the storage functionality of the Heil and Nakayama combination. *Id.*, ¶166-168.

Thus, a POSITA would combine Heil, Nakayama, Srinivasan, and Macklem by implementing the specific protocols for managing storage access disclosed by Srinivasan and Macklem in the combined system of Heil and Nakayama. As Heil

already discloses that a distributed lock manager should be used to limit access to shared storage (Ex.1006, 2:18-32), a POSITA would merely look to Srinivasan and Macklem for specific implementations of how to limit access. Ex.1004, ¶¶166-168.

2. Claim 5

Heil in view of Nakayama, Srinivasan and Macklem renders claim 5 obvious. Ex.1004, ¶¶169-198.

i. Claim 5.a

Heil in view of Nakayama, Srinivasan and Macklem discloses element 5.a. Ex.1004, ¶¶170-175. As discussed in Section VI.E.2.iii, Heil discloses configuration manager software including I/O redirector software 240, and as discussed in Section VI.E.5, further discloses a volume access table in the form of a directory that maps and manages data stored in the distributed storage system. Heil further describes that techniques for maintaining data consistency in a distributed storage architecture were already well known, providing as an example the “distributed lock manager (DLM)” of Oracle Parallel Database which regulates access to disk blocks. Ex.1006, 2:18-32. Heil expressly describes the importance of data consistency in distributed data storage systems, and a POSITA would have understood that the I/O redirector and cache management functionalities described in Heil would provide management of data consistency in a manner similar to known techniques like those implemented in the Oracle Parallel Database Server.

Ex.1004, ¶171. Based on Heil’s acknowledgement of the importance of data consistency and the extended description of access modes in the Oracle Parallel Database Server, a POSITA would have found it obvious that the directory in Heil’s system would likewise include access modes and would thus have been motivated to seek examples of specific implementations of access modes such as those found in Nakayama. Ex.1004, ¶¶171-175. Combining the teachings of Nakayama with Heil would have been straight forward to a POSITA, and would have resulted in the incorporation of access modes in the directory of Heil, which would be utilized by the I/O redirector to facilitate data consistency. Ex.1004, ¶¶171-175.

Nakayama discloses storing data indicating whether a host has exclusive access to a drive (i.e., the host reserves the drive, thereby controlling access to it and excluding other host computers from accessing the drive): “The interhost exclusive information 403 is used when the host computer 300 or 303 exclusively controls the drive access from the other host computer and sets the information 403 ‘on’ when the processor reserves access of the specified drive number and sets it ‘off’ when the host computer releases the access right.” *Id.* (citing Ex.1007, 5:43-47, 5:31-34.). A POSITA would have understood the inclusion of Nakayama’s exclusivity indicator in Heil’s directory to indicate an access mode. Ex.1004, ¶173. As a POSITA at the time of invention would recognize volumes and blocks are

alternative manners of partitioning a disk or cache, the POSITA would likewise have found it obvious for the I/O redirector to utilize access mode information to ensure data consistency by checking the exclusivity indicator to determine taught by Nakayama. Ex.1004, ¶173. For example, the POSITA would have understood that Heil's disks or caches could be partitioned in volumes and a directory controlling access to volumes is a volume access table. *Id.* The POSITA would have understood that the exclusivity indicator for a block or volume must be checked prior to accessing disk storage to maintain data consistency. *Id.*

A POSITA would have found it obvious to combine Heil and Nakayama for the reasons stated in Section VI.E.1. The combination of Heil and Nakayama is consistent with Heil's teaching that access to disk blocks must be regulated by a distributed lock manager (DLM). Ex.1004, ¶¶171-175; Ex.1006, 2:18-32. Thus, Heil in view of Nakayama, Srinivasan and Macklem renders obvious element 5.a.

ii. Claim 5.b

Heil in view of Nakayama, Srinivasan and Macklem discloses element 5.b. Ex.1004, ¶¶176-186. As discussed in Section VI.F.2.i above, Heil in view of Nakayama discloses checking an access mode in a volume access table. Both Srinivasan and Macklem further disclose an exclusive mode that causes reads and writes to be stored in the cache memory. Ex.1004, ¶¶176-186.

Srinivasan describes methods of cache management where several clients may access the same data in the same cache. Ex.1018, 45. In such situations, Srinivasan teaches that cache consistency and performance may be improved by implementing protocols that limit concurrent client access to the same cached file. *Id.*, 45-46. Thus, Srinivasan describes at least three protocols that limit read and write operations on a cache : a “single-writer” “non-write-shared” mode, a “read-only” “non-write-shared” mode, and a “write-shared” mode. Ex.1018, 46. A POSITA would understand that a cache or disk storage may operate under each of these modes. Ex.1004, ¶179 (“While Srinivasan discloses this mode in connection with exclusive access to files, it would have been a simple design choice to exclude access to an entire volume, cache, or drive, to improve consistency as taught by Heil and Srinivasan.”). An access mode in a volume access table denotes a mode of access of volumes, caches, or storage identified by the volume access table: for example, the volume access table may identify a particular volume is in exclusive mode, then that volume would be operating in exclusive mode. Ex.1004, ¶179. Thus, the volume access table combining Srinivasan with Heil and Nakayama may also denote the access modes of Srinivasan (such as “single-writer” “non-write-shared” mode) indicating that the identified volume, cache, or storage is operating in that mode. *Id.* In addition, Srinivasan describes a flag or value indicating which of the three modes the cache is set to. *Id.*, ¶180. Because each of the values of the

flag indicates a different access mode of the cache, a POSITA would have understood that indicating a particular access mode of the cache to the system may be achieved by setting the value of an indicator in the volume access table to indicate a mode of Srinivasan (i.e., the “single-writer” “non-write-shared” mode). Ex.1004, ¶¶178-180.

In the “single-writer” “non-write-shared” mode, a client has both read and write access to a file stored in cache. Ex.1018, 48. In this mode, Srinivasan utilizes a cacheEnabled flag “to tell the client whether it is allowed to cache data for this file.” *Id.*, 48. Because the cacheEnabled flag indicates a “single-writer” “non-write-shared” mode in which the client has both read and write access to a file stored in cache, it would have been obvious to a POSITA to modify an indicator in the volume access table of Heil and Nakayama to indicate the “single-writer” “non-write-shared” mode and cacheEnabled flag which corresponds to an access mode that causes both reads and writes to be stored in the cache memory. Ex.1004, ¶¶178-180. Further, because “single-writer” “non-write-shared” mode indicates that access to the file is limited to the client that currently has access (i.e., a “single-writer”) and writes is not shared with other clients (“non-write-shared”), this mode grants the client exclusive access to the cached file. *Id.* Srinivasan discloses that these modes are advantageous to maintain data consistency. Ex.1018, 45 (“[C]aching implies the potential existence of multiple copies of the same data, and

keeping these multiple copies consistent is a challenge”). It would thus have been obvious to a POSITA when combining Srinivasan with Heil and Nakayama to modify an indicator in the volume access table of the combination of Heil and Nakayama to assign a value for the indicator to indicate Srinivasan’s non-write-shared access mode. In doing so, the access mode indicator would then indicate the cache accessibility corresponding to Srinivasan’s non-write-shared mode. *Id.* As discussed above, the POSTIA would have modified the indicator in the volume access table to indicate the access modes of the cache to the system taught by Srinivasan. *Id.* This would be a logical modification to enable the access mode indicator of the volume access table to be able to indicate access modes such as those described by Srinivasan to improve access control to cached data. *Id.*

Additionally, a POSITA would understand that controlling access to storage and granting exclusive storage access to a specific client achieves Srinivasan’s described procedures for limiting access to files. For example, Srinivasan states that “by tracking open and close operations the Sprite file server knows not only which clients are currently using a file, but whether any of them are potentially writers,” indicating that its caches are read and write controlled. Ex.1018, 46-47; Ex.1004, ¶¶178-180. Similarly, the “cacheEnabled” flag indicates that the client has access to the cache. *Id.*

Additionally, Srinivasan discloses that in a “single-writer” “non-write-shared” mode, the system provides the server having stored the noncached version of the file a writeMode flag indicating that the file is in use and written to cache. Ex.1018, 48 (“The open RPC call takes a "file handle" (an identifier returned by the lookup operation) and a flag indicating if the client intends to write the file... The close procedure tells the server that the client is no longer using the specified file handle. The client passes the same writeMode flag that it provided for the corresponding open operation.”). In “single-writer” “non-write-shared” mode, because only a single client has access to a cached file, no other clients would be permitted access to that cached file. Srinivasan may return the cacheEnabled flag indicating the file is locked to other clients, which is an invalidate message sent indicating that reads and writes are to be stored in cache rather than in server memory. Ex.1004, ¶¶178-180. Furthermore, because Heil teaches a distributed storage system where data responsive to an I/O request may be found on a remote HBA rather than the local HBA, a local host attempting to access data stored on a remote HBA would cause the invalidate message to be sent to the remote storage system of the remote HBA. *Id.*

Furthermore, Macklem also describes protocols that limit read and write operations on a cache. Ex.1009, 264. Macklem similarly describes a distributed file system in which the system must manage accesses by multiple clients to the same

file. Ex.1009, 261 (“[M]aintaining consistency between these caches is a problem whenever write sharing occurs; that is, when a process on a client writes to a file and one or more processes on other client(s) read the file.”). Macklem likewise teaches that cache consistency and performance may be improved by implementing protocols that limit concurrent client access to the same cached file. *Id.*, 261 (“This protocol provides full cache consistency during normal operation, while permitting more effective client-side caching in an effort to improve performance”). Macklem describes multiple modes of operation that restrict client access to cache, including “write-caching lease,” “read-caching lease,” and a “non-caching” mode. *Id.*, 264. In the “write-caching lease” mode, a client is permitted to both read and write to cache. Ex.1009, 264. Thus, Macklem’s “write-caching lease” corresponds to an access mode that causes both reads and writes to be stored in the cache memory. Ex.1004, ¶¶181-183. Additionally, as described above for Srinivasan, it would have been advantageous for an access mode indicator to indicate Macklem’s “write-caching lease” access mode to inform the system of the access mode limitations on the volume, cache, or drive. *Id.* Like for Srinivasan, it would similarly be obvious for a POSITA to assign an access mode indicator value in an access mode table to indicate Macklem’s write-caching leasing mode among other access modes. *Id.* Furthermore, because each indicator value indicates a different access mode of the cache, a POSITA would have understood that

indicating a particular access mode of the cache to the system may be achieved by modifying the indicator in the volume access table such that it indicates Macklem's write-caching lease mode. *Id.*

Furthermore, Macklem discloses that in "write-caching lease," the system periodically renews the lease by transmitting a "Get write lease" from the client to the storage system having stored the non-cached version of the file indicating that the file is in use and written to cache, as shown in Diagram 2. Ex.1004, ¶181-183. The system replies with a lease grant that indicates the client has access to the cache (i.e., read and write are stored in the cache memory). Macklem's lease renewal process therefore involves an invalidate message sent to the storage system indicating that reads and writes are to be stored in cache rather than in server memory. Ex.1004, ¶¶181-183. Furthermore, because Heil teaches a distributed storage system where data responsive to an I/O request may be found on a remote HBA rather than the local HBA, a local host attempting to access data stored on a remote HBA would cause the invalidate message to be sent to the remote storage system of the remote HBA. *Id.*

A POSITA would have found it obvious to combine Heil, Nakayama, Srinivasan, and Macklem for the reasons stated in Section VI.F.1. Both Srinivasan and Macklem describe aspects of caching in a known network file system (Sprite), and a POSITA would implement Srinivasan's cache management system that

permits read and write to cache as well as Macklem's cache management system that permits read and write to cache in order to improve cache consistency and performance as disclosed by Srinivasan and Macklem. Ex.1018, 45-46; Ex.1009, 261; Ex.1004, ¶¶184-185. As discussed above, an indicator in the volume access table may be modified to indicate Srinivasan's or Macklem's access modes. Heil describes benefits to maintaining data consistency (Ex.1006, 2:19-35), whereas Srinivasan and Macklem describe specific methods of maintaining cache consistency by using access modes to control access to disks or volumes. Ex.1004, ¶¶184-185. Thus, the POSITA would be motivated to use cache consistency protocols as described by Srinivasan and Macklem in the context of a distributed file storage system. Ex.1004, ¶¶184-185. Though Srinivasan and Macklem modified NFS to implement their cache consistency protocols, the techniques could be regularly extended to other distributed file system software. *Id.*, ¶185. A POSITA would expect success as Heil describes a distributed file storage system and Srinivasan and Macklem both describe alternative solutions to the cache consistency problem that was a well-known problem and had a number of non-available solutions, including these. Ex.1004, ¶¶184-185. Indeed, Srinivasan describes well-known protocols (Sprite protocols) that a POSITA would understand are applicable to cache storage systems. *Id.* Thus, Heil in view of Nakayama, Srinivasan, and Macklem suggests element 5.b.

iii. Claim 5.c

Heil in view of Nakayama, Srinivasan, and Macklem suggests element 5.c. Ex.1004, ¶¶187-192. As discussed in Section VI.F.2.i above, Heil in view of Nakayama suggests checking an access mode in a volume access table. To the extent neither Heil nor Nakayama states that the access mode is a shared access mode that causes only reads to be stored in the cache memory, both Srinivasan and Macklem suggest this modification. *Id.*

As discussed in the preceding section, Srinivasan discloses indicators for multiple modes of accessing a cache, including a “non-write shared” “read-only” mode. Ex.1018, 46. Because each indicator value indicates a different access mode of the cache, a POSITA would have understood that indicating a particular access mode of the cache to the system may be achieved by storing such an indicator in the volume access table. Ex.1004, ¶189. In “read-only” “non-write-shared” mode, a host only has read access to file stored in cache. Ex.1018, 48. In this mode, the cacheEnabled flag indicates that the client is not permitted to write to cache, i.e., read only. *Id.*, 48. As explained with respect to element 5.b, a POSITA would understand that controlling access to storage and granting exclusive storage access to a specific client achieves Srinivasan’s described procedures for limiting access to files. Ex.1004, ¶189. For example, Srinivasan states that “by tracking open and close operations the Sprite file server knows not only which clients are currently

using a file, but whether any of them are potentially writers,” indicating that its caches are read and write controlled. Ex.1018, 46-47; Ex.1004, ¶189. Further, because the client only has read access to a file stored in cache in “non-write shared” “read-only” mode, the “single-writer” “non-write-shared” mode and cacheEnabled flag corresponds to an access mode that causes only reads to be stored in the cache memory. Ex.1004, ¶189. A POSITA would thus, when combining Srinivasan with Heil and Nakayama, modify an indicator in a volume access table as taught by Heil and Nakayama to further indicate Srinivasan’s non-write-shared access mode such that the access mode indicator can indicate the cache accessibility corresponding to Srinivasan’s non-write-shared mode. *Id.* This would be a logical modification to enable the access mode indicator of the volume access table to be able to indicate access modes such as those described by Srinivasan to improve access control to cached data. *Id.*

In addition to the write-caching lease discussed in the previous section, Macklem further discloses a “read-caching lease” mode in which the client is permitted only to cache for read purposes. Ex.1009, 264 (“A read-caching lease allows for client data caching but no modifications may be done.”). Because “read-caching lease” mode limits the client to only reading the cache and does not permit writes to cache, Macklem’s “read-caching lease” corresponds to an access mode that only reads to cache memory. Ex.1004, ¶190. Similar to above, the POSITA

would further modify the indicator in the volume access table taught by Heil and Nakayama to indicate Macklem's "read-caching lease" mode to indicate to the system the access mode limitations on the indicated volume, cache, or drive when that volume, cache, or drive is in "read-caching lease" mode. Ex.1004, ¶190.

A POSITA would have found it obvious to combine Heil, Nakayama, Srinivasan, and Macklem for the reasons stated in Section VI.F.1. A POSITA would implement either Srinivasan's cache management system that permits only reads to cache and Macklem's cache management system that permits only reads to cache in order to improve cache consistency and performance as disclosed by Srinivasan and Macklem. Ex.1018, 45-46; Ex.1009, 261; Ex.1004, ¶191. Indeed, Heil already discloses that a distributed lock manager should be used to limit access to shared storage (Ex.1006, 2:18-32), and a POSITA would merely look to Srinivasan and Macklem for specific implementations of how to limit access. Ex.1004, ¶191. Finally, the POSTIA would have understood that storing an indicator value indicating Srinivasan's and Macklem's access mode in the volume access table would inform the system of the access mode of the cache. *Id.* Thus, Heil in view of Nakayama, Srinivasan, and Macklem suggests element 5.c.

iv. Claim 5.d

Heil in view of Nakayama, Srinivasan, and Macklem suggests element 5.d. Ex.1004, ¶¶193-198. As discussed in Section VI.F.2.i above, Heil in view of

Nakayama suggests checking an access mode in a volume access table. In addition, both Srinivasan and Macklem suggest that, if the access mode is a value other than exclusive or shared, reads and writes are performed directly to a disk without using the cache memory.

As discussed in the preceding section, Srinivasan discloses multiple modes of accessing a cache, including a “write-shared” mode. Ex.1018, 46. In “write-shared” mode, because multiple clients may have access to the same file, “none of the clients (writers or readers) are allowed to cache it.” Ex.1018, 47. “For writers, this reverts to the write-through policy of NFS, and provides the same single-copy consistency between a writer and the server. For readers, this is stricter than the NFS mechanism; each read operation goes directly to the server.” *Id.* Thus, Srinivasan discloses that both read and write operations under “write-shared” mode are performed directly to disk without using the cache memory. Ex.1004, ¶195. As discussed above, a POSITA would understand that controlling access to storage and granting exclusive storage access to a specific client achieves Srinivasan’s described procedures for limiting access to files. For example, Srinivasan states that “by tracking open and close operations the Sprite file server knows not only which clients are currently using a file, but whether any of them are potentially writers,” indicating that its caches are read and write controlled. Ex.1018, 46-47; Ex.1004, ¶195. Similar to above, the POSITA would further modify the indicator

in the volume access table taught by Heil and Nakayama to indicate Srinivasan's "write-shared" mode that reads and writes are made to disk rather than cache. *Id.*

In addition to the read-caching lease and write-caching lease discussed in the preceding sections, Macklem also describes a non-caching mode. Ex.1009, 264 ("Whenever a client wishes to access a file's data it must hold one of three types of lease: read-caching, write-caching or non-caching. The latter type requires that all file operations be done synchronously with the server via the appropriate RPCs."); Ex.1004, ¶196. In non-caching mode, the client does not use cache data. Ex.1009, 263 ("The basic principle is that the server disables client caching of files whenever concurrent write sharing could occur"), 264 ("Diagram 3 depicts how read and write leases are replaced by a non-caching lease when there is the potential for write sharing."). Reads and writes are therefore performed to disk without using cache. Similar to above, the POSITA would further modify the indicator in the volume access table taught by Heil and Nakayama to indicate Macklem's non-caching mode that reads and writes are made to disk rather than cache. Ex.1004, ¶196.

A POSITA would have found it obvious to combine Heil, Nakayama, Srinivasan, and Macklem for the reasons stated in Section VI.F.1. A POSITA would implement either Srinivasan's cache management system that reads and writes directly to a disk without using cache memory and Macklem's cache

management system that reads and writes directly to a disk without using cache memory in order to improve cache consistency and performance as disclosed by Srinivasan and Macklem. Ex.1018, 45-46; Ex.1009, 261; Ex.1004, ¶197. Indeed, Heil already discloses that a distributed lock manager should be used to limit access to shared storage (Ex.1006, 2:18-32), and a POSITA would merely look to Srinivasan and Macklem for specific implementations of how to limit access. Ex.1004, ¶¶197.

Thus, Heil in view of Nakayama, Srinivasan, and Macklem suggests element 5.d.

VII. SECONDARY CONSIDERATIONS

Petitioner is unaware of any evidence related to secondary considerations, whether identified by the prosecution history or by PO in the parallel district court litigation, known to the industry, or identified by Dr. Franzon. Ex.1004, ¶¶199-201. To the extent PO makes any arguments regarding secondary considerations, Petitioner reserves the right to further address them at that time.

VIII. DISCRETION

A. *Fintiv* Factors Favor Institution

An evaluation of the factors under *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020), weighs against discretion to deny institution. Moreover, this Petition presents compelling evidence of unpatentability and, per the PTO

interim guidance dated 6/21/2022 (Ex.1011) as clarified by *CommScope Technologies, LLC v. Dali Wireless Inc.*, IPR2022-01242, Paper No.23 (PTAB Feb. 27, 2023), demonstrates that the PTAB should not discretionarily deny institution.

Factor 1 is whether the court granted a stay or whether evidence exists that one may be granted if IPR is instituted. Petitioner has filed a motion to stay the parallel district court litigation based on Petitioner's IPR2023-00120 filed against U.S. Patent 10,154,092. Ex.1017. Although the Court has not yet acted on Petitioner's Motion, the district court litigation has already been stayed once. Ex.1011. Thus, this factor weighs against discretionary denial.

Factor 2 is the proximity of the court's trial date to the projected statutory deadline for the PTAB's FWD. Judge Yeakel's scheduling order set a *Markman* hearing for November 9, 2023 and has not yet set a trial date. Ex.1019. The case has since been reassigned to Judge Pitman. The median time to trial for civil cases before Judge Pitman is 27.2 months (Ex.1009), meaning that any trial in the litigation likely will not occur until well after a FWD issues in this IPR. Thus, this factor weighs strongly against discretionary denial.

Factor 3 is the investment in the parallel proceeding by the court and the parties. The Court and the parties have not yet invested significant resources in the litigation. Following Petitioner's answer to PO's original complaint, the case was

transferred from the Waco Division to the Austin Division of the Western District of Texas and thereafter assigned to Judge Lee Yeakel. Following an initial case management conference on November 3, 2022, Judge Yeakel entered a pretrial schedule, including briefing on a motion to stay pending IPR of related U.S. Patent No. 10,154,092 (“’092 patent”) and a claim construction hearing for the ’092 patent scheduled for November 9, 2023. On November 10, 2022, the case was reassigned to Judge Robert Pitman. No trial date has been set, fact discovery has not commenced, and a motion to strike and dismiss PO’s amended complaint, in addition to Petitioner’s motion to stay pending IPR, remain pending. Thus, this factor weighs strongly against discretionary denial.

Factor 4 is the overlap between issues raised in the petition and in the parallel proceeding. Factor 4 weighs against discretionary denial. At present, it is unclear how much overlap-if any-there will be between the two proceedings as invalidity contentions are not due until June 1, 2023. However, there “should not be any overlap by the time of trial” as this Petition challenges all claims of the ’988 Patent and the final written decision will likely precede the trial and eliminate any overlap. *Google v. EcoFactor*, IPR2022-00473, Paper 7 at 12 (PTAB Aug. 3, 2022).

Factor 5 is whether petitioner and defendant in the parallel proceeding are the same party. Petitioner is defendant in one of the three remaining district court

litigations. However, “due to the statutory estoppel provision of section 315(e)(2), since the Final Written Decision will precede trial, this factor weighs strongly against” discretionary denial. *Google*, IPR2022-00473, Paper 7 at 12-13.

Finally, **factor 6** are other circumstances that impact the Board’s exercise of discretion, including the merits. Factor 6 weighs strongly against discretionary denial. Petitioner respectfully submits that the unpatentability challenge presented in this Petition is compelling. As discussed above, factors 1-5 already favor denial of institution, and Section VI.E presents a compelling case of unpatentability by showing that every element of every challenged claim was already well known in the art and that a person of ordinary skill could readily combine those elements to arrive at the challenged claims based on the knowledge of the art at the time of invention. *Commscope Tech. LLC v. Dali Wireless, Inc.*, IPR2022-01242, Paper 23 at *5 (PTAB Feb. 27, 2023) (“In circumstances where, however, the Board’s analysis of Fintiv factors 1–5 favors denial of institution, the Board shall then assess compelling merits.”). As *Fintiv* noted, “if the merits of a ground raised in the petition seem particularly strong...institution of a trial may serve...overall system efficiency and integrity.” *Fintiv*, 14-15.

Further, as discussed above in Section V, the ’988 Patent issued without considering any of the art presented in this Petition. Institution is thus consistent

with the significant public interest against “leaving bad patents enforceable.”

Thryv, Inc. v. Click-To-Call Techs., LP, 140 S. Ct. 1367, 1374 (2020).

Accordingly, all *Fintiv* factors favor institution. Based on a “holistic view of whether efficiency and integrity of the system are best served,” the facts here weigh against exercising discretionary denial. *Samsung Elecs. Co. Ltd. v. Dynamics Inc.*, IPR2020-00505, Paper 11 at 15 (PTAB Aug. 12, 2020).

B. *General Plastic* is Inapplicable Here

This is the first IPR Petition filed by Google to challenge the ’988 Patent. Thus, *General Plastic* is not applicable. See *General Plastic Industrial Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper No.19 at 3, 8, 15-19 (PTAB Sept. 6, 2017). While Microsoft has also filed a Petition challenging the ’988 Patent, Google prepared its IPR independently of Microsoft with no knowledge of the contents of the Microsoft petition, as evidenced by the short time between Google’s and Microsoft’s filings. There is also no overlap in the prior art between the Petitions.

C. 35 U.S.C. §325(d) Favors Institution

This is the first post-grant proceeding filed by Google to challenge the ’988 Patent and, as discussed above, the ’988 Patent was issued without considering any of the art presented in this Petition. Thus, the grounds presented in this IPR are not

cumulative or duplicative of any prior art or arguments previously presented to the Patent Office.

To the extent PO makes any arguments regarding discretionary denial, Petitioner reserves the right to further address them at that time.

IX. CONCLUSION

For the foregoing reasons, Petitioner requests institution of IPR for claims 1-6 based on the above-specified grounds.

Respectfully submitted,

/Minghui Yang/

Minghui Yang
Reg. No. 71,989
Counsel for Petitioner
CARMICHAEL IP, PLLC
8607 Westwood Center Drive, Suite 270
Tysons, VA 22182
Email: mitch@carmichaelip.com
Telephone: (703) 646-9248

Date: March 24, 2023

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITS

This Petition for *Inter Partes* Review (the “Petition”) consists of 13,888 words, excluding table of contents, table of authorities, certificate of service, this certificate, or table of exhibits. The Petition complies with the type-volume limitation of 14,000 words as mandated in 37 C.F.R. § 42.24. In preparing this certificate, counsel has relied on the word count of the word-processing system used to prepare the paper (Microsoft Word).

Respectfully submitted,

/Minghui Yang/

Minghui Yang

Date: March 24, 2023

CERTIFICATE OF SERVICE

The undersigned hereby confirms that the foregoing Petition for *Inter Partes* Review, associated exhibits, and Power of Attorney were caused to be served on March 24, 2023 via overnight courier upon the following counsel of record for Patent Owner in the related district court litigation:

William P. Ramey, III
Ramey LLP
5020 Montrose Blvd., Suite 800
Houston, Texas 77006
(713) 426-3923
wramey@rameyfirm.com

The foregoing were also caused to be served on March 24, 2023 via overnight courier upon the following counsel of record for Patent Owner:

Toler Law Group
8500 Bluffstone Cove
Suite A201
Austin, Texas 78759

Respectfully submitted,

/Minghui Yang/

Minghui Yang

Date: March 24, 2023