

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

LS CLOUD STORAGE TECHNOLOGIES, LLC,
Patent Owner.

U.S. Patent No. 6,549,988
Issued: April 15, 2003
Application No.: 09/236,409
Filed: January 22, 1999

Title: DATA STORAGE SYSTEM COMPRISING
A NETWORK OF PCS AND METHOD USING SAME

DECLARATION OF JON B. WEISSMAN

MICROSOFT CORP.
EXHIBIT 1003

TABLE OF CONTENTS

	Page(s)
I. INTRODUCTION AND ENGAGEMENT	4
II. BACKGROUND AND QUALIFICATIONS	4
III. MATERIALS CONSIDERED AND INFORMATION RELIED UPON REGARDING '988 PATENT	8
IV. UNDERSTANDING OF PATENT LAW	9
V. SUMMARY OF OPINIONS	13
VI. LEVEL OF SKILL IN THE ART	13
VII. THE STATE OF THE ART	13
VIII. OVERVIEW OF THE '988 PATENT	21
IX. CLAIMS 1 AND 3 ARE ANTICIPATED BY BERGSTEN	25
A. Bergsten	25
B. Claim 1	31
C. Claim 3	47
X. CLAIMS 2, 4, AND 6 ARE OBVIOUS OVER BERGSTEN IN VIEW OF THE KNOWLEDGE OF A POSITA	48
A. Claim 2	49
B. Claim 4	57
C. Claim 6	58
XI. CLAIMS 2, 4, AND 6 ARE OBVIOUS OVER BERGSTEN IN VIEW OF BLUMENAU	72
A. Blumenau	72
B. Rationale For Combining Bergsten And Blumenau	74

C.	Claim 2	77
D.	Claim 4	82
E.	Claim 6	83
XII.	CLAIM 5 IS OBVIOUS OVER BERGSTEN IN VIEW OF SMITH	96
A.	Smith.....	96
B.	Rationale For Combining Smith With Bergsten	98
C.	Claim 5	99
XIII.	CLAIM 5 IS OBVIOUS OVER BERGSTEN IN VIEW OF BLUMENAU AND SMITH.....	106
A.	Rationale For Combining Smith With Bergsten And Blumenau.....	106
B.	Claim 5	107

I, Jon B. Weissman, do hereby declare as follows:

I. INTRODUCTION AND ENGAGEMENT

1. I have been retained as an independent expert on behalf of Microsoft Corporation in connection with the above-captioned Petition for Inter Partes Review (“IPR”) to provide my analyses and opinions on certain technical issues related to U.S. Patent No. 6,549,988 (hereinafter “the ’988 Patent”).

2. I am being compensated at my usual and customary rate of \$700 per hour for the time I spent in connection with this IPR. My compensation is not affected by the outcome of this IPR.

3. Specifically, I have been asked to provide my opinions regarding whether claims 1-6 (“Challenged Claims”) of the ’988 Patent would have been obvious to a person having ordinary skill in the art (“POSITA”) as of January 22, 1999. It is my opinion that each Challenged Claim would have been obvious to a POSITA after reviewing the prior art discussed herein.

II. BACKGROUND AND QUALIFICATIONS

4. I am an expert in the field of computer science. My research is in the broad area of distributed systems. I work predominantly on cloud computing, mobile computing, and high-performance computing, developing techniques to improve performance, reliability, and energy efficiency. In much of my work on these platforms, I have focused on how to build distributed storage systems.

5. In formulating my opinions, I have relied upon my training, knowledge, and experience in the relevant art. A copy of my curriculum vitae is appended to this declaration as Appendix A and provides a description of my professional experience, including my academic and employment history, publications, conference participation, awards and honors, and more. The following is a brief summary of my relevant qualifications and professional experience.

6. I have held faculty positions at the Department of Computer Science and Engineering University of Minnesota since 1999. I am currently Professor of Computer Science, a position I have held since Summer of 2012. I was Assistant Professor of Computer Science from 1999 to 2003, and Associate Professor of Computer Science from Fall 2003 to Spring 2012.

7. I have more than 25 years of technical experience in the computer science field. I have consulted on a range of technical/legal projects including expert analysis, patent examination, intellectual property, and industrial projects, in the area of Internet applications, mobile computing, distributed systems, storage systems, and the cloud. I am Steering Committee Chair of the International ACM/IEEE Symposium on High-Performance Parallel and Distributed Computing and have held associate editorships of IEEE Transactions on Parallel and Distributed Systems and of IEEE Transactions on Computers.

8. I graduated from the Carnegie-Mellon University with a B.S. in Applied Mathematics and Computer Science in 1984. I earned an M.S. in Computer Science and a Ph.D. in Computer Science from the University of Virginia in 1989 and 1995, respectively.

9. During the 25-plus years since receiving my Ph.D., I have held multiple positions as a professor and researcher in the field of computer science. I have held faculty positions in Computer Science at the University of Minnesota since 1999 as detailed at paragraph 2, above. I also was a Visiting Researcher and Distinguished Visitor at the National e-Science Center at the University of Edinburgh from 2007 to 2008, and Assistant Professor, Computer Science at the University of Texas, San Antonio from 1995 to 1999.

10. As described in my curriculum vitae, I have extensive experience in the field of data storage and file systems.

11. At the University of Minnesota, I have been an Investigator with the Center for Research in Intelligent Storage (CRIS), consisting of faculty colleagues and both graduate and undergraduate students. CRIS is a partnership between universities and industry, and does research in several relevant areas, including storage and memory technologies, big data technologies, cloud computing, backup/restore technology, data deduplication technology, and file systems.

12. Currently, I am developing a distributed cloud storage system called Wiera that virtualizes data into containers, contents of which may be distributed across several storage locations hidden from the user. Wiera allows data to be stored in multiple storage tiers including volatile and non-volatile memory. Wiera policies allow data to be replicated in memory (i.e. cache), which requires cache consistency. I am also developing the Armada edge cloud, a dispersed cloud with an edge-based data store for persistent data. The data store may be replicated data across distributed locations for efficiency, requiring data consistency techniques.

13. In addition to my research, I have been teaching for nearly 25 years. I have taught numerous undergraduate and advanced graduate classes in distributed systems, networking, and operating systems, all with a major emphasis on storage, pertinent to the subject matter in this case. I have also supervised numerous Ph.D. and M.S. students while at the University of Minnesota, in areas such as cloud computing, cloud storage, and networking.

14. In total, I have been an author or co-author on more than 100 scholarly publications, dating back to 1994. These publications include articles in highly competitive rigorously reviewed refereed peer-reviewed journals, papers in selective conferences, books, and book chapters, many relating to data storage. I have also served as a program committee member for many conferences, including

IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing,
IEEE International Conference on Big Data and many others.

III. MATERIALS CONSIDERED AND INFORMATION RELIED UPON REGARDING '988 PATENT

15. In preparing this declaration, I have reviewed the following materials bearing Exhibit Nos. that I understand are being referenced in the IPR to which my declaration accompanies:

LIST OF EXHIBITS

No.	Description
1001	U.S. Patent No. 6,549,988 (“ the '988 Patent ”)
1002	File History of U.S. Patent No. 6,549,988
1004	U.S. Patent No. 6,073,209 (“Bergsten”)
1005	U.S. Patent Application No. 08/828,888
1006	U.S. Patent No. 6,421,711 (“Blumenau”)
1007	U.S. Patent Application No. 09/106,299
1008	Spritely NFS: Experiments with Cache-Consistency Protocols” (“Srinivasan”)
1009	R.H. Katz, Network-Attached Storage Systems, 1992 (“Katz”)
1010	Gang Ma, et al., An evaluation of storage systems based on network-attached disks, 1998 (“Ma”)
1011	Jim Handy, The Cache Memory Book, 1998 (excerpts) (“Handy”)
1012	Microsoft Computer Dictionary, 2d ed., 1994 (excerpts)

1013	Kinariwala et al., Programming in C, 1993 (“Kinariwala”)
1014	Parsaye, et al., Intelligent Database Tools & Applications, 1993 (“Parsaye”)
1015	Alan J. Smith, Cache Memories, 1982 (“Smith”)
1016	U.S. Patent No. 4,500,954 (“Duke”)
1017	U.S. Patent No. 4,433,374 (“Hanson”)
1018	Smith Copyright Registration

IV. UNDERSTANDING OF PATENT LAW

16. I am not an attorney. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to my opinions. My understanding of the law was provided to me by the Petitioner’s attorneys.

17. I understand that when considering the scope of the claims of a patent that the patent claim terms should generally be given the ordinary meaning that the terms would have to a person of ordinary skill in the art in question after reading the patent as of the earliest claimed priority date.

18. I understand that the person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the term appears, but in the context of the entire patent, including the specification. I further understand that the principal considerations regarding the scope and meaning of the claims are the plain language of the claim (including the surrounding claim language and context), the patent specification, and the prosecution history. I understand that

while a claim is to be read in light of the specification, one must generally avoid importing limitations into the claim from the specification. I am also informed that the prosecution history can often inform the meaning of the claim by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim scope narrower than it would otherwise be. I applied these understandings when considering the scope and meaning of the claims of the '988 patent.

19. I understand that a prior art reference anticipates an asserted claim, and thus renders the claim unpatentable, if all elements of the claim are disclosed in that prior art reference, either explicitly or inherently (i.e., necessarily present or implied).

20. I further understand that a claim is unpatentable if it would have been obvious. Obviousness of a claim requires that the claim would have been obvious from the perspective of a POSITA at the time the alleged invention was made. I understand that a claim could have been obvious from a single prior art reference or from a combination of two or more prior art references.

21. I understand that an obviousness analysis requires an understanding of the scope and content of the prior art, any differences between the alleged invention and the prior art, and the level of ordinary skill in evaluating the pertinent art.

22. I further understand that a claim would have been obvious if it unites old elements with no change to their respective functions, or merely substitutes one element for another known in the field, and that combination yields predictable results. While it may be helpful to identify a reason for this combination, I understand that there is no strict requirement of finding an express teaching, suggestion, or motivation to combine within the references. When a product is available, design incentives and other market forces can prompt variations of it, either in the same field or different one. If a POSITA can implement a predictable variation, obviousness likely bars its patentability. For the same reason, if a technique has been used to improve one device and a POSITA would recognize that it would improve similar devices in the same way, using the technique would have been obvious. I understand that a claim would have been obvious if common sense directs one to combine multiple prior art references or add missing features to reproduce the alleged invention recited in the claims.

23. I further understand that certain factors may support or rebut the obviousness of a claim. I understand that such secondary considerations include, among other things, commercial success of the patented invention, skepticism of those having ordinary skill in the art at the time of invention, unexpected results of the invention, any long-felt but unsolved need in the art that was satisfied by the alleged invention, the failure of others to make the alleged invention, praise of the

alleged invention by those having ordinary skill in the art, and copying of the alleged invention by others in the field. I understand that there must be a nexus—a connection—between any such secondary considerations and the alleged invention. I also understand that contemporaneous and independent invention by others is a secondary consideration tending to show obviousness.

24. I am not aware of any allegations by the named inventors of the '988 patent or any assignee of the '988 patent that any secondary considerations tend to rebut the obviousness of any Challenged Claim of the '988 patent.

25. I understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the descriptions found in the patent being considered.

26. I understand that other challenges to the patentability of a patent, including patent ineligibility, enablement, written description, and definiteness or clarity of claim language, cannot be raised in IPR proceedings before the Board.

27. I understand that Petitioner has the burden of proving unpatentability by a preponderance of evidence, which means that the claims are more likely than not unpatentable.

28. The analysis in this declaration is in accordance with the above-stated legal principles.

V. SUMMARY OF OPINIONS

29. As detailed below, it is my opinion that claims 1-5 of the '988 patent were obvious to a person of ordinary skill in the art ("POSITA"), at least in view of the prior art discussed below.

VI. LEVEL OF SKILL IN THE ART

30. A POSITA in January 1999 would have a Bachelor's degree in Computer Science, Computer Engineering, or Electrical Engineering, or an equivalent course of study, as well as two or more years of academic or industry experience in the field of computer networking, computer architecture and storage management. Lack of work experience could have been offset by additional education, and vice versa.

VII. THE STATE OF THE ART

31. By the early 1990s, computer scientists had recognized that computing performance could be improved by using distributed computer servers and workstations connected by high performance networks over the more traditional centralized mainframe systems. EX1009, Katz, 68. A 1992 paper titled "Network-Attached Storage Systems," authored by Randy Katz from the University of California, Berkeley, discussed distributed storage systems, including the common architecture used to improve data storage systems. EX1009.

32. Katz describes two channel architectures to connect “storage devices” to “I/O controllers”, including the “low end” SCSI (SCSI stands for “Small Computer System Interface”). “Channels provide the logical and physical pathways between I/O controllers and storage devices.” EX1009, 69. “SCSI is the channel interface most frequently encountered in small form factor (5.25” diameter and smaller) disk drives. SCSI treats peripheral devices in a largely independent fashion. For example, a disk drive is viewed as a linear byte stream.” EX1009, 70.

33. Figure 3, reproduced below, shows the internal organization of an I/O Controller. EX1009, Figure 3.

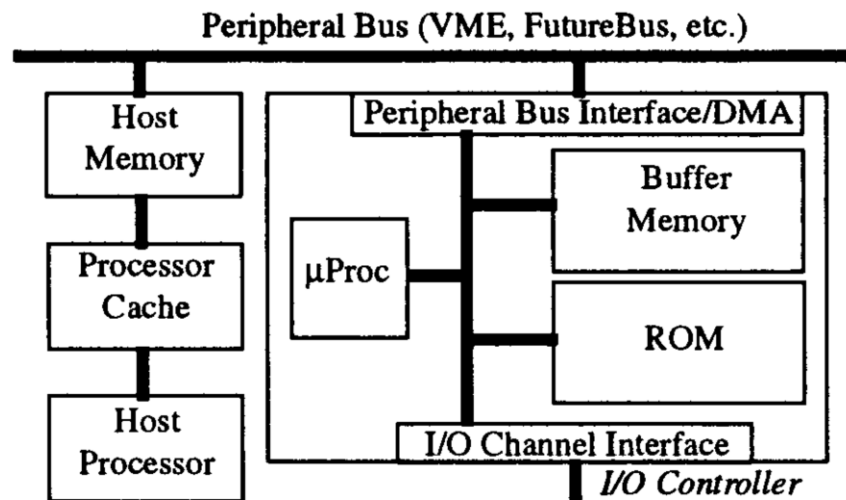


Figure 3: Internal Organization of an I/O Controller

34. Katz describes the controller as the element on the right of this figure, “the controller contains a microprocessor, a modest amount of memory dedicated to buffers and run-time data structures, a ROM to hold the controller firmware, a DME/peripheral bus interfaces, and I/O channel interface.” It’s clear from the

discussion above that the I/O channel interface is connected to the disk drives, e.g., via a SCSI channel, and that the connection to the host is via a separate path.

EX1009, 72.

35. Thus, a POSITA as of 1999 would be familiar with storage systems using dedicated I/O channels, including SCSI, between the host and storage, as well as distributed network communications between storage servers.

36. Caching was also common in this time frame. Ma and Reddy of Texas A&M University published a paper entitled, “An evaluation of storage systems based on network-attached disks” that included a discussion of cached network storage systems. EX1010, “Ma”. I understand that the Ma paper was published in 1998 by IEEE, and specifically was presented at the August 14, 1998 at the International Conference on Parallel Processing.

37. Figure 2 of the paper, replicated below, shows network-based storage, including a distributed cache.

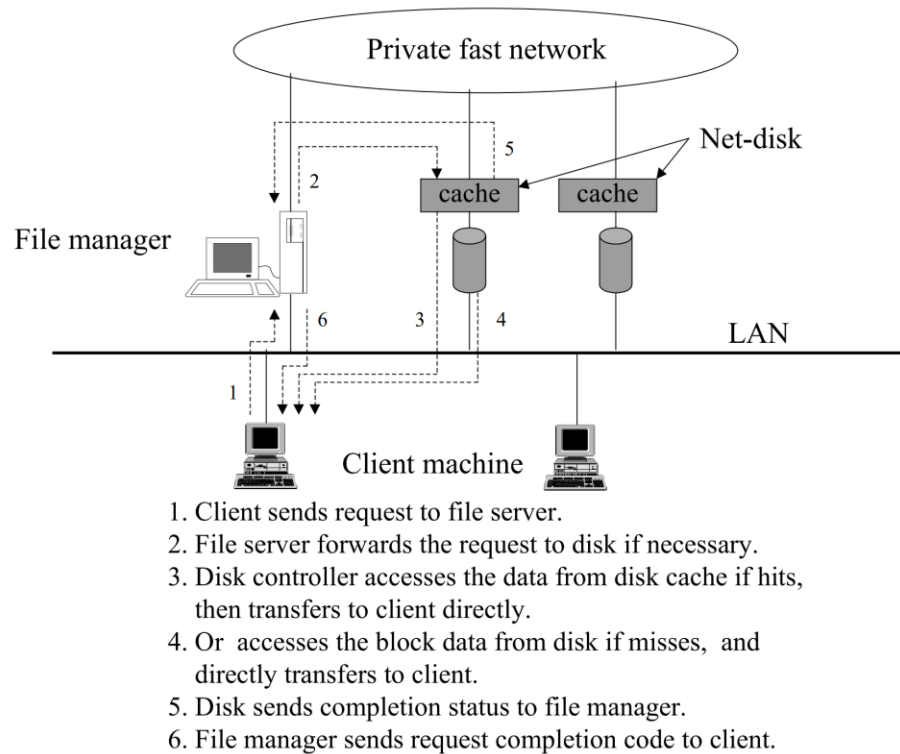


Figure 2: Net-disk:Storage system based on Network-attached disks.

38. Ma describes the operation of this distributed cache as follows:

Net-disk and server disk organizations employ caching at different places in the system. A single large cache space in the file manager in server-disk enables efficient utilization of cache space across all the data sets in the system. The smaller caches at each disk in net-disk organizations can lead to inefficiencies if disks are accessed non-uniformly. Our assumption of system-wide disk striping [10] however reduces this disadvantage.

EX1010, 3.

39. A POSITA would know that memory systems were designed to reduce the expense of memory, while maximizing speed and capacity. A POSITA understood the value of cache memory as a high-speed memory array for increasing the effective memory speed of a system. A basic primer of the use of cache memory is described in the 1998 treatise titled *THE CACHE MEMORY BOOK*, authored by Jim Handy (EX1011, “Handy”). Among other things, Handy describes considerations when designing cache systems, including considerations for multi-processor cached systems and maintaining coherency in cached systems. EX1011, *e.g.*, 138-155. Handy explains that caching improves speed, as it is faster to retrieve data from a cache memory than a disk. EX1011, 7-8; *see also* EX1012, 60 (defining “cache” as a “special memory subsystem in which frequently used data values are duplicated for quick access”).

40. Figure 1.12 of the treatise, replicated below, shows two examples of multiprocessor systems: a loosely coupled multiprocessor system or a distributed memory multiprocessor (DMM) in the top diagram and a tightly coupled system or shared memory machine in the bottom diagram.

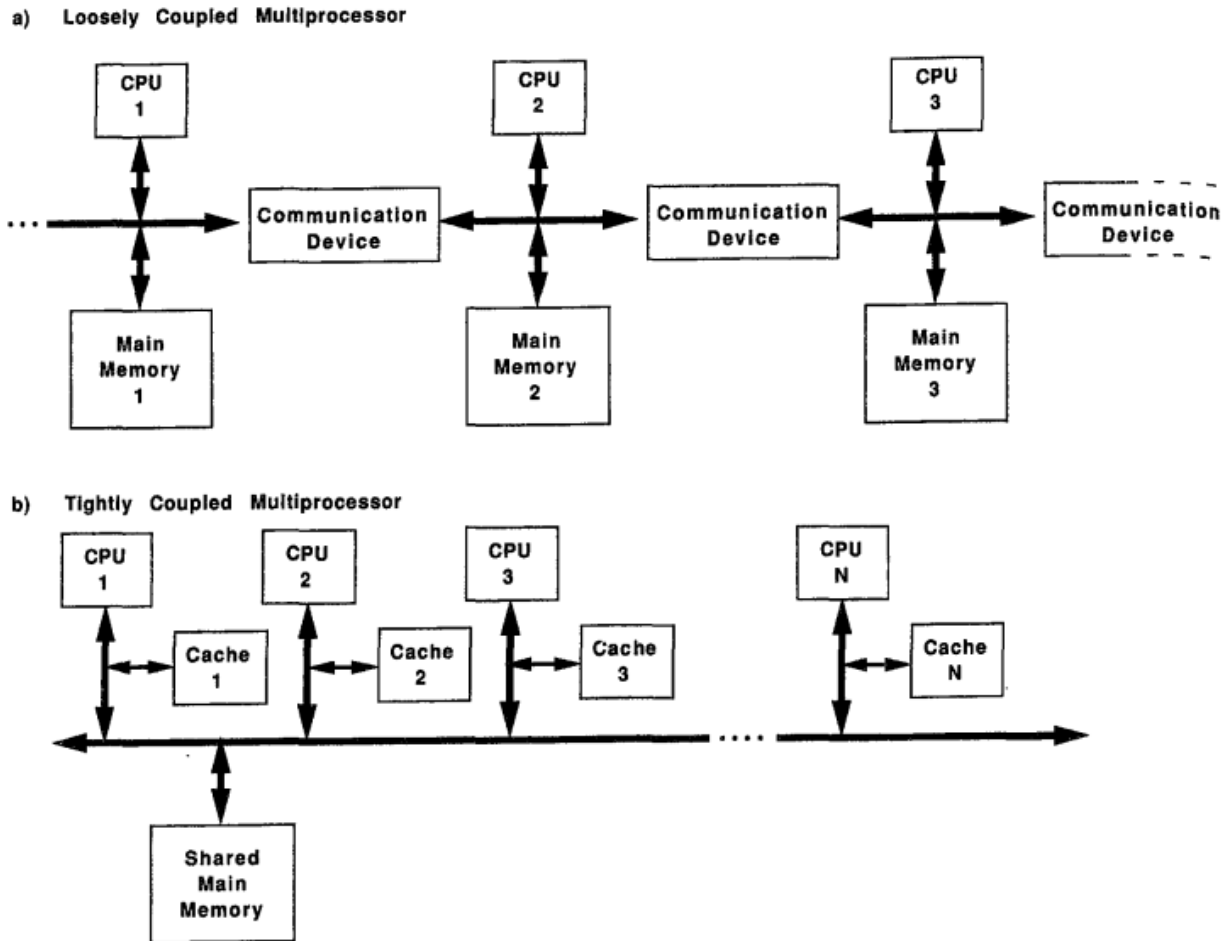


Figure 1.12. (a) A loosely coupled multiprocessing system. Each CPU has its own main memory and system bus. The processors communicate via the communication devices, which are usually accessed through each processor's system bus. (b) A tightly coupled multiprocessing system. All processors (or processor/cache complexes) communicate to a single main memory via a single system bus.

41. Caching was known to improve performance (*e.g.*, speed, etc.), but also presented issues related to cache coherency. This known concern, as well as techniques for addressing it, are described in the 1989 ACM paper titled “Spritely NFS: Experiments with Cache-Consistency Protocols,” authored by V. Srinivasan and Jeffrey C. Mogul (EX1008, “Srinivasan”).

Cache management strategies are central to performance, reliability, and correctness of distributed file services. Caching improves performance by avoiding unnecessary disk traffic, network traffic, and server use, but caching implies the potential existence of multiple copies of the same data, and keeping these multiple copies consistent is a challenge. This is especially true when the caches are kept by the clients of a distributed file service, which might be attempting concurrent access to the same file.

EX1008, 45; *see also* EX1011, 124 (“Cache coherency is the term given to the problem of assuring that the contents of the cache memory and those of main memory for *all* caches in a multiple cache system are either identical or under tight enough control that stale and current data are not confused with each other.”).

42. Cache policies are the rules of operation of the cache. (EX1011, 49. Cache-consistency strategies or protocols were known and implemented to assure “that two or more caches in a system stay coherent with each other.” EX1011, 142. The MESI Protocol is an example of such a protocol. EX1011, 156-158. The MESI protocol uses four states—Modified, Exclusive, Share, and Invalid—to maintain coherency of multiple cache systems. EX1011, 156.

43. Table 4.1, replicated below, shows the four MESI states and responses for different read and write operations. EX1011, 157.

Table 4.1. MESI

	Invalid	Shared	Exclusive	Modified
From CPU Bus				
Read Miss	Update line from main memory. Update status to Shared.	Update line from main memory. No status change.	Update line from main memory. Update status to Shared.	Evict modified line to main memory. Update line from main memory. Update status to Shared.
Read Hit	Does not occur.	Read cache data. No status change.	Read cache data. No status change.	Read cache data. No status change.
Write Miss	Write to cache line. Write to system bus. Update status to Exclusive.	Write to cache line. Write to system bus. Update status to Exclusive.	Write to cache line. Write to system bus. No status change.	Evict modified line to main memory. Write to cache line. Write to system bus. Update status to Exclusive.
Write Hit	Does not occur.	Write to cache line. Write to system bus. Update status to Exclusive.	Write to cache line. Update status to Modified.	Write to cache line. No status change.
From System Bus				
Read Miss	No response. No status change.	No response. No status change.	No response. No status change.	No response. No status change.
Read Hit	Does not occur.	No response. No status change.	Update status to Shared.	Abort snooped cycle. Write modified line to main memory. Update status to Shared.
Write Miss	No response. No status change.	No response. No status change.	No response. No status change.	No response. No status change.
Write Hit	Does not occur.	Update status to Invalid.	Update status to Invalid.	Update status to Invalid.

44. There were known strategies for handling write cycles in caches.

EX1011, 64. One example is write-through caching which causes disk to be updated during all write cycles. EX1011, 65. Another example is write-back caching which causes writes to be stored in cache and periodically written back to disk with the fresher data which is in the cache. EX1011, 65. Another strategy is write invalidation which causes invalidations during write cycles for consistency purposes. EX1011, 64.

45. Thus, a POSITA as of 1999 would be familiar with cache management strategies, including those that use different modes (e.g., exclusive, shared, etc.) and invalidations for consistency purposes.

VIII. OVERVIEW OF THE '988 PATENT

46. The '988 is directed to a data storage system that consists of volumes (i.e. named logical area(s) of the physical disk(s) denoted by a volume #, 1, 2, etc.). In the '988, the volume is akin to a storage device (i.e. a place where data is stored). The data within a volume may be cached in any number of distributed data caches for performance, where the same data may be kept in different caches (i.e. multiple cache copies). This introduces the issue of data consistency where the caches may contain different values for the same data, and different data from the authoritative copy of that data on disk. Hosts run user programs that issue I/O requests for data (e.g. read/write) within the overall data storage system. Hosts may be channel-attached (i.e. direct or locally attached) to data storage systems that can cache data locally, and remote or network attached to other data storage systems. Figure 1, shown below, illustrates “[t]he PC data storage system 131 services a plurality of channel attached host processors 111, 112 using channels 121, 122, and a plurality of network attached host processors 106, 107 using network link 151, and a plurality of network attached data storage systems 132, 133 using network

links 152, 153. PC storage system 132 services channel attached hosts 157, 158.”

EX1001, 4:67-5:6.

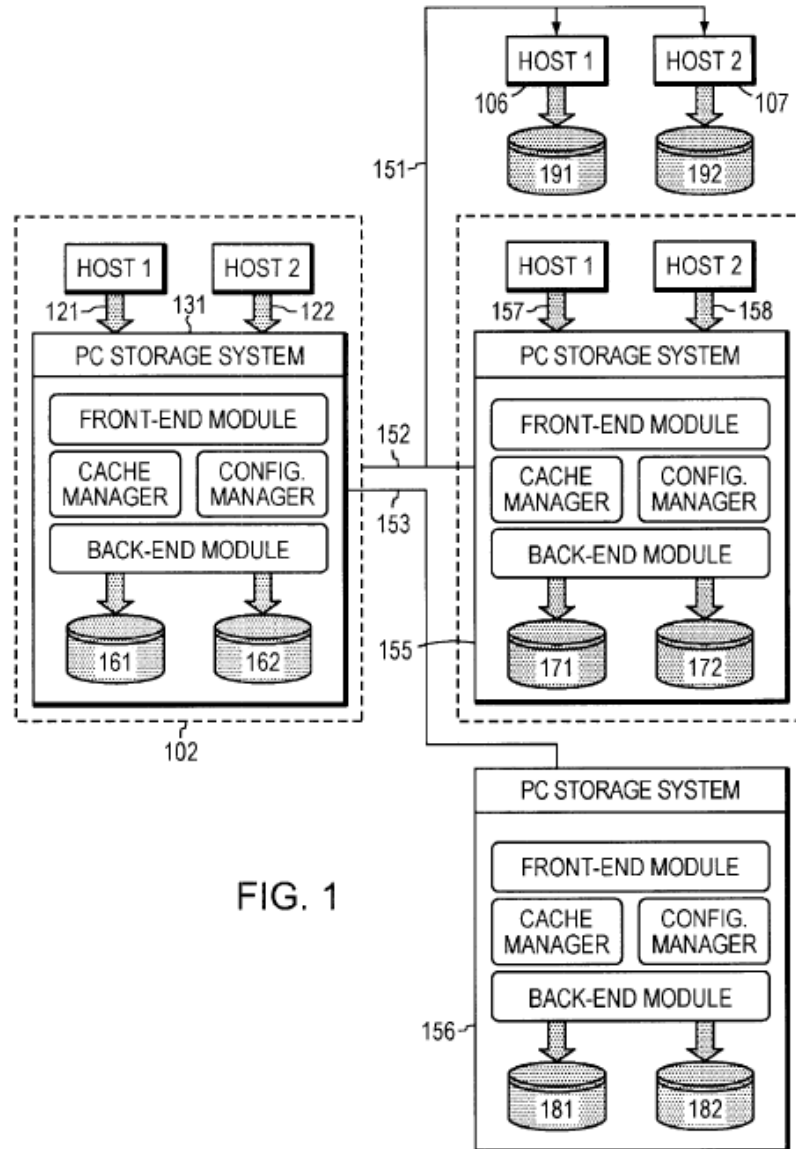


FIG. 1

EX1001, '988 Patent Fig. 1

The preferred embodiment of the alleged invention uses well-established technologies such as SCSI channels for I/O traffic and Ethernet link for network

traffic. In FIG. 2, the alternate embodiment of the alleged invention uses fiber channel technology for both I/O traffic and network traffic. The fiber channel connects computers and hard disks into one logical network.

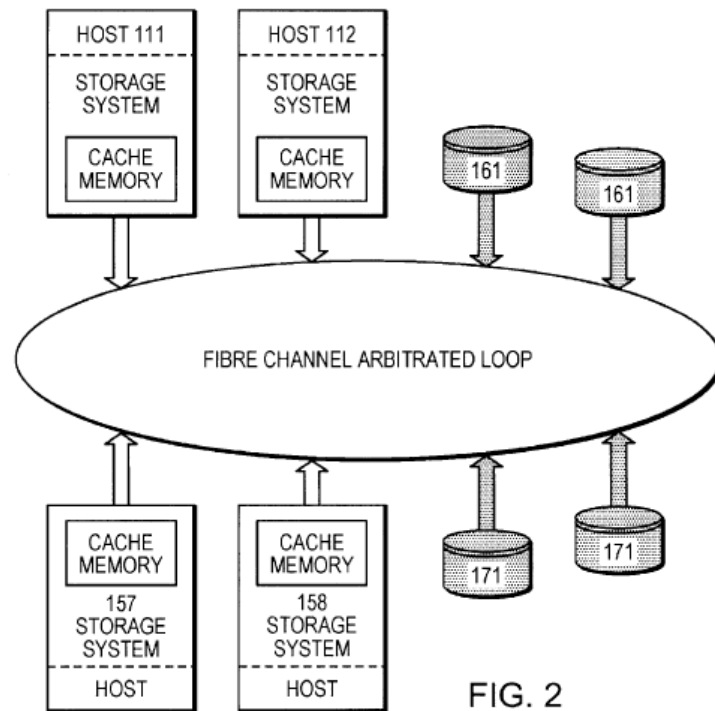


FIG. 2

EX1001, '988 Patent Fig. 2

47. Although not clearly stated in the '988, it appears that the volume associated with a requested data would be known to the system. A host may access data within a volume subject to four different access modes *defined at the volume level*. These access modes are stored in a host volume table that specifies that host N may access volume M (i.e. any of its contained data) using access mode A. The

access modes correspond to the way in which I/O operations will be performed with respect to consistency. The four modes are as follows:

Exclusive write means that when a host writes the data, the cache manager will ensure that this new value becomes the accepted global value of the data at the cache local to the host. To accomplish this, all (older) copies of this data in other caches must be invalidated to eliminate inconsistent copies. To do this, invalidate messages are sent to the hosts that are locally connected to the caches that store copies.

Shared write means that when a host writes the data, the cache manager will allow it to be written locally, and also sent to the remote host so it can be written to the exclusive cache copy held at its local cache. As above, this may cause invalidation of other previously cached versions of the data.

Exclusive read means that when a host that has exclusive read access reads the data locally, the cache manager will enable it to be read and cached locally. Otherwise, remote read requests will be directed to this host.

Shared read means that when a host reads the data held locally in its cache, the read proceeds from the cache. If the data is not in the cache, it is retrieved from the host that has the data in exclusive read mode and then cached for subsequent access.

IX. CLAIMS 1 AND 3 ARE ANTICIPATED BY BERGSTEN

48. As explained below, claims 1 and 3 are anticipated by U.S. patent No. 6,073,209 (“Bergsten”) in view of U.S. Patent No. 6,421,711 (“Blumenau”).

A. Bergsten

49. Bergsten, titled “Data Storage Controller Providing Multiple Hosts With Access To Multiple Storage Subsystems,” issued from U.S. Patent Application No. 08/828,888, filed on March 31, 1997. Bergsten’s 1997 application discloses those aspects of the issued Bergsten patent relied on herein. EX1005.

50. Bergsten is directed to “a storage controller which allows multiple host computer systems at different locations to access any of multiple copies of stored data.” EX1004, 3:2-4. The storage controller is not dependent upon any particular hardware or software and it emulates a local storage array and a host computer system for the storage array. *Id.*, 3:11-17. The host computer accesses “stored data using virtual device addresses, which are mapped to real device addresses by the storage controller.” *Id.*, 3:17-19.

51. Bergsten’s Figure 1 illustrates multiple storage controllers (8) providing multiple host computers (7) access to multiple storage arrays (4-1, 4-2, 4-M).

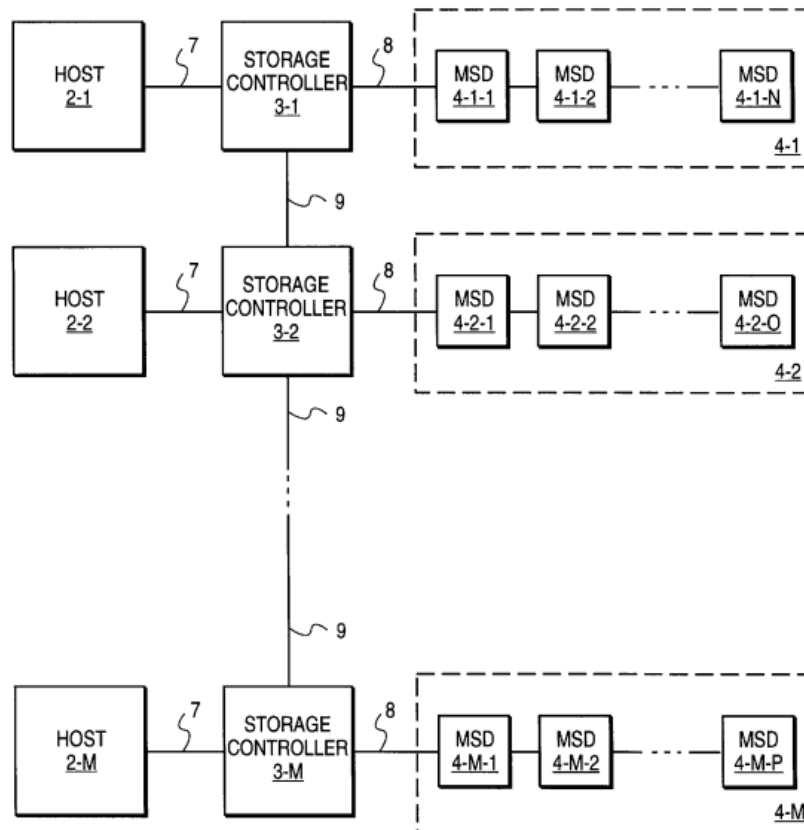


FIG. 1

EX1004, Bergsten Fig. 1

52. Bergsten explains that each of the storage arrays (4-1, 4-2, 4-M) includes multiple mass storage devices. *Id.*, 3:28-30. Bergsten explains that the communication paths between hosts and storage controllers may be implemented via a local area network (LAN), Fast Ethernet, Asynchronous Transfer Mode, T1 (or T3), Fiber Distributed Data Interface, “or any other suitable type of link.” *Id.*, 3:35-45.

53. Bergsten further discloses that the data is accessed from storage “using virtual addressing, such that a host computer system has no knowledge of which physical storage device is being accessed.” *Id.*, 4:48-50. More specifically, Bergsten recites:

A local host computer accesses data by transmitting a (virtual) host address to its local storage controller. The host address is then mapped to a real address representing a location on one or more physical MSDs, as will be described below in detail. The mapping is completely transparent to all of the host computers. To improve performance, the storage controller may distribute a single copy among multiple MSDs, which may be located in different storage arrays. Hence, in the above described mapping process, a single host address may map to multiple physical addresses, which may be distributed among multiple MSDs, and such MSDs may further be located in different storage arrays.

Id., 6:10-21.

54. Bergsten’s Figure 3 illustrates the hardware architecture of a storage controller. Notably “[t]he host device interface 14 connects communication path 7 to the bus 17 in order to connect a local host computer to the storage controller 3.” *Id.*, 6:41-43. Bergsten provides that the host device interface can be a SCSI adapter. *Id.*, 6:46-52. Bergsten explains that “[t]he controller device interface 16

connects communication path 9 to the bus 17 in order to connect a remote storage controller to a storage controller 3,” and that the interface may be of the same types noted above. *Id.*, 6:56-61

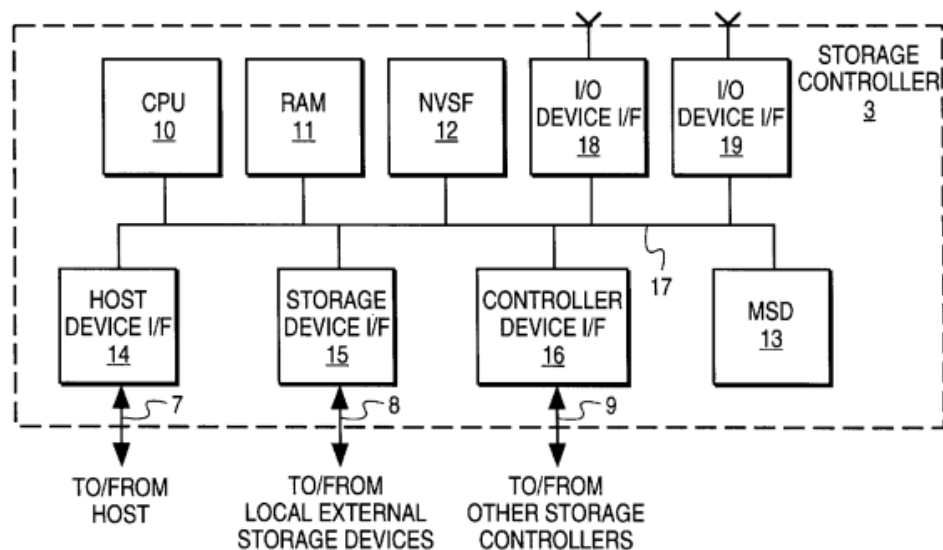


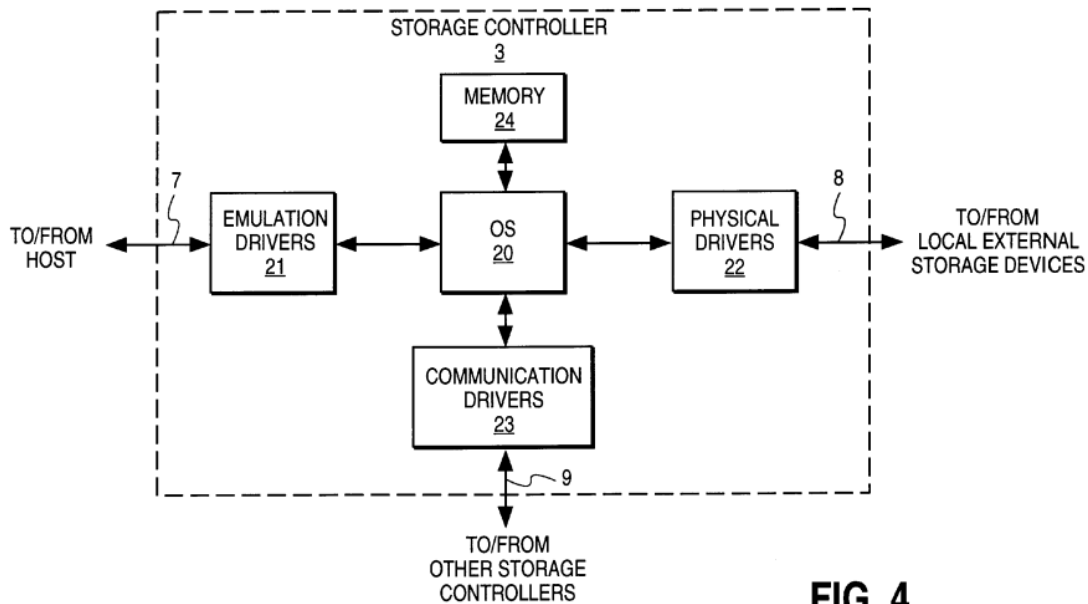
FIG. 3

EX1004, Bergsten Fig. 3

55. Bergsten’s Figure 4, illustrates the functional modules within a storage controller, including “an operating system (OS) 20 coupled to emulation drivers 21, physical drivers 22, communication drivers 23, and local memory 24.” *Id.*, 7:24-26.

The OS 20 controls the overall operation of the storage controller, including emulation, data flow, caching of data in the local memory 24, control of data mirroring, and error recovery. Emulation drivers 21 are controlled by the

OS 20 to provide communication of data between the storage controller 3 and the local host computer 2 via communication path 7. In particular, the emulation drivers 21 receive read and write requests from the host computer 2 and convert the read and write requests into a format recognized by the OS 20. The emulation drivers 21 further function to transfer any data received from the local storage array or a remote storage array to the local host computer. The communication drivers 23 are controlled by the OS 20 to provide communication of data between the storage controller 3 and remote storage controllers via communication path 9. The physical drivers 22 are controlled by the OS 20 to transfer data between the storage controller 3 and its local storage array. Such transfer includes the transformation of commands and control data from the format used by the storage controller 3 to the format recognized by the local storage array (e.g., SCSI). *Id.*, 7:29-49.

**FIG. 4****EX1004, Bergsten Fig. 4**

56. Bergsten further discloses using cache storage:

At various times, the storage controller must read data from one or more MSDs into its internal memory (“the cache”) or transfer data from the cache to one or more MSDS, these procedures are known as “staging” and “destaging,” respectively. Staging is generally performed, for example, when executing a host read request. Destaging is generally performed when executing a host write request. FIG. 10 illustrates a routine for staging data, which can be performed by a storage controller of the present invention. In step 1001, the storage controller selects the “nearest” free device housing the data of interest. In this context, the term “nearest” refers to the mass storage device associated with the smallest overall

access time or earliest availability, rather than referring to physical distance. (Note, however, that physical distance may be a factor in the total access time.) In a given situation, the nearest storage device may be the local memory within the local storage controller, an MSD within the local storage array, or an MSD within a remote storage array. After selecting the nearest free device, in step 1002 the logical block address of the data is mapped to a physical block address in the manner described above. In step 1003, the block or blocks are read from the selected device using the physical block address. If the read is successful (step 1004), then in step 1005 a read error recovery routine is called if any block was bad.

Id., 10:23-47.

57. Among other things, Bergsten discloses a tree hierarchy used by the storage controller to map host information to information regarding logical devices. *Id.*, Figs. 7, 8, 9; 9:9-10:22.

B. Claim 1

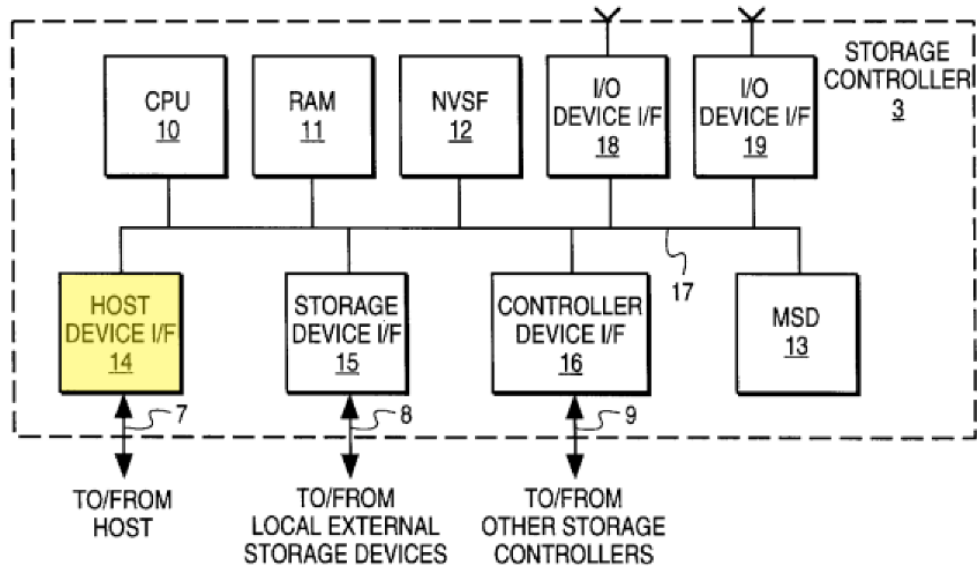
[1.1]¹ A computer suitable for use in a data storage system comprising a network interconnecting a plurality of such computers, the computer comprising:

¹ Reference numbers in the format of [claim#.limitation#] are added throughout for ease of reference.

58. Bergsten discloses this limitation by disclosing a storage controller (a computer) as part of a computer network of interconnected storage controllers (plurality of computers), each coupled to a storage array, and each storage controller coupled to at least one host processing system and to at least one other storage controller to control access of the host processing systems to the storage device. *Id.*, Abstract; see also 3:20-39, Fig. 1.

[1.2] an I/O channel adapter for accepting an incoming I/O request from a host;

59. A POSITA would understand that Bergsten's networked storage controllers receive I/O requests from the hosts. Specifically, Bergsten includes an I/O adapter accepting incoming I/O requests from a host in the form of host device interface 14 of a storage controller, which receives I/O requests from the host over communication path 7. *Id.*, 6:41-55; see also *Id.*, 8:24-26 (discussing a storage controller receiving I/O read or write requests from a host.) Bergsten discloses this limitation because, as noted above, Bergsten expressly discloses that the host device interface can be a SCSI adapter. *Id.* The host device interface 14 connecting the storage controller "to/from host" via communication path 7 is shown in Fig. 3, reproduced below. *Id.*, 2:13, Fig. 3.



EX1004, Bergsten Fig. 3 (highlighting added)

60. The claimed “I/O channel adapter” corresponds to the ’988 patent’s recitation of “I/O adapter driver software has been modified to accept target SCSI I/O requests from host 111 and 112.” EX1001, 5:52-54. Thus, this limitation is broad enough to encompass Bergsten’s disclosure of using an SCSI adapter as the host device interface.

[1.3] 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;

61. Bergsten discloses this limitation. As discussed above, Bergsten discloses functional modules within the storage controllers, including controlling the flow and caching of data. EX1004, 7:29-35. Specifically, the storage controller of Bergsten includes configuration manager software as part of its operating system, OS 20, which “controls the overall operation of the storage controller,

including emulation, *data flow, caching of data in the local memory 24*, control of data mirroring, and error recovery.” *Id.*, 7:29-32 (emphasis added).) The functional modules of Bergsten, including OS 20, are depicted in Fig. 4, reproduced below.

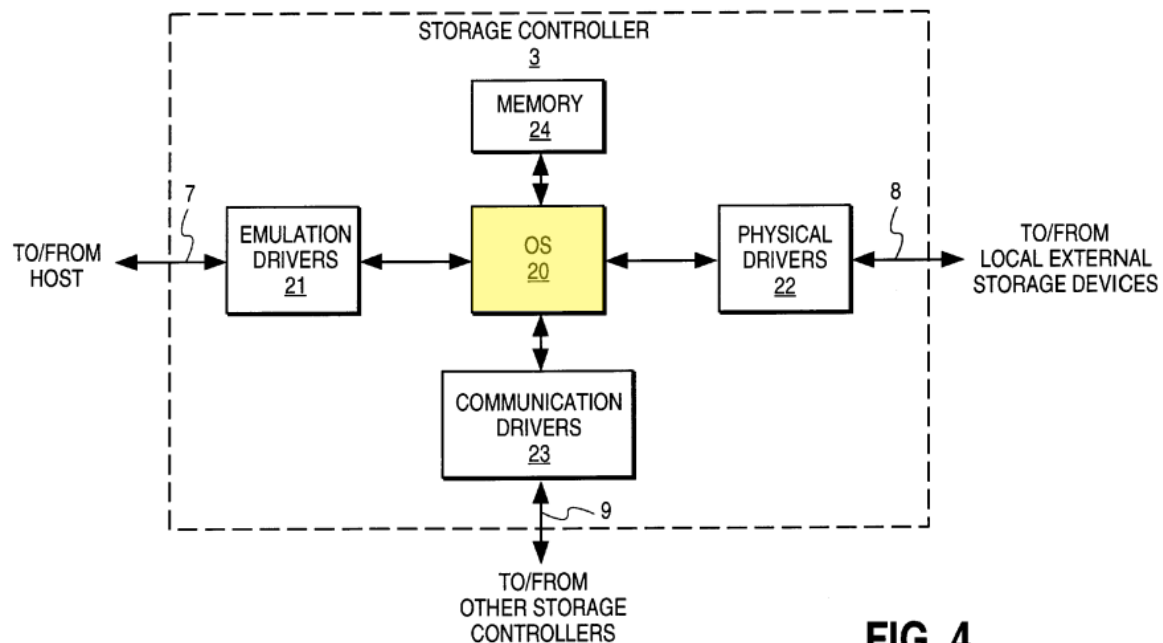


FIG. 4

EX1004, Bergsten Fig. 4 (highlighting added)

62. Bergsten discloses routines that satisfy each of the claimed alternative routing decisions (i.e., route request to cache, route request to disk or reject request) and because the storage controller’s OS 20 controls the operations of data flow and memory caching a POSITA would understand software components of the OS 20 module to control how the controller’s host device interface routes requests received from a host. For example, with regard to routing a host request to cache, Bergsten discloses performing a standard read operation in response to a

read request (*id.*, 12:37-38), including deciding to route the request to cache if the block is cached, and if so, transferring that cached data to the host. *Id.*, 12:47-56, Fig. 14 at 1404-1406. Bergsten's standard read operation performed by the storage controller is depicted in Fig. 14, reproduced below. *Id.*, 2:39-41, Fig. 10.

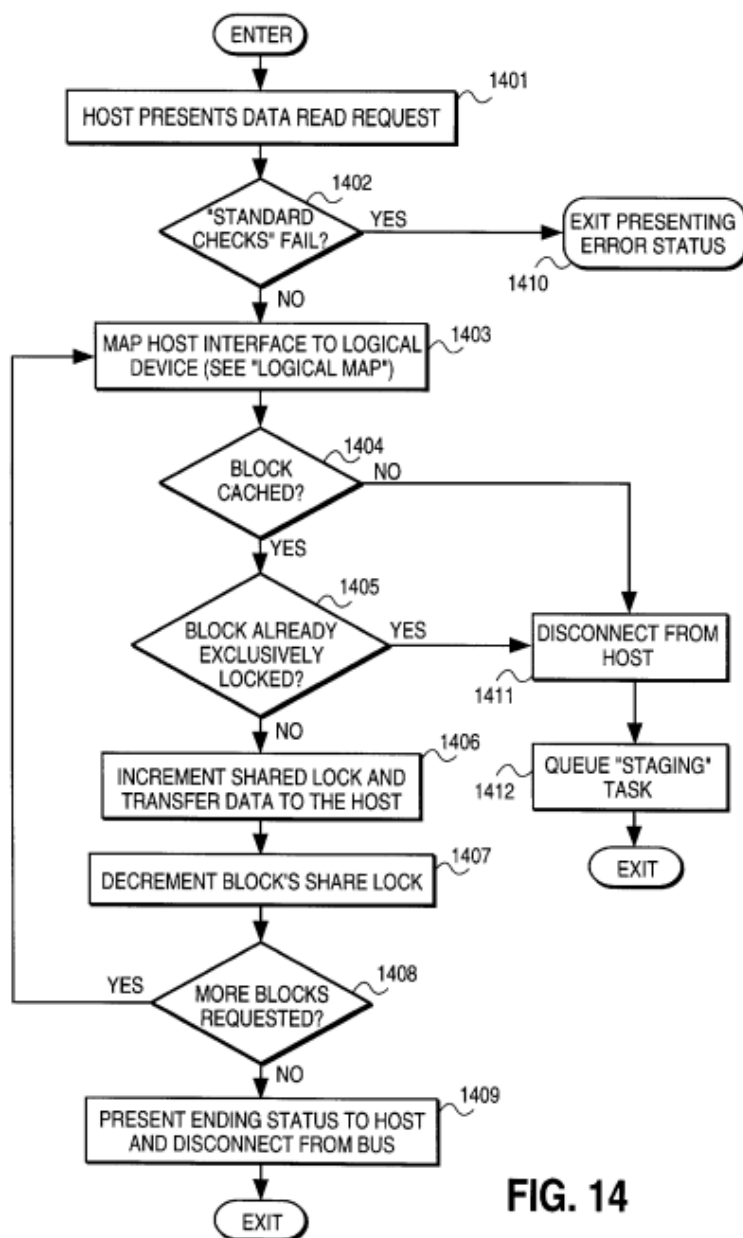
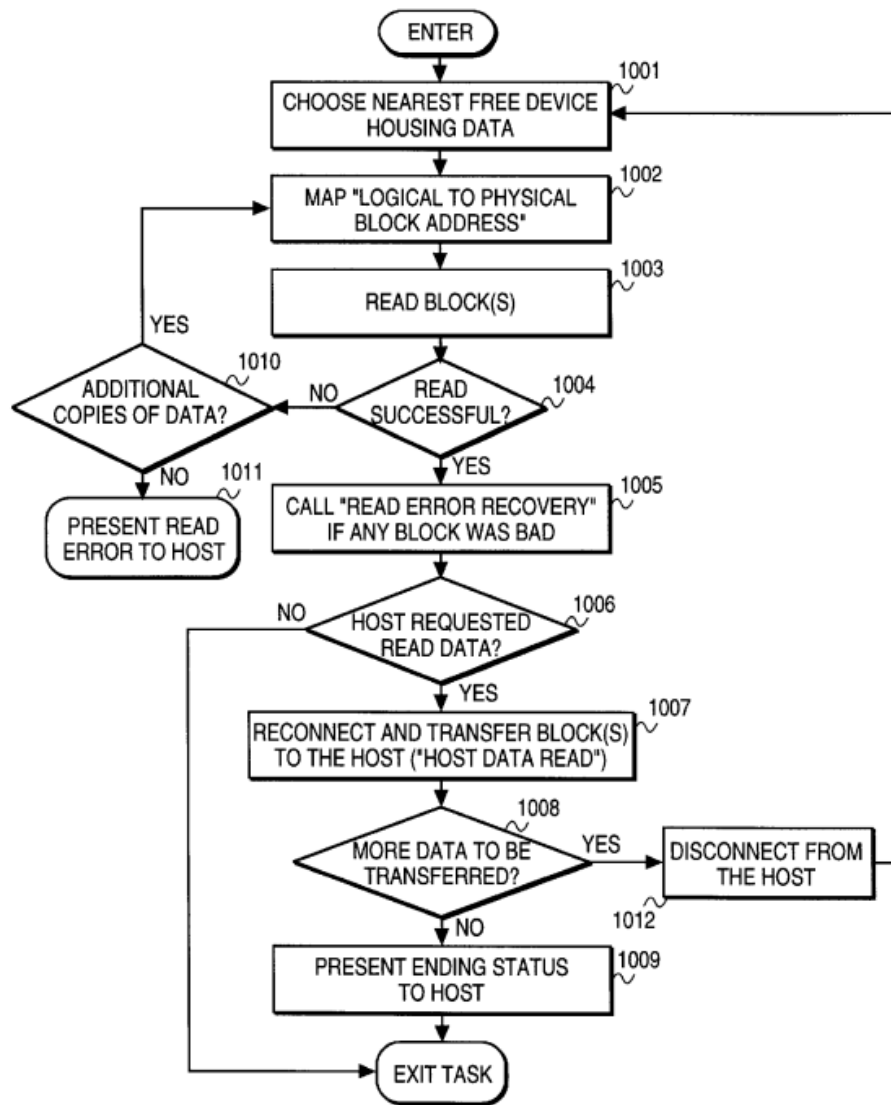


FIG. 14

EX1004, Bergsten Fig. 14

63. Bergsten further discloses routing the request to disk where, if a requested data block to be read is not cached at the storage controller, the storage controller initiates a staging routine. *Id.*, 12:47-53, Fig. 14 at 1404, 1411-1412. During the staging routine, the storage controller determines the nearest free storage drive housing the requested data – “nearest” refers the mass storage device having the smallest access time or earliest availability and may be a local MSD or a remote MSD connected to a remote storage controller – and routes the read request to that drive to read the requested block and transfer the read data back to the host. *Id.*, 10:30-45, 55-61. Fig. 10 depicting the staging routine performed by the storage controller is reproduced below. *Id.*, 2:29-31, Fig. 10.

**FIG. 10****EX1004, Bergsten Fig. 10**

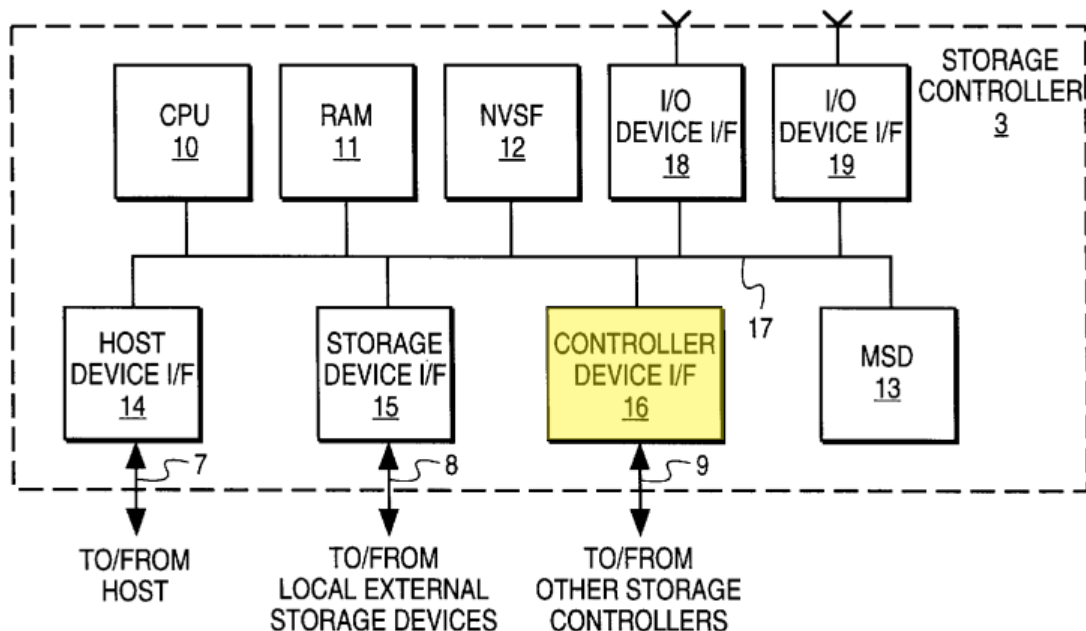
64. Finally, with regard to determining whether to reject a request, “[i]n response to each I/O request (read or write) from a host, the storage controller performs a series of standard checks before carrying out the operation.” *Id.*, 8:24-26. For example, when performing a standard read routine, “in step 1402 the standard checks routine (see Fig. 6) is performed. If any of the standards checks

fail, then the routine exits in step 1410, presenting an error status message to the host.” *Id.*, 12:40-43. Fig. 6 of Bergsten shows a list of the standard checks performed that may result in an error message presented to the host, including checking if the device is off-line, ready, or busy, or if the requested data block exists on the device, is reserved, or exceeds the device capacity. *Id.*, 8:24-61, Fig. 6. A POSITA would understand that, because the storage controller exits its routine and sends an error message to the host in response to any of the checks failing, this results in the storage controller rejecting the request by not completing the requested read or write operation.

[1.4] a network adapter for handling network control traffic;

65. Bergsten discloses a network adapter for handling network control traffic in the form of its “controller device interface 16,” which “connects communication path 9 to the bus 17 in order to connect a remote storage controller to storage controller 3.” *Id.*, 6:56-58. Controller device interface 16 may be an Ethernet, ATM, T1, T3, or FDDI adapter, or any other suitable device, depending upon the nature of the communication link 9.” *Id.*, 6:56-61. A POSITA would understand this disclosure to include the claimed network adapter for handling network control traffic because it enables communications between the networked storage controllers of Bergsten. The controller device interface 16 connecting the

storage controller “to/from other storage controllers” via communication path 9 is shown in Fig. 3, reproduced below. *Id.*, 2:13, Fig. 3.



EX1004, Bergsten Fig. 3 (highlighting added)

66. A POSITA would understand the claimed “network adapter” to be broad enough to encompass Bergsten’s disclosures about using “an Ethernet, ATM, T1, T3, or FDDI adapter” because the only structure recited by the ’988 patent related to this claim limitation is that the ’988 patent uses either an “Ethernet link” or fiber channel technology for network traffic. EX1001, 5:25-29.

[1.5] a cache memory;

67. Bergsten discloses a cache memory in the form of its local memory 24 (alternatively called cache 24 and internal memory 24). This local memory “may include any combination of RAM 11, NVSF 12, or MSD 13 (see FIG. 3)” and is

depicted as a functional block in Fig. 4 of Bergsten, reproduced below. EX1004, 7:20-28, Fig. 4.

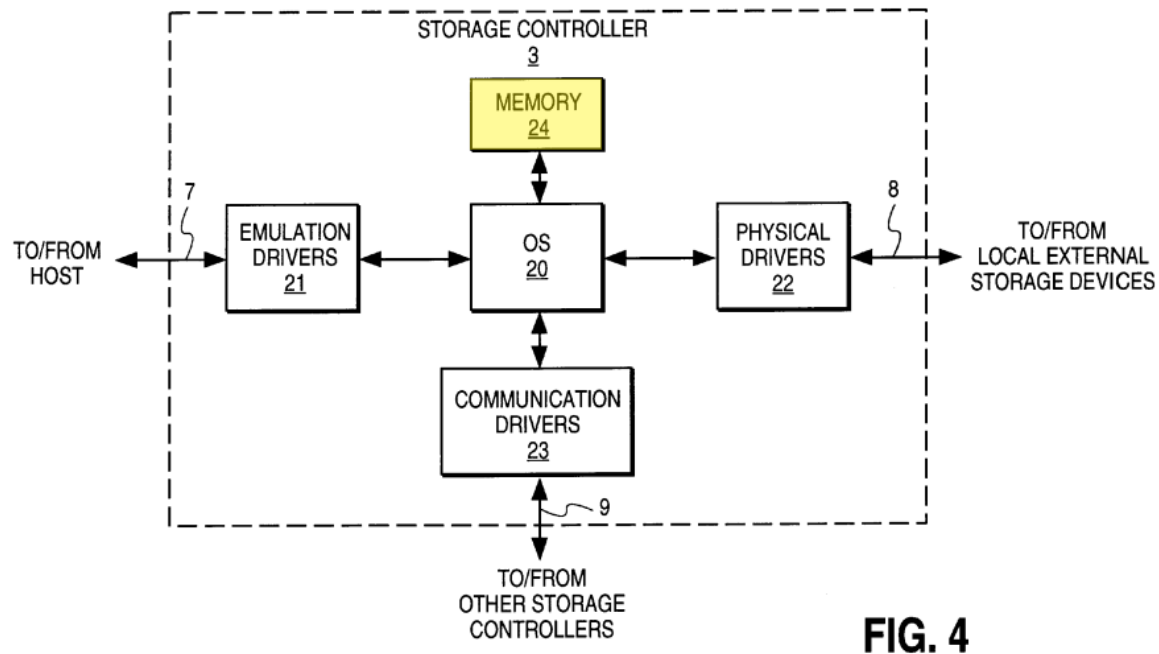


FIG. 4

EX1004, Bergsten Fig. 4 (highlighting added)

68. In performing various operations in response to a host request, such as a standard read or write operation, the storage controller connected to the host checks whether the requested block is cached by determining whether the block is stored in the storage controller's internal memory 24. *Id.*, 12:47-51, 13:20-23. In addition, "at various times the storage controller must read data from one or more MSDs into its internal memory ('the cache') or transfer data from the cache to one or more MSDs; these procedures are known as 'staging' and 'destaging', respectively. Staging is generally performed, for example, when executing a host read request. Destaging is generally performed when executing a host write

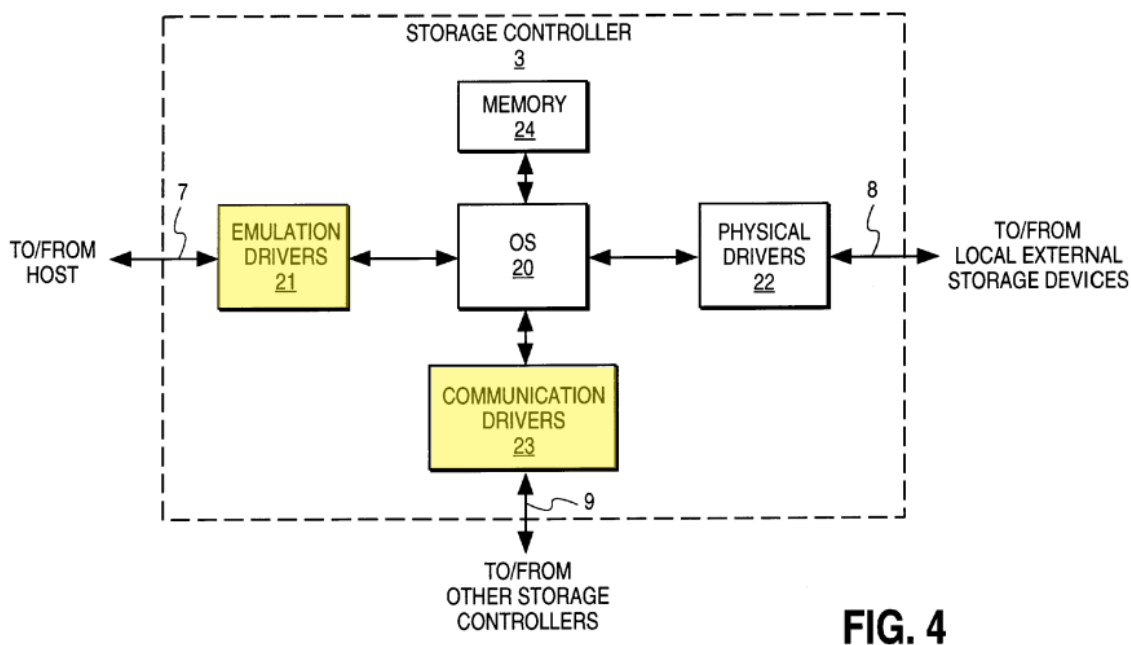
request.” *Id.*, 10:23-29. Accordingly, a POSITA would understand the internal memory of Bergsten’s storage controller to be a cache memory.

[1.6] front-end software for handling I/O requests arriving at the I/O channel adapter or the network adapter;

69. Bergsten discloses this limitation. A POSITA understood the term “front-end” to refer to the local, user-facing aspects of a networked computer system, e.g., the host processors of the ‘988 patent or Bergsten. Accordingly, a POSITA would recognize a storage controller’s emulation drivers 21 and communication drivers 23, in conjunction with OS 20, to constitute front-end software because, as explained below, they include software for handling requests from the system’s front-end components, namely a host directly connected to the storage controller or a host indirectly connected via a separate storage controller.

70. For handling I/O requests at its I/O channel adapter (i.e., host device I/F 14), Bergsten includes front-end software in the form of emulation drivers 21, which “are controlled by the OS 20 to provide communication of data between the storage controller 3 and the local host computer 2 via communication path 7. In particular, “[t]he emulation drivers 21 receive read and write requests from the host computer 2 and convert the read and write requests into a format recognized by the OS 20.” *Id.*, 7:35-38. For handling I/O requests arriving at its network adapter (i.e., controller device I/F 16), Bergsten’s “communication drivers 23 are controlled by

the OS 20 to provide communication of data between the storage controller 3 and remote storage controllers via communication path 9.” *Id.*, 7:40-43. Emulation drivers 21 handling I/O requests “to/from host” and communication drivers 23 handling I/O requests “to/from other storage controllers” are depicted as functional block in Fig. 4 of Bergsten, reproduced below. *Id.*, 7:20-28, Fig. 4.



EX1004, Bergsten Fig. 4 (highlighting added)

[1.7] cache manager software, responsive to said front-end software, for handling data stored in said cache memory; and

71. Bergsten discloses this limitation because its “OS 20 controls the overall operation of the storage controller, including emulation, data flow, *caching of data in the local memory 24*, control of data mirroring, and error recovery.” *Id.*, 7:29-35 (emphasis added). OS 20 of a storage controller therefore includes

software for managing the storage controller's cache, i.e., "cache manager software."

72. One example of Bergsten's cache manager software responsive to front-end software is the routine performed by the storage controller in "performing a standard read operation in response to a request by a host." *Id.*, 12:37-40. As part of this routine, the storage controller determines whether the requested block of data is stored in cache, and if not, initiates a staging task to retrieve the requested block from a mass storage drive. *Id.*, 12:47-58. If the block is stored in cache, "the data is transferred to the host from the cache." *Id.*, 12:56.

73. Bergsten's standard read operation is analogous to read operations described in the '988 patent. For example, Fig. 4B shows a flowchart for a cache manager of a storage system processing a read request for a volume that it holds in an exclusive mode (EX1001, 6:65-66, Fig. 4B), and Fig. 4D shows a similar flowchart for a cache manager of a storage system processing a read request for a volume it holds in a shared mode (*id.*, 7:46-47, Fig. 4D). As with Bergsten, both of the '988 patent's read routines performed by the software of the cache manager handle data stored in the cache memory by first checking whether the requested read data is stored in the cache memory (steps 420, 450), and either initiating a disk read in response to a cache miss (steps 425, 455) or transmit cached data to

the host in response to a cache hit (steps 424, 451). *Id.*, 6:65-7:8, 7:46-61, Figs. 4B, 4D.

74. Another example of Bergsten's cache manager is the routine performed by the storage controller in "performing a standard write operation in response to a request from a host." EX1004, 13:6-7. As part of this write routine the storage controller performs a cache check, determining "in step 1506 whether the block is already cached in the storage controller's internal memory 24." *Id.*, 13:20-23, Fig. 15. If the block is not in cache (i.e., a cache miss), "in step 1520 an empty block is allocated in the internal cache." *Id.*, 13:43-45, Fig. 15 and, subsequently at step 1508, the storage controller transfers write data from the host to the allocated block in the cache (*id.*, 13:51-53, Fig. 15). In case of a cache hit, the necessary block is already cached so no new cache block needs to be allocated, so the storage controller proceeds directly (after exclusively locking the block) to step 1508 to transfer data from the host to the appropriate block in cache. *Id.*, 13:23-30, Fig. 15.

75. As with read operations, the '988 patent's disclosure of the handling of data stored in cache memory for write operations is analogous to Bergsten. For example, Fig. 4A shows a flowchart for a cache manager of a storage system processing a write request for a volume that it holds in an exclusive mode (EX1001, 6:43-44, Fig. 4A), and Fig. 4C shows a similar flowchart for a cache

manager of a storage system processing a write request for a volume it holds in a shared mode (*id.*, 7:18-19, Fig. 4C). As with Bergsten, both of the '988 patent's write routines performed by the software of the cache manager handle data stored in the cache memory by first checking whether the requested buffer is stored in the cache memory (steps 411, 430), allocating a new buffer in the cache in response to a cache miss (steps 412, 431), and finally transferring data from the host to the appropriate cache buffer for both a cache hit or miss (steps 413, 432). *Id.*, 6:45-50, 7:19-25, Figs. 4A, 4C².

76. A POSITA would understand that because OS 20 controls the operations of caching of data in the storage controller, the OS would contain software instructions for performing the described read and write routines. A POSITA would also understand such software instructions to be responsive to the front-end software because the read and write routines are performed in response to a host read or write request and, as explained for limitation [1.6], such requests

² Fig. 4C of the '988 patent includes an apparent error indicating that after the "check cache" step 430, in case of a cache hit the flow bypasses step 432 and resumes at step 433. However, the specification explains that "[f]or a cache hit, the cache manager branches directly to step 432 where data is copied into the newly allocated buffer." EX1001, 7:23-25. Therefore, a POSITA would understand that the flow arrow corresponding to a cache hit in Fig. C should point to the start of "copy data" step 432 instead of step 433. A POSITA would also recognize this to be consistent with Fig. 4A which correctly has the flow arrow for a cache hit pointing toward the start of the "copy data" step.

are received by Bergsten's channel adapter and handled by the corresponding front end software.

[1.8] 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.

77. Bergsten discloses back-end software in the form of “physical drivers 22 [that] are controlled by the OS 20 to transfer data between the storage controller 3 and its local storage array. Such transfer includes the transformation of commands and control data from the format used by the storage controller 3 to the format recognized by the local storage array (e.g., SCSI).” *Id.*, 7:44-49. As can be seen in Bergsten's Fig. 4, reproduced below, the physical driver operations, which handle the reads and writes to local storage disks, are separate from the communication drivers and emulation drivers, that handle the network and I/O traffic, respectively. Because the physical drivers 22 communicating with local storage drives are separate from the emulation drivers 21 communicating with a host computer, a POSITA would understand that the back-end software of Bergsten does not communicate over Bergsten's I/O channel adapter (i.e., host device I/F 14) that receives I/O requests from the host.

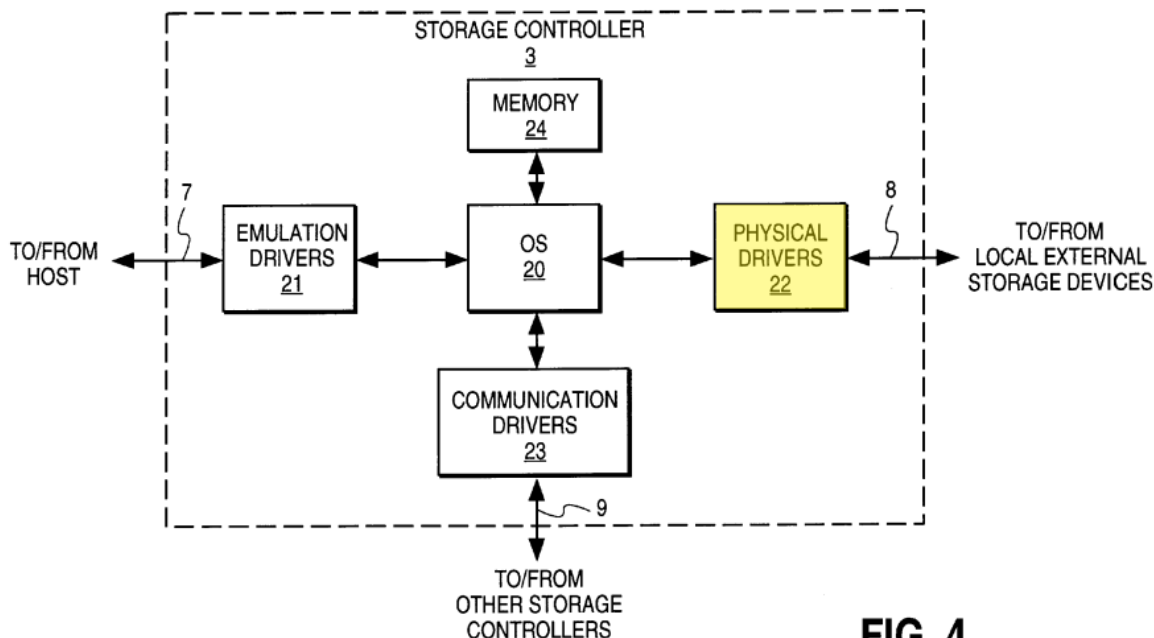


FIG. 4

EX1004, Bergsten Fig. 4 (highlighting added)

78. Thus, Bergsten discloses the claimed back-end software that handles the reads and writes to disks without communicating over the I/O channel adapter and a POSITA would understand that the disk operations of Bergsten are separate from network and I/O traffic.

C. Claim 3

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

79. A POSITA would understand that Bergsten discloses the added limitation of claim 3. A POSITA understood “distributed cache memory” to refer to cache memory allocated among multiple devices over a network. See EX1010, 3; EX1012, 128 (defining “distribute” in computing as “[t]o share out among

locations or facilities”). For example, the ’988 patent, describes multiple PC storage systems connected over a network with each system including its own cache memory. EX1001, 5:38-45, Fig. 3. Thus, each cache of each PC storage system constitutes a portion of a distributed cache memory across the network of storage systems. Bergsten teaches that each of its storage controllers includes a portion of distributed cache memory in the same way as described in the ’988 patent. As discussed above, Bergsten discloses multiple storage controllers interconnected by a network, with each storage controller including its own cache memory (local memory 24). EX1004, 3:20-39, 7:23-28, Fig. 1. Accordingly, each storage cache of each storage controller constitutes a portion of total cache memory distributed across the interconnected network of storage controllers. In addition, Bergsten recognizes the importance and “common practice” of storing back-up documents on separate back-up devices. Accordingly, a POSITA would understand that back-up copies of documents would be distributed across multiple storage caches belonging to multiple storage controllers in the system of Bergsten.

**X. CLAIMS 2, 4, AND 6 ARE OBVIOUS OVER
BERGSTEN IN VIEW OF THE KNOWLEDGE OF A POSITA**

80. As explained below, claims 2, 4, and 6 are obvious over Bergsten in view of the knowledge of one of ordinary skill in the art.

A. Claim 2

[2.1] The system of claim 1 wherein the computers comprise off-the-shelf hardware and operating systems and further comprise:

81. Bergsten discloses all limitations of claim 1 as discussed above for Ground 1.

82. A POSITA would understand that Bergsten can be implemented with off-the-shelf hardware and operating systems. Specifically, during prosecution the Applicant stated that “[i]t is well known to those skilled in the art that PCS refer to standard off-shelf computers that can be purchased at the retail store.” EX1002, 98. Further, Bergsten explains that its computer systems, storage controllers, interfaces, and storage devices are all conventional systems implementing standard computing protocols. EX1004, 3:50-53 (“Each of the host computer systems 2 may be any conventional computer system. For example, a host computer system may be a personal computer (PC), a mini-computer, or a mainframe.”), 4:18-25 (“In one embodiment, each storage controller communicates with its local host computer system and its local storage array using standard SCSI (Small Computer Systems Interface) protocols. Consequently, operation of a storage controller of the present invention in the manner described herein is not dependent upon the particular hardware or software configuration of any host computer or storage array, as long as those devices are SCSI-compatible.”), 4:29-37 (“Thus, because data paths 7 and

8 are conventional interfaces, a storage controller of the present invention can be used concurrently with host computers and MSDs having different configurations.”), 6:32-34 (“MSD 13 may be any conventional device that is suitable for non-volatile storage of large volumes of data, such as any of those discussed above”). Moreover, a POSITA would be motivated to implement the system of Bergsten using off-the-shelf hardware and operating systems because such systems would be widely available to a POSITA and a POSITA would recognize an off-the-shelf solution to be more economical and convenient than building the system of Bergsten *ad hoc* from the ground up. Nothing in Bergsten would indicate to a POSITA that it could not be implemented using off-the-shelf components and operating systems.

[2.2] an adapter I/O software for accepting incoming I/O requests from a host; and

83. Bergsten discloses this limitation for the reasons discussed above for limitation [1.6]. Specifically, Bergsten discloses emulation drivers 21 “controlled by the OS 20 to provide communication of data between the storage controller 3 and the local host computer 2 via communication path 7. In particular, the emulation drivers 21 receive read and write requests from the host computer 2 and convert the read and write requests into a format recognized by the OS 20.” *Id.*, 7:32-38.

[2.3] a volume access table employed by the configuration manager to ensure consistency of data stored on the network.

84. Dependent claim 2 introduces the '988 patent's "volume access table" feature. The '409 application does not expressly define "volume access table," instead it discloses that "[t]he flowchart uses a volume access table 450 (see also FIG. 5) and controlled by the configuration manager. ... Depending upon the status of the value in a volume access table 450 the [read/write] requests are routed" consistent with the particular value in the volume access table. EX1002, 7-8. The application also provides that "[t]he configuration manager uses a volume access table 450 that has been shown in Fig. 4." *Id.*, 8. Thus the '409 application discloses only that the volume access table is used by the configuration manager to determine an access mode for a read/write request.

85. The phrase "volume access table" was not a commonly known term in the art at the time of filing of the '409 application. A search of patent and publication references shows a single reference (EX1006) filed prior to the '409 application to use the phrase in its entirety. Thus, a POSITA would understand the phrase to mean what a POSITA would understand the individual terms to mean in the context of the '409 application's specification. The Microsoft Computer Dictionary provides reasonable definitions as follows (EX1012, 414, 8, 383):

- volume - “Another name for a disk or tape that stores computer data.
Each separate disk or tape is a volume, with a unique volume name or number. In some instances, a large hard disk can be divided into separate volumes, each of which is treated as a separate disk, even though all volumes reside physically in the same drive.”
- access – “To obtain entry to, or to locate, read into memory, and make ready for, some operation. Access is used with regard to disks, files, records and network entry procedures.”
- table – “In programming, a data structure usually consisting of a list of entries, each entry being identified by a unique key and containing a set of related values. A table is often implemented as an array of records, a linked list, or (in more primitive languages) several arrays of different data types, all using a common indexing scheme. See also array, list, record. In relational databases, a data structure characterized by rows and columns, with data occupying or potentially occupying each cell formed by a row column intersection. The table is the underlying structure of a relation.”

Thus, a reasonable interpretation of volume access table in the context of the '409 application is a table used to store access characteristics of storage volumes.

86. Bergsten recognizes the need to maintain consistency of multiple copies of data that may be located throughout a distributed storage system. EX1004, 5:61-62, 12:62-64, 14:42-43. A POSITA would be motivated to implement the storage controller of Bergsten, utilizing the conventional knowledge in the art, to include a volume access table to further ensure consistency of data stored on the network. Bergsten suggests such an approach in that it already discloses mapping volume access mode values between a host and volumes that the host has access to. Specifically, Bergsten teaches implementing “Exclusive Lock” and “Shared Lock” access modes to ensure data consistency as follows:

Associated with each block of data are the two single-bit binary quantities, Exclusive Lock and Shared Lock. Only a single device is permitted access to a given block of data for purposes of performing a write operation; this constraint is implemented using Exclusive Lock. A write on a block is performed by obtaining an Exclusive Lock on that block. For purposes of writing to a block, Exclusive Lock for that block can be set for only if neither Shared Lock nor Exclusive Lock is already set. When a write operation on a block is initiated, Exclusive Lock is set for that block, so that no other device may access that block during the write operation. In contrast, multiple devices are permitted to read from a given block of data at the same time. A read on a block is performed by obtaining a Shared Lock on that block. A device may obtain a Shared

Lock on a data block, regardless of the current status of Shared Lock, provided Exclusive Lock for that block is not already set. When a read is initiated on a given block, Shared Lock for that block is set, so that no device may write to that block during the read operation.

Id., 11:23-42. Thus, Bergsten discloses storing data structures in the form of “two single-bit binary quantities” (i.e., a binary numeral expressing a 0 or 1 (EX1012, 42, 44-45)) associated with each block of stored data, indicating whether that block is subject to either an Exclusive Lock or a Shared Lock. A POSITA understands a “block” in the context of disk storage to constitute a “collection of consecutive bytes of data that are read from or written to a device (such as a disk) as a group.” *Id.*, 48.

87. A POSITA would recognize that Bergsten’s disclosed Exclusive Lock and Shared Lock access modes are implemented to ensure consistency of data across a network. Specifically, the use of the Exclusive Lock mode ensures that only a single device may write to a given block at any one time, while the use of the Shared Lock allows for multiple devices to read a given block at the same time while also preventing devices from writing to that block while others are reading it.

88. While Bergsten discloses that its access modes are binary quantities associated with each data block, Bergsten does not explicitly disclose how the binary access modes are structured and stored in the system of Bergsten. Thus,

Bergsten leaves it to a POSITA to implement its access modes according to solutions known in the art. A POSITA would find it obvious to implement the Exclusive/Shared Lock access mode values of Bergsten using a table mapping each data block with the access mode value associated with that block, because a table is a common, long used, and well-understood data structure for associating one element with another (e.g., for associating a data block and access mode). A table is a basic data structure consisting of rows and columns creating cells and is particularly useful for storing related data. EX1012, 383. In computing, tables are one of the most fundamental data structures for storing data. For example, in programming an array is an implementation of table (*id.*, 26, 383), and at least one programming textbook used for university-level instruction (Kinariwala, “Programming in C”) teaches arrays as the very first type of derived data structure for storing base data (EX1013, 293). Another publication (Parsaye, “Intelligent Database Tools & Applications”) explains that the “basis of database technology consists of methods for structuring and organizing data” and that the relational database model “is presently the most widely used data model for databases.” EX1014, 22. The basis of this model, the relational database consists of tables containing rows and columns. *Id.* As known to a POSITA, and explained in Kinariwala, “[t]here is no best data structure for all tasks; suitable data structures must be selected for the specific task.” EX1013, 293. A POSITA would implement

a table as a particularly suitable data structure for storing the association of data blocks to the Exclusive/Shared Locks of Bergsten. One reason for the suitability of a table would be its simplicity and ease of implementation, consisting only of specified rows and columns holding the associated data. Another reason a POSITA would recognize such a table to be suitable is because it would allow for the fast and efficient search for relevant information via a table lookup routine. Given an index into the table, a lookup can be done in a few computer instructions. Thus, such a table would allow a storage controller receiving a request for a specific data block to easily access the table for the Exclusive/Shared Lock value associated with that block.

89. Providing the above-described association between stored data blocks and Exclusive/Shared Lock values constitutes a volume access table. A POSITA would understand that the data blocks stored in a mass storage system are stored in volumes, i.e., the physical disks comprising the mass storage. Therefore, by controlling how hosts can access specific stored blocks based on the Exclusive/Shared Lock value, the table holding such values also controls how the host can access the specific volume on which that block is stored. For example, where a set Exclusive/Shared Lock permits a host to perform a read or write operation for a specific data block, it likewise permits the host to perform the operation to the specific volume to which the data block is stored.

90. To the extent it is determined that a “volume access table” should be construed to require a mapping between hosts and volumes, a POSITA would also find it obvious to include in such a table what host or hosts hold a specific lock on a given block. A POSITA would do so in order for the system to know which devices are currently writing to or reading from a data block and also to know which devices are currently preventing other devices in the network from writing to or reading to a block. Similarly, if all blocks in a volume are to be treated uniformly, then it would be obvious to represent this information at the volume level for efficiency reasons. Finally, a POSITA would implement the storage controllers of Bergsten to utilize such a table because the storage controller’s “OS 20 controls the overall operation of the storage controller, including emulation, data flow, caching of data in the local memory 24, control of data mirroring, and error recovery,” (*id.*, 7:29-35) and would therefore control access by a host to a given block based on the access value for that block.

B. Claim 4

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

91. Bergsten discloses all limitations of claims 1 and 3 as discussed above for Ground 1.

92. Bergsten discloses the necessity of ensuring consistency of data stored in the distributed cache. EX1004, 5:61-62, 12:62-64, 14:42-43, Fig. 18. As explained above for claim 2 , it would have been obvious to a POSITA to implement the storage controllers of Bergsten to include a volume access table including Exclusive/Shared Lock access values to ensure consistency. Further, Bergsten discloses that data blocks may be located at either mass storage drives or in a storage controllers cache storage. A POSITA would recognize that Bergsten's use of exclusive and shared access modes would ensure data consistency by preventing writes to data blocks by a device while other devices are writing to or reading from that block and a POSITA would recognize that such access modes and resulting data consistency would be desirable regardless of whether the data block is stored in cache or in a mass storage drive.

C. Claim 6

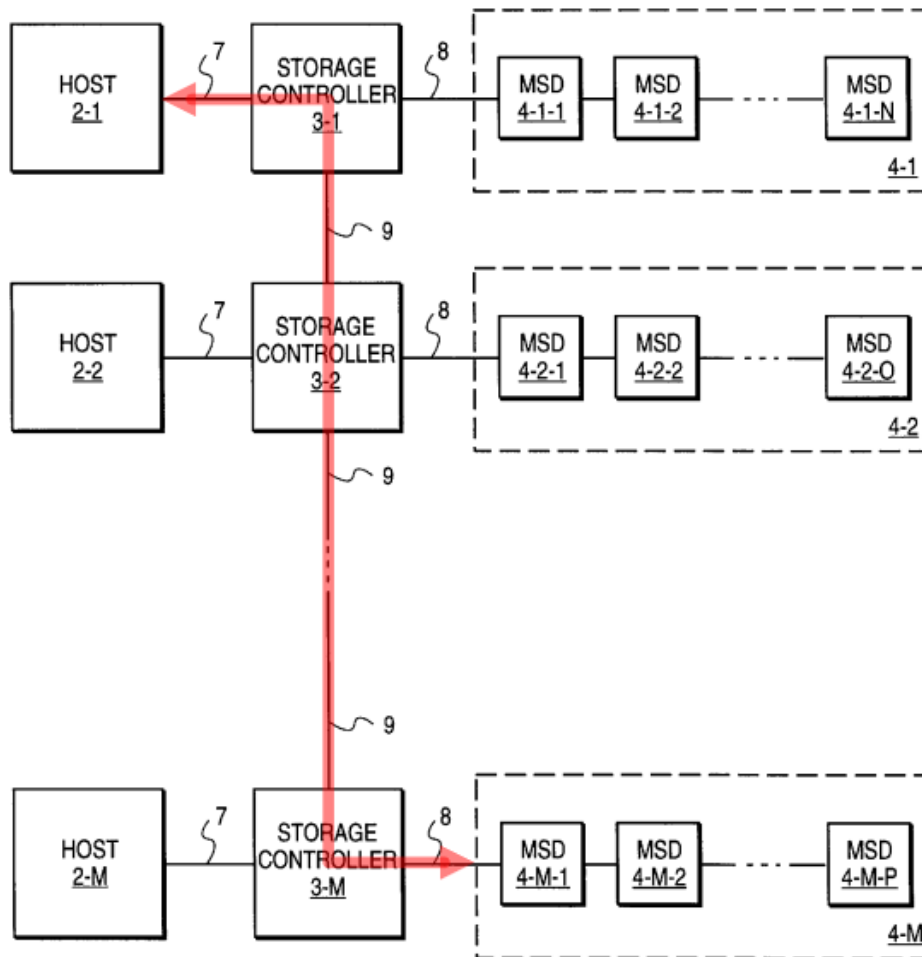
[6.1] A method of accessing a remote disk over a computer network without incurring network overhead, the method comprising the steps of:

93. Bergsten discloses this limitation by disclosing a computer network of interconnected storage controllers (plurality of computers), each coupled to a storage array, and each storage controller coupled to at least one host processing system and to at least one other storage controller to control access of the host

processing systems to the storage device. EX1004, Abstract; see also 3:20-39, Fig.

1.

94. This computer network allows a local host computer to access any connected remote disk as “[t]he storage controllers in the computing system cooperate to allow any of the host computer systems to access data located in any of the storage arrays.” *Id.*, 4:39-41. With reference to Fig. 1, Bergsten provides the example of host computer system 2-1 accessing data located in remote storage array 4-M “by communicating with remote storage controller 3-M via storage controllers 3-1, 3-2, etc.” *Id.*, 4:31-47. Such communication between the local computer system and remote storage array via intermediary storage controllers is depicted in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

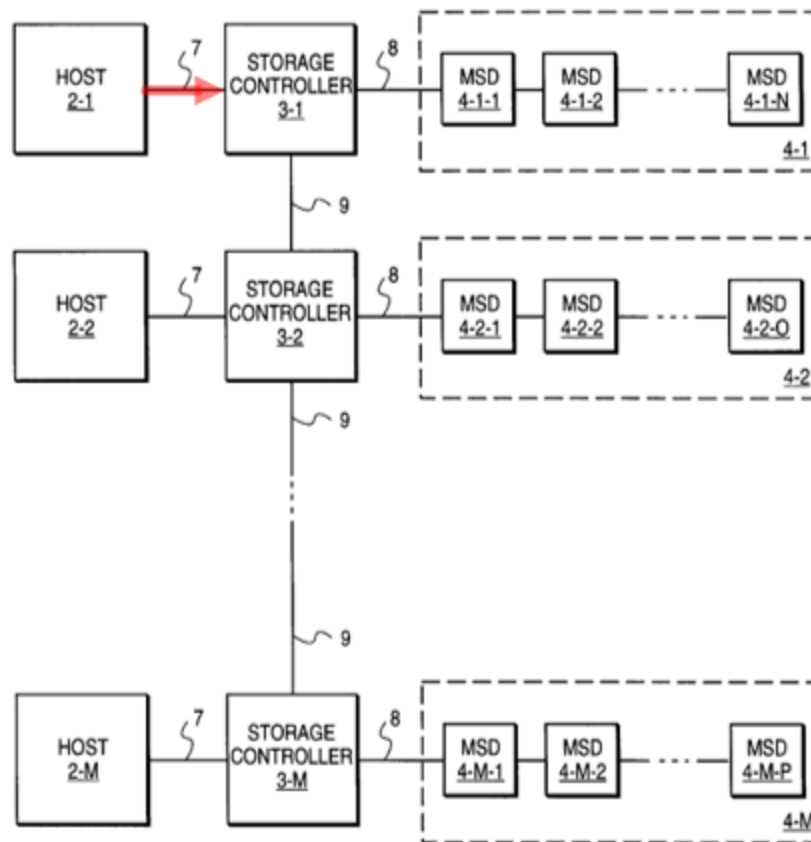
95. In addition, the local host computer of Bergsten accesses a remote disk without incurring network overhead in the same way as described in the '988 patent. With regard to its Fig. 1, the '988 patent describes a computer system interconnected over a network. The network includes, *inter alia*, a data storage system 131 including host processors 111, 112 using channels 121, 122. EX1001, 4:66-5:2. Data storage system 131 connects to network attached data storage system 132 using network link 152, with storage system 132 servicing hosts 157,

158. *Id.*, 5:2-6. Thereby, “[h]osts 157 and 158 access a data storage system 131 indirectly via network attached data storage system 132, thereby off loading communications protocol overhead from remote hosts 157, 158.” *Id.*, 5:7-10. On the other hand, separate hosts 106, 107 are directly connected to storage system 131 via network link 151. *Id.*, 5:10-14, Fig. 1. This direct access to storage system 131 “incurs communications protocol overhead on hosts 106, 107 and therefore decreasing performance of applications running on said hosts.” *Id.*

96. Bergsten performs remote access without incurring network overhead at the host in the same as the '988 patent. Specifically, as demonstrated in annotated Fig. 1 of Bergsten, reproduced above, a local host processor indirectly connects to a remote disk storage array 4-M via a series of storage controllers 3-1 through 3-M. EX1004, 4:39-50. As with the '988 patent, Bergsten does not incur network overhead for a host computer, because network communications protocols are handled by the intermediary storage controllers. As Bergsten explains, “[c]ommunication between individual storage controllers takes place on communication links 9 using a common protocol implemented by all storage controllers in the system.” *Id.*, 4:55-58. Such communications between a local storage controller and remote storage controllers are provided by communication driver 23 and controlled by OS 20 of the storage controller. Thus, processing of network communications in Bergsten are off-loaded to the storage controllers.

[6.2] causing a local host to issue a request over an I/O channel to a local computer;

97. Bergsten discloses a local host issuing a request over an I/O channel to a local computer. Specifically, in Bergsten local host 2-1 is directly coupled to and communicates over data communication path 7 with a local computer in the form of storage controller 3-1. EX1004, 4:10-15, Fig. 1. A POSITA would understand the storage controller to constitute a computer as it includes its own computer processing unit, memory, and interface devices. *Id.*, 6:22-28, Fig. 3. The local storage controller 3-1 also is directly coupled to a corresponding local data storage array 4-1. *Id.*, 4:15-17, Fig. 1. A POSITA would recognize data communication path 7 to constitute an I/O channel as it transmits I/O requests using the standard SCSI protocol. *Id.*, 6:41-55. The system of Bergsten causes the local host to issue I/O requests over the data communication path 7. *Id.*, 8:24-26 (“In response to each I/O request (read or write) from a host, the storage controller performs a series of standard checks before carrying out the operation.”) This operation of a local host sending a read request to its local storage controller is demonstrated in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

[6.3] providing a configuration manager on the local computer, the configuration manager routing the request to a remote computer via the computer network;

98. As discussed for claim limitation [6.2], Bergsten teaches a local computer in the form of storage controller 3-1. As discussed above, Bergsten discloses functional modules within the storage controllers, including controlling the flow and caching of data. EX1004, 7:29-35. Specifically, the storage controller of Bergsten includes a configuration manager as part of its operating system, OS 20, which “controls the overall operation of the storage controller, including

emulation, *data flow*, caching of data in the local memory 24, control of data mirroring, and error recovery.” *Id.*, 7:29-32 (emphasis added).) As part of controlling data flow, OS 20 controls communication drivers 23 “to provide communication of data between the storage controller 3 and remote storage controllers via communication path 9.” EX1004, 7:40-43.

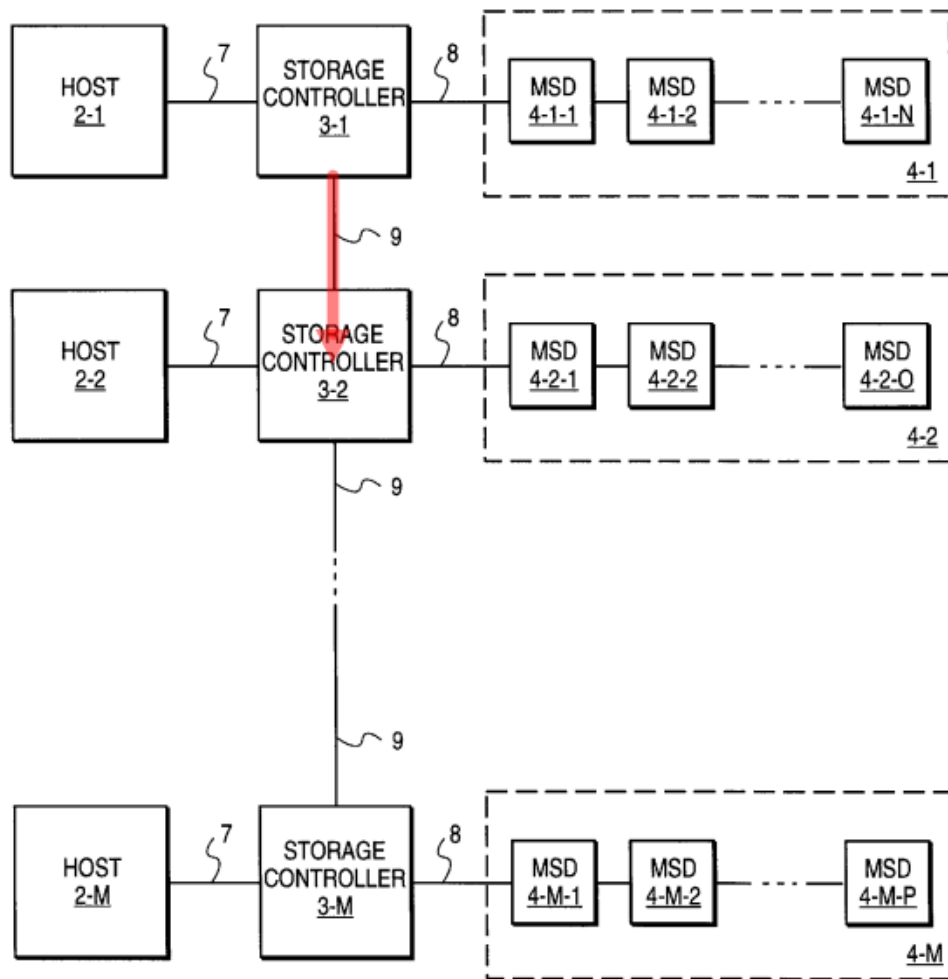
99. As Bergsten explains, a local storage controller will read data from or write data to a remote mass storage device connected to a remote storage controller according to a staging/destaging routine:

At various times, the storage controller must read data from one or more MSDs into its internal memory (“the cache”) or transfer data from the cache to one or more MSDs; these procedures are known as “staging” and “destaging”, respectively. Staging is generally performed, for example, when executing a host read request. Destaging is generally performed when executing a host write request.

Id., 10:23-29.

100. Fig. 10 of Bergsten demonstrates the staging procedure performed by a storage controller upon receiving a read request from its local host. *Id.*, 2:29-31, 10:30-31, Fig. 10. At the beginning of the destaging process, at step 1001, the storage controller determines the “nearest” free mass storage device storing the data to be read, where “nearest” refers to the device with the smallest overall

access time or earliest availability. *Id.*, 10:32-36. Bergsten explicitly recognizes that the nearest device holding the requested data may be “an MSD within a remote storage array.” *Id.*, 10:38-41. Next, at steps 1002 and 1003, the logical block address of the requested data is mapped to the physical block address holding that data in the identified storage device and the data is read from that device. *Id.*, 10:41-43. A POSITA would recognize that, in the case of a local storage controller reading requested data from a remote storage drive, the local storage controller would route the read request for the data through the remote storage controller associated with the storage drive, because, as shown in Fig. 1, the local storage controller (e.g., 3-1) connects with the remote storage array (e.g., 4-2) holding the remote drive indirectly via an intermediary remote storage controller (e.g., 3-2). *Id.*, Fig. 1. Thus, in Bergsten the configuration manager controlling the flow of data in a local storage controller routes a read request to a remote storage controller to read data from a remote storage device. This operation of the local storage controller routing a read request to a remote storage controller is demonstrated in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

[6.4] causing the remote computer to check the request against a volume access table;

101. For the reasons discussed above at ¶¶86-90, a POSITA would have been motivated to implement Bergsten with a volume access table to ensure consistency of data stored on the network, including causing the remote storage controller to check the request discussed above in ¶¶98-100. Specifically, in the

system of Bergsten, a read on a data block “is performed by obtaining a Shared Lock on that block” and may be obtained by any device “provided Exclusive Lock for that block is not already set.” EX1004, 11:35-40. Accordingly, a remote storage controller (e.g., storage controller 3-2) receiving a remote write request pursuant to Bergsten’s destaging procedure, as discussed above for claim limitation 6.4, would determine whether the requested data block in its connected storage array (e.g., storage array 4-2) has an Exclusive Lock. The storage array would do so by checking the corresponding Shared/Lock access value associated with the requested block. This procedure would ensure operation consistent with Bergsten’s prescription of allowing data reads only for blocks not exclusively locked by another device. *Id.*

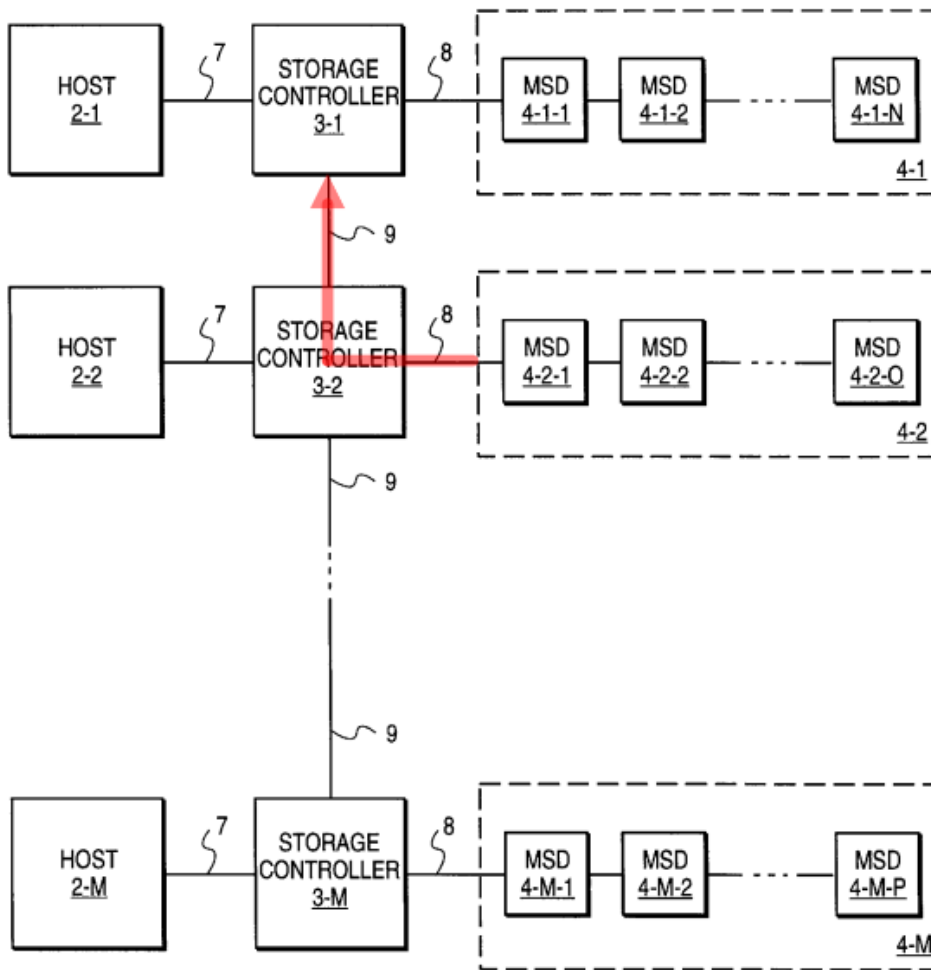
[6.5] causing the remote computer to perform an I/O operation on a disk located on the remote computer and to return data to the local computer;

102. Bergsten discloses this limitation because, as explained for limitation [6.3], in order to read data from a remote storage drive, a local storage controller routes a read request to the intermediary remote storage controller associated with the remote storage drive. As Bergsten explains, “[t]he storage controllers in the computing system cooperate to allow any of the host computer systems to access data located in any of the storage arrays. . . . For example, host computer system 2-1 may access data located in remote storage array 4-M by communicating with

remote storage controller 3-M via storage controllers 3-1, 3-2, etc.” *Id.*, 4:38-47.

Storage controllers perform I/O operations with their associated MSDs over communication path 8 using a standard communications protocol such as SCSI.

Id., 4:15-28. Accordingly, a POSITA would understand that in Bergsten in response to a remote read request (e.g., from host 2-1 via local storage controller 3-1), the remote storage controller (e.g., storage controller 3-2) would perform an I/O operation over communication path 8 to read the requested data from its associated mass storage array holding the data (e.g., 4-2). And the POSITA would further understand the remote storage controller to then transmit the read data back to the local storage controller over communication path 9 in order for the data to be routed to the local host requesting it. This operation of the remote storage controller retrieving read data from its mass storage array and transmitting that data to the storage controller requesting it is demonstrated in annotated Fig. 1, below.

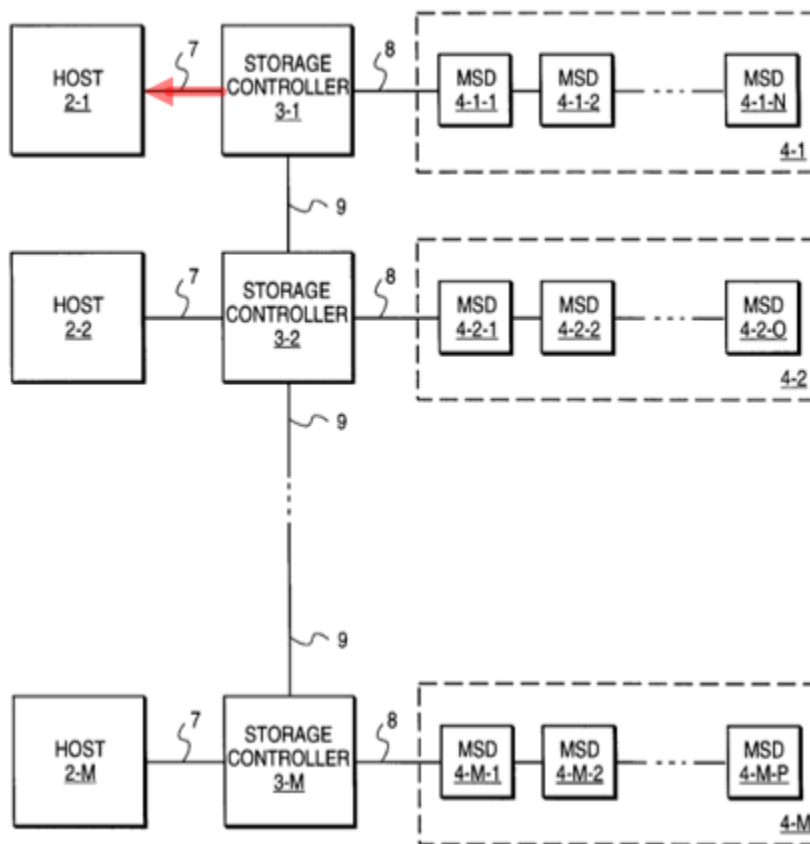


EX1004, Bergsten Fig. 1 (annotated)

[6.6] causing the local computer to provide the returned data to the local host via the I/O channel; and

103. Bergsten discloses this limitation because, after receiving the requested read data, at step 1007 of the destaging process “the storage controller reconnects to the host (i.e., acquires exclusive use of communication link 8, which may be a SCSI bus) and transfers the block or blocks to the host in step 1007.” *Id.*,

10:55-61, Fig. 10. The step of the storage controller transferring the requested data blocks to the requesting host is demonstrated in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

[6.7] 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.

104. For the same reasons as discussed above at ¶¶86-90, 101, a POSITA would have been further motivated to implement Bergsten to cause the local computer (i.e., the local storage controller) to check the data against the volume access table to ensure consistency with the remote computers (remote storage

controllers). Specifically, in the system of Bergsten a write operation is performed by obtaining an Exclusive Lock on the block of data to be accessed. *Id.*, 11:25-2. A POSITA would therefore recognize the need for a local storage controller, when receiving retrieved data, to check the volume access table for the data block associated with that data to determine that no host computer has obtained an Exclusive Lock on the associated data block because such an Exclusive Lock would indicate a potential write to the block and an editing of the data in that block. A check of the lock value in the volume access table for the data block associated with the retrieved data constitutes a check of that data against the volume access table, the presence of an Exclusive Lock would indicate a potential change to the block from which the data was retrieved. In such a case a POSITA would recognize a potential inconsistency between the data received at the local storage controller and the newly written data for that block and would prevent the data to be transferred to the requesting host in order to ensure data consistency. Conversely, where the local storage controller determines from its volume access table that no new Exclusive Lock has been attained the storage controller would be assured of consistency of the data across computers and transfer the received data to the requesting host.

**XI. CLAIMS 2, 4, AND 6 ARE OBVIOUS
OVER BERGSTEN IN VIEW OF BLUMENAU**

105. As explained below, claims 2, 4, and 6 are obvious over Bergsten in view of U.S. Patent No. 6,421,711 (“Blumenau”).

A. Blumenau

106. Blumenau, titled “Virtual Ports For Data Transferring Of A Data Storage System,” issued from U.S. Patent Application No. 09/106,299, filed on June 29, 1998. Blumenau’s 1998 application discloses those aspects of the issued Blumenau patent relied on herein. EX1007.

107. Blumenau is directed to a data storage system, including a storage controller that provides virtual ports for storage access and a virtual switch for routing storage access requests from the physical port to the virtual ports. EX1006, Abstract.

108. Like Bergsten, Blumenau discloses multiple host computers accessing multiple storage systems. More specifically, Blumenau discloses a cached storage subsystem that includes storage volumes and a storage controller for controlling access of the hosts to the storage volumes. The storage volumes are logical units of storage distributed over one more storage devices. *Id.*, 7:2-3. Blumenau further discloses “stor[ing] information defining a correspondence between hosts in the data processing system and the set of volumes accessible to each host through the port adapter. For example, a volume access table 80 and volume lists 81 are stored

in the memory 77. The volume access table specifies a correspondence between hosts and respective lists of volumes accessible to the hosts.” *Id.*, 12:66-13:6, *See also, e.g.*, Figs. 8-10, shown below.

Fig. 8

VOLUME GROUP NAME	HOST CONTROLLER WWN	PRIVATE /SHARED FLAG	VOLUME BITMAP	VOLUME LIST	PORT ADAPTER INDEX LIST
—	—	—	—	—	—
—	—	—	—	—	—
⋮	⋮	⋮	⋮	⋮	⋮

88

Fig. 9

HOST CONTROLLER S_ID	PRIVATE /SHARED FLAG	VOLUME BITMAP	PERSISTENT TABLE INDEX
—	—	—	—
—	—	—	—
⋮	⋮	⋮	⋮

80

Fig. 10

SID MODULUS	LISTS OF S_IDs AND TRANSIENT TABLE INDICES
0	(S_ID, INDEX) , (S_ID, INDEX) , ...
1	(S_ID, INDEX) , (S_ID, INDEX) , ...
⋮	⋮

97

EX1006, Blumenau Figs. 8, 9, 10

109. Referring to Fig. 8 of Blumenau, the volume access table 88 includes columns for volume group name, a host controller WWN, and a private/shared flag. *Id.*, 15:34-42, Fig. 8. The volume group name “provides a stable and unique identification number for a group of logical storage volumes to be accessed from a host port” (*id.*, 13:38-40), with the specific volume group names and the volumes in each volume list defined by a system administrator (*id.*, 13:58-59). Each host

computer of Blumenau includes a host controller (*id.*, Fig. 4) and each such controller includes a World Wide Name (“WWN”) (*id.*, 9:58-59), which is listed in the volume access table. Thereby, the volume access table of Blumenau provides a mapping of each host and the data volumes to which the host has access. With regard to the private/shared flag of the volume access table, Blumenau explains that in data storage, “[r]estricted access is desirable for security of private data. For example, private volumes should be assigned to each host for storage of private data, and other hosts should not be permitted to see or modify the private volumes of other hosts.” *Id.*, 1:48:-51. Therefore, when the table’s private/shared flag is set, this “indicates that all of the logical volumes in the volume list are private and no permission is needed from a lock manager before the host controller port can access[] any logical volume in its assigned volume list.” *Id.*, 13:53-57.

B. Rationale For Combining Bergsten And Blumenau

110. For at least the following reasons, Bergsten and Blumenau would have been a natural combination to a POSITA in January, 1999. Specifically, as explained in detail below, a POSITA would have been motivated to implement the distributed multi-host, multi-storage system as shown in Fig. 1 of Bergsten, and would have been motivated to implement the volume access table of Blumenau into the storage controllers of Bergsten.

111. First, Bergsten and Blumenau relate to the same technology. Bergsten discloses, among other things, an architecture very similar to the architecture of the '988 patent, consisting of multiple network-connected host processors, storage control systems (i.e., storage controllers), and mass storage disks (*see e.g.*, EX1001, Fig. 1; EX1004, Fig. 1). Blumenau similarly discloses hosts connected over a network to a storage controller and remote storage volumes (EX1006, Fig. 1) and also recognizes multi-host, multi-storage environments (*id.*, 41:34-39). Thus, the teachings in these references relate to the same technology.

112. Second, a POSITA reading Bergsten would understand the need to improve security. For example, Bergsten recognizes that “[a] storage controller of the present invention further allows data blocks to be write protected, so that a block cannot be modified from any host computer. Write protection may be desirable for purposes such as virus protection or implementation of security firewalls.” EX1004, 15:40-46. Blumenau provides techniques related to improving security in these types of storage systems. For example, Blumenau recognizes that “[i]t may be possible for any host to have access to any volume in a data storage subsystem to which the host has access. However, it may be desirable to restrict the set of volumes that can be seen by any one host. Restricted access is desirable for security of private data. For example, private volumes should be assigned to

each host for storage of private data, and other hosts should not be permitted to see or modify the private volumes of other hosts.” EX1006, 1:43-51.

113. Third, as noted above, each of Bergsten and Blumenau track information regarding host and how that information can be mapped to corresponding logical devices. Thus, a POSITA implementing Bergsten would be motivated to look to similar references such as Blumenau for ways to improve performance, including incorporating teachings related to security that Bergsten recognizes is a need. Blumenau describes the use of a table to manage access modes such as private access (EX1006, 15:34-42), which is a very efficient data structure (as described earlier), consistent with Bergsten’s goals of improving system performance (e.g., EX1004, 6:15-17).

114. Fourth, combining the teachings of Blumenau’s volume access table with Bergsten would be one way in which a POSITA would be able to improve security in Bergsten’s storage system. Specifically, as taught by Blumenau, using a volume access table with specified access modes would restrict access to data, which, as stated by Blumenau, “is desirable for security of private data.”

115. Finally, the techniques taught by Bergsten were entirely compatible with the volume access table taught by Blumenau, and a POSITA would have had a reasonable expectation of success to combine these teachings. Specifically, a POSITA would have combined the distributed storage network architecture as

depicted in Fig. 1 of Bergsten with the volume access table software solution taught by Blumenau. A POSITA would recognize that Bergsten's storage controller already include memory 24 (EX1004, 7:23-28) capable of storing the volume access table as well as an operating system (OS 20) that controls the operation of the storage controller (*id.*, 7:29-32) and that could easily be programmed by a POSITA to access the volume access table and control data flow based on data stored in the table. It would be well within the skill of a POSITA to program the storage controller of Bergsten to implement the volume access table of Blumenau. Bergsten already teaches that its storage controller performs checks of data stored in its internal memory such as an Exclusive/Shared Lock value in response to a read/write request from a host computer and, as explained herein, a table is a basic form of data structure known to and easily incorporated by a POSITA. A POSITA therefore would have a reasonable expectation of success in combining the software volume access table taught by Blumenau with the hardware distributed storage system taught by Bergsten.

C. Claim 2

[2.1] The system of claim 1 wherein the computers comprise off-the-shelf hardware and operating systems and further comprise:

116. Bergsten discloses all limitations of claim 1 as discussed above for claim 1. The combination of Bergsten with Blumenau does not affect the operation of Bergsten with respect to the limitations of claim 1.

117. A POSITA would understand that Bergsten can be implemented with off-the-shelf hardware and operating systems. Specifically, during prosecution the Applicant stated that “[i]t is well known to those skilled in the art that PCS refer to standard off-shelf computers that can be purchased at the retail store.” EX1002, 98. Further, Bergsten explains that its computer systems, storage controllers, interfaces, and storage devices are all conventional systems implementing standard computing protocols. EX1004, 3:50-53 (“Each of the host computer systems 2 may be any conventional computer system. For example, a host computer system may be a personal computer (PC), a mini-computer, or a mainframe.”), 4:18-25 (“In one embodiment, each storage controller communicates with its local host computer system and its local storage array using standard SCSI (Small Computer Systems Interface) protocols. Consequently, operation of a storage controller of the present invention in the manner described herein is not dependent upon the particular hardware or software configuration of any host computer or storage array, as long as those devices are SCSI-compatible.”), 4:29-37 (“Thus, because data paths 7 and 8 are conventional interfaces, a storage controller of the present invention can be used concurrently with host computers and MSDs having different

configurations.”), 6:32-34 (“MSD 13 may be any conventional device that is suitable for non-volatile storage of large volumes of data, such as any of those discussed above”). Moreover, a POSITA would be motivated to implement the system of Bergsten using off-the-shelf hardware and operating systems because such systems would be widely available to a POSITA and a POSITA would recognize an off-the-shelf solution to be more economical and convenient than building the system of Bergsten *ad hoc* from the ground up. Nothing in Bergsten would indicate to a POSITA that it could be implemented using off-the-shelf components and operating systems.

[2.2] an adapter I/O software for accepting incoming I/O requests from a host; and

118. Bergsten discloses this limitation for the reasons discussed above at ¶¶69-70 for limitation [1.6]. Specifically, Bergsten discloses emulation drivers 21 “controlled by the OS 20 to provide communication of data between the storage controller 3 and the local host computer 2 via communication path 7. In particular, the emulation drivers 21 receive read and write requests from the host computer 2 and convert the read and write requests into a format recognized by the OS 20.” *Id.*, 7:35-40.

[2.3] a volume access table employed by the configuration manager to ensure consistency of data stored on the network.

119. Notwithstanding the reasonable interpretation discussed above regarding volume access table, it should be noted that the applicant added a definition for “volume access table” during prosecution of the patent that was not included in the original specification as-filed:

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.

EX1002, 160/280.

120. Thus, I understand it is possible that the phrase could be interpreted more narrowly by the Board to mean a mapping between hosts and volumes specifying an access mode value. A POSITA would further understand that to be known in the prior art.

121. As discussed above, Blumenau expressly discloses a volume access table that includes access values for host computers and their corresponding volumes. EX1006, 13:4-6 (“The volume access table specifies a correspondence between hosts and respective lists of volumes accessible to the hosts.”); *see, also*, Figs. 5, 8-10.

↖ 82

VOLUME GROUP NAME	HOST CONTROLLER WWN	HOST CONTROLLER S_ID	PRIVATE /SHARED FLAG	VOLUME LIST POINTER
HOST 22-1	0276ADF1	0000B8	1	0472
HOST 23-2	0276ADC2	0000B9	0	0476
HOST 24-2	0276AE32	0001A7	0	048E
HOST 25-1	0276AE51	0001A6	1	04AO
⋮	⋮	⋮	⋮	⋮

Fig. 5

EX1006, Blumenau Fig. 5

Fig. 8

↖ 88

VOLUME GROUP NAME	HOST CONTROLLER WWN	PRIVATE /SHARED FLAG	VOLUME BITMAP	VOLUME LIST	PORT ADAPTER INDEX LIST
—	—	—	—	—	—
—	—	—	—	—	—
⋮	⋮	⋮	⋮	⋮	⋮

Fig. 9

↖ 80

HOST CONTROLLER S_ID	PRIVATE /SHARED FLAG	VOLUME BITMAP	PERSISTENT TABLE INDEX
—	—	—	—
—	—	—	—
⋮	⋮	⋮	⋮

Fig. 10

↖ 97

SID MODULUS	LISTS OF S_IDs AND TRANSIENT TABLE INDICES
0	(S_ID, INDEX) , (S_ID, INDEX) , ...
1	(S_ID, INDEX) , (S_ID, INDEX) , ...
⋮	⋮

122. Moreover, the volume access table of Blumenau includes a private/shared flag for an identified mapping of data volumes and host, and a POSITA would understand that such a flag is used to ensure data consistency of

data stored in a network. Blumenau states that “[c]ertain ‘private’ volumes should be assigned to each host and other hosts should not be permitted to see or modify the private volumes of other hosts.” *Id.*, 12:2-5. Thus, the system of Bergsten implementing Blumenau’s volume access table would utilize Bergsten’s storage controller (which controls system data flows) to check the stored private/shared flag for a volume sought to be accessed by a host read/write request to determine whether the host is permitted access to that volume. Therefore, setting a private/shared flag to indicate a private volume ensures the consistency of that data across the network by only allowing it to be modified by a single host.

123. For the reasons discussed above, a POSITA would have been motivated to incorporate Blumenau’s teachings of a volume access table into Bergsten to combine Bergsten’s disclosed distributed storage system architecture with Bergsten’s access information in a table structure such as Blumenau’s Fig. 8.

D. Claim 4

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

124. Bergsten discloses all limitations of claims 1 and 3 as discussed above.

125. Bergsten discloses the necessity of ensuring consistency of data stored in the distributed cache. EX1004, 5:61-62, 12:62-64, 14:42-43, Fig. 18. As

explained above for claim 2 at ¶¶110-115, 121-123 , it would have been obvious to a POSITA to combine the distributed storage system of Bergsten with the volume access table of Blumenau. Further, Bergsten discloses that data blocks may be located at either mass storage drives or in a storage controllers cache storage. A POSITA would recognize that the use of a private access mode as disclosed in Blumenau would ensure data consistency by ensuring only a single device can modify a data block flagged as private and a POSITA would recognize that such an access mode and resulting data consistency would be desirable regardless of whether the data block is stored in cache or in a mass storage drive.

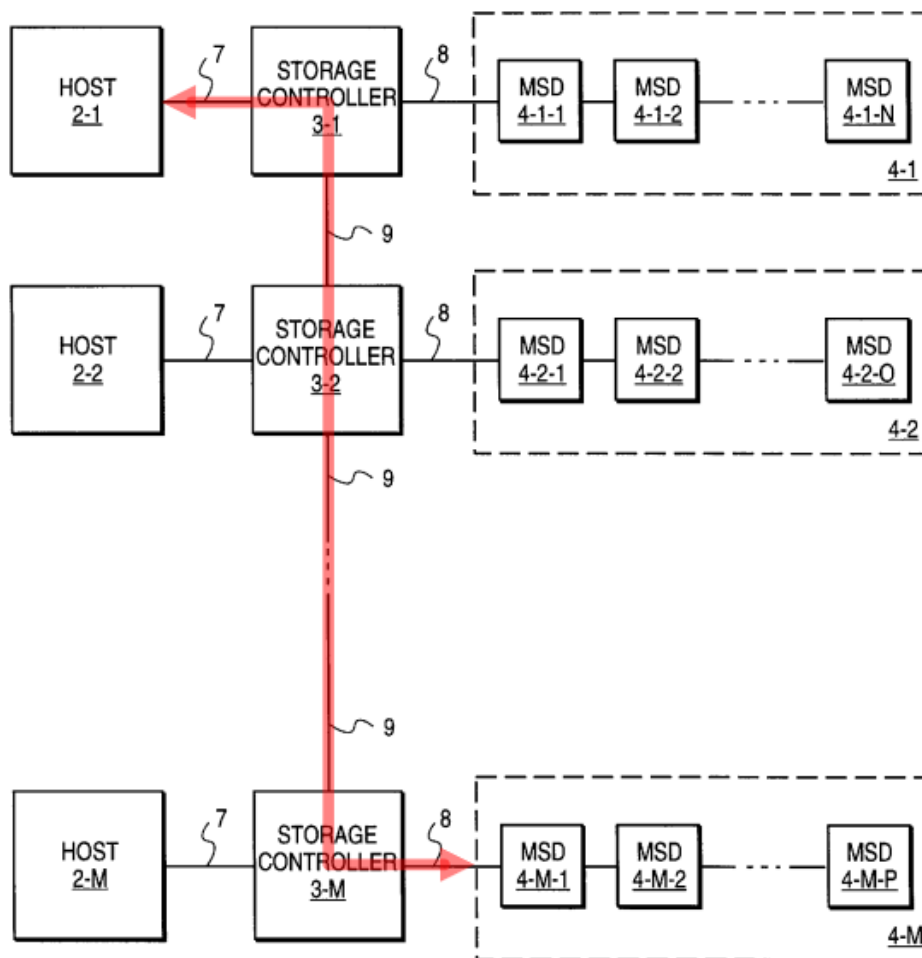
E. Claim 6

[6.1] A method of accessing a remote disk over a computer network without incurring network overhead, the method comprising the steps of:

126. To the extent the preamble is limiting, Bergsten discloses this limitation by disclosing a computer network of interconnected storage controllers (plurality of computers), each coupled to a storage array, and each storage controller coupled to at least one host processing system and to at least one other storage controller to control access of the host processing systems to the storage device. EX1004, Abstract; see also 3:20-39, Fig. 1.

127. This computer network allows a local host computer to access any connected remote disk as “[t]he storage controllers in the computing system

cooperate to allow any of the host computer systems to access data located in any of the storage arrays.” (*Id.*, 4:39-41.) With reference to Fig. 1, Bergsten provides the example of host computer system 2-1 accessing data located in remote storage array 4-M “by communicating with remote storage controller 3-M via storage controllers 3-1, 3-2, etc.” (*Id.*, 4:31-47.) Such communication between the local computer system and remote storage array via intermediary storage controllers is depicted in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

128. In addition, the local host computer of Bergsten accesses a remote disk without incurring network overhead in the exact same way as described in the '988 patent. With regard to its Fig. 1, the '988 patent describes a computer system interconnected over a network. The network includes, *inter alia*, a data storage system 131 including host processors 111, 112 using channels 121, 122. EX1001, 4:66-5:2. Data storage system 131 connects to network attached data storage system 132 using network link 152, with storage system 132 servicing hosts 157, 158. *Id.*, 5:2-6. Thereby, "[h]osts 157 and 158 access a data storage system 131 indirectly via network attached data storage system 132, thereby off loading communications protocol overhead from remote hosts 157, 158." *Id.*, 5:7-10. On the other hand, separate hosts 106, 107 are directly connected to storage system 131 via network link 151. *Id.*, 5:10-14, Fig. 1. This direct access to storage system 131 "incurs communications protocol overhead on hosts 106, 107 and therefore decreasing performance of applications running on said hosts." *Id.*

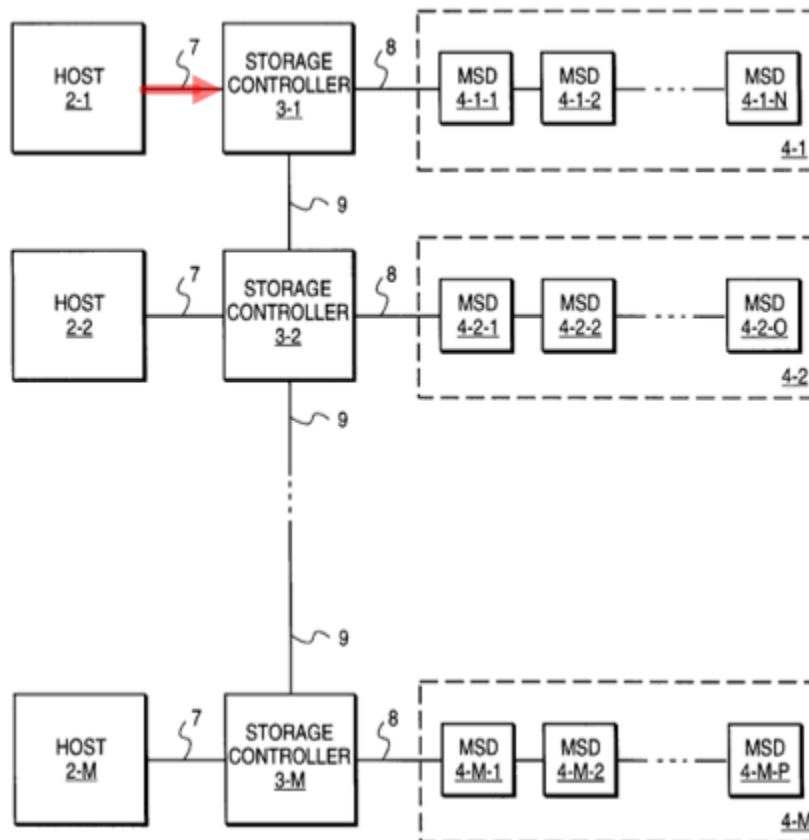
129. Bergsten performs remote access without incurring network overhead at the host in the same was the '988 patent. Specifically, as demonstrated in annotated Fig. 1 of the Bergsten, reproduced above, a local host processor indirectly connects to a remote disk storage array 4-M via a series of storage controllers 3-1 through 3-M. EX1004, 4:39-50. And as with the '988 patent, Bergsten does not incur network overhead for a host computer, because network

communications protocols are handled by the intermediary storage controllers. As Bergsten explains, “[c]ommunication between individual storage controllers takes place on communication links 9 using a common protocol implemented by all storage controllers in the system.” *Id.*, 4:55-58. Such communications between a local storage controller and remote storage controllers are provided by communication driver 23 and controlled by OS 20 of the storage controller. Thus, processing of network communications in Bergsten are off-loaded to the storage controllers.

[6.2] causing a local host to issue a request over an I/O channel to a local computer;

130. Bergsten discloses a local host issuing a request over an I/O channel to a local computer. Specifically, in Bergsten local host 2-1 is directly coupled to and communicates over data communication path 7 with a local computer in the form of storage controller 3-1. EX1004, 4:10-15, Fig. 1. A POSITA would understand the storage controller to constitute a computer as it includes its own computer processing unit, memory, and interface devices. *Id.*, 6:22-28, Fig. 3. The local storage controller 3-1 also is directly coupled to a corresponding local data storage array 4-1. *Id.*, 4:15-17, Fig. 1. A POSITA would recognize data communication path 7 to constitute an I/O channel as it transmits I/O requests using the standard SCSI protocol. *Id.*, 6:41-55. The system of Bergsten causes the

local host to issue I/O requests over the data communication path 7. *Id.*, 8:24-26 (“In response to each I/O request (read or write) from a host, the storage controller performs a series of standard checks before carrying out the operation.”) This operation of a local host sending a read request to its local storage controller is demonstrated in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

[6.3] providing a configuration manager on the local computer, the configuration manager routing the request to a remote computer via the computer network;

131. As discussed for claim limitation [6.2], Bergsten teaches a local computer in the form of storage controller 3-1. As discussed above, Bergsten discloses functional modules within the storage controllers, including controlling the flow and caching of data. EX1004, 7:29-35. Specifically, the storage controller of Bergsten includes a configuration manager as part of its operating system, OS 20, which “controls the overall operation of the storage controller, including emulation, *data flow*, caching of data in the local memory 24, control of data mirroring, and error recovery.” *Id.*, 7:29-32 (emphasis added).) As part of controlling data flow, OS 20 controls communication drivers 23 “to transfer data received from the local storage array to provide communication of data between the storage controller 3 and remote storage controllers via communication path 9.

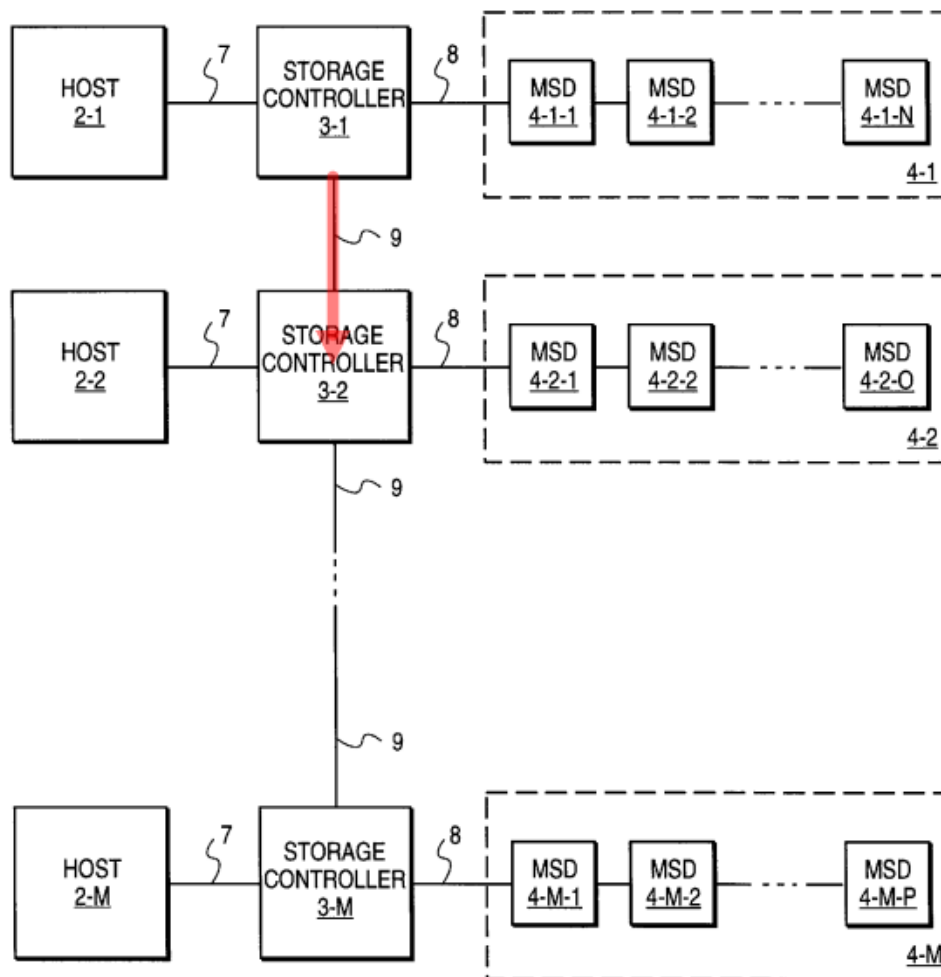
132. As Bergsten explains, a local storage controller will read data from or write data to a remote mass storage device connected to a remote storage controller according to a staging/destaging routine:

At various times, the storage controller must read data from one or more MSDs into its internal memory (“the cache”) or transfer data from the cache to one or more MSDs; these procedures are known as “staging” and “destaging”, respectively. Staging is generally performed, for example, when executing a host read request. Destaging is generally performed when executing a host write request.

Id., 10:23-29.

133. Fig. 10 of Bergsten demonstrates the staging procedure performed by a storage controller upon receiving a read request from its local host. *Id.*, 2:29-31, 10:30-31, Fig. 10. At the beginning of the destaging process, at step 1001, the storage controller determines the “nearest” free mass storage device storing the data to be read, where “nearest” refers to the device with the smallest overall access time or earliest availability. *Id.*, 10:32-36. Bergsten explicitly recognizes that the nearest device holding the requested data may be “an MSD within a remote storage array.” *Id.*, 10:38-41. Next, at steps 1002 and 1003, the logical block address of the requested data is mapped to the physical block address holding that data in the identified storage device and the data is read from that device. *Id.*, 10:41-43. A POSITA would recognize that, in the case of a local storage controller reading requested data from a remote storage drive, the local storage controller would route the read request for the data through the remote storage controller associated with the storage drive, because, as shown in Fig. 1, the local storage controller (e.g., 3-1) connects with the remote storage array (e.g., 4-2) holding the remote drive indirectly via an intermediary remote storage controller (e.g., 3-2). *Id.*, Fig. 1. Thus, in Bergsten the configuration manager controlling the flow of data in a local storage controller routes a read request to a remote storage controller to read data from a remote storage device. This operation

of the local storage controller routing a read request to a remote storage controller is demonstrated in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

[6.4] causing the remote computer to check the request against a volume access table;

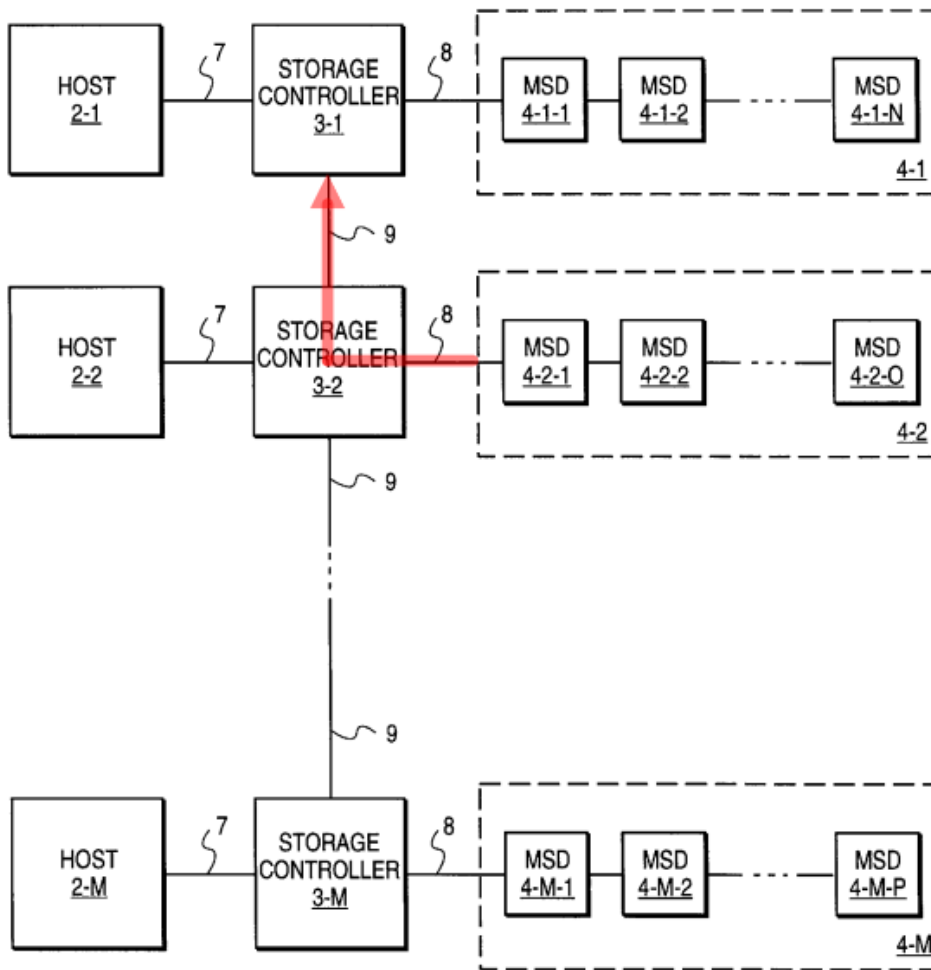
134. For the reasons discussed above at ¶¶86-90, a POSITA would have been motivated to implement Bergsten with a volume access table as taught by

Blumenau to ensure consistency of data stored on the network, including causing the remote storage controller to check the request discussed above in ¶¶131-33. Specifically, as explained above for claim element [2.3] (¶¶84-90), a POSITA would utilize Bergsten's storage controller, which controls system data flows (EX1004, 7:29-32), to check the stored private/shared flag (in the volume access table as taught by Blumenau) for a volume sought to be accessed by a host read/write request to determine whether the host is permitted access to that volume and thereby ensure consistency of data across the network. It would have been obvious for a POSITA to implement such a check of the volume access table to be performed at the remote storage controller in response to a read request for data stored on the storage array directly connected to that remote storage device. A POSITA would recognize a defined, limited number of options for implementation (e.g., the private/shared flag could be checked in response to the request at either the local storage controller 3-1 or at the remote storage controller 3-2) and would therefore find it an obvious implementation to perform the check at the remote storage controller (i.e, the remote computer).

[6.5] 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;
--

135. Bergsten discloses this limitation because, as explained for limitation [6.3], in order to read data from a remote storage drive, a local storage controller

routes a read request to the intermediary remote storage controller associated with the remote storage drive. As Bergsten explains, “[t]he storage controllers in the computing system cooperate to allow any of the host computer systems to access data located in any of the storage arrays. . . . For example, host computer system 2-1 may access data located in remote storage array 4-M by communicating with remote storage controller 3-M via storage controllers 3-1, 3-2, etc.” *Id.*, 4:38-47. Storage controllers perform I/O operations with their associated MSDs over communication path 8 using a standard communications protocol such as SCSI. *Id.*, 4:15-28. Accordingly, a POSITA would understand that in Bergsten in response to a remote read request (e.g., from host 2-1 via local storage controller 3-1), the remote storage controller (e.g., storage controller 3-2) would perform an I/O operation over communication path 8 to read the requested data from its associated mass storage array holding the data (e.g., 4-2). And the POSITA would further understand the remote storage controller to then transmit the read data back to the local storage controller over communication path 9 in order for the data to be routed to the local host requesting it. This operation of the remote storage controller retrieving read data from its mass storage array and transmitting that data to the storage controller requesting it is demonstrated in annotated Fig. 1, below.

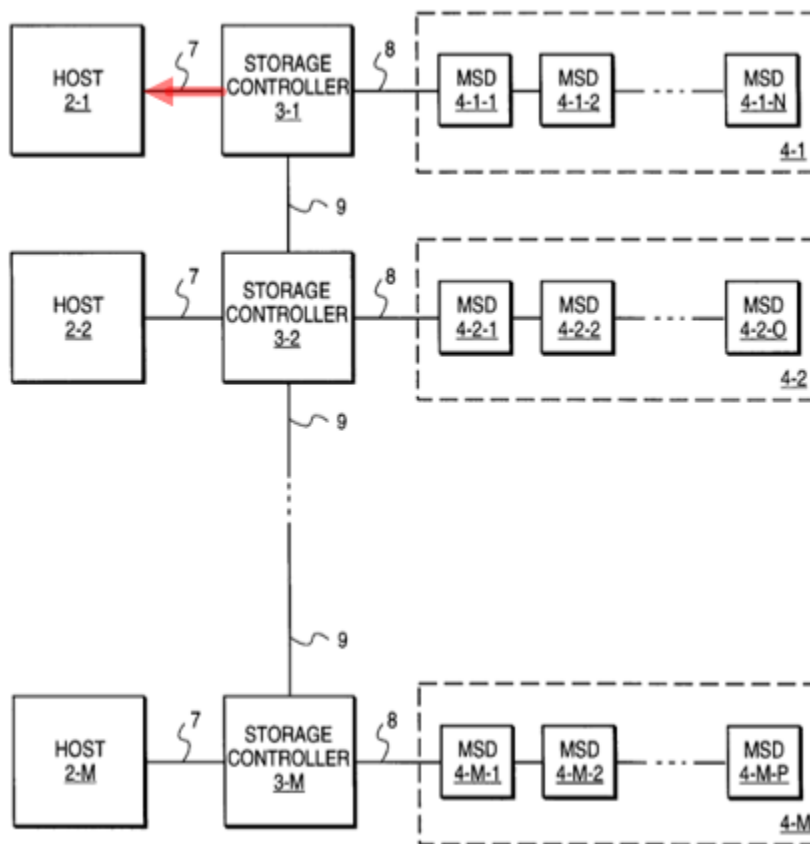


EX1004, Bergsten Fig. 1 (annotated)

[6.6] causing the local computer to provide the returned data to the local host via the I/O channel; and

136. Bergsten discloses this limitation because, after receiving the requested read data, at step 1007 of the destaging process “the storage controller reconnects to the host (i.e., acquires exclusive use of communication link 8, which may be a SCSI bus) and transfers the block or blocks to the host in step 1007.” *Id.*,

10:55-61, Fig. 10. The step of the storage controller transferring the requested data blocks to the requesting host is demonstrated in annotated Fig. 1, below.



EX1004, Bergsten Fig. 1 (annotated)

[6.7] 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.

137. For the same reasons as discussed above at ¶¶86-90, 134, a POSITA would have been further motivated to implement Bergsten to cause the local computer (i.e., the local storage controller) to check the data against the private/shared flag in the volume access table of Blumenau. And as explained

above at ¶¶86-90, 134, determining whether a host has private or shared access to a data volume ensures data consistency across the network, including between local and remote computers (storage controllers). Specifically, a POSITA would implement the local storage controller (e.g., storage controller 3-1) to check the private/shared flag for the volume associated with the retrieved data prior to transferring the data to the requesting host processor (e.g., host 2-1). By doing so, the system would ensure that the private/shared flag has not changed for the requesting host prior to transmitting the data. For example, if the flag has changed (e.g., via a system administrator) to indicate that the volume associated with the requested data has been set to be privately held by a different host than the requesting host, then the requesting host would no longer be permitted access to that data and the transmission of the data should not be completed. A POSITA would further understand that performing such a check of the volume access table at the local storage controller (before transferring data to the requesting host) and the remote storage controller (in response to a request, as explained for claim element 6.4) would ensure system redundancy and further promote system security.

XII. CLAIM 5 IS OBVIOUS OVER BERGSTEN IN VIEW OF SMITH

138. As discussed below, claim 5 is obvious over Bergsten in view of “Cache Memories,” Alan Jay Smith, Computing Surveys, Vol. 14, No. 3, Sept. 1982. EX1015, “Smith”.

A. Smith

139. Smith, cited on the face of the '988 patent, provides a summary of very aspects of cache memory design and operation. Smith discusses using cache memory in “modern computer systems” (circa 1982) to temporarily hold contents of main memory that are currently in use. EX1015. According to Smith, the advantage of cache memory is that less time is used to access information stored in cache memory than is stored in main memory. *Id.* Smith describes various common design techniques used in cache memory systems, including broadcasting invalidate messages to maintain cache consistency (*id.*, 505) and bypassing the cache for operations unlikely to result in the reuse of data (*id.*, 503).

140. Smith was published and available to the public by the end of 1982 at the latest. Smith was published in “Computing Surveys” a periodical journal published by the Association of Computing Machinery ("ACM"), was published in the September 1982 issue of the journal (Vol. 14, No. 3), and bears a 1982

copyright. EX1015, (i).³ Further, Smith and the Computing Surveys issue it was published in were identified by the inventor of the '988 patent, Ilya Gertner, as a known research paper in an Information Disclosure Statement filed with the original application for the patent. EX1002, 65-66. The Computing Surveys journal and the specific September 1982 issue have a registered copyright with an ISSN Number of 0010-4892 and Registration Number TX0001037896. EX1018.⁴ The copyright registration states that the September 1982 issue was created in 1982 and first published December 7, 1982. I am personally familiar with ACM, being a Member of the association and having my own research papers published in numerous ACM journals, symposia, and conference publications. ACM is “the world’s largest educational and scientific computing society,” and has as its mission “serving both professional and public interests **by fostering the open exchange of information.**” <https://www.acm.org/about-acm>; <https://www.acm.org/about-acm/mission-vision-values-goals> (emphasis added). As part of its mission of fostering the open exchange of information, ACM disseminates its publications as soon as the publications are prepared. See <https://www.acm.org/publications/about-publications> (“ACM’s emphasis on rapid

³ Smith was retrieved from the Association of Computing Machinery’s online publication archive, available at <https://dl.acm.org/toc/csur/1982/14/3>.

⁴ The copyright registration for Smith was retrieved from the United States Copyright Office’s online Public Catalog, and is available at https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=TX0001037896&Search_Code=REGS&PID=gznOs1JG05vPU66A4qziuOgfgZERI&SEQ=20230315124336&CNT=25&HIST=1.

publication ensures minimal delay in communication of exciting new ideas and discoveries.”). In light of the above evidence, I can conclude that Smith constitutes a printed publication as of no later than the end of 1982.

B. Rationale For Combining Smith With Bergsten

141. Smith is a background reference generally discussing cache memory. A POSITA would be very familiar with ACM Computing Surveys publications such as Smith, ACM has been publishing articles related to computing for over 50 years. As Bergsten discloses a system using cache memory, a POSITA would naturally use Smith as a reference paper for understanding the details of cache memory when implementing the Bergsten system. First, Smith specifically describes multicache systems. EX1015, 504-55. Smith describes “[l]arge modern computer systems...hav[ing] several independent processors, consisting sometimes of several CPUs” where each processor may have its own assigned cache and “with a main memory behind the cache.” *Id.*, 503-05. A POSITA would understand this description to be applicable to the system of Bergsten, which also constitutes a system of networked independent processors (hosts), with each individual host assigned its own cache (storage controller) and main memory (storage array). A POSITA would therefore understand the teachings of Smith to be applicable to and compatible with the networked multicache system of Bergsten. Second, Smith teaches various design techniques to improve the function

of cache memories and multicache systems. Therefore, applying these techniques to the multicache system of Bergsten constitute nothing more than use of known techniques to improve a similar device in the same way, as well as applying known techniques to a known device ready for improvement to yield predictable results. For example, one such technique taught by Smith is the broadcast of invalidate messages, and as explained in detail below Smith provides an express motivation for a POSITA to use the technique to ensure consistency of multiple copies of information stored across a distributed storage system. Another such technique taught by Smith is the bypass of the cache, and as explained below Smith provides an express motivation for implementing the technique in order to avoid displacing from the cache data that is likely to be reused in favor of non-reused data.

C. Claim 5

[5.1] The system of claim 4, wherein the configuration manager includes software that checks an access mode in the volume access table and
--

142. As explained above for claim 4 at ¶92, it would have been obvious for a POSITA to implement Bergsten to contain a volume access table storing Exclusive Lock and Shared Lock signals. The storage controller OS of Bergsten, in managing data caching and data flow between networked host computers and storage devices (EX1004, 7:29-31), checks the Exclusive Lock/Shared Lock access signals stored in this table to determine an appropriate access mode for a request

from a given host. For example, Bergsten's storage controller, in response to a read request (Fig. 14), at step 1405 checks the stored access signal associated with the requested data to determine whether Exclusive Lock is set. *Id.*, 12:47-51.

Similarly, in response to a write request (Fig. 15), the storage controller at step 1507 determines whether Exclusive Lock is possible by checking that neither Shared Lock nor Exclusive Lock are set. *Id.*, 13:23-27.

[5.2] (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;

143. Bergsten discloses that if the access signal associated with requested data is set to Exclusive lock, then both reads and writes are stored in cache memory. Specifically, Bergsten teaches both reads stored in cache memory and writes stored in cache memory. *Id.*, Fig. 14 at step 1404, Fig. 15 at 1506.

Regarding write operations, Bergsten teaches that “[o]nly a single device is permitted access to the given block of data.” *Id.*, 11:25-28. Therefore, in order to perform a write of requested data, the associated Exclusive Lock signal must be set “so that no other device may access that block during the write operation. *Id.*, 13:28-34. For read operations, by contrast, there is no need to allow only a single device to access at a given time and, therefore, “multiple devices are permitted to read from a given block of data at the same time” because there is no concern about data being overwritten while being accessed by other devices. *Id.*, 11:34-36.

In contrast, a POSITA would recognize that while a host holds a given volume in an exclusive access mode then that host would be allowed to perform both read and write operations as disclosed in Figs. 14 and 15 of Bergsten, because no other device would be able to overwrite data in the volume due to the Exclusive Lock held by the host.

144. Bergsten does not explicitly disclose causing invalidate messages to be sent to remote storage systems. However, invalidate messages were a well-known procedure associated with distributed storage systems and are explicitly taught by Smith. Smith specifically describes multicache systems. EX1015, 504-55. Smith describes “[l]arge modern computer systems hav[ing] several independent processors, consisting sometimes of several CPUs” where each processor may have its own assigned cache and “with a main memory behind the cache.” *Id.*, 504-05. A POSITA would understand this description to be applicable to the system of Bergsten, which also constitutes a system of networked independent processors (hosts), with each individual host assigned its own cache (storage controller) and main memory (storage array).

145. Bergsten distributes data throughout its system, as “the storage controllers automatically provide multiple back-up copies of stored data.” EX1004, 5:24-26. “For example, when a given data file is saved by a local host computer to its local storage array, the local storage controller causes one or more back-up

copies of the data file to be stored in one or more of the remote storage arrays, without any intervention from or interruption to the local host computer.” *Id.*, 5:27-33. And because copies of the same file may exist in multiple storage arrays, a POSITA would also understand that copies may also be present in the storage cache of each associated storage controller to facilitate efficient access to that data. Bergsten describes that the storage controller can cache (or stage) data by selecting the “nearest” free device housing the data of interest, where “nearest” refers to the mass storage device associated with the smallest overall access time or earliest availability. *Id.*, 10:30-35. A POSITA would understand that the “nearest” could change over time and thus different copies of the same data may be stored in different caches.

146. Smith describes a known problem with such distributed storage systems:

Unfortunately, in such a multiple processor system, a given piece of information may exist in several places at a given time, and it is important that all processors have access (as necessary) to the same, unique (at a given time) value.

EX1015, 505. That is, it is important that each copy of a file distributed across the system is the same, so that different processors accessing the same file are truly operating with the same data. It would be problematic if multiple processors

accessed separate file copies that they believed to be the same, but one file had been edited over overwritten while the other was not, so that the processors were actually operating on different sets of data.

147. Smith describes sending invalidate messages as a solution to this problem of potential inconsistencies among multiple file copies across a distributed storage system. Smith explains as follows:

2. *Broadcast Writes*. Every time a CPU performs a write to the cache, it also sends that write to all other caches in the system. If the target of the store is found in some other cache, it may be either updated or invalidated. Invalidation may be less likely to create inconsistencies, since updates can possibly "cross," such that CPU A updates its own cache and then B's cache. CPU B simultaneously updates its own cache and then A's. Updates also require more data transfer. The IBM 370/168 and 3033 processors use invalidation.

EX1015, 505. Thus, Smith describes a specific solution to maintain data consistency for multiple file copies across a distributed storage system in response to a write request. Bergsten already recognizes the need to maintain data consistency (EX1004, 5:61-62, 12:62-64, 14:42-43), and a POSITA would therefore be motivated to implement the known solution of invalidate messages as taught by Smith in order to further ensure distributed data consistency. Therefore, a

POSITA would modify Bergsten to cause invalidate messages to be sent in response to a write operation editing a stored file copy.

[5.3] (ii) if the access mode is set to shared, causes only reads to be stored in the cache memory; and

148. Bergsten teaches that when its Shared Lock access mode is set, only reads are stored in cache memory. As explained above for limitation [5.2], Fig. 14 of Bergsten, at step 1404, discloses reads stored in cache memory and write operations are only allowed for an Exclusive Lock mode. For read requests, however, “[w]hen a read is initiated on a given block, “[s]hared Lock for that block is set, so that no device may write to that block during the read operation.” *Id.*, 11:40-42. Therefore, a POSITA would understand that where the access value table contains a Shared Lock access signal, this would allow reads to be stored in the cache memory according to the read operation of Bergsten, but would not allow write operations.

[5.4] (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.

149. Smith further discloses the “idea of bypassing the cache for operations unlikely to result in the reuse of data. For example, long vector operations such as a very long move character (e.g., IBM MVCL) could bypass the cache entirely . . . and thereby avoid displacing other data more likely to be reused.” EX1015, 503. A

POSITA was aware of various types of operations, including read and write operations, that would be unlikely to result in the reuse of data and should therefore bypass the cache. Duke (U.S. Patent No. 4,500,954), for example, also recognizes that “it is not always in the best interest of total data processing performance and integrity to always use a caching buffer interposed between a using unit and a backing store.” EX1016, 2:19-22. Duke teaches that certain commands, including reads and writes, to a storage device “require access to the device irrespective of whether or not a cache is present. Such commands include read IPL, write home address, write RO and write COUNT KEY DATA.” *Id.*, 2:42-61. A POSITA would therefore understand that caching such requests would negatively affect system performance as caching them would provide no benefit while potentially displacing data in the cache that is likely to be reused. Similarly, Hanson (U.S. Patent No. 4,433,374) teaches bypassing cache memory for long data transfers because “statistically, it has been found that the data transferred during a long data transfer is not likely to be used again soon.” EX1017, 1:35-42.

Accordingly, a POSITA would have been motivated to modify the Bergsten to further incorporate the teaching of Smith to provide a cache bypass functionality for data that a POSITA would understand to be unlikely to be reused, and therefore prevent the displacement in the cache of more likely reused data. A POSITA would further find it obvious to implement such a solution by including an

additional access mode in the volume access table of Bergsten to indicate that the reads or writes of data associated with a given volume in the table should bypass the cache, i.e., they should be performed directly to disk without using the cache. This choice is advantageous given the performance objectives of Bergsten as only one signal needs to be read to determine that the data is not cacheable.

**XIII. CLAIM 5 IS OBVIOUS
OVER BERGSTEN IN VIEW OF BLUMENAU AND SMITH**

A. Rationale For Combining Smith With Bergsten And Blumenau

150. As explained above at ¶¶110-115, a POSITA would have been motivated to combine the volume access table taught by Blumenau with the distributed storage architecture disclosed by Bergsten. I explain above at ¶¶139-147 that a POSITA would have implemented two discrete cache management techniques—broadcasting invalidate messages for distributed cache consistency and bypassing the cache for data unlikely to be reused—into the distributed storage architecture of Bergsten. A POSITA would have been further motivated to combine the same teachings of Smith with the proposed combination of Bergsten and Blumenau for the same reasons as explained above at ¶141. The addition of Blumenau’s volume access table to the architecture of Bergsten would not dissuade a POSITA from implementing the additional teachings of Smith in order to attain their known benefits.

B. Claim 5

[5.1] The system of claim 4, wherein the configuration manager includes software that checks an access mode in the volume access table and

151. As explained above at ¶125, claim 4 is met by the combination of Bergsten and Blumenau, and it would have been obvious for a POSITA to implement Bergsten to contain a volume access table as taught by Blumenau.

152. Bergsten teaches implementing “Exclusive Lock” and “Shared Lock” access modes to ensure data consistency and that these modes are stored in the form of “two single-bit binary quantities” associated with each block of stored data. EX1004, 11:23-42. However, Bergsten does not explicitly disclose how the binary access modes are structured and stored in the system of Bergsten. Thus, Bergsten leaves it to a POSITA to implement its access modes according to solutions known in the art. A POSITA, having implemented the volume access table of Blumenau into the distributed storage system of Bergsten, would be motivated to store the Exclusive/Shared lock values in this volume access table. As explained above for claim 2 at ¶¶86-90, a POSITA would recognize a table as a suitable data structure for storing the mode values and a POSITA would implement the values in the already available volume access table of Blumenau rather than generate and store an entirely new data structure.

[5.2] (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;

153. Bergsten discloses that if the access signal associated with requested data is set to Exclusive lock, then both reads and writes are stored in cache memory. Specifically, Bergsten teaches both reads stored in cache memory and writes stored in cache memory. EX1004, Fig. 14 at step 1404, Fig. 15 at 1506. Regarding write operations, Bergsten teaches that “[o]nly a single device is permitted access to a given block of data.” *Id.*, 11:25-28. Therefore, in order to perform a write of requested data, the associated Exclusive Lock signal must be set “so that no other device may access that block during the write operation. *Id.*, 11:28-34. For read operations, by contrast, there is no need to allow only a single device to access at a given time and, therefore, “multiple devices are permitted to read from a given block of data at the same time” because there is no concern about data being overwritten while being accessed by other devices. *Id.*, 11:34-36. Accordingly, a POSITA would recognize that, in the combined volume access table of Blumenau and Bergsten, while a host holds a given volume in an exclusive access mode then that host should be allowed to perform both read and write operations as disclosed in Figs. 14 and 15 of Bergsten, because no other device would be able to overwrite data in the volume due to the Exclusive Lock held by the host.

154. Neither Bergsten nor Blumenau explicitly disclose causing invalidate messages to be sent to remote storage systems. However, invalidate messages were a well-known procedure associated with distributed storage systems and are explicitly taught by Smith. Smith specifically describes multicache systems. EX1015, 504-55. Smith describes “[l]arge modern computer systems hav[ing] several independent processors, consisting sometimes of several CPUs” where each processor may have its own assigned cache and “with a main memory behind the cache.” *Id.*, 504-05. A POSITA would understand this description to be applicable to the system of Bergsten, which also constitutes a system of networked independent processors (hosts), with each individual host assigned its own cache (storage controller) and main memory (storage array). Bergsten distributes data throughout its system, as “the storage controllers automatically provide multiple back-up copies of stored data.” EX1004, 5:24-26. “For example, when a given data file is saved by a local host computer to its local storage array, the local storage controller causes one or more back-up copies of the data file to be stored in one or more of the remote storage arrays, without any intervention from or interruption to the local host computer.” *Id.*, 5:27-33. And because copies of the same file may exist in multiple storage arrays, , a POSITA would also understand that copies may also be present in the storage cache of multiple storage controllers to facilitate data back-up and efficient access to that data. See EX1004, 1:34-37 (“[T]here is a need

for the capability to store multiple back-up copies of data in geographically separate locations, while still permitting quick and easy access by a host computer to any back-up copy.”) Smith describes a known problem with such distributed storage systems:

Unfortunately, in such a multiple processor system, a given piece of information may exist in several places at a given time, and it is important that all processors have access (as necessary) to the same, unique (at a given time) value.

EX1015, 505. That is, it is important that each copy of a file distributed across the system is the same, so that different processors accessing the same file are truly operating with the same data. It would be problematic if multiple processors accessed separate file copies that they believed to be the same, but one file had been edited over overwritten while the other was not, so that the processors were actually operating on different sets of data.

155. Smith describes sending invalidate messages as a solution to this problem of potential inconsistencies among multiple file copies across a distributed storage system. Smith explains as follows:

2. *Broadcast Writes.* Every time a CPU performs a write to the cache, it also sends that write to all other caches in the system. If the target of the store is found in some other cache, it may be either updated or invalidated. Invalidation

may be less likely to create inconsistencies, since updates can possibly "cross," such that CPU A updates its own cache and then B's cache. CPU B simultaneously updates its own cache and then A's. Updates also require more data transfer. The IBM 370/168 and 3033 processors use invalidation.

EX1015, 505. Thus, Smith describes a specific solution to maintain data consistency for multiple file copies across a distributed storage system in response to a write request. Bergsten already recognizes the need to maintain data consistency (EX1004, 5:61-62, 12:62-64, 14:42-43), and a POSITA would therefore be motivated to implement the known solution of invalidate messages as taught by Smith in order to further ensure distributed data consistency. Therefore, a POSITA would modify the combination of Bergsten and Blumenau to cause invalidate messages to be sent in response to a write operation editing a stored file copy.

[5.3] (ii) if the access mode is set to shared, causes only reads to be stored in the cache memory; and

156. Bergsten teaches that when its Shared Lock access mode is set, only reads are stored in cache memory. As explained above for limitation [5.2], Fig. 14 of Bergsten, at step 1404, discloses reads stored in cache memory and write operations are only allowed for an Exclusive Lock mode. For read requests, however, "[w]hen a read is initiated on a given block, "[s]hared Lock for that block

is set, so that no device may write to that block during the read operation.” *Id.*, 11:40-42. Therefore, a POSITA would understand that in the combination of Bergsten and Blumenau, where the access value table as taught by Blumenau contains a Shared Lock access value as taught by Bergsten, this would allow reads to be stored in the cache memory according to the read operation of Bergsten, but would not allow write operations.

[5.4] (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.

157. For the same reasons as explained above at ¶149, a POSITA would have been motivated to modify the combination of Bergsten and Blumenau to further incorporate the teaching of Smith to provide a cache bypass functionality for data that a POSITA would understand to be unlikely to be reused, and therefore prevent the displacement in the cache of more likely reused data. A POSITA would further find it obvious to implement such a solution by including an additional access mode in the volume access table of the proposed combination of Bergsten and Blumenau to indicate that the reads or writes of data associated with a given volume in the table should bypass the cache, i.e., they should be performed directly to disk without using the cache.

158. Smith further discloses the “idea of bypassing the cache for operations unlikely to result in the reuse of data. For example, long vector operations such as

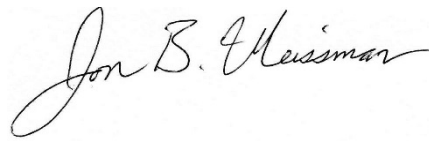
a very long move character (e.g., IBM MVCL) could bypass the cache entirely . . . and thereby avoid displacing other data more likely to be reused.” EX1015, 503. A POSITA was aware of various types of operations, including read and write operations, that would be unlikely to result in the reuse of data and should therefore bypass the cache. Duke (U.S. Patent No. 4,500,954), for example, also recognizes that “it is not always in the best interest of total data processing performance and integrity to always use a caching buffer interposed between a using unit and a backing store.” EX1016, 2:18-23. Duke teaches that certain commands, including reads and writes, to a storage device “require access to the device irrespective of whether or not a cache is present. Such commands include read IPL, write home address, write RO and write COUNT KEY DATA.” *Id.*, 2:43-62. A POSITA would therefore understand that caching such requests would negatively affect system performance as caching them would provide no benefit while potentially displacing data in the cache that is likely to be reused. Similarly, Hanson (U.S. Patent No. 4,433,374) teaches bypassing cache memory for long data transfers because “statistically, it has been found that the data transferred during a long data transfer is not likely to be used again soon.” EX1017, 1:40-56.

Accordingly, a POSITA would have been motivated to modify the combination of Bergsten and Blumenau to further incorporate the teaching of Smith to provide a cache bypass functionality for data that a POSITA would understand to be unlikely

to be reused, and therefore prevent the displacement in the cache of more likely reused data. A POSITA would further find it obvious to implement such a solution by including an additional access mode in the volume access table of the proposed combination of Bergsten and Blumenau to indicate that the reads or writes of data associated with a given volume in the table should bypass the cache, i.e., they should be performed directly to disk without using the cache. This choice is advantageous given the performance objectives of Bergsten as only one signal needs to be read to determine that the data is not cacheable.

I, Jon B. Weissman, do hereby declare and state, that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: March 23, 2023

A handwritten signature in black ink, reading "Jon B. Weissman". The signature is written in a cursive, flowing style. The "J" is large and loops around the "B". The "W" is also large and loops around the "e". The signature is positioned above a horizontal line.

Jon B. Weissman

APPENDIX A

JON B WEISSMAN

CURRICULUM VITAE

February 15, 2023

Work Address:

Department of Computer Science and Engineering
University of Minnesota
4-192 Keller Hall, 200 Union Street SE
Minneapolis, MN 55455-0159

Phone: (612) 626-0044
Fax: (612) 625-0572
Email: jon@cs.umn.edu
URL: <http://www.cs.umn.edu/~jon>

EDUCATION

Ph.D., Computer Science, University of Virginia, 1995.

M.S., Computer Science, University of Virginia, 1989.

B.S., Applied Mathematics and Computer Science, Carnegie-Mellon University, 1984.

PROFESSIONAL EXPERIENCE

Professor, Computer Science	Univ. of Minnesota (Summer 2012 to date)
Associate Professor, Computer Science	Univ. of Minnesota (Fall 2003 to Spring 2012)
Visiting Researcher and Distinguished Visitor	National e-Science Center, University of Edinburgh (2007-2008)
Assistant Professor, Computer Science	Univ. of Minnesota (1999-2003)
Assistant Professor, Computer Science	Univ. of Texas San Antonio (1995-1999)
Member of Technical Staff	Mitre Corporation, McLean Virginia (1989-1991)
Software Engineer	Software A&E, Arlington Virginia (1984-1987)

PROFESSIONAL ACTIVITIES

Tau Beta Engineering Honor Society, Member ACM, IEEE Senior Member

ACADEMIC AWARDS AND HONORS

- **George W. Taylor Distinguished Teaching Award** 2022.
- **Best paper nominee**, IEEE International Conference on Cloud Engineering (IC2E) 2015.
- **Best paper**, IEEE Grid conference, 2009.
- **Honorary Fellow**, College of Science and Engineering, University of Edinburgh, 2007-2008.
- **IEEE Senior Member**, promoted 2003.
- **Success Story**, ARL HPC (HPCMO), Virtual Data Grid Project, 2002.
- **CAREER Award**, National Science Foundation, 1996.
- **Supercomputing Award** for “High-Performance Computing with Legion”, SC, 1995.
- **Teaching Award**, University of Virginia Teaching Medal of Excellence, 1995.

LEGAL SUPPORT EXPERIENCE

Testifying Expert

- client: Fisch Sigler LLP, case: Juniper Networks Inc. v. Swarm technology LLC, area: parallel processing, project: IPR declarations and ex-parte reexaminations, *4 depositions*, represented: petitioner, status: finished, 2023.
- client: Quinn Emmanuel LLC, case: Juniper Networks Inc. v. WSOU INVESTMENTS, LLC D/B/A BRAZOS LICENSING AND DEVELOPMENT, area: networking, project: IPR, *deposition*, represented: petitioner, status: finished, 2023.
- client: Klarquist, case: LS CLOUD STORAGE TECHNOLOGIES, LLC, INC., vs. Microsoft, area: data management, project: IPR declaration, represented: defense, status: finished, 2023.
- client: Skadden, Arps, Slate, Meagher & Flom LLP, and Keystone, case: Carbonite, Securities Class Action, area: backup/restore, project: testifying expert and report, represented: defense, status: on-going.
- client: Quinn Emanuel Urquhart & Sullivan UK LLP, Keystone, case: IBM UNITED KINGDOM LIMITED vs. LZLABS GmbH, WINSOPIA LIMITED, LZLABS UK LIMITED, Intellectual Property and license dispute, area: IBM mainframe runtime, project: patent analysis, represented: claimant, status: on-going.
- client: Quinn Emanuel Urquhart & Sullivan LLP, case: WSOU INVESTMENTS, LLC d/b/a BRAZOS LICENSING AND DEVELOPMENT, vs. SALESFORCE.COM, INC., area: Internet services, project: testifying expert and report, represented: defense, status: on-going.
- client: Marton Ribera Schumann & Chang LLP, case: Zoho Corporation vs. BCS Software, LLC, area: file sharing, project: patent analysis, represented: defense, status: finished.
- client: Jones Day, case: Commvault v. Cohesity, area: data deduplication, project: patent analysis, represented: defense, status: finished.
- client: DLA Piper LLP, case: Intellectual Ventures Co et al. v. Hewlett-packard, area: archival file systems, project: Ex-Parte Reexamination declaration, represented: defense, status: finished, 2021.
- client: ROPES & GRAY LLP, case: PALO ALTO NETWORKS, INC. v. CENTRIPETAL NETWORKS, INC., area: networking, project: IPR declaration, represented: petitioner, status: finished, 2021.
- client: Quinn Emmanuel LLC, case: Juniper Networks Inc. v. WSOU INVESTMENTS, LLC D/B/A BRAZOS LICENSING AND DEVELOPMENT, area: networking, project: IPR declaration, *deposition*, represented: petitioner, status: finished, 2022.
- client: Unified Patents, LLC, area: virtualization, project: IPR declaration, represented: petitioner, status: finished, 2021.
- client: Klarquist, case: TERADATA US, INC., v. SAP SE, area: data management, represented: defense, status: finished, 2021.
- client: case: VEEAM SOFTWARE CORP., v. HYBIR INC., area: data backup, project: IPR declaration, *deposition*, represented: patent owner, status: finished, 2021.
- client: Farella Braun + Martel LLP, case: Syncloud Technologies LLC vs. Adobe, area: storage, Web: IPR declaration + litigation, represented: petitioner, status: finished, 2021.
- client: Lee Sullivan Shea & Smith LLP and Orrick, case: Sonos vs. Google Inc., area: network audio, project: validity expert, represented: plaintiff, *claim construction deposition*, **testifying ITC court**, status: finished, 2021.
- client: Marton Ribera Schumann & Chang LLP, case: Zoho Corporation vs. Sentius International, LLC, area: web databases, project: patent analysis, expert reports, represented: defense, **claim construction tutorial to district court**, status: stay, 2021.

- client: Kilpatrick Townsend & Stockton LLP, FACEBOOK, INC. vs. BCS SOFTWARE, LLC, area: IT monitoring, project: IPR declaration, represented: petitioner, status: finished, 2020.
- client: Fisch Sigler LLP, case: Packet Intelligence vs. Juniper Networks and Palo Alto Networks, area: networking, project: IPR declaration, represented: petitioner, status: finished, 2020.
- client: DLA Piper LLP, case: TMI Solutions, LLC vs. CVS, Officemax, Home Depot Product Authority LLC, area: security, network services, project: IPR declaration, represented: petitioner, status: finished, 2020.
- client: Reed Smith LLP, case: Uniloc vs. Paychex, area: distributed applications, project: prior art analysis, represented: defense, status: finished, 2020.
- client: Orrick, case: SecureWave vs. Micron Technology, Inc., area: security, project: claim construction, represented defense, status: finished, 2019.
- client: Fenwick & West LLP, case: PersonalWeb vs. Amazon, area: cloud, data storage, project: patent analysis, expert reports, represented: defense, status: settled, 2019.
- client: Jenner & Block LLP, case: Kove IO, Inc., vs. Amazon Web Service, area: cloud, data storage, project: patent analysis, IPR declaration, represented: defense, status: finished.
- client Cooley LLP, case: Zak Inc vs. Facebook, area: web, project: patent analysis, expert reports, represented: defense, status: active.
- client Cooley LLP, case: Blackberry Inc vs. Facebook, area: mobile computing, project: patent analysis, expert reports, *deposition*, represented: defense, status: active.
- client: Faegre Baker Daniels LLP, case: ICN Acquisition, LLC vs. ipDataTel, LLC, area: home alarm systems and networks, IPR declarations, represented: petitioner, *deposition*, status: settled prior to trial, 2019.
- client: Finnegan, Henderson, Farabow, Garrett & Dunner, LLP, case: KOM Software Inc. vs. NetApp, Inc. and HP, area: Distributed file systems, 5 IPR declaration, represented: petitioner, status: finished, 2019.
- client: Cooley LLP, case: Vaporstream Inc vs. Snapchat Inc, area: mobile computing, project: patent analysis, expert reports, *deposition*, represented: defense, status: stay, 2018.
- client: Fenwick & West LLP, case: IBM vs. Groupon, area: security, web, client-server, project: patent analysis, expert reports, *deposition*, **testifying district court**, represented: defense, status: finished, 2017-2018.
- client: DLA Piper LLP, Workspot, Inc. vs. Citrix Systems, INC., area: security, IPR declaration, *deposition*, represented: petitioner, status: active.
- client: Kecker, Van Nest & Peters LLP, BMC Software, INC., vs. Cherwell Software, LLC, area: software systems, IPR declaration, represented: petitioner, status: active.
- client: Cooley LLP, case: Sound View Inc vs. Facebook, area: databases, web, project: patent analysis, expert reports, *deposition*, represented: defense, status: settled, 2018.
- client: Fenwick & West LLP, case: IBM vs. Groupon, area: security, IPR declaration, represented: patent owner, status: finished, 2017.
- client: Knobbe Martens, case: Ancora vs. HTC, area: security, CBM declaration, *deposition on claim construction 2019*, represented: petitioner, status: finished, 2017.
- client: Fisch Sigler LLP, case: Broadcom vs. Amazon, area: multimedia, IPR declarations, represented: petitioner, status: finished, 2017. No associated litigation.
- client: Finnegan, Henderson, Farabow, Garrett & Dunner, LLP, area: Internet applications, IPR declaration, represented: petitioner, status: finished, 2017. No associated litigation.
- client: McDonnell Boehnen Hulbert & Berghoff LLP, case: Trading Technologies Inc., area: distributed systems, project: Patent Examination, declaration, represented: patent owner, status: finished, **examiner interview**, 2016.

- client: Fenwick & West LLP, case: Actifio vs. Delphix, area: cloud storage, project: patent analysis, expert reports, represented: defense, status: finished (settled prior to trial), 2016.
- client: Fisch Sigler LLP, case: Kaavo vs. Amazon and Tier3, area: cloud computing, IPR declarations, *deposition*, represented: petitioner, status: finished, 2016.
- client: Morrison and Foerster LLP, case: Ancora vs. Apple Inc, area: security, CBM declaration, represented: petitioner, status: finished (settled prior to trial), 2016.
- client: Finnegan, Henderson, Farabow, Garrett & Dunner, LLP, area: mobile computer systems, 3 IPR declarations, represented: petitioner, status: finished, 2016. No associated litigation.
- client: Cooley LLP, case: BMC Inc vs. ServiceNow Inc, area: computer systems, project: patent analysis, expert reports, *deposition*, represented: defense, status: finished (settled prior to trial), 2016.
- client: Fish & Richardson P.C., case: Ericsson Inc vs. Apple Inc, area: mobile systems, project: patent analysis, represented: defense, status: finished, 2015.
- client: Morrison and Foerster LLP, case: Good Technology Inc vs. Airwatch Inc, area: mobile systems, project: patent analysis, 3 expert reports, *deposition*, represented: defense, status: finished (settled prior to trial), 2015.
- client: Troutman Sanders, Intellectual Ventures I LLC vs. Capitol One Financial Corp, claim construction declarations, area: mobile interfaces, represented: defense, status: finished, 2015.
- client: Bryan Cave LLP, case: Symantec vs. Veeam Software Corporation, area: storage systems, IPR declaration, *deposition*, represented: patent owner, status: finished, 2014.
- client: Finnegan, Henderson, Farabow, Garrett & Dunner, LLP, case: Clouding IP vs. Rackspace, area: storage systems, project: patent analysis, IPR and CBM declarations, *two depositions*, represented: petitioner, status: finished, 2014.
- client: Sidley Austin LLP, case: Motorola vs. Microsoft, area: mobile instant messaging, project: software patent analysis, expert reports, *deposition*, represented: defense, status: finished (settled prior to trial), 2011-2012.
- client: Baker & McKenzie LLP, case: WMR e-PIN vs. Wells Fargo, area: Internet e-commerce applications, project: software, patent analysis, expert reports, *deposition*, **testifying 3 judge panel**, represented: defense, status: closed, 2008-2009.

Consulting Expert

- client: William G. Osborne, Esq., Superior Edge vs. Monsanto, area: distributed software, project: code analysis, report, represented: plaintiff, status: closed, 2015.
- client: Arnold & Porter LLP, Rosebud v. Adobe, area: distributed software, project: patent analysis, represented: defense, status: closed, 2015.
- client: Jones Day, Summit 6 vs. Apple, area: web technologies, project: patent analysis, represented: defense, status: closed, 2014.
- client: Ropes & Gray LLP, case: Parallel Iron vs. EMC, area: storage systems, project: patent analysis, represented: defense, status: closed, 2012-2013.
- client: Bridges & Mavrakakis LLP, case: Nokia/HTC vs. Apple, area: operating systems, project: software, patent analysis, represented: defense, status: closed, 2011.

Professional references

- Jim Day (Farella Braun + Martel): <https://www.fbm.com/james-l-day>
- Andrew Mace (Cooley): www.cooley.com/people/andrew-mace
- Larissa Bifano: (DLA Piper): www.dlapiper.com/en/us/people/b/bifano-larissa/
- Timothy Sullivan: (Faegre Baker Daniels LLP): www.faegrebd.com/en/professionals/s/sullivan-timothy-m
- Phil Haack: (Martin Ribera Schumann LLP): <https://www.martonribera.com/haack>
- Diek Van Nort (Morrison & Foerster LLP): www.mofo.com/people/v/van-nort-diek-o
- Joshua Goldberg (Finnegan LLP): <http://www.finnegan.com/>

MAJOR RESEARCH AFFILIATIONS

1. **Distributed Computing Systems Group, University of Minnesota, Founder and Co-Director..** The Distributed Computing Systems Group (DCSG) is collection of faculty and students working on topics in parallel and distributed computing, Internet/Web, mobile, cloud, edge, IoT, big data and machine learning. Cross-cutting topics include storage, networking, operating systems/middleware, security, and applications. Research centers on system building with a focus on resource management, efficiency, reliability, and enabling newly emerging applications.
2. **Digital Technology Center, University of Minnesota, Investigator.** The Digital Technology Center (DTC)'s goal is to create, promote, and coordinate cooperative interdisciplinary advanced technology initiatives between University, government, industry, and to serve as a point of entry into research and development partnerships with various partners. The DTC is a hub of innovation and excellence at the University of Minnesota in the digital technologies serving the industrial, educational, and public needs of the state of Minnesota and the nation. The DTC integrates research, education, and outreach in digital design, computer graphics and visualization, telecommunications, intelligent data storage and retrieval systems, multimedia, datamining, scientific computation, and other digital technologies. See dcs.cs.umn.edu for details.
3. **CRIS:Center for Research in Intelligent Storage, University of Minnesota, Investigator.** The Center for Research in Intelligent Storage (CRIS) is a partnership between universities and industry, featuring high-quality, industrially relevant fundamental research, strong industrial support of collaboration in research and education, and direct transfer of university developed ideas, research results, and technology to U.S. industry. Research areas include storage and memory technologies, big data, cloud computing, backup/restore, data deduplication, and file systems. See cris.cs.umn.edu.

GRANT FUNDING

1. **Joint Principal Investigator**, Edge Computing, Cisco, 2022-2023, \$165K (with Chandra).
2. **Principal Investigator**, CNS Core: Small: Scaling the IoT with Constellation, NSF, 2019-2022, \$500K (PI Chandra).
3. **Co-Principal Investigator**, I/UCRC Phase II: Center on Intelligent Storage, NSF, 2016-2019, \$895,516 (PI Du, Co-PIs Chandra, Lilja, Mokbel).
4. **Co-Principal Investigator**, CSR: Small: Location, location, location (L3): Support for Geo-Centric Applications, NSF, 2016-2019, \$528K (PI Chandra).
5. **Co-Principal Investigator**, II-NEW: One Cloud Does Not Fit All: Minnesota Integrated cloud Systems research Testbed (MIST), NSF, 2013-2016, \$350K, (PI DU, co-PIs Chandra, Zhang).
6. **Principal Investigator**, An Integrated Middleware Framework to Enable Extreme Collaborative Science, DOE, 2012-2015, \$342K UMn share, collaborative with Rutgers and University of Chicago.
7. **Principal Investigator**, CSR: Medium: Enriching Mobile User Experience Through The Cloud, NSF, 2012-2015, \$700K (co-PI Chandra, Karypis).
8. **Principal Investigator**, DC: Small: One Thousand Points of Light: Accelerating Data-Intensive Applications By Proxy, NSF, 2009-2012, \$482K, including REU supplement \$36K (co-PI Chandra).
9. **Co-Investigator**, Collaborative Research: A Multi-University I/UCRC Center on Intelligent Storage, NSF, 2009-2014, \$400K, (DU PI, many co-PIs).
10. **Principal Investigator**, Grid Computing for E-Science, National E-Science Centre, University of Edinburgh, 2007-2008, \$52K.
11. **Co-Investigator**, ePCRN: Electronic Primary Care Research Network, NIH Roadmap, 2006-2008, \$50K (CSE share annual), (Peterson PI, Delaney PI, several co-investigators).
12. **Co-Principal Investigator**, ITR: A Data Mining and Exploration Middleware for Grid and Distributed Computing, NSF ITR, 2003-2007, \$1.5M, (Kumar PI, Weissman co-PI).
13. **Investigator**, Intelligent Storage Consortium, DTC, 2003-2009, \$250K (Du PI, Weissman, Lilja, Tewfik, Kim co-PIs).
14. **Investigator**, MRI: Development of a System for Interactive Analysis and Visualization of Multi-Terabyte Datasets, NSF MRI, 2004-2006, \$300K (Woodward PI).
15. **Contributor**, NIH NCRR, Intelligent Data Storage Support for Microarray data, equipment, (Mayo Clinic PI, with Du), 2004, \$222K.
16. **Co-Principal Investigator**, Intelligent Storage Consortium - Engenio, Engenio Information Technologies, 2004-2005, \$45K, (Du PI, Weissman co-PI).
17. **Principal Investigator**, Making Parallel Computing Easy, Department of Energy (Office of Advanced Scientific Computing Research), 2002-2007, \$250K, sole PI.
18. **Principal Investigator**, A Framework for Adaptive Grid Services, NSF ACR-CNS, 2003-2008, \$240K, sole PI.
19. **Principal Investigator**, Towards Community Services: Putting Parallel Network Services On-line, NSF EIA-NGS Program, 2002-2003, \$30K, sole PI.
20. **Co-Principal Investigator**, Collaborative Data Analysis and Visualization, NSF EIA Research Resources (Collaborative Research), 2002-2005, \$500K (incl. 2 RA's shared with Du), (Woodward PI, Weissman, Du, Retzel, Wetherby co-PIs.)

21. **Investigator**, Metacomputing: Enabling Technologies and the Virtual Data Grid, Army High Performance Computing and Research Center (AHPCRC), 2000-2004, \$260K, original contributor to center proposal.
22. **Principal Investigator**, Resource Management for Parallel and Distributed Systems, NSF CAREER award ACR, 1996-2001, \$200K, REU supplement \$5K (2000), sole PI.
23. **Principal Investigator**, Smart File Objects: An Application-directed File Access Paradigm, Texas Advanced Research Program ARP-010115-226, 1997-2000, \$106K, sole PI.
24. **Participant**, Telecommunication Networks for Mobile and Distributed Computing and Communications, AFOSR-F49620-96-1-0472, 1996-2000, \$2M, (PI Hiromoto).
25. **Co-Principal Investigator**, Building a Pipeline for Minority Scholars, NSF CISE MI-I CDA-9633299, 1996 - 1999, \$1.3M. (Hiromoto PI, Weissman, Das, Boppana, Psarris, Jeffery, Rosen co-PIs).
26. **Fellowship**, Applying Parallel Object-Oriented Computing to Scientific Applications, NASA Graduate Student Researchers Program Fellowship, Jet Propulsion Laboratory, 1991-1995, \$66K.

PUBLICATIONS

Journal Articles: for up-to-date-list, see www.cs.umn.edu/~jon

1. Kwangsung Oh, Abhishek Chandra, and **Jon Weissman**, Network Cost-aware Geo-distributed Data Analytics System *IEEE Transactions on Parallel and Distributed Systems*, 2022.
2. Kwangsung Oh, Nan Qin, Abhishek Chandra, and **Jon Weissman**, Wiera: Policy-Driven Multi-Tiered Geo-Distributed Cloud Storage System, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 31, No. 11, Feb 2020.
3. Albert Jonathan, Mathew Ryden, Kwangsung Oh, Abhishek Chandra, and **Jon Weissman**, Nebula: Distributed Edge Cloud for Data Intensive Computing, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, No. 11, Nov 2017.
4. Benjamin Heintz, Abhishek Chandra, Ramesh Sitaraman, and **Jon B. Weissman**, End-to-end Optimization for Geo-Distributed MapReduce, *IEEE Transactions on Cloud Computing*, Vol. 4, No. 3, July 2016.
5. Jinoh Kim, Seonho Kim, and **Jon B. Weissman**, A Security-enabled Grid System for MINDS Distributed Data Mining, *Journal of Grid Computing*, 2014.
6. Murray Cole, Shantenu Jha, Daniel S. Katz, Manish Parashar, Omer Rana, and **Jon B. Weissman**, Distributed Computing Practice for Large-Scale Science and Engineering Applications, *Concurrency: Practice and Experience*, Volume 25, Issue 11, September 2013.
7. Adam Barker, **Jon B. Weissman**, Jano van Hemert, Accelerating Service-Oriented Workflows, *IEEE Transactions on Services Computing*, 5(3), 2012.
8. Jinoh Kim, Abhishek Chandra and **Jon B. Weissman**, Passive Network Performance Estimation for Large-scale, Data-intensive Computing, *IEEE Transactions on Parallel and Distributed Systems*, 22(8), August 2011.
9. Jia Jingxi, Bharadwaj Veeravalli, and **Jon B. Weissman**, Scheduling Multi-source Divisible Loads on Arbitrary Networks, *IEEE Transactions on Parallel and Distributed Systems*, 21(4), April 2010.
10. Jinoh Kim, Abhishek Chandra and **Jon B. Weissman**, Using Data Accessibility for Resource Selection in Large-scale Distributed Systems, *IEEE Transactions on Parallel and Distributed Systems*, 20(6), June 2009.
11. Byoung-Dai Lee, **Jon B. Weissman**, Young-Kwang Nam, Adaptive middleware supporting scalable performance for high-end network services, *Journal of Network and Computer Applications*, 32(3), May 2009.
12. Adam Barker, **Jon B. Weissman** and Jano van Hemert, The Circulate Architecture: Avoiding Workflow Bottlenecks Caused By Centralised Orchestration, *Journal of Cluster Computing*, 12(2), 2009.
13. Jason D. Sonnek, Abhishek Chandra, and **Jon B. Weissman**, Adaptive Reputation-Based Scheduling on Unreliable Distributed Infrastructures, *IEEE Transactions on Parallel and Distributed Systems*, 18(11), November 2007.
14. Darin England, Bharadwaj Veeravalli, and **Jon B. Weissman**, A Robust Spanning Tree Topology for Data Collection and Dissemination in Distributed Environments, *IEEE Transactions for Parallel and Distributed Systems*, 18(5), May 2007.
15. Rahul Trivedi, Abhishek Chandra, and **Jon B. Weissman**, Heterogeity-Aware Workload Distribution in Donation Based Grids, invited to the *International Journal for High Performance Computing and Applications* (IJHPCA), 20(4), 2006.
16. Darin England and **Jon B. Weissman**, A Resource Leasing Policy for On-demand Computing, invited to the *International Journal for High Performance Computing and Applications* (IJHPCA), 20(1), 2006.
17. **Jon B. Weissman**, Darin England, and Lakshman Abburi Rao, "Integrated Scheduling: The Best of Both Worlds," *Journal of Parallel and Distributed Computing*, 63(6), 2003.

18. **Jon B. Weissman** and Byoung-Dai Lee, "The Virtual Service Grid: An Architecture for Delivering High-End Network Services", *Concurrency: Practice and Experience*, Vol. 14, No. 4, April 2002.
19. **Jon B. Weissman**, "Predicting the Cost and Benefit of Adapting Parallel Applications in Clusters", *Journal of Parallel and Distributed Computing*, Vol. 62, No. 8, August 2002.
20. **Jon B. Weissman**, Mike Gingras, and Mahesh Marina, "Optimizing Remote File Access for Parallel and Distributed Network Applications" *Journal of Parallel and Distributed Computing*, Vol. 61, No. 11, November 2001.
21. **Jon B. Weissman**, "Prophet: Automated Scheduling of SPMD Programs in Workstation Networks", *Concurrency: Practice and Experience*, Vol. 11, No. 6, August 1999.
22. **Jon B. Weissman** "Gallop: The Benefits of Wide-Area Computing for Parallel Processing", *Journal of Parallel and Distributed Computing*, Vol. 54, No. 2, November 1998.
23. **Jon B. Weissman** and Xin Zhao, "Scheduling Parallel Applications in Distributed Networks," *Journal of Cluster Computing*, Vol. 1, No. 1, May 1998, invited paper.
24. Andrew S. Grimshaw, **Jon B. Weissman**, and W. Timothy Strayer, "Portable Run-time Support for Dynamic Object-Oriented Parallel Processing," *ACM Transactions on Computer Systems*, Vol. 14, No. 2, May 1996.
25. **Jon B. Weissman** and Andrew S. Grimshaw, "A Framework for Partitioning Parallel Computations in Heterogeneous Environments", *Concurrency: Practice and Experience*, Vol. 7, No. 5, August 1995.
26. Andrew Grimshaw, **Jon B. Weissman**, Emily West, and Edmond Loyot, Jr., "Metasystems: An Approach Combining Parallel Processing and Heterogeneous Distributed Computing Systems", *Journal of Parallel and Distributed Computing*, Vol. 21, No. 3, June 1994.
27. **Jon B. Weissman**, Andrew S. Grimshaw, and Robert R. Ferraro, "Parallel Object-Oriented Computation Applied to a Finite Element Problem," *Journal of Scientific Programming*, Vol. 2, No. 4, 1993.

Books and Book Chapters

1. Benjamin Heintz, Abhishek Chandra and **Jon Weissman**, Cross-Phase Optimization in MapReduce, in Cloud Computing for Data-Intensive Applications, Springer, 2014.
2. **Jon B. Weissman** and Jinoh Kim, Network Awareness in Volunteer Networks, in Desktop Grid Computing, CRC Press, 2013.
3. Steve Chapin and **Jon B. Weissman**, Distributed and Multiprocessor Scheduling (Operating Systems Area), The Computer Science and Engineering Handbook, 2004.

Highly Selective Articles : for up-to-date-list, see www.cs.umn.edu/~jon

1. Lei Huang, Zhiying Liang, Nikhil Sreekumar, Sumanth Kaushik Vishwanath, Cody Perakslis, Abhishek Chandra, and **Jon Weissman**, Armada: A Robust Latency-Sensitive Edge Cloud in Heterogeneous Edge-Dense Environments, *42nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2022.
2. Kwangsung Oh, Abhishek Chandra, and **Jon Weissman**, A Network Cost-aware Geo-distributed Data Analytics System *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021.
3. Albert Jonathan, Abhishek Chandra, and **Jon Weissman**, Wide-area Adaptive Stream Processing *ACM Middleware*, 2020.
4. Nikhil Sreekumar, Abhishek Chandra, and **Jon Weissman**, Towards a Robust Edge-Native Storage System *Fifth ACM/IEEE Symposium on Edge Computing (SEC)*, 2020.
5. Zach Leidall, Abhishek Chandra, **Jon Weissman**, An Edge-based Framework for Cooperation in Internet of Things Applications, 2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge), Renton, Washington, July 2019.
6. Albert Jonathan, Abhishek Chandra, and **Jon Weissman**, Multi-Query Optimization in Wide-Area Streaming Analytics, *ACM Symposium on Cloud Computing 2018 (SOCC)*, Carlsbad, California, October 2018.
7. Feng Liu, Kate Keahey, Pierre Riteau, and **Jon Weissman**, Dynamically Negotiating Capacity between On-Demand and Batch Clusters, *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'18)*, Nov 2018, Dallas, TX.
8. Albert Jonathan, Abhishek Chandra, and **Jon B. Weissman**, Rethinking Adaptability in Wide-Area Stream Processing Systems, *Workshop on Hot Topics in Cloud Computing (HotCloud '18)*, June 2018.
9. Kwangsung Oh, Abhishek Chandra and **Jon Weissman**, TripS: Automated Multi-tiered Data Placement in a Geo-distributed Cloud Environment, *10th ACM International Systems and Storage Conference (SYSTOR)*, May 2017, Haifa Israel.
10. Kwangsung Oh, Abhishek Chandra, and **Jon Weissman**, Wiera: Towards Flexible Multi-Tiered Geo-Distributed Cloud Storage Instances, *25th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, June 2016, Kyoto, Japan.
11. Matteo Turilli, Feng Liu, Zhao Zhang, Andre Merzky, Michael Wilde, **Jon Weissman**, Daniel S. Katz, and Shantenu Jha, Integrating Abstractions to Enhance the Execution of Distributed Applications, *30th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, May 2016, Chicago, IL.
12. Albert Jonathan, Abhishek Chandra, and **Jon Weissman**, Awan: Locality-aware Resource Manager for Geo-distributed Data-intensive Applications, *IEEE International Conference on Cloud Engineering*, April 2016, Berlin, Germany.
13. Francis Liu and **Jon Weissman**, Elastic Job Bundling: An Adaptive Resource Request Strategy for Large-Scale Parallel Applications, to appear in the *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'15)*, Nov 2015, Austin, TX.
14. John Kolb, Prashant Chaudhary, Alexander Schillinger, Abhishek Chandra and **Jon Weissman**, Cloud-Based, User-Centric Mobile Application, *IEEE International Conference on Cloud Engineering*, March 2015, Tempe, AZ.
15. Ajaykrishna Raghavan, Abhishek Chandra, and **Jon B. Weissman**, Tiera: Towards Flexible Multi-Tiered Cloud Storage Instances, *ACM Middleware*, 2014.
16. John Kolb, Will Myott, Thao Nguyen, Abhishek Chandra and **Jon B. Weissman**, Exploiting User Interest in Data-Driven Cloud-based Mobile Optimization, *IEEE Conference on Mobile Cloud Computing*, April 2014, Oxford, UK

17. Mathew Ryden, Kwangsung Oh, Abhishek Chandra and **Jon B. Weissman**, Nebula: Distributed Edge Cloud for Data Intensive Computing, *IEEE International Conference on Cloud Engineering*, March 2014, Boston, MA
18. Abhishek Chandra, **Jon Weissman**, and Benjamin Heintz Decentralized Edge Clouds, *IEEE Internet Computing*, 2013.
19. Benjamin Heintz, Chenyu Wang, Abhishek Chandra, and **Jon B. Weissman**, Cross-Phase Optimization in MapReduce, *IEEE International Conference on Cloud Engineering*, March 2013, San Francisco, CA
20. Chonglei Mei, Daniel Taylor, Chenyu Wang, Abhishek Chandra, and **Jon B. Weissman**, Sharing-aware Cloud-based Mobile Outsourcing *IEEE 5th International Conference on Cloud Computing*, 2012.
21. Atul Katiyar and **Jon B. Weissman**, ViDeDup: An Application-Aware Framework for Video De-duplication, *Third Usenix Workshop on Hot Topics in Storage and File Systems (HotStorage'11)*, Portland, OR 2011.
22. Louis-Claude Canon, Emmanuel Jeannot, and **Jon B. Weissman**, A Scheduling and Certification Algorithm for Defeating Collusion in Desktop Grids, *31st IEEE International Conference on Distributed Computing Systems (ICDCS)*, Minneapolis, MN, June 2011.
23. Louis-Claude Canon, Emmanuel Jeannot, and **Jon B. Weissman**, A Dynamic Approach for Characterizing Collusion in Desktop Grids, *24th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2010.
24. Abhishek Chandra and **Jon B. Weissman**, Nebulas: Using Distributed Voluntary Resources to Build Clouds, *Workshop on Hot Topics in Cloud Computing (HotCloud '09)*, June 2009.
25. **Jon B. Weissman** and Siddharth Ramakrishnan, Using Proxies to Accelerate Cloud Applications, *Workshop on Hot Topics in Cloud Computing (HotCloud '09)*, June 2009.
26. Adam Barker, **Jon B. Weissman** and Jano van Hemert, Eliminating the Middle-Man: Peer-to-Peer Dataflow, *17th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, Boston, MA, June 2008.
27. Jinoh Kim, Abhishek Chandra and **Jon B. Weissman**, Accessibility-based Resource Selection in Loosely-coupled Distributed Systems, *28th IEEE International Conference on Distributed Computing Systems (ICDCS)*, Beijing, China, June 2008.
28. Krishnaveni Budati, Abhishek Chandra, and **Jon B. Weissman**, RIDGE: Combining Reliability and Performance in Open Grid Platforms, *16th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, June 2007.
29. Jason D. Sonnek, Mukesh Nathan, Abhishek Chandra, and **Jon B. Weissman**, Reputation Based Scheduling on Unreliable Distributed Infrastructures, *The 26th International Conference on Distributed Computing Systems (ICDCS)*, July 2006.
30. Darin England, Jayashree Sadagopan, and **Jon B. Weissman**, A New Metric for Robustness with Application to Job Scheduling, *Fourteenth IEEE International Symposium on High Performance Distributed Computing (HPDC)*, 2005.
31. **Jon B. Weissman** and Xin Zhao, "Run-time Support for Scheduling Parallel Applications in Heterogeneous NOWs," *Proceedings of the Sixth IEEE International Symposium on High Performance Distributed Computing (HPDC)*, August 1997.
32. **Jon B. Weissman** and Andrew S. Grimshaw, "A Federated Model for Scheduling in Wide-Area Systems," *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing (HPDC)*, August 1996.
33. **Jon B. Weissman** and Andrew S. Grimshaw, "Network Partitioning of Data Parallel Computations," *Proceedings of the Third IEEE International Symposium on High Performance Distributed Computing (HPDC)*, August 1994.

Selective Articles: : for up-to-date-list, see www.cs.umn.edu/~jon

34. Albert Jonathan, Abhishek Chandra, and **Jon Weissman**, Locality-Aware Load Sharing in Mobile Cloud Computing *IEEE/ACM International Conference on Utility and Cloud Computing*, 2017.
35. Albert Jonathan, Abhishek Chandra, and **Jon Weissman**, Ensuring Reliability in Geo-Distributed Edge Cloud, *IEEE RWS (Resilience Week)*, Wilmington, Delaware, 2017.
36. Kwangsung Oh, Ajaykrishna Raghavan, Abhishek Chandra and **Jon Weissman**, Redefining Data Locality for Cross-Data Center Storage, *International Workshop on Software-Defined Ecosystems (BigSystem 2015)*, June 2015, Portland, OR.
37. William Myott, Thao Nguyen, Abhishek Chandra, George Karypis, and **Jon B. Weissman**, Opportunities for Data-Driven Cloud-based Mobile Optimization, invited to 2014 *International Conference on Collaboration Technologies and Systems*, Minneapolis, MN, May 2014.
38. Siddharth Ramakrishna, Robert Reutiman, Abhishek Chandra, and **Jon B. Weissman**, Accelerating Distributed Workflows With Edge Resources, *Second International Workshop on Workflow Models, Systems, Services and Applications in the Cloud (CloudFlow)*, Boston MA, May 2013.
39. Mike Cardosa, Anshuman Nangia, Chenyu Wang, Abhishek Chandra and **Jon B. Weissman**, Exploring MapReduce Efficiency with Highly-Distributed Data, *The Second International Workshop on MapReduce and its Applications (MapReduce '11)*, June 2011.
40. Pradeep Sundarrajan, Abhishek Gupta, Matthew Ryden, Rohit Nair, Abhishek Chandra, and **Jon B. Weissman**, Early Experience with the Distributed Nebula Cloud, *The Fourth International Workshop on Data Intensive Distributed Computing (DIDC '11)*, June 2011.
41. Shantenu Jha, Daniel S. Katz, Manish Parashar, Omer Rana, and **Jon B. Weissman**, Critical Perspectives on Large-Scale Distributed Applications and Production Grids, *10th IEEE/ACM Conference on Grid Computing (Grid 2009)*, Banff, Canada, 2009, winner best paper award.
42. Adam Barker, Paolo Besanay, David Robertson, and **Jon B. Weissman**, The Benefits Of Service Choreography For Data-Intensive Computing, *Proceedings of the 2009 Workshop in Challenges of Large Applications in Distributed Environments (CLADE)*, 2009.
43. Adam Barker, **Jon B. Weissman** and Jano van Hemert, Orchestrating Data-centric Workflows, *IEEE/ACM CCGrid International Symposium on Cluster Computing and the Grid (CCGrid)*, Lyon, France, May 2008.
44. Vasumathi Sundaram, Abhishek Chandra and **Jon B. Weissman**, Exploring the Throughput-Fairness Tradeoff of Deadline Scheduling in Heterogeneous Computing Environments, *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, short paper, Annapolis, MD, June 2008.
45. Krishnaveni Budati, Jinoh Kim, Abhishek Chandra and **Jon Weissman**, NGS: Service Adaptation in Open Grid Platforms, *Proceedings of the IPDPS NSF Next Generation Software Workshop*, March 2007.
46. **Jon B. Weissman**, Vipin Kumar, Varun Chandola, Eric Eilertson, Levent Ertöz, Gyorgy Simon, Seonho Kim, and Jinoh Kim, DDDAS/ITR: A Data Mining and Exploration Middleware for Grid and Distributed Computing, *Workshop on Dynamic Data Driven Application Systems - DDDAS 2007*, Beijing, China, May 2007.
47. Jinoh Kim, Abhishek Chandra, and **Jon B. Weissman**, Exploiting Heterogeneity for Collective Data Downloading in Volunteer-based Networks, *IEEE/ACM CCGrid International Symposium on Cluster Computing and the Grid (CCGrid)*, 2007.
48. **Jon B. Weissman**, Seonho Kim, and Darin England, A Framework for Dynamic Service Adaptation in the Grid, *Proceedings of the IPDPS NSF Next Generation Software Workshop Denver*, Colorado, April 2005.
49. Jason D. Sonnek and **Jon B. Weissman**, A Quantitative Comparison of Reputation Systems in the Grid, *6th IEEE/ACM International Workshop on Grid Computing (GRID)*.

50. **Jon B. Weissman**, Seonho Kim, and Darin England, Supporting the Dynamic Grid Service Lifecycle, *IEEE/ACM CCGrid International Symposium on Cluster Computing and the Grid* (CCGrid), 2005.
51. Darin England and **Jon B. Weissman**, "A Stochastic Control Model for the Deployment of Dynamic Grid Services", *5th IEEE/ACM International Workshop on Grid Computing* (GRID), 2004.
52. Darin England and **Jon B. Weissman**, "A Load Sharing Model for the Computational Grid," *Workshop on Job Scheduling Strategies for Parallel Processing* (JSSPP) with Sigmetrics 2004.
53. Seonho Kim and **Jon B. Weissman**, "A GA-based approach for scheduling decomposable data grid applications," *International Conference on Parallel Processing* (ICPP) 2004.
54. Byoung-Dai Lee and **Jon B. Weissman**, "Adaptive Resource Selection for Grid-enabled Network Services," *IEEE 2nd International Symposium on Network Computing and Applications* (NCA), 2003.
55. Byoung-Dai Lee and **Jon B. Weissman**, "Adaptive Resource Scheduling for Network Services," *IEEE 3rd International Workshop on Grid Computing*, 2002.
56. Byoung-Dai Lee and **Jon B. Weissman**, "Community Service Toolkit for Rapid Deployment of Network Services," *IEEE 2002 Workshop on Cluster Computing*, 2002.
57. **Jon B. Weissman** and Pramod Srinivasan, "Ensemble Scheduling: Resource Co-Allocation on the Computational Grid", *IEEE 2nd International Workshop on Grid Computing* (GRID), November 2001.
58. Byoung-Dai Lee and **Jon B. Weissman**, "An Adaptive Service Grid Architecture for Dynamic Replication," *IEEE 2nd International Workshop on Grid Computing* (GRID), November 2001.
59. **Jon B. Weissman** and Byoung-Dai Lee, "The Service Grid: Supporting Scalable Heterogenous Services in Wide-Area Networks," *The IEEE 2001 Symposium on Applications and the Internet* (SAINT), January 2001.
60. **Jon B. Weissman**, "Scheduling Multi-Component Applications in Heterogeneous Wide-area Networks," *Heterogeneous Computing Workshop*, International Parallel and Distributed Processing Symposium, May 2000.
61. **Jon B. Weissman**, "Fault Tolerant Wide-Area Parallel Computing," *IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems*, International Parallel and Distributed Processing Symposium, May 2000.
62. Mike Gingras and **Jon B. Weissman**, "Smart Multimedia File Objects," *IEEE Workshop on Internet Applications* (WIA), July 1999.
63. **Jon B. Weissman**, "Smart File Objects: A Remote File Access Paradigm," *Sixth ACM Workshop on I/O in Parallel and Distributed Systems* (IOPADS), May 1999.
64. **Jon B. Weissman**, "The Interference Paradigm for Network Job Scheduling," *Heterogeneous Computing Workshop*, Tenth International Parallel and Distributed Processing Symposium, April 1996.
65. **Jon B. Weissman**, Andrew S. Grimshaw, and Robert R. Ferraro, "Parallel Object-Oriented Computation Applied to a Finite Element Problem," *Object-Oriented Numerics*, 1993.
66. Lisa Sokol, **Jon B. Weissman**, and Paula Mutchler, "The Role of Event Granularity in Parallel Simulation Design," *Proceedings of the Parallel and Distributed Simulation Conference* (PADS), 1992.
67. Richard Weatherly, David Seidel, and **Jon B. Weissman**, "Aggregate Level Simulation Protocol," *Proceedings of the 1991 Summer Computer Simulation Conference*, July 1991.
68. Lisa Sokol, **Jon B. Weissman**, and Paula Mutchler, "An Empirical Comparison of MTW and Time Warp in a Shared Memory Environment," *Proceedings of the SCS Multiconference on Work in Progress (WSMC)*, January 1991.
69. Lisa Sokol, **Jon B. Weissman**, and Paula Mutchler, "MTW: An Empirical Performance Study," *Proceedings of the Winter Simulation Conference* (WSC), December 1991.

Short Papers

1. Zach Leidall, Abhishek Chandra, **Jon Weissman** Creating an Abstraction to Exploit IoT Synergies, *2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI 2017)*, Extended Abstract/Poster, April 2017.
2. Arvanitis TN, Taweel A, Zhao L, Delaney BC, Peterson KA , Speedie SM, Sim I, **Weissman J**, Fontaine P, Lange C, Janowiec M, and Stone J, Supporting e-trials over distributed networks: a tool for capturing randomised control, *MEDNET 2007*.
3. Seonho Kim, **Jon B. Weissman**, Kevin A. Peterson, A Secure Federated Health Data Query System for Primary Care Clinical Trials on the Grid, *2007 Annual Meeting NIH Cancer Grid (CABig)*, February 2007.
4. Speedie SM, Peterson K, Fontaine P, **Weissman J**, Delaney B, Arvanitis T, Taweel A, Sim I, Lange C, Janowiec M, Stone J, Wolff, A., An Infrastructure for Conducting Clinical Trials in Primary Care. "Current Issues in Clinical Research: Latest Trends in Clinical Research" (abstract) October 4-5, 2006, Minneapolis, MN.
5. Rahuk Trivedi, Abhishek Chandra, and **Jon B. Weissman**, Platform-of-Platforms: A Modular Integrated Resource Framework for Large-Scale Services, *2nd Symposium on Networked System Design and Implementation (NSDI)*, 2005, (poster paper).
6. **Jon B. Weissman** and Zhi-li Zhang, "Smart Environments: Middleware Building Blocks for Pervasive Network Computing," *NSF workshop on Infrastructure for Mobile and Wireless Systems*, Springer-Verlag Research Notes, 2003.
7. Byoung-Dai Lee and **Jon B. Weissman**, "Dynamic Replica Management in the Service Grid," *10th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, August 2001.
8. Mike Karo, Christopher Dwan, John Freeman, **Jon B. Weissman**, Miron Livny, and Ernest Retzel, "Applying Grid Technology to Bioinformatics", with Mike Karo et. al, *10th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, August 2001.
9. **Jon B. Weissman**, "Fault Tolerant Computing on the Grid: What are my options?," *Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC)*, August 1999.
10. **Jon B. Weissman**, "Metascheduling: A Scheduling Model for Metacomputing Systems," *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC)*, 1998.
11. Andrew Grimshaw, **Jon B. Weissman**, and Emily A. West, "Overhead versus Application Granularity in Mentat," *Proceedings of the Workshop on Efficient Implementation of Concurrent Object-Oriented Programs (OOPSLA)*, Washington D.C., September, 1993, extended abstract.
12. Andrew Grimshaw, **Jon B. Weissman**, and Emily A. West, "Experiences with the Mentat Metasystems Testbed," *1992 Workshop on Cluster Computing*, December 1992, extended abstract.

Technical Reports and Other Papers

1. Mitch Terrell, Yixuan Wang, Matt Dorow, Soumya Agrawal, Bhaargav Sriraman, Zach Leidall, Abhishek Chandra, **Jon Weissman**, Constellation: An Edge-Based Semantic Runtime System for Internet of Things Applications, *arXiv:2201.12394*, 2022.
2. Lei Huang, Zhiying Liang, Nikhil Sreekumar, Sumanth Kaushik Vishwanath, Cody Perakslis, Abhishek Chandra, and **Jon Weissman**, Armada: A Robust Latency-Sensitive Edge Cloud in Heterogeneous Edge-Dense Environments, *arXiv:2111.12002*, 2021.
3. Aravind Alagiri Ramkumar, Rohit Sindhu, and **Jon Weissman**, Constellation Plan B Report, University of Minnesota Computer Science TR-20-001, January 2020.
4. Ajaykrishna Raghavan, Abhishek Chandra, and **Jon B. Weissman**, Tiera: Towards Flexible Multi-Tiered Cloud Storage Instances, University of Minnesota Computer Science TR-14-003, January 2014.
5. Mathew Ryden, Abhishek Chandra, and **Jon Weissman**, Nebula: Data Intensive Computing over Widely Distributed Voluntary Resources, University of Minnesota Computer Science TR-13-07, March 2013.
6. Chonglei Mei, Daniel Taylor, Chenyu Wang, Abhishek Chandra, and **Jon B. Weissman**, Mobilizing the Cloud: Enabling Multi-User Mobile Outsourcing in the Cloud, University of Minnesota Computer Science TR-11-029, November 2011.
7. Dan Katz, Shantenu Jha, Manish Parashar, Omer Rana, and **Jon Weissman**, Analysis of Production Distributed Computing Infrastructures, Technical Report CI-TR-7-0811. Computation Institute, University of Chicago & Argonne National Laboratory, September 2011.
8. Chonglei Mei, James Shimek, Chenyu Wang, Abhishek Chandra and **Jon Weissman**, Dynamic Outsourcing Mobile Computation to the Cloud, University of Minnesota Computer Science TR-11-006, March 2011.
9. Louis-Claude Canon, Emmanuel Jeannot, and **Jon B. Weissman**, A Scheduling Algorithm for Defeating Collusion, INRIA, Research Report RR-7403, Oct. 2010.
10. Siddharth Ramakrishnan, Robert Reutiman, Abhishek Chandra, and **Jon B. Weissman**, Standing on the Shoulders of Others: Using Proxies to Opportunistically Boost Distributed Applications, University of Minnesota Computer Science TR-10-012, 2010.
11. Siddharth Ramakrishnan, Robert Reutiman, Harlan Iverson, Shimin Lian, Abhishek Chandra, and **Jon B. Weissman**, Early Experience with Mobile Computation Outsourcing, University of Minnesota Computer Science TR-10-020, 2010.
12. Vasumathi Sundaram, Abhishek Chandra, **Jon B. Weissman**, Exploring the Throughput-Fairness Tradeoff of Deadline Scheduling in Heterogeneous Computing Environments, University of Minnesota Computer Science TR-08-003, 2008.
13. Jinoh Kim, Abhishek Chandra, and **Jon B. Weissman**, OPEN: Passive Network Performance Estimation for Data-intensive Applications, University of Minnesota Computer Science TR-08-041, 2008.
14. Seonho Kim, Jinoh Kim, and **Jon B. Weissman**, A Security-Enabled Grid Middleware for Distributed Data Mining, University of Minnesota Computer Science TR-07-001, 2007.
15. Abhishek Chandra, Rahul Trivedi, and **Jon B. Weissman**, Hosting Services on the Grid: Challenges and Opportunities, University of Minnesota Computer Science TR-05-026, 2005.
16. Lakshman Abburi Rao, **Jon B. Weissman**, MPI-based Adaptive Parallel Grid Services, University of Minnesota Computer Science TR-03-032, 2003.
17. **Jon B. Weissman**, "The Virtual Data Grid: Just-in-time Data Delivery and Data Production for HPC Applications", DOD High Performance Computing Modernization Program Success Story Book, 2002.
18. **Jon B. Weissman**, "Metacomputing Research at the AHPCRC", AHPCRC Bulletin, 2002.

19. **Jon B. Weissman** and Colin Gan, “The Virtual Data Grid: An Infrastructure to Support Distributed Data-Centric Applications,” *DOD High Performance Computing Modernization Conference Program* (Users Group Conference 2002), June 2002.
20. **Jon B. Weissman**, “Grids in the Classroom,” *IEEE Concurrency*, education feature, July 2000.

INVITED TALKS

(keynotes in bold)

1. Cloud Data Centers & Edge (CDCE) Workshop UMN, CRIS, Edge Computing Infrastructure, 2021.
2. Media, WCCO Radio Live Interview, Can the Internet Withstand Covid-19?, March 20, 2020.
3. Invited Talk, Hamline University, What does the future of the Internet Hold?, March 2019.
4. Invited Talk, IUCRC Meeting, talk title: Geo-distributed Cloud Storage, May 2017.
5. Invited **Inaugural Keynote**, Open Cloud Institute, University of Texas at San Antonio, talk title: Geo-Distributed Clouds, September 2015.
6. Invited Talk, NITRD MAGIC, talk title: AIMES: Abstractions and Integrated Middleware For Extreme-Scale Science, March 2015.
7. Invited Talk, Inria-ENS, Lyon, France, talk title: Job Shaping on the Cloud, July 2014.
8. Invited Talk, 2014 International Conference on Collaboration Technologies and Systems (CTS 2014), talk title: Opportunities for Data-Driven Cloud-based Mobile Optimization, May 2014.
9. Invited **Keynote**, NSF/ARL Workshop on Cloud Robotics: Challenges and Opportunities, Drexel University, Feb 2013.
10. Invited Panelist, HPDC, The Value of HPDC, June 2012.
11. Invited Talk, CMU, talk title: Extending Mobile Devices into the Cloud, 2012.
12. Invited Talk, Inria-PGH, talk title: TBD, June 2012.
13. Invited Talk, NSF US-China Workshop on Collaborative Software Research, talk title: TBD, 2012.
14. Invited Talk, Thomson-Reuters Inc, talk title: A Tour of MapReduce/Hadoop, 2012.
15. Invited Talk, Adventium Labs Inc, talk title: Minnesota Cloud Research, November 2011.
16. Invited Talk, Georgia Tech (Open Cirrus Summit), talk title: Expanding the Horizons of Cloud Computing Beyond the Data Center, October 2011.
17. Invited Talk, UMn CS Colloquium, talk title: Scheduling Edge Resources to Build Better Distributed Systems, October 2011.
18. Invited **Keynote**, The Fourth International Workshop on Data Intensive Distributed Computing (DIDC 2011), San Jose, California, talk title: Wide-Area Clouds, 2011.
19. Invited Talk, Microsoft, Cloud Futures Workshop, talk title: Expanding the Horizons of Cloud Computing Beyond the Data Center, June 2011.
20. Invited Talk, Inria-Grenoble, Grenoble, France, talk title: Cloud Computing Using Edge Resources, May 2011.
21. Invited Talk, Inria-sponsored Workshop Scheduling for Large Scale systems, Aussois, France, talk title: Scheduling in the Cloud, May 2011.
22. Invited Talk, UMn CS Colloquium, talk title: Cloud Computing: Research Perspectives, April 2011.
23. Invited Talk, University of Minnesota, Science & Engineering Day, talk title: Charting the Cloud Landscape, March 2011.
24. Invited Talk, NSF Workshop: The Science of Cloud Computing, talk title: Minnesota Cloud Research Vision, March 2011.
25. Invited Talk, University of Edinburgh, Informatics/CS, talk title: New Cloud Models for Distributed Dynamic Data-Intensive Programming Application Systems, November 2010.

26. Invited Talk, Inria-sponsored Workshop: Scheduling for Large Scale systems, Aussois, France, talk title: Living on the Edge: Scheduling Edge Resources Across the Cloud, May 2010.
27. Invited Talk, UMn IT Dean Series, talk title: CloudSystems@UMn, April 2010.
28. Invited Talk, Louisiana State University, talk title: Living on the Edge: Broadening the Vision of Cloud Computing, Baton Rouge, Louisiana, March 2010.
29. Invited **Keynote**, Third Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid), Rome, Italy, talk title: Cloud@Grid, 2009.
30. Invited Talk, University of Florida, Workshop on Data Mining for Biomedical Applications, talk title: Communication is Key to a Happy Marriage-Linking Biomedical Applications and the Open Grid, September 2008.
31. Invited Talk, Sabbatical lecture: University of Edinburgh, Informatics/CS, talk title: Grid Computing for E-Science, July 2008.
32. Invited Talk, University of Edinburgh, Informatics/CS, Edinburgh, talk title: Challenges in Reliable Grid Computing, July 2008.
33. Invited Talk, CLADE Workshop, Boston, MA, talk title: Scalable Computing on Open Distributed Systems, June 2008.
34. Invited **Keynote**, ASCI conference, Delft, Netherlands, talk title: Challenges in Reliable Grid Computing, June 2008.
35. Invited Talk, NeSC Science Advisory Board, talk title: Grid Computing for E-Science, May 2008.
36. Invited Talk, University of Edinburgh, Informatics/CS, NeSC Research Seminar, talk title: Harnessing Volunteer Grids, September 2007.
37. Invited Talk, University of Edinburgh, E-Science Institute Public Lecture: talk title: Decentralizing Grids, November 2007.
38. Invited Talk: Vrije University, Amsterdam, Netherlands, talk title: The Dynamic Grid, November 2007.
39. Invited Talk, University of Edinburgh, Informatics/CS, talk title: Achieving Reliability in Open Grids, April 2007.
40. Invited Talk, University of Chicago, talk title: Achieving Reliability in Open Grids, March 2007.
41. Invited Talk, Vrije University, Amsterdam, Netherlands, talk title: Trust-Sensitive Scheduling on the Open Grid, March 2006.
42. Invited Education Talk, UMn Institute of Technology, talk title: Recruiting Workshop talk, Winter 2006.
43. Invited Talk, Inria/NSF-sponsored Workshop: Scheduling for Large Scale distributed platforms, San Diego, California, talk title: Scheduling Challenges in Hosting Services on the Public Grid, 2005.
44. Invited Education Talk, University of Wisconsin, Eau Claire, talk title: Graduate Studies in Computer Science at the "U", 2005.
45. Invited Talk, DOE-ANL, talk title: Future Technology Challenges for Collaboration: Customizing the Grid, 2004.
46. Invited Talk, Inria/NSF-sponsored Workshop: Scheduling for Large-Scale Distributed Platforms, Aussois, France, talk title: Robust Grid Scheduling, 2004.
47. Invited Talk, Digital Technology Center, Open House, talk title: Emerging Technologies: Grid Computing, 2003.
48. Invited Talk, Minnesota IT Leadership Forum, Minneapolis, MN, talk title: Emerging Technology: Grid Computing, October 2002.

49. Invited Talk, Digital Technology Center, Grid Computing: Collaborative Opportunities within the DTC, December 2002.
50. Invited demonstration, HPDC, title: Community Services Project, July 2002.
51. Invited Talk, University of Indiana, Argonne National Laboratory/University of Chicago, talk title: Community Services: Putting High-end Network Services On-line, University of Tennessee, University of Minnesota, 2002.
52. Invited Talk, Dagstuhl Seminar on Management of Metacomputers,” Saarbrücken, Germany, talk title: Scheduling Computational Grids: A Five-Year Retrospective, June 2001.
53. Invited Talk, UMn CS Colloquium, talk title: Turning Hype into Reality: Achieving High Performance in a Computational Grid, Fall 1999.
54. Invited Talk, North Carolina State University, Virginia Polytechnic Institute, Northwestern, Indiana University, University of Kentucky, University of Oregon, talk title: Smart File Objects, Spring 1999.
55. Invited Talk, Southwest Research Institute, San Antonio, TX, talk title: Metacomputing Systems and Tools, 1998.
56. Invited Talk, Southwest Research Institute, San Antonio, TX, talk title: Tools for Application Scheduling in Parallel and Distributed Systems, 1997.
57. Invited Talk, UT San Antonio Seminar, talk title: A Survey of Wide-Area File Systems: Issues, Challenges, and Solutions, 1997.
58. Invited Talk, National Cancer Institute, NIH, talk title: Legion: The Next Logical Step Toward a World-Wide Virtual Computer, 1995.
59. Invited Talk, Nasa Goddard, talk title: Legion: The Next Logical Step Toward a World-Wide Virtual Computer, 1995.
60. Invited Talk, Nasa Jet Propulsion Laboratory, talk title: Scheduling Parallel Computations in a Heterogeneous Environment, 1994.
61. Invited Talk, Nasa Jet Propulsion Laboratory, talk title: Multigranular Scheduling of Data Parallel Programs, Nasa Jet Propulsion Laboratory, 1993.

TEACHING

Courses Taught

University of Minnesota

- CSci 2021 Machine Organization and Architecture: 2011.
- CSci 8101 Advanced Operating Systems: 2011, 2008, 2007, 2002.
- CSci 4211 Introduction to Computer Networks: 2009.
- CSci 4061 Operating Systems: 2017, 2016, 2014, 2013, 2011, 2010, 2006, 2001.
- CSci 8980 Special Topics in Distributed Systems: 2015, 2012, 2004
- CSci 5103 Graduate Operating Systems: 2015, 2012, 2009, 2000-2005.
- CSci 5105 Introduction to Distributed Systems: 2014, 2013.
- CSci 4131/5131 Internet Programming: 2004, 2001.
- CSci 5980 Special Topics (Metacomputing): 2000.

STUDENT ADVISING

Ph.D:

- Mitch Terrel, area: Internet of Things, started Fall 2021, past WPE.
- Yixuan Wang, area: Security/Privacy, started Fall 2021.
- Nikhil Sreekumar, area: Edge Storage, started Fall 2019, past WPE.
- Zachary Leidall, area: Internet of Things, (WPE Spring 2019), left program.
- Albert Jonathan *Multi-Tenant Geo-Distributed Analytics*, 2019.
- Kwangsung Oh, *Exploiting Heterogeneous Resources in a Multi-cloud Environment*, 2019.
- Francis Liu, *Elastic Scheduling in HPC Resource Management Systems*, 2019.
- Zachary Leidall, area: Internet of Things, (WPE Spring 2019).
- Jinoh Kim, Ph.D. 2010, *Data Dissemination for Distributed Computing*
Post-doc Lawrence Berkeley Research Lab,
Associate Professor Texas A&M
- Darin England, Ph.D. 2006, *Robust Design for Distributed Computing Systems*, Assistant Teaching Professor,
University of Minnesota.
- Byoung-Dai Lee, Ph.D. 2003, *Adaptive Middleware for High-End Network Services*
Assistant Professor at Kyonggi University, South Korea.

MS Plan A/B/C:

- Sumanth Kaushik Vishwanath

Graduated:

- Lei Huang, Spring 2021, Plan A
- Saurabh Gupta, Spring 2020, Plan B
- Soumya Agrawal, Spring 2020
- Aravind Alagiri Ramkumar, Spring 2019, Plan B
- Rohit Sinhu, Spring 2019, Plan B
- Albert Jonathan (co-advised Chandra), Fall 2016, Plan B
- Prashant Chaudhary (co-advised Chandra), Spring 2015, Plan B
- Ajaykrishna Raghavan (co-advised Chandra), Spring 2014, Plan B
- Mathew Ryden (co-advised Chandra), Spring 2013, Plan B
- Chenyu Wang (co-advised Chandra), Spring 2012, Plan A
- Atul Katiyar, Spring 2010, Plan B
- Siddarth Ramakrishnan, Spring 2010, Plan B
- Omer Jilani, Summer 2009, Plan A-Edinburgh
- Vasumathi Sundaram, Summer 2008, Plan B

- Krishnaveni Budati, Summer 2007, Plan B
- Rahul Trivedi, Fall 2006, Plan B
- Jayashree Sadagopan, Summer 2005, Plan B
- Jason Sonnek, Summer 2005, Plan B
- Anusha Iyer, Spring 2004, Plan B
- Murali Sangubhatla, Summer 2003, Plan B
- Zhaoxin Ding, Fall 2003, Plan B
- Lakshman Abburi, Fall 2003, Plan B
- Pramod Srinivasan, Summer 2001, Plan B

SERVICE

Professional Service

Leadership

- Steering Committee Chair, HPDC, 2010-to-date
- PC Vice-Chair, IPDPS, St. Petersburg Florida, 2023
- General Chair, HPDC, Minneapolis Minnesota, 2022
- General Chair, HPDC, Phoenix Arizona, 2019
- General Chair, HPDC, Washington D.C., 2017
- Co-Organizer The Science of Cyberinfrastructure: Research, Experience, Applications and Models Workshop, e-Science, 2016.
- Co-Organizer The Science of Cyberinfrastructure: Research, Experience, Applications and Models Workshop, HPDC, 2015.
- General Chair, HPDC, New York City, NY, 2013
- Local Chair, PACT, Minneapolis, MN, 2012
- Panel co-organizer and panelist, Cloud Computing, UMn Open House, 2011.
- Steering Committee Chair, HPDC, 2010-to-date.
- International Liaison Co-Chair, ICDCS, 2011.
- Co-Organizer Cloud panel, ICDCS, 2011.
- Co-Organizer Dynamic Distributed Data Programming Abstractions Workshop, HPDC, 2011.
- Member advisory board, NSF FutureGrid, 2010-to-date.
- Global Program Chair, EuroPar, 2009.
- Co-Organizer, NSF Workshop on Data Mining for Biomedical Applications, University of Florida, 2008.
- Co-Organizer, Distributed Programming Abstractions Workshop at E-Science Conference, 2008.
- Program Co-Chair, HPDC 2008.
- Session Chair, HPDC: 2007, 1999,
- Session Chair, GRID, 2004.
- Panel organizer, Intelligent Storage Workshop, UMn DTC, 2006.
- Publicity Chair, ICPP, 2004, 2000.
- Co-Organizer, Intelligent Storage Workshop, UMn DTC, 2004.
- Co-Organizer, Data Mining and the Grid Workshop, UMn MSI, 2003.
- Exhibits and Poster Chair, HPDC, 2003.
- Program Vice-Chair, ICPP, 2002.
- Tutorial Chair, HPDC: 2001, 2000.
- Session Chair, IPDPS 2000.

Program Committee Member

- The International ACM Workshop on Software-Defined Ecosystem (BigSystems), 2015.
- IEEE International Workshop on Cloud Analytics 2014.
- IEEE International Conference on Cloud Engineering (IC2E) 2016, 2015, 2014, 2013.
- IEEE/ACM International Symposium on High Performance Distributed Computing: 2016-1997.
- IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing: 2016, 2015, 2014, 2012, 2011, 2010, 2009, 2007, 2003.
- EuroPar 2014.
- 2013 IEEE International Conference on Big Data 2013, 2023.
- IEEE International Parallel and Distributed Processing Symposium 2012.
- SC 2012, 2011, 2010, 2009, 2008, 2007, 2006.
- Workshop on Job Scheduling Strategies For Parallel Processing: 2011, 2010, 2007, 2006.
- Workshop on Large-Scale, Volatile Desktop Grids: 2011, 2010, 2009, 2007.
- IEEE International Conference on Grid Computing: 2010, 2009, 2007, 2006, 2005, 2004, 2003, 2002.
- Cluster Computing and the Grid: 2011, 2010, 2009, 2007, 2003.
- Challenges of Large Applications in Distributed Environments: 2010, 2009, 2006, 2005, 2004, 2003.
- International Conference on Parallel Processing: 2009, 2005, 2004, 2003.

- International Conference on Distributed Computing Systems: 2008.
- IEEE International Conference on e-Science and Grid Technologies: 2005.
- IEEE International Conference on Services Computing: 2004.
- Heterogeneous Computing Workshop: 2004, 2003, 2002, 2001, 2000, 1999.
- International Workshop on Active Middleare Services (Autonomic Computing): 2004, 2003, 2002.
- International Workshop on Heterogeneous and Adaptive Computing: 2003.
- International Conference on High Performance Computing: 2002.

Editorships

- Editor HPDC 1992-2012 special issue
- Editorial board: IEEE Transactions on Computers.
- Editorial board: IEEE Transactions on Parallel and Distributed Systems.
- Editorial board: Journal of Parallel and Distributed Computing.
- Special issue editor for Grid Computing, Journal of Parallel and Distributed Computing.
- Special issue editor for Grid and P2P computing, ACM Sigmetrics newsletter, Performance Evaluation Review.
- Technical editor of IEEE TPDS newsletter.