

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

DATABRICKS, INC.
Petitioner

v.

BYTEWEAVR, LLC
Patent Owner.

Inter Partes Review No. IPR2025-00716
Patent 8,275,827 B2

**DECLARATION OF JOHN D. KUBIATOWICZ, PH.D.
REGARDING U.S. PATENT NO. 8,275,827**

TABLE OF CONTENTS

I.	BACKGROUND	4
II.	LEGAL PRINCIPLES	9
III.	PERSON OF ORDINARY SKILL IN THE ART	10
IV.	OVERVIEW OF THE ‘827 PATENT	11
	A. Purported Invention of the ‘827 Patent	11
V.	SUMMARY OF THE PROSECUTION HISTORY	18
VI.	CLAIM CONSTRUCTION	21
	A. “NAS device” and “dedicated NAS device” (Claims 2 & 14)	22
	B. “stand-alone dedicated NAS device” (Claims 2 & 14)	23
	C. “client agent” (all claims)	25
VII.	THE CHALLENGED CLAIMS ARE UNPATENTABLE	25
	A. Overview of Prior Art	26
	B. Ground 1: Claims 2, 10-12, 14, and 22-24 are rendered obvious by Carter	34
	i. Overview of Carter	35
	ii. Claim 1	53
	iii. Claim 13	74
	iv. Claims 2 and 14	77
	v. Claims 9 and 21, which claims 10-12 and 22-24 depend upon	87
	vi. Claim 10	89
	vii. Claim 11	92
	viii. Claim 12	95
	ix. Claims 22	97

x.	Claims 23-24.....	98
C.	Ground 2: Claims 2 and 14 are Obvious over OceanStore	98
i.	Overview of OceanStore.....	98
ii.	Claim 1.....	115
iii.	Claim 13.....	127
iv.	Claims 2 and 14	130
D.	Motivation to Combine Carter and Gibson	138
E.	Ground 3: Claims 2 and 14 are Obvious over Carter in view of Gibson	141
i.	Overview of Gibson.....	141
ii.	Claims 2 and 14	143
VIII.	AVAILABILITY FOR CROSS-EXAMINATION	148
IX.	RIGHT TO SUPPLEMENT	149

I declare as follows:

1. My name is John D. Kubiatowicz.

I. BACKGROUND

2. I received a Ph.D. in Electrical Engineering and Computer Science in 1998, an M.S. in Electrical Engineering and Computer Science in 1993, and two B.S. degrees, one in Electrical Engineering and the other in Physics in 1987, all from the Massachusetts Institute of Technology (MIT). While at MIT, I was the chief architect of the Alewife Multiprocessor, one of the first massively-parallel distributed multiprocessors to exploit both cache-coherent shared memory and message-passing in the same machine. As part of the Alewife research project, I helped to design the cache-coherence protocol and hardware for both the cache controller and Sparcle processor (a derivative of the Spare processor). I also designed aspects of the operating system that interacted directly with the hardware and shared cache. The design and implementation of Alewife became the focal-point of my PhD thesis.

3. Since receiving my doctorate, I have devoted my professional career to the research, design, development, study, and teaching of numerous aspects of computer systems and architecture. I have studied, taught, practiced, and researched in the field of computer science for over thirty years.

4. I am currently Full Professor of Computer Science at the University of California, Berkeley (UC Berkeley), where I have taught and supervised graduate students since 1998. I teach and have taught a number of courses at UC Berkeley, including advanced courses in computer architecture and operating systems for upper-level undergraduates and graduate students. I have also taught advanced graduate seminars about various computer science topics. In my graduate systems class, I teach about a variety of file systems and databases – in both local and distributed architectures. I also cover distributed networked systems in general, cloud-computing infrastructures, and edge-computing systems.

5. My research specialties include Internet-scale distributed storage systems, data caching and cache coherence, parallel operating systems, multiprocessor computer architecture, chip design, reconfigurable computing, and quantum computing. My research interests also include security, privacy, and resilience to faults and denial of service attacks in Internet-scale systems.

6. In addition to my experience in academia, I have over 25 years of industrial experience designing computer systems. For example, from 1987 to 1989, I worked with IBM on Project Athena, to produce a campus-wide distributed computing environment. From 1989-1993, I consulted from IBM on operating systems/device driver development, and from 2001-2003, worked with IBM on storage and automatic computing. From 1996-1998, I worked with a small

company called CLAM associates, to help develop cluster computing systems that automatically recover from node failures. From 2000-2001, I consulted with Sun Microsystems on RAS design techniques (specifically error correction codes for DRAM), and in 2006, I worked with Cisco Systems on network protocol design.

7. During the 1999-2018 timeframe, I was a frequent committee member for HotChips, an annual industrial conference where companies announce and discuss their latest computer chips-often for the first time. My duties involved soliciting and selecting talks for the conference; in 2006 and 2018, I was one of the chairs of the program committee.

8. I am an expert in the field of distributed storage systems, and have particular knowledge of the technologies most relevant to the '827 patent. I am well known in the field of distributed peer-to-peer storage systems as a result of OceanStore, a research project that explored mechanisms for providing secure access to storage on disparate computers spread around the Internet. I am the first author on the OceanStore paper. As a result of my research, I was selected as a top-50 researcher and policy maker by Scientific American in 2002 and as a “person to watch” for 2004 by US News & World-Report. During the early 2000s, I participated in many workshops devoted to peer-to-peer computing, storage, and networking, one of the most exemplary being the International Workshop on Peer-to-Peer Systems (IPTPS). I also taught a special seminar course for graduate

students on Peer-to-Peer systems. The expertise that I gained during this time supplemented with my work on Alewife (during graduate school) is directly relevant to the field of the '827 patent.

9. I am a member of a number of professional organizations, including the IEEE, ACM, and the American Association for the Advancement of Science (AAAS). I have authored or co-authored over 85 papers in peer-reviewed journals, conferences, and workshops in the field of computer science and engineering. In addition, I have been a referee for a number of first-tier academic conferences, including the International Symposium on Computer Architecture (ISCA), the Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS), the Symposium on High Performance Computer Architecture (HPCA), the Symposium on Operating Systems Design and Implementation (OSDI), and the Symposium on Operating Systems Principles (SOSP). I was a general chair of the IEEE Edge conference in 2019 and a program co-chair for the ACM/IEEE Symposium on Edge Computing in 2020. I was an editor for a special issue of the IEEE Micro Journal devoted to the best participants from HotChips 2006 as well as HotChips 2018. I have also served as a reviewer for proposals for the National Science Foundation (NSF).

10. My curriculum vitae, attached as Appendix A, includes a list of publications on which I am a named author. It contains further details regarding

my experience, education, publications, and other qualifications to render an expert opinion in connection with this proceeding.

11. I have reviewed the specification, claims, and file history of U.S. Patent No. 8,275,827 (“the ‘827 patent,” EX1001).

12. I have also reviewed the following references, which I understand to be prior art to the ‘827 patent:

- International Application WO 98/22881 to Carter *et al.*, titled “Remote Access in a Globally Addressable Storage Environment,” published on May 28, 1998 (“Carter,” EX1005).
- Kubiawicz, John et al., “OceanStore: An Architecture for Global-Scale Persistent Storage,” stamped November 20, 2000 by the University of Michigan, and appearing in ASPLOS-IX proceedings: Ninth International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, Mass., Nov. 12-15, 2000 (190-201) (“OceanStore”; EX1006)
- Garth A. Gibson et al., File Server Scaling with Network-Attached Secure Disks (“Gibson”; EX1007)

13. I have also reviewed the other exhibits and references cited in this Declaration.

14. I am being compensated at my normal consulting rate for my work.

15. My compensation is not dependent on and in no way affects the substance of my statements in this Declaration.

16. I have no financial interest in the Petitioners. I similarly have no financial interest in the '827 patent.

II. LEGAL PRINCIPLES

17. I have been informed that a claim is invalid as anticipated under pre-AIA 35 U.S.C. § 102(a) if “the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for patent.” I have also been informed that a claim is invalid as anticipated under 35 U.S.C. § 102(b) if “the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States.” Further I have been informed that a claim is invalid as anticipated under 35 U.S.C. § 102(e) if “the invention was described in ... an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent....” It is my understanding that for a claim to be anticipated, all of the limitations must be present in a single prior art reference, either expressly or inherently.

18. I have been informed that a claim is invalid as obvious under 35 U.S.C. § 103(a):

if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which [the] subject matter pertains.

19. I understand that a claimed invention would have been obvious, and therefore not patentable, if the subject matter claimed would have been considered obvious to a person of ordinary skill in the art at the time that the invention was made. I understand that when there are known elements that perform in known ways and produce predictable results, the combination of those elements is likely obvious. Further, I understand that when there is a predictable variation and a person would see the benefit of making that variation, implementing that predictable variation is likely not patentable. I have also been informed that obviousness does not require absolute predictability of success, but that what does matter is whether the prior art gives direction as to what parameters are critical and which of many possible choices may be successful.

III. PERSON OF ORDINARY SKILL IN THE ART

20. The '827 patent relates to the field of distributed processing. I have been informed by counsel that the Board in the previous IPRs found that, at the

time the ‘827 patent was filed, a person of ordinary skill in the art (“POSITA”) would have had at least a bachelor’s degree in computer science, computer engineering, electrical engineering, or equivalent, with 1–2 years of work experience in the field of computer networking, storage systems, or a related field, and that such a person would have been familiar with computer systems architecture and design, and would have been aware of design trends relating to distributed processing systems. IPR2017-00374, Paper 53 (EX1019) at 18; IPR2017-00439, Paper 50 (EX1020) at 18. I agree that this is the appropriate level of skill in the art to apply in this proceeding. Based on my experience above, I am able to assess how a POSITA would have understood the ‘827 patent and the various cited exhibits.

IV. OVERVIEW OF THE ‘827 PATENT

A. Purported Invention of the ‘827 Patent

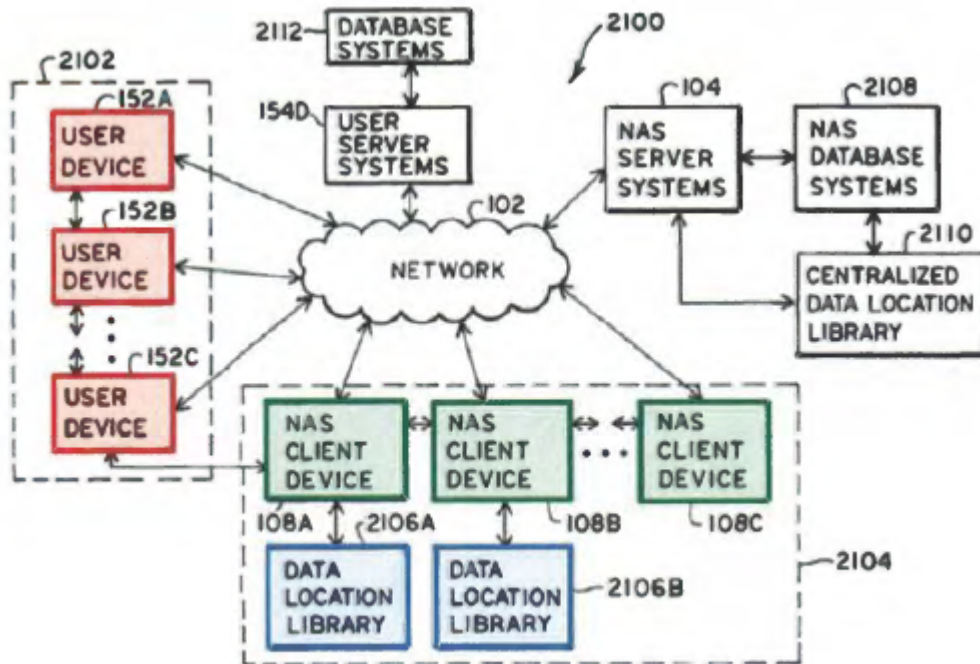
21. The ‘827 patent describes systems and methods for software-based network attached storage (NAS) services hosted on a massively-distributed network of devices. EX1001, Title, Abstract.

22. The Background describes an “NAS device” as generally referring to “a specialized file server that connects to the network to provide storage capacity to network-connected users.” EX1001, 2:18–20. But, in the context of the ‘827 patent, the Background explains that an “NAS device” “broadly refers to a device that makes data storage resources available to network-connected user devices”

(*id.*, 2:31–34). The Background further describes an NAS device as typically taking the form of a standalone dedicated hardware device that contains its “own storage, processing, connectivity, and management resources.” *Id.*, 2:12–14. The Background observes that, while ordinary desktop computers often sit idle for long periods of time, these computers have the same specifications (e.g., a Pentium processor, a hard drive, and an Ethernet connection) and often greater functional capabilities than dedicated NAS devices. *Id.*, 2:46-55.

23. In view of the foregoing, the ‘827 patent proposes ways in which these otherwise unused storage resources may be made available (as so-called NAS services) to other users connected to the network. *Id.*, 2:66-3:1, 42:36-41.

24. As will be clear from the description below, the ‘827 claims focus on subject matter added to the specification when the ‘827 application was filed (as a continuation-in-part application) on April 13, 2001. In particular, the ‘827 claims are concerned with the embodiments described in connection with and depicted in new Figure 21, which provides “a distributed processing system that provides network attached storage (NAS) services, according to the present invention.” *Id.*, 42:29-32. The entirety of this new disclosure is found at col. 42:29-46:10.



25. In Figure 21, shown annotated above, “the NAS embodiment 2100” includes a network 102 that connects user devices (depicted in red), and NAS client devices (depicted in green). *Id.*, 43:6-11. Each of the distributed NAS client devices 108A, 108B, and 108C runs a client agent (not shown) to provide NAS functionality. *Id.*, 43:26-30. Each of the user devices 152A, 152B, and 152C can then use the “software-based NAS resources” provided by the distributed computing system. *Id.*, 44:60-65.

26. The client devices 108A, 108B, and 108C are each configured to “mimic” dedicated NAS devices, even though “the software-based NAS devices 108A, 108B ... 108C are actually non-dedicated devices that rely on spare and/or unused capabilities of the existing devices to provide NAS services.” EX1001, 43:47-56; *see also* 44:30-32 and 5:64-6:2. These devices can be personal

computers systems (PCs), internet appliances, notebook computers, servers and the like. Note that the specification is very light on details about what exactly constitutes “mimicking” a dedicated NAS device. It does not place requirements on the type of protocol¹ supported by a single node nor does it specify how multiple NAS components should work with one another. Instead, the specification claims that “‘NAS device’ broadly refers to a device that makes data storage resources available to network-connected user devices.” *Id.*, 2:31-34.

27. Some of the distributed devices providing software-based NAS services can provide “both storage and data location functionality, thereby acting both to provide data storage services and distributed data location services.” *Id.*, 44:9-12. For example, the data location libraries 2106 (depicted in blue in Fig. 21) may provide links or directories to where desired data is stored among the NAS client devices. *Id.*, 43:59-62, 44:1-3. Once again, the specification provides little in the way of specific detail for what constitutes distributed data location services.

¹ By “protocol” I mean the format and sequencing of network messages required to access storage on a mimicked NAS device (e.g. the Network File System (NFS) protocol). The ‘827 specification provides no restrictions on the network protocols that a “NAS device” must use.

28. The NAS client devices can also be combined with NAS server systems 104 which may include a file storage database system 2108 and a centralized data location library 2110, which can provide services for the “files and other data being stored on the multitude of software-based NAS devices.” *Id.*, 44:27-37.

29. The software architecture for providing NAS functionality, as described in the specification, is extremely light on details but straightforward. It relies on the notion of “client agent.” There is little detail provided about client agents with regard to NAS functionality. Instead, much of the functionality declared for “client agents” seems to be left over from previous filings. *See generally* EX1002 (prosecution history). Two figures, namely Fig. 15 and Fig. 20, give some notion of what a client agent should contain. To perform its activities, the client agent communicates over the network (shown as the “communication interface” 1508 in Fig. 15). Three components within the client agent include the “core agent” (1502), “project component” (1504), and “user interface component” (1510). The “core agent” provides execution environment for the project and user interface components: “In effect, the core agent component 1502 can provide a backbone upon which is attached the project component 1504, the user interface component 1510, and any other desired component.” EX1001, 41:54-57; *see also id.*, 33:19-21 (“The task module 1512 and the work unit 1514 make up the separate

project or workload component 1504. The task module 1512 operates on top of the core agent module 1502 to provide processing of each work unit 1514.”).

30. The user interface component is not well described but seems to be code that interacts with users on the client device: “And the service provider is represented as providing at least a portion of the interface development and code 2010 for the user interface component 1510.” *Id.*, 42:5-7; *see also id.*, 42:18-28. This aspect of the embodiment of Fig. 15 and Fig. 20 does not seem to have been exploited by the NAS embodiment or claimed in the ‘827. As described above, it is also a separable component of the client agent architecture. Thus, the use of “client agent” in the ‘827 patent does not seem to require a user interface component.

31. Returning now to the ‘827 embodiments, the specification states that: “the client devices 108A, 108B ... 108C run a client agent, as described above, that provides NAS functionality, for example, by running a NAS component as one of the projects for the distributed device.” *Id.*, 43:26-30. We can now interpret this as saying that the “client agent” is providing a runtime environment (through a “core agent”) that is executing the NAS component. Since there is only one “project” and no “user interface component” in the ‘827 embodiments, much of the complexity of the client agent architecture is eliminated. Any system that could connect to the network and support the execution of a “NAS component” would meet this “client agent” limitation.

32. The specification also describes “a client agent” as “provid[ing] a protected data storage area accessible to outside devices, which is represented by the SD block within each client system in FIG. 14B.” EX1001, 31:46-49, Fig. 14B. The specification further explains that “[t]hese shared data (SD) blocks provide mechanisms that enable a wide variety of possible interactions among the client system.” EX1001, 31:54-56. “For example, “the data sharing mechanism may provide a space for a cache of other device addresses attached to the network” and “a simple indexing system that is automatically re-indexed when content is added or removed from the storage area,” among other functionalities. EX1001, 31:57-60, 31:60-62.

33. A POSITA would understand that any device that provided NAS service would have a software component that responded to network requests for storage (by definition). Such a software component would be running in an environment that invoked it on arrival of appropriate messages. For instance, UNIX and Linux operating systems typically launch daemons at boot time (such as “nfsd” that handles NAS requests using the NFS protocol). These daemons register to receive certain types of messages when they start. From that point on, the operating system will direct appropriate incoming messages to that daemon. Consequently, this software component would meet the limitation of a “NAS component,” while the environment would meet the limitation of “core agent.”

34. The specification also explains that “[i]t is [] advantageous to facilitate the full use of available storage space.” EX1001, 45:2-3. “To help achieve and ensure these objectives, the NAS implementation may undertake a number of controls to minimize interference with user storage operations while maximizing the storage value of the software-based NAS solution.” *Id.*, 45:3-7. “These controls may include user or server controlled management features and may include, for example, user specified constraints, standard bit, block or file priority levels and direct bit or file priority markings.” *Id.*, 45:7-11. For example, the ’827 patent’s disclosed “user specified constraints” may set storage priority controls, such as providing “a maximum allowed percentage for use of existing storage space,” prioritizing files that are more frequently accessed over those that are less frequently accessed, and setting “quality of service.” EX1001, 45:2-17; 45:18-67.

V. SUMMARY OF THE PROSECUTION HISTORY

35. I have been informed that the ’827 patent issued from U.S. Patent Appl. No. 09/834,785, which was filed on April 13, 2001, and is a child of fourteen prior continuations-in-part, each of which added new subject matter. *See* EX1001, Cover.

36. I have reviewed these filings and note the following:

37. I have been informed that the March 30, 2000 applications (IPR2017-00439, EX1121-EX1127) had disclosures through Figure 8; June 23, 2000 applications added disclosures through Figure 15 (IPR2017-00439, EX1128-EX1132); an August 25, 2000 application added disclosures through Figure 18b; and a February 27, 2001 application added disclosures through Figure 20 (IPR2017-00439, EX1134). It was not until April 13, 2001 that the ‘785 application (which issued as the ‘827 patent) added disclosure of Figure 21 and its description. IPR2017-00439, EX1106, 59-65, 104. Thus, the earliest application in which Figure 21 of the ‘827 patent or the accompanying text at col. 42, line 29 through col. 44, line 65 (in particular the discussion of “mimicking dedicated stand-alone NAS devices” at 43:49-50 and “location distributed devices” at 43:30-33, 43:59-63) appear was filed in April 13, 2001.

38. The ‘785 application (which issued as the ‘827 patent) initially sought claims directed to “configuring a client agent program to have a software-based NAS component and to take advantage of unused or under-utilized resources of a plurality of network-connected distributed devices put into operation for purposes distinct from operating the client agent program to provide NAS services.” (IPR2017-00439, EX1104, claim 1.) The Applicant pointed to pages 59-65 of the specification and relative to Figure 21 for the support. EX1002, 287.

39. The Patent Office properly recognized that this subject matter was not patentable, and rejected these claims and various versions thereof several times. *See generally* EX1002, 254-278, 293-303, 321-325, 345-357 (office actions).

40. I have also been informed that during prosecution, the Applicant confirmed that “[s]ince the client agent programs configure the devices to mimic NAS devices by relying on spare and/or unused capabilities of the existing devices, then it follows that the client agent programs must be capable of assessing what capabilities are spare and/or unused (unused or under utilized).” EX1002, 358 (emphasis in original). Pointing to Figure 21 and the Specification (page 61, lines 13-21), the Applicant asserted that “even though the word ‘assess’ was not specifically recited in this passage, it would have been obvious to one of ordinary skill in the art that the client agent programs must assess unused or under utilized capabilities for it to be able to configure the devices to mimic stand-alone NAS devices using their spare and/or unused capabilities.” *Id.*

41. Further, I have been informed that the independent claims stood rejected until the Applicant amended them to recite a limitation concerning a “location distributed device,” specifically: “enabling at least one of the selected distributed devices to function as a location distributed device to store location information associated with data stored by the selected distributed devices through

use of the respective client agents for the particular distributed device.” EX1002, 904 (claim amendment).

42. Both the specification and the prosecution history thus confirm that the claims are entitled to a priority date no earlier than the filing date of April 13, 2001. The Applicant cited to the new matter for support for their claims. The April 2001 specification is the first in the family to contain the subject matter concerning “mimicking” NAS devices and “location distributed devices” discussed in connection with new Figure 21 and during prosecution. The earlier disclosures did not properly support the issued claims. Accordingly, the claims are entitled to a priority date no earlier than the filing date of April 13, 2001.

VI. CLAIM CONSTRUCTION

43. I have been informed and understand that for purposes of this proceeding, the claim terms subject to *inter partes* review are to be “construed using the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. § 282(b), including construing the claim in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” 37 C.F.R. § 42.100(b). For purposes of this Petition, the challenged claims are interpreted according to their plain and ordinary meaning. I have reviewed the constructions proposed by Petitioner in the Petition for *Inter Partes*

Review, which I understand correspond to the Board's findings in the prior IPRs, IPR2017-00374 and IPR2017-00439. It is my opinion that Petitioner's proposals are correct. I have used Petitioner's proposals in the analysis that follows, and it is my opinion that the claims are invalid under each construction. Every other term should be considered under its ordinary and customary meaning in light of the specification, as commonly understood by those of ordinary skill in the art.

A. "NAS device" and "dedicated NAS device" (Claims 2 & 14)

44. While the Boards in the prior IPRs did not construe the terms "NAS device" and "dedicated NAS device," they explained in their Final Written Decisions that a "'NAS device' as defined in the '827 patent 'broadly refers to a device that makes data storage resources available to network-connected user devices" and a "'dedicated NAS device' is a device whose primary operational purpose is for providing NAS services.'" IPR2017-00439, Paper 50 (EX1020) at 36; IPR2017-00374, Paper 53 (EX1019) at 32-33. I understand that in co-pending litigation against Petitioner, Patent Owner and Petitioner agree that "NAS device" be construed as "a device that makes data storage resources available to network-connected user devices" and "dedicated NAS device" be construed as "a device whose operational purpose is for providing NAS services." EX1015 (Dkt. 55) at 1. A POSITA would have understood that the plain and ordinary meanings of "NAS device" and "dedicated NAS device" are consistent with the Boards' interpretations

of these terms in the prior IPRs and the agreed constructions of the parties in the co-pending litigation.

B. “stand-alone dedicated NAS device” (Claims 2 & 14)

45. In prior IPRs, the Boards did not directly construe “stand-alone dedicated NAS device” but, while considering Patent Owners’ proposed constructions for the “representing . . .” term of claim 1, the Boards suggested that “stand-alone dedicated NAS device” means a “distributed device that separately appears to network-connected devices as a dedicated NAS device.” In particular, the Patent Owners advanced a substantively identical construction—that “each selected distributed device [] separately appear to network-connected user devices as a dedicated NAS device”—in an attempt to limit the “representing . . .” term of claim 1 to the ’827 patent’s “stand-alone dedicated NAS device” implementation. IPR2017-00374, Paper 53 (EX1019) at 9-14; IPR2017-00439, Paper 50 (EX1020) at 9-14. Patent Owner argued that its proposed construction was correct based on the patent’s stand-alone dedicated NAS device implementation, described below, in which “each individual NAS device ‘brings its resources to the network with the appearance of a dedicated NAS device.’” IPR2017-00374, Paper 53 (EX1019) at 13; IPR2017-00374, Paper 21 at 10; see also IPR2017-00439, Paper 50 (EX1020) at 13. And the Board understood Patent Owner’s construction as referring to a stand-alone dedicated NAS device. *Id.*

46. To provide some context, the '827 patent specification discloses “three different NAS architectures...: a stand-alone only NAS device implementation, a NAS device fabric implementation, and a server assisted implementation.” EX1001, 43:37-40. The '827 patent discloses that in the stand-alone dedicated NAS device implementation, distributed devices separately appear as dedicated NAS devices. EX1001, 43:44-50. Specifically, the '827 patent discloses that a “large number of desktop PCs in the enterprise (Intranet) or on the Internet can each act as a NAS device by . . . bring[ing] its resources to the network with the appearance of a dedicated NAS device,” and thus, “mimic the functionality of stand-alone NAS devices.” EX 1001, 42:49-43:1; *id.* 43:44-50 (discussing “NAS devices . . . mimicking dedicated stand-alone NAS devices”). And because these NAS devices “each act as” or separately “appear as a single or dedicated NAS device” to network-connected devices “user device[s] . . . can directly access them as such.” EX1001, 42:59-65; 43:44-47. The patent further explains why a “stand-alone dedicated NAS device” separately appears as a dedicated NAS device. While these devices “operate[] like a standard dedicated NAS device” they are actually “software-based NAS devices that rely on spare and/or unused capabilities of the existing devices to provide [the stand-alone dedicated] NAS services.” *Id.* at 43:50-58.

47. I understand that in co-pending litigation against Petitioner, Patent Owner proposes that “stand-alone dedicated NAS device” be construed as “a dedicated NAS device that contains its own storage, processing, connectivity, and management resources” while Petitioner proposes it be construed as “a distributed device that separately appears to network connected user devices as a dedicated NAS device.” EX1016 (Dkt. 55-1) at 8; EX1017 (Dkt. 55-2) at 4. In my analysis, I have applied both proposed constructions and, as set forth below, it is my opinion that the prior art invalidates under either construction.

C. “client agent” (all claims)

48. I understand that in co-pending litigation against Petitioner, Patent Owner proposes that “client agent” be construed as “a program that manages the workloads and processes of a distributed processing system” while Petitioner proposes it be construed as “a software program configured to run on each distributed device to provide what appears to client devices as dedicated NAS functionality.” EX1016 (Dkt. 55-1) at 7; EX1017 (Dkt. 55-2) at 5. I have applied both constructions and, as set forth below, it is my opinion that the prior art invalidates under either construction.

VII. THE CHALLENGED CLAIMS ARE UNPATENTABLE

49. I have applied the above-discussed claim construction standard in my analysis below. However, based on my reading of the ‘827 patent’s specification

and the plain and ordinary meanings of the claim terms, the prior art teaches each claim limitation under any reasonable construction of the claim terms.

50. The challenged claims recite and claim features that were known in the art prior to the critical date, and are obvious in view of the prior art.

A. Overview of Prior Art

51. As explained above, the ‘827 patent concerns a plurality of distributed devices, which determine the amount of their unused or under-utilized storage, and which represent themselves as NAS devices having such unused storage available for other devices to use. However, there was absolutely nothing new about either this general idea or the specific techniques used in the ‘827 patent. These kinds of systems and the opportunities they seek to exploit were known years before the ‘827 patent.

52. It began with the 1980s, which ushered in a rapid push to exploit networking, away from the traditional “mainframe” environment (in which all resources were centrally located) toward widely distributed computing and storage (in which resources were geographically distributed). Academic research projects such as Project Athena² (initiated in 1983 at the Massachusetts Institute of

² EX1014.

Technology) and Project Andrew³ (initiated in 1982 at Carnegie Mellon University), as well as companies such as Sun Microsystems paved the way for distributed computing environments by developing remote windowing systems (X10 and X11), remote storage systems (“NFS”, and “AFS”), and remote authentication systems (“Kerberos”). As discussed in the Project Athena paper, both Athena and Andrew had similar scaling goals: “Both systems [Andrew and Athena] support on the order of 1,000 midrange workstations in a Unix and Ethernet environment, with plans to scale to 10,000 workstations.” EX1014, 42. Both systems were targeted at improving the educational experience through distributed computing.

53. As a specific example, the Andrew File System (AFS) was a distributed file system that grew out of Project Andrew to support its distributed goals. This first version of AFS, called Vice (the implementation), was described in 1985 by Satyanarayanan in a paper, entitled “The ITC Distributed File System: Principles and Design.” (EX1009.) This file system provides a shared namespace which holds files spread throughout the system. In principle, any file in the shared space may be accessed from anywhere. Further, each workstation has local storage that is unique to that workstation and reserved for private use. EX1009, Fig. 3-2.

³ EX1008.

54. In Satyanarayanan's proposal for implementation (depicted below in Figure 2-2 of the paper), files are stored and distributed throughout a network on multiple "cluster servers."

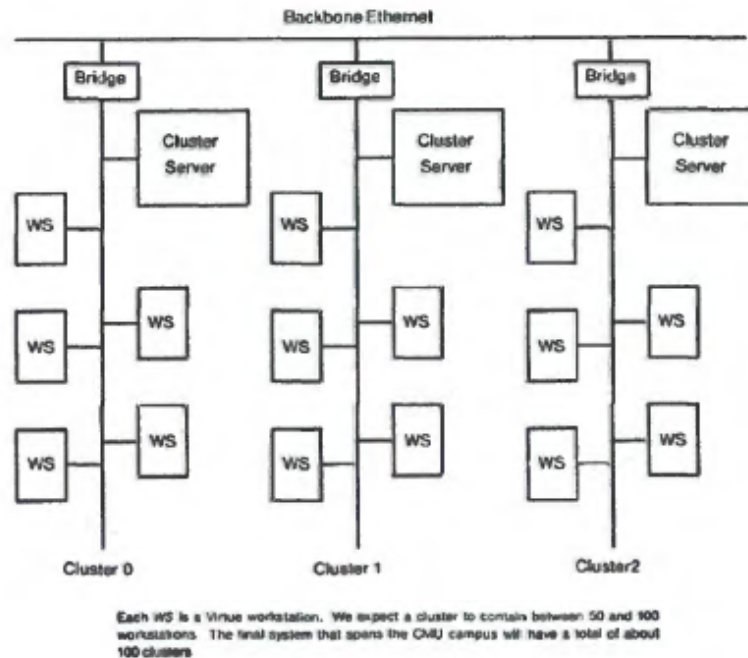


Figure 2-2: Vice Topology

55. EX1009, 38. Users (sitting at workstations (WS)) may access any file in the shared space. Consequently, the space of storage on all servers forms a large, distributed storage system.

56. For performance, files are cached extensively throughout the network: "Part of the disk on each workstation is used to store local files, while the rest is used as a cache of files in Vice." EX1009, 41. Consequently, Vice uses excess storage on each workstation for caching. A user on a workstation requests files using normal UNIX naming conventions. Local file requests are served by the local disk, while requests to the shared space first check the local cache, then send

requests to a local, nearby cluster server, which may or may not have the file. The storage server determines if it has the file, and if it does it returns the file in response; if not, the server consults its stored location information to return the identity of the appropriate server. The workstation will continue by contacting the new server.

57. In this way, the storage servers contain not only the file storage but location information to direct file requests accordingly, if necessary. EX1009, 40-41 (“If a server receives a request for which it is not the custodian, it will respond with the identity of the appropriate custodian.”).

58. The second version of the Andrew File System (called Vice-II in the Andrew paper (EX1008)) incorporated a number of improvements discussed in the Andrew paper. AFS became sufficiently stable that project Athena eventually utilized it for all of its storage. As reported in the Athena retrospective paper, “A distributed file system call the Andrew File System (AFS), developed as part of Andrew, is now being used as the basis of a nationwide file system experiment. The AFS file space is separated into two parts: local and shared space. The local space is accessible to the user (generally on local disk) but not publicly shared. The shared space can exist anywhere in the network and is publicly accessible.” EX1014, 42.

59. I would like to point out that the mechanisms developed in AFS were quite successful at sharing information across wide geographic areas and using caching to improve performance. As a staff-member at Project Athena (an IBM assignment), I had extensive experience with AFS. AFS illustrated that it would be possible to move away from the mainframe storage model of previous decades to a much more distributed storage model. It was one of the few research prototypes that graduated to a production system and was widely used. In fact, it was even picked up by IBM and morphed into a product called “DFS” (for Distributed File System).

60. With respect to the ‘827, I will note that AFS supported a managed storage model in which the local disk resources of each workstation was reserved for local users (either as local storage or as a cache on the shared address space that benefited local users).

61. In the same timeframe as Project Athena and Project Andrew, other lines of research sought to make it possible to share computation cycles spread across numerous workstations. One exemplary system was Condor, which dated back to at least 1987. Condor was famous for its approach of making otherwise unused, networked computer resources (i.e., so-called “idle cycles”) available for use by other network devices. EX1010, 1 (“In most cases, the resources of the workstation are under utilized. The processing demands of the owner are much

smaller than the capacity of the workstation he/she owns. However, very often some of the users face the problem that the capacity of their workstations is much too small to meet their processing demands. These users would like to take advantage of any available capacity they can access that can support their needs.”). In short, with Condor, network devices could “volunteer” their unused capacity to others connected to the network and run applications on their behalf (as a lower priority task) when the volunteer computers were otherwise idle. EX1010, 2 (“The Condor system schedules long running background jobs at idle workstations.”).

62. Figure 1 depicts the “Condor Scheduling Structure” in which a “central coordinator allocates capacity from idle workstations to local schedulers on workstations that have background jobs waiting.” EX1010, 6.

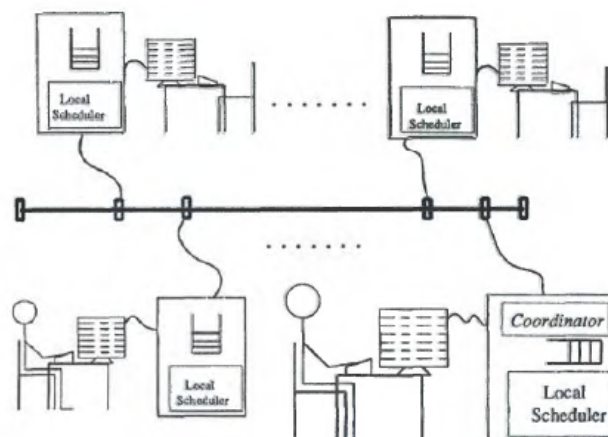


Figure 1: The Condor Scheduling Structure.

63. What is perhaps less well known about Condor is that the volunteer computers needed to assess not only their extra CPU capacity, but also their unused disk space to confirm they had sufficient storage resources to run these

applications and their corresponding check point files. EX1010, 13 (“A user has to devote some local capacity to support the placement and checkpointing of remote jobs and the execution of system calls.”); 16 (“The implementation of the Condor system brought a clearer understanding of several issues. Many of these issues relate to the nature of background jobs and the large amount of memory needed for their remote execution. For example, if a job is to be executed remotely, it must be placed on the remote station’s disk. Because users of workstations often do little to manage their own disk space, users let their disk become full. When a disk is full, a remote job cannot be placed on the workstation for remote execution. Even if a workstation is idle so that its processor is available for executing remote jobs, the disk might be full so that no remote job can execute there. The coordinator must know not only which workstation’s processor is available, but must know how much disk space is available on each workstation.”).

64. Consequently, the idea of computers having software based components to assess their unused computer resources, including storage, predates the ‘827 by well more than a decade.⁴

⁴ *Compare* EX1001, 2:50-56 (“Considering the specifications of an average corporate desktop computer, the capabilities of these desktop computers often exceed the capabilities [of] computers being sold as dedicated NAS devices,

65. More recent in time, several researchers specifically proposed making unused storage resources available to others - in a manner similar to how Condor made CPU resources available. For example, in Chapter 17 of the book “The Grid, Blueprint for a New Computing Infrastructure,” (1999) Andrew Chien wrote, “[t]he rapid decrease in the cost of storage has significant implications for computational grids. Not only do technology improvements make it possible and cost-effective to store large quantities of data, but the *exploding storage capability in lowest-cost systems dictates that huge amounts of storage will be available on networks of any size, and the improving capabilities of high-speed networks will*

and these desktop computers often sit idle for significant amounts of time each day.” *with* EX1010 (Condor), 1 (“users would like to take advantage of any available capacity” from “resources of the workstation[s] [that] are under utilized”) *with* EX1006 (OceanStore), 191 (“In a time when desktop workstations routinely ship with tens of gigabytes of spinning storage, the management of data is far more expensive than the media to store it on. OceanStore hopes to take advantage of this excess of storage space to make the management of data seamless and carefree.”) *with* EX1005, 4 (“the structured store of data is stored in an addressable shared memory space, which allows the nodes to transparently access portions of the structured store using standard memory access commands.”).

enable grids to use that storage flexibly. For example, in our local PC-based instructional laboratory at the University of Illinois, the file server is configured with 16 GB of disk. However, the 100 desktop machines associated with that server contain a far larger quantity-over 200 GB.” EX1011, 432-433 (emphasis added).

66. Similarly, James Plank, a researcher at the University of Tennessee, proposed an internet-based storage system they described as an “Internet Backplane Protocol” for controlling storage that is implemented as part of the network fabric itself. EX1012, Abstract. Using the Internet Backbone Protocol (IBP), “client calls may be made by *anyone* who can attach to an IBP server.” EX1012, Section 2, 3. These “IBP servers” (also called “IBP depots”) “require only storage and networking resources,” and “may be restricted to use only idle physical memory and disk resources.” EX1012, Section 2, 3.

67. Consequently, the idea of computers having software based components to assess their unused computer resources, including storage resources, and allocating those unused resources to other network devices, predates the ’827 patent by more than a decade.

B. Ground 1: Claims 2, 10-12, 14, and 22-24 are rendered obvious by Carter

68. International Application WO 98/22881 to Carter *et al.*, titled “Remote Access in a Globally Addressable Storage Environment,” published on

May 28, 1998 (“Carter,” EX1005), is a continuation-in-part of two U.S. Applications (U.S. ANos. 08/754,481 and 08/827,534), and explicitly incorporates the contents of both of these applications by reference. EX1005, 1.

69. I have been informed that even if the ‘827 patent were accorded the benefit of its earliest-claimed priority date, Carter is prior art to the ‘827 patent under 35 U.S.C. §§ 102(a) and (b). I have been further informed that Carter was not disclosed to the Patent Office during original prosecution of the ‘827 patent.

i. Overview of Carter

70. As explained in detail below, Carter describes a distributed storage system that allows a group of computers connected by a network to aggregate their unused storage resources together and share storage resources. Each computer in the group runs a software application that allocates a portion of the computer’s local storage resources to be included in the aggregated collection of storage resources. The software application both (a) responds to requests from other computers in the group that want to store/access the contents of that computer node’s local storage; and (b) sends requests to other computers in the group in order to store/access the contents of their portion of the shared storage. The Carter software presents a standard storage interface to the user of each computer. Specifically, to an end-user of one of the computers in the group, the aggregated, shared storage space looks like a standard computer hard drive (e.g., the “C:” drive

connected to a computer running Windows). However, the data stored on this “drive” actually is scattered throughout the storage resources of the computers in the group, and managed “behind the scenes” by software applications running on each computer.

71. More specifically, Carter concerns a distributed storage system in which a group of computers (also called “nodes”) are interconnected via local- and/or wide-area networks. *See* EX1005, Abstract. Each node is a conventional computer, including, *e.g.*, “a processor 30a.” EX1005, 10. Further, each node is equipped to run a software package taught by Carter that includes various software components, including “a data control program 32a, and a shared memory subsystem 34a.” EX1005, 10; *see also id.*, Fig. 1; *see also id.*, 28 (“[t]he depicted shared memory subsystem 220 is a **software module**”). The data control program provides an “interfac[e] [for] a shared memory subsystem 34a” to “access and store data within the addressable shared memory space 20.” EX1005, 10, Fig. 1. Each of the nodes runs this software to manage a structured store of data that is stored in an addressable shared memory space. *See* EX1005, 9 (“Each of the nodes includes at least a data control program which accesses and manages the structured store of data.”). In particular, the software taught by Carter manages a portion of each node’s available storage space (*e.g.*, unused hard drive space) and communicates with other nodes over the network in order to combine each node’s

unused space together to create a shared storage space that is accessible and usable by each of the participating computers. *See* EX1005, 9 (“The structured store of data may be stored in an ***addressable shared memory*** or the structured store may be stored in a more traditional fashion. For example, ***each node*** may be responsible for storing a particular element or elements of the structured store of data.”) (emphasis added); *see also id.*, 24 (“In a system that distributes the storage across the memory devices of the network, the persistent memory device will be understood to include a plurality of local persistent memory devices that each couple to a respective one of the plural computers.”).

72. The unused storage space attached to each computer in the system is pooled and managed by the software taught by Carter so that it may be available for use and access by any node. *See* EX1005, 4 (“the structured store of data is stored in an addressable shared memory space, which allows the nodes to transparently access portions of the structured store using standard memory access commands.”) (emphasis added).

73. The shared memory space is logically divided into sections of fixed size (e.g., “4k bytes”), each of which is associated with a globally unique identifier (or “global address signal.”) *See* EX1005, 4 (“the engine or system can create or receive, a global address signal that represents a portion, for example 4k bytes, of the virtual memory space.”). Using the global address signal corresponding to a

particular block (what Carter refers to as a “4k page”), any node connected to the system can access the data stored in that block or write data to that block for later use. EX1005, 31-32 (“Each page can be located within the shared memory space which is designated by a global address signal for that page of memory. The system can receive address signals from an application program or, optionally, can include a global address generator that generates the address signals.”).

74. The global address signal is a *virtual* address, because it is “decoupled from (*i.e.*, unrelated to) the physical and virtual address spaces of the underlying computer hardware.” EX1005, 4. Using the teachings of Carter, the system is able to translate a global address signal into a physical address, which pinpoints where the relevant 4k page is physically located. *See* EX1005, 5 (“The system provides physical storage for each portion of the virtual space in use by mapping (*i.e.*, assigning) each such portion to a physical device such as RAM or a hard disk.”). By maintaining the mapping between global addresses, and physical locations, the systems taught by Carter are able to balance the storage workload among the participating nodes in the system, to avoid overtaxing a particular node or group of nodes. *See* EX1005, 5 (“In general, the mapping provides a level of indirection that facilitates data migration, fault-tolerant operation, and load balancing.”).

75. As shown in Fig. 1, the shared memory subsystem is the software component that is responsible for communicating with the other nodes over a

network. EX1005, Fig. 1; *see also id.*, 11, “FIG. 1 illustrates that shared memory subsystems provide the network nodes with access to an addressable shared memory space.” Together, the shared memory subsystems cooperate to manage the transfer and storage of data among the various physical storage devices that make up the distributed system. As Carter explains, the subsystems can include logic to assess the “availability of resources, such as printer capabilities, hard-disk space, available RAM and other such resources.” EX1005, 36-37. The shared memory subsystem may also include a “policy controller [that] can employ any of the suitable heuristics to direct the elements, such as the paging controller, disk directory manager, and other elements to dynamically distribute the available resources.” EX1005, 36-37.

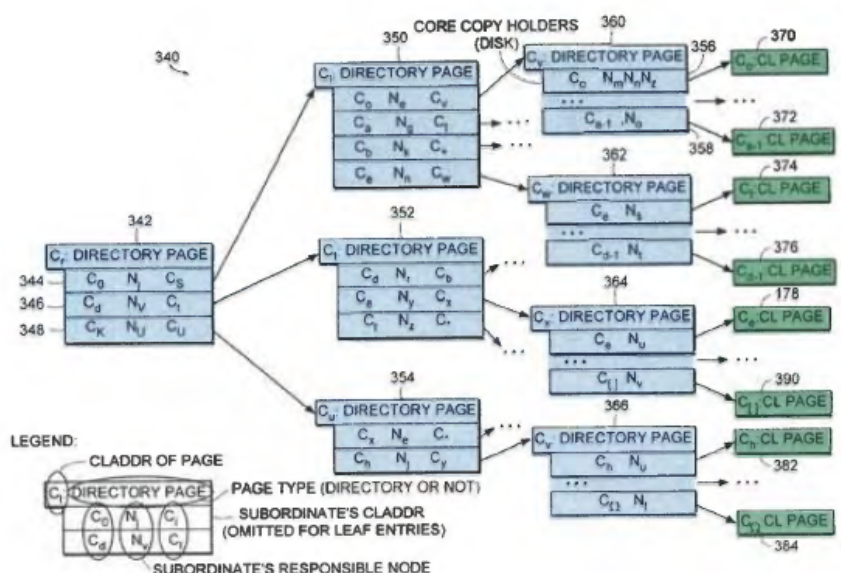
76. In Carter, a global address signal is translated into a physical address using a hierarchical structure of directory pages⁵⁷ that is stored *in* the shared memory system. EX1005, 38-40. This data structure is used to map a global address to the specific node providing the corresponding storage resources for that address. When the location of the data has been identified, the requesting node’s shared memory subsystem may access or modify its contents. *Id.*, 29.

⁵ This directory is for storage pages, and is separate from the file directory of the file system.

Specifically, Carter explains that “[e]ach node 212a-212c through the shared memory subsystem 220 can access each page 224a-224d stored in the virtual shared memory 222. Each page 224a-224d represents a unique entry of computer data stored within the virtual shared memory 222. Each page 224a- 224d is accessible to each one of the nodes 212a-212c, and alternatively, each node can store additional pages of data within the virtual shared memory 222. Each newly stored page of data can be accessible to each of the other nodes 212a-212c.”

EX1005, 29.

77. Fig. 10, reproduced below, illustrates an example of a hierarchical directory structure 340, as it exists at a particular point in time (the directory pages are shaded in blue):



Ex. 1140, Fig. 10 (annotated)

78. The directory structure 340 shown in Fig. 10 includes root directory page 342, first level directory pages 350, 352, and 354, and second level directory pages 360, 362, 364, and 366 in a hierarchical structure, where each subsequent level of directory pages subdivides the shared memory space into smaller sections. EX1005, 39-40.⁶ In this example, the root directory page 342 divides the entire shared memory space into three global address ranges: (1) the range of addresses between global address Co and global address Cd; (2) the range of addresses between global address Cd and global address CK; and (3) the range of addresses between global address CK and high highest global address of the address space.⁷ EX1005, 39. Information related to these three address ranges is stored in the corresponding respective directory entries 344, 346, and 348 within the directory page 342, respectively. EX1005, 39. In particular, each of the directory entries identifies the “responsible node” corresponding to the subentry (the variables N_j ,

⁶ A person of ordinary skill would understand that this hierarchical directory structure would contain more or less than three levels and more or less than the illustrated number of directory entries within each directory page. In fact, see discussion of Fig. 11, below.

⁷ Note that in the text of Carter, C_k is called “ C_g ,” following the notation used in an early version of Fig. 10.

N_v , and N_u shown in Fig. 10, above), and the global address of the subentry (the variables C_s , C_t , and C_u shown in Fig. 10, above).

79. The directory entries 344, 346, and 348 respectively point to a first level of “subordinate” directory pages 350, 352, and 354. Each of the first level directory pages further subdivides the address range associated with the corresponding directory entry. EX1005, 39. Specifically, in the example directory page structure of Fig. 10, each of the first level directory pages 350, 352, and 354 includes directory entries that point to a second level of “subordinate” pages 360, 362, 364, and 366 as shown in Fig. 10. EX1005, 39-40. The format of these directory entries is similar to the format of the directory entries included in the root directory page 342, described above. EX1005, 39-40.

80. The second level directory pages 360, 362, 364, and 366 are called “leaf entries,” because they represent the “leaves” of the tree of directory pages. The “leaf entries” identify, for each of the global addresses in the corresponding address range, “responsible nodes” for the virtual memory pages (shaded in green) at the given address. EX1005, 40.

81. The “responsible node stores information regarding the location and attributes of data associated with a respective global address.” EX1005, 32. The responsible node “optionally stores a copy of that page’s data” and “tracks ownership and responsibility” for the page’s data. EX1005, 32, 40. The node that

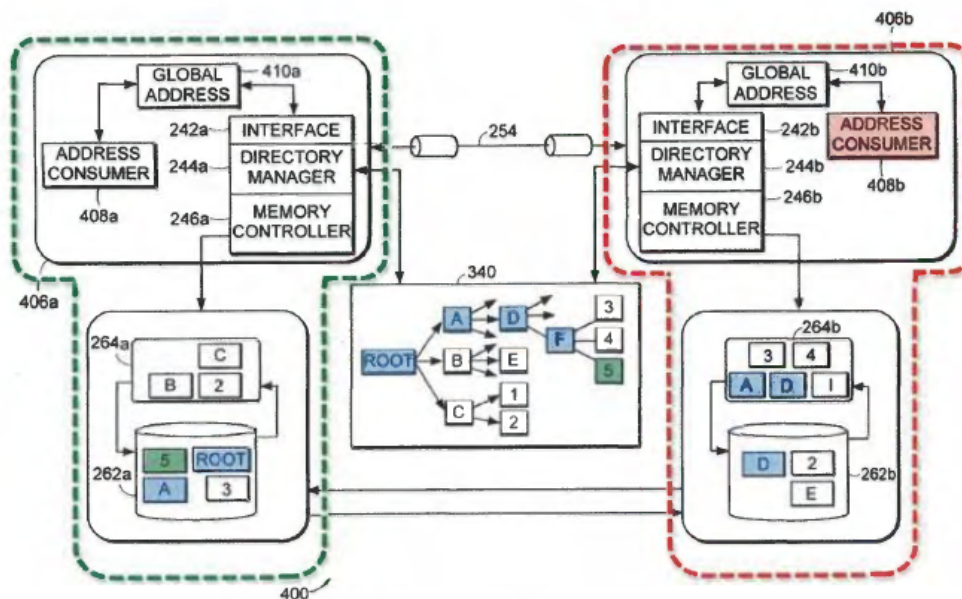
has write privileges to the page at that physical location is called an owner node. EX1005, 37. Thus, once the responsible node of the page of data being located is determined, the owner node (*i.e.*, the physical location of the page of the data being located) can also be determined. EX1005, 40.⁸ For example, “during a memory access operation, the memory subsystem [] passes to the responsible node . . . the address of the page being accessed” and “the shared memory system of that [responsible] node can identify a node that stores a copy of the page being accessed, including the owner node.” *Id.*, 40, 41.

82. The directory pages described above “are stored in the distributed address space and maintained by the distributed shared memory system.” EX1005, 41. A root, such as the directory page 342, is stored in a known location “to allow for bootstrap of the system.” EX1005, 41. Even without this explicit teaching, the need for a well-defined (node) location for the root directory would have been

⁸ Note that a person of skill would recognize the language and description of “cop[ies] of a page’s data,” “owner nodes”, “write privileges”, etc. as referring to page-based data coherence or distributed shared memory (DSM). In general, the embodiments described by Carter are straightforward/easy to interpret by one of ordinary skill.

obvious to one of ordinary skill in the art, since data could not be located without the initial information stored in the root directory 342.

83. Fig. 11, reproduced below, illustrates a snapshot in time for an implementation of a system 400 which includes such a hierarchical directory page structure 340, that is accessed by nodes 406a and 406b (colors will be described subsequently):



84. This single figure shows much of the NAS functionality of Carter. Nodes 406a and 406b each include “address consumers” 408a and 408b, respectively, each of which may be “an application program, file system, hardware device, or any other such element that requests access to the virtual memory,” as well as shared-memory subsystems consisting of “interfaces” (242a and 242b respectively), “directory managers” (244a and 244b respectively), and “memory controllers” (246a and 246b respectively). Ex 1005, 41) Note that these shared

memory components correspond to those presented in Fig. 7. EX1005, 33-34.

Address consumers “request an address, or range of addresses, and the directory manager can include a global address generator that provides the consumer with the requested address, or a pointer to the requested address.” EX1005, 41. Further, each node contains “volatile memory devices” (e.g. RAM), 264a and 264b respectively, and “persistent memory devices” (e.g. Disk), 262a and 262b respectively. EX1005, 41. Finally, box 340 depicts the distributed directory *implemented* by this system (recall discussion of Fig. 10, above); said directory is *not* stored on any one particular node, but rather distributed throughout the RAM and Disk storage of the various nodes in the system. EX1005, 41.

85. I have colored this figure to illustrate how one of skill in the art would understand the process of satisfying a request for page “5” from node 406b. A POSITA would understand that a request for block “5” could originate in a number of ways, such as from the “data control program” as part of a file-system access on node 406b. Target page “5” is colored green and is stored in the persistent storage (262a) of node 406a. Since page “5” is stored on the disk (262a) of node 406a, I have outlined this node in green, to indicate that it will serve as the NAS device for this scenario. Node 406b is outlined in red to indicate that it will serve as a client

device for this scenario.⁹ Further, client device (406b) has its “address consumer” (408b) colored red to indicate that the request in question starts by passing through this component.

86. In keeping with the discussion of Fig. 10, elements of the directory (340) leading to block “5” are colored blue. As the request for block “5” makes its way to the directory manager of node 406b (i.e. 244b), the directory manager will make a series of queries to the distributed directory in order to locate block “5” (EX1005, 32-33, Fig. 10), ultimately querying node 406a over the network connection (254) to retrieve block “5.” The shared-memory protocol that permits retrieval of block “5” easily satisfies the definition of “NAS” from the ‘827 patent, as discussed above.

87. For this particular example, directory manager 244b will first traverse the directory entries (ROOT → A → D → F) to discover which node is responsible for block “5.” Since the “ROOT” directory entry is on node 406a, in its Disk (262a), node 406b will request this entry over the network from 406a.¹⁰ Next,

⁹ Note that the same physical node can serve as both client and NAS device depending on the particular request that is flowing through the system.

¹⁰ As discussed previously, the location of the root data block must be known to all nodes in the system so that they can perform directory lookups for data; without

directory entries “A” and “D” will be discovered in the local cache (264b) and fetched quickly. Alternatively, “A” could be fetched from node 406a’s disk and “D” could be fetched from node 406b’s disk. Finally, directory entry “F” will be fetched over the network from some third node, not shown in the figure. Once entry “F” was fetched, it would become clear that node 406a should be queried to request the data: “[o]nce the directory manager 244 identifies which node is responsible for maintaining a particular address, the directory manager 244 can identify a node that stores information for locating a copy of the page, and make the call to the memory controller 246 of that node and pass to that node’s memory controller the memory request provided by the memory interface 242.” EX1005, 33. In this example, node 406a would serve as both “responsible node” and final storage repository for block “5.” Further, as clearly illustrated by this example, the shared memory subsystem running on node 406a is mimicking a NAS device by servicing storage requests for data on its local disk (*e.g.* directory block “ROOT” and data block “5”) over the network (254) initiated by node 406b (acting as a client of the NAS).

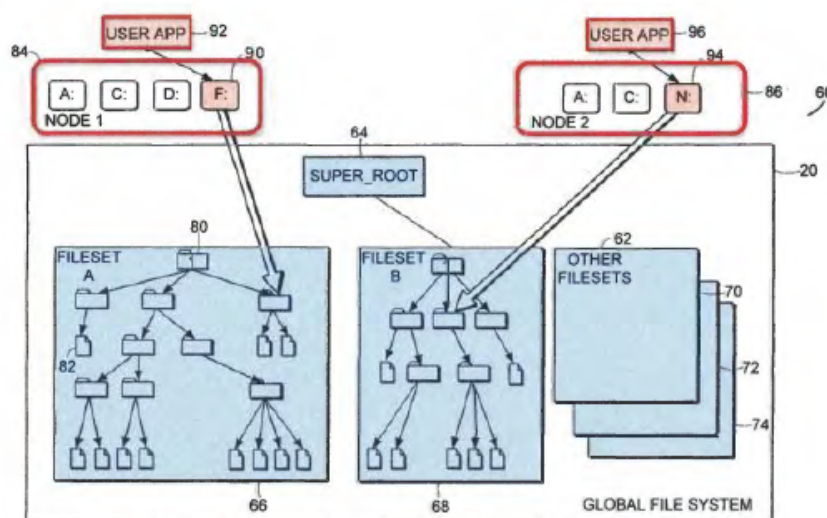
this information, the translation process could not occur. Thus, node 406a is serving as a centralized server for other nodes attempting to locate data.

88. Thus, the shared memory subsystem makes it possible for any one of the computers in the networked group to store/access a particular 4K page of memory in the shared memory space using its global address. Like the '827 patent, Carter accomplishes this by distributing location information among the computers in the group, independent of the location of the underlying data - indeed, as illustrated in Figs. 10 and 11 and by the above example, it is common for one node in the system to store information about the location of data that is stored on different node.

89. Further, as explained below, the “data control program,” another component of the software application taught by Carter, implements a filesystem structure that uses the page-level access to the shared memory space provided by the shared memory subsystem. This filesystem stores both the file data itself and also the metadata describing the structure and location of the files in the shared memory space. As further explained below, the software taught by Carter is able to mimic a standalone network hard drive; the distributed nature of the shared memory pool is transparently hidden from the user by the data control program, acting in conjunction with the shared memory subsystem. The filesystem provided by the “data control program” provides yet another level of “location information” that is distributed across the network. Just like shared memory subsystem uses distributed location information to translate global addresses into physical ones, the

data control program uses distributed location information to locate files and directories of files that are stored in the shared memory space.

90. For example, Figure 2, reproduced below, shows “a structured file system 60” that “employs the properties of the addressable shared memory space 20 to implement what looks to all network nodes like a coherent, single file system.” EX1005, 12. Thus, while the file system *looks* like a coherent filesystem that is stored on a single physical device, “in fact it spans all network nodes coupled to the addressable shared memory space 20.” EX1005, p.12.



Ex. 1140, Fig. 2 (annotated)

91. As shown above in Figure 2, the distributed addressable shared memory system 20 includes a “global file system” that is shared by all of the nodes on the network. EX1005, 13. The global file system is organized into multiple “filesets” (shaded in blue), which effectively partition the shared memory space into multiple independent file stores, just like partitions on a conventional hard

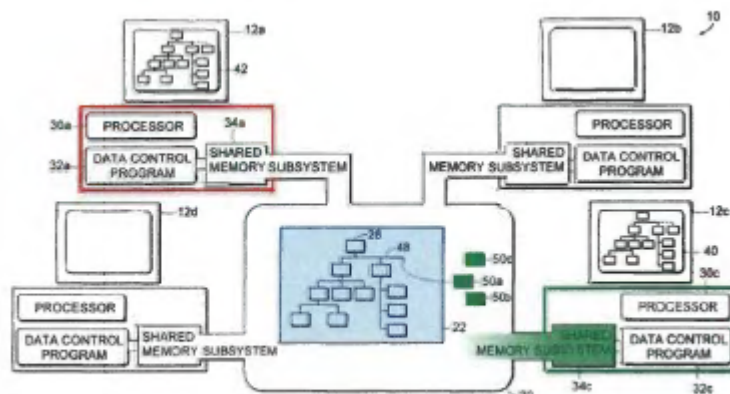
disk. *See* EX1005, 13 (“A fileset can be thought as logically equivalent to a traditional file system partition.”).

92. The location of the various filesets is stored in a structure called a “super root,” which “can be stored in a static and well-known memory location in the addressable shared memory space 20, and it can be accessed via a distributed shared memory system program interface.” EX1005, 14. To access a particular file in the filesystem, a node starts by accessing the super root structure, “to determine the identifier associated with the fileset, *e.g.*, the shared memory address used to access the fileset,” which provides the location of the root directory of the fileset containing the desired file. EX1005, 14. From the root directory, “it then can traverse the entire fileset directory tree to locate the desired file.” EX1005, 14.

93. As shown in the figure above, the shared memory space not only contains the file data itself, but also the metadata describing the structure and location of the files in the shared memory space formed by pooling the nodes’ collective storage resources. *See* EX1005, 12 (“[i]n contrast to known physical file systems which map a file organization onto disk blocks, the file system 60 manages the mapping of a directory and file structure onto a distributed addressable shared memory system 20”) (emphasis added). Thus, location information pertaining to the files in the filesystem is distributed across the nodes in the network.

94. To software applications running on the nodes of the distributed storage system, the filesystem that overlays the virtual shared storage space appears as a “logical drive,” just like a standard network attached storage device. As shown in Fig. 2, above, user application 92 (shaded in red) running on node 84 sees fileset 66 of the global file system as the “F:” drive (also shaded in red). Similarly, user application 96 running on node 86 sees fileset 68 of the global file system as the “N:” drive. EX1005, Fig. 2. Carter explains that these devices may act like “DOS devices” that connect to the shared file system using specialized filesystem drivers. EX1005, 13. However, the details of the underlying storage system are invisible to the user applications 92 and 96, and is even “transparent to the operating system” running on the nodes. EX1005, 30.

95. This transparency is achieved by means of a “data control program” that interfaces with the shared memory subsystem. EX1005, Fig. 1. Figure 1, reproduced below, shows the multilayer architecture of the Carter system.



Ex. 1140, Fig. 1 (annotated)

96. Figure 1 above shows four nodes connected to the distributed storage system of Carter. EX1005, 10. Each node includes a “data control program” the purpose of which is to “operate as a structured storage system, such as a file system, that is adapted to maintain a structured store of data and to employ the shared memory system as an addressable memory device that can store a structured store of data.” EX1005, 10. Specifically, the “data control program” on each node interfaces with the “shared memory subsystem” which is able to “access and store data within the addressable shared memory space 20.” EX1005, 10. For example, in response to a user application, data control program 32a (shaded in red) running on node 12a (outlined in red) might request data that is stored in data object 50c (shaded in green), which may be physically located on node 12c (outlined in green). In this case, the shared memory subsystem running on node 12c is responsible for locating and retrieving the data stored in block 50c and sending it to data control program 32a, which provides it to the requesting user application.

97. Thus, as Carter explains, the data control program and the shared memory subsystem are “cooperating elements”: the data control program provides a transparent interface to user applications running on the node, making the shared storage space appear to be a dedicated hard drive that is attached to the local network. EX1005, 10. To accomplish this, the data control program uses the lower-level capabilities of the shared memory subsystem, which has the ability to load

and store blocks of data from the shared pool of memory using global addresses, as explained above. EX1005, 10, 29. The shared memory subsystem receives global addresses from the data control program and translates them into physical locations using distributed location information, as explained above with reference to Figs. 10 and 11. Ultimately, when the physical location of the required memory page is located, the shared memory subsystem is responsible for retrieving data from and/or writing data to this physical location. EX1005, 29.

ii. Claim 1

98. As described below, Carter teaches every limitation of claim 1 of the ‘827 patent. I am informed by counsel that claims 1 and 13, which claims 2 and 14 depend upon, were invalidated over Carter in IPR2017-00439.

1(pre) “A computer-implemented method comprising ...”

99. I have been informed that this is a preamble term and does not limit the claim. Nevertheless, Carter teaches a “computer-implemented method.”

100. For example, Carter teaches an invention that “relates in general to distributed **computer** workgroups and, more specifically, to a globally addressable storage environment that accommodates remote access and two or more interconnected **computer** networks.” EX1005, Abstract (emphasis added).

101. Specifically, Carter explains that “the invention ... can be understood as **methods** for providing a **computer** system having a addressable shared memory space. The **method** can include the steps of providing a network for carrying data

signals representative of **computer** readable information, providing a hard-disk, coupled to the network, and having persistent storage for data signals, providing plural **computers**, each having an interface, coupled to the data network, for exchanging data signals between the plural **computers**, and assigning a portion of the addressable memory space to a portion of the persistent storage of the hard disk to provide addressable persistent storage for data signals.” EX1005, 27 (emphasis added).

1(a) “configuring a distributed processing system of a plurality of distributed devices coupled to a network, wherein the plurality of distributed devices include respective client agents configured to process respective portions of a workload for the distributed processing system”

102. Carter teaches this limitation. Carter teaches a “system that is intended to “provide computer network systems that have adaptable system configurations for dynamically exploiting distributed network resources.” EX1005, 4 (emphasis added). The distributed systems of Carter include “a plurality of network nodes (i.e., “a plurality of distributed devices coupled to a network”) that access a memory space storing a structured store of data, such as a structured file system or a database.” EX1005, 9 (emphasis added).

103. One of the primary goals of the teachings of Carter is to distribute processing and storage workloads more effectively. In particular, maintaining a shared memory space accessible by a plurality of devices “allows the networked computers to appear to have a single memory, and therefore can **allow application**

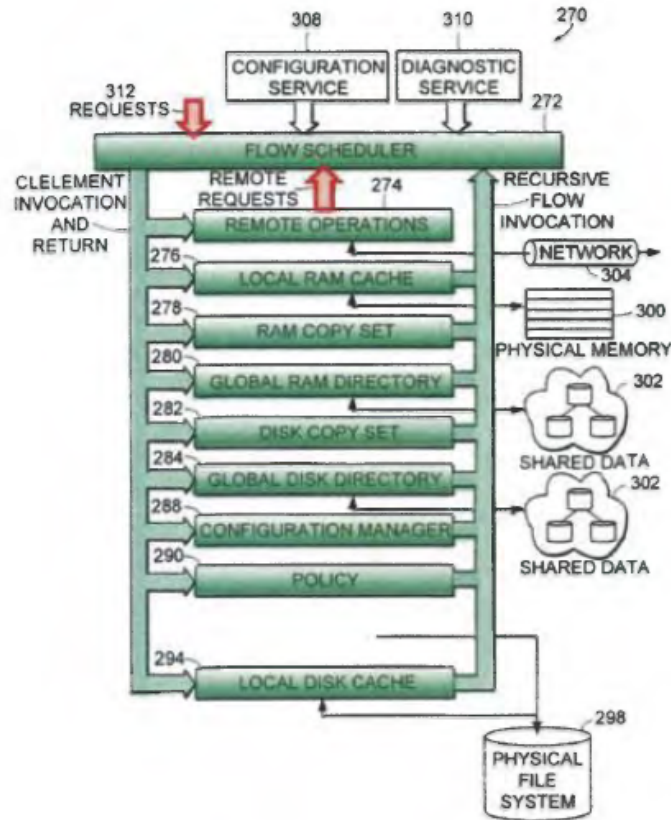
programs running on different computers to communicate using techniques *currently employed to communicate between applications running on the same machine.*” EX1005, 5 (emphasis added).

104. As explained above, the systems of Carter are implemented using a computer program that comprises various software components that run on each of the distributed devices in the system, including the “shared memory subsystem” and the “data control program” described above, as well as additional components including, among others, a “policy controller” or “policy element 290,” “distributor” component, a “disk directory manager,” a “migration controller,” and a flow scheduler. EX1005, 10, 24-25, Figs. 2, 8. Collectively, this software is a “client agent” for processing portions of a distributed workload as discussed above.

105. In particular, storage or access workloads are processed by the “shared memory subsystem,” which is a component of the client agent taught by Carter. *See* EX1005, Fig. 2, reproduced above. The shared memory subsystems running on the distributed nodes communicate with each other over the network, and cooperate to manage the transfer and storage of data among the various physical storage devices that make up the distributed system. They serve to export storage resources to client devices over the network and are “NAS components,” as explained in the analysis of the next limitation below.

106. The shared memory subsystem is responsible for processing storage workloads generated by the distributed devices. Specifically, Carter explains that “[e]ach node 212a-212c ***through the shared memory subsystem*** 220 can access each page 224a-224d stored in the virtual shared memory 222.” EX1005, 29, emphasis added. Each page 224a-224d represents a unique entry of computer data stored within the virtual shared memory 222 and thus, using the shared memory subsystem, each page 224a- 224d can be accessed or modified by each one of the nodes 212a-212c. EX1005, 29. Each node can also “store additional pages of data within the virtual shared memory 222,” ***and*** “[e]ach newly stored page of data can be accessible to each of the other nodes 212a-212c.” EX1005, 29. All of these storage and access capabilities are provided by the shared memory subsystem, which is a component of the software package described by Carter (*i.e.*, the “client agent”).

107. As explained above, the shared memory subsystem is a core component of the “client agent” provided by the Carter system as a whole. Carter illustrates an exemplary shared memory subsystem in Figure 8, reproduced below:



Ex. 1140, Fig. 8 (annotated)

108. As shown in the figure above, a shared memory subsystem includes a “flow scheduler” 272 that “can orchestrate the controls provided by an API of the subsystem 270.” EX1005, 35. The flow scheduler is a semi-autonomous software agent that routes and services requests (red arrows) from the distributed nodes. It operates as a “state machine that *monitors and responds to the requests 312 and remote requests through network 304* which can be instructions for memory operations and which can include signals representative of the global addresses being operated on. These memory operation requests 312 can act as op- codes for primitive operations on one or more global addresses. They can be read and write

requests, or other memory operations” that are part of a storage or access workload EX1005, 35.

109. In some of the embodiments taught by Carter, the storage and access capabilities of the shared memory subsystem are used by the “data control program,” another component of Carter’s software package, which uses the low-level storage services provided by the shared memory subsystem to maintain a “structured storage system, such as a file system, that is adapted to maintain a structured store of data,” which is provided to “external *clients*.” EX1005, 10, 35 (emphasis added). In this way, the software system described by Carter comprises various components that work in tandem to provide access to storage resources to other nodes on the network. These components include, among others, a shared memory subsystem that processes storage and access workloads, and a data control program that provides file-structured data to clients. Carter’s software package is a “client agent” under either of the co-pending litigation constructions (“a program that manages the workloads and processes of a distributed processing system” or “a software program configured to run on each distributed device to provide what appears to client devices as dedicated NAS functionality”).

110. Thus, for the reasons given above, the software that runs on each of the distributed nodes described in Carter is a “client agent configured to process respective portions of a workload for the distributed processing system.”

1(b) “wherein the respective client agents for particular distributed devices of the plurality of distributed devices have corresponding software-based network attached storage (NAS) components configured to assess unused or under- utilized storage resources in selected distributed devices of the plurality of distributed devices”

111. Carter teaches this limitation. Specifically, as explained below, the “client agent” software taught by Carter includes, as one of its components, the “shared memory subsystem” that functions as a “software-based NAS component” as required by the claim.

112. *First, the shared memory subsystem is a “NAS component”*: As explained above, the shared memory subsystem is a component of the larger “client agent” provided by Carter, which allows low-level access to a shared addressable memory. By communicating over the network with other nodes in the Carter system, the shared memory subsystem makes data storage resources available to network-connected user devices, and thus, it is a “NAS component.” EX1005, 39-42, Figs. 10, 11; *see also id.*, Fig. 7.

113. *Second, the shared memory subsystem is implemented in software*: Carter explains that “[t]he depicted shared memory subsystem 220 is a **software module** ...” EX1005, 28 (emphasis added). Thus, the “shared memory subsystem” is a “software-based NAS component,” as the claim requires.

114. *Third, the shared memory subsystem “assesses unused or under-utilized storage resources”*: Carter teaches that nodes may have an “optional

private memory device 218” that “provides the node with local storage that can be kept inaccessible to the other nodes on the network. Typically, the private memory device 218 includes a RAM, or a portion of a RAM memory, for temporarily storing data and application programs and for providing the processor 214 with memory storage for executing programs. The private memory device 18 can also include persistent memory storage, typically a hard disk unit or a portion of a hard disk unit, for the persistent storage of data.” EX1005, 28. Thus, Carter teaches systems that distinguish between “unused” storage space (*i.e.*, the storage space that is pooled to form the shared virtual address space) and “used” storage space (*i.e.*, private storage that is reserved for the use of a particular node).

115. Further, the shared memory subsystem is configured to “allow portions of the distributed shared memory to move between persistent and electronic memory depending on predetermined conditions...” Carter explains that “the memory subsystems 232a-232c can include a hierarchy manager that provides hierarchical control for the distribution of data. This includes controlling the migration controller, and policy controller ... to perform hierarchical data migration and load balancing, such that data migrates primarily between computers of the same group, and passes to other groups in hierarchical order. **Resource distribution** is similarly managed.” EX1005, 31 (emphasis added).

116. The shared memory subsystem may also include “a policy element
290 that can be a software module that acts as a controller to determine the
availability of resources, such as printer capabilities, hard-disk space, available
RAM and other such resources.” EX1005, 36. Specifically, the policy controller
evaluates the hard disk space on the local node, and evaluates storage throughout
the system by communicating with other nodes. Moreover, the policy element
determines the availability of resources by, for example, assessing whether or not
hard disk space of various types is available, what volume of space is available for
allocation, and using that information to quickly determine whether to proceed
with allocation processing. Additionally, the shared memory subsystem assesses
the extent to which storage resources are being utilized and uses load balancing
and resource distribution techniques to prevent underutilization. EX1005, 30.
Specifically, the “policy controller can employ any of the suitable heuristics to
direct the elements, such as the paging controller, disk directory manager, and
other elements to dynamically *distribute the available resources*.” and is
configured to “allow portions of the distributed shared memory to move between
persistent and electronic memory depending on predetermined conditions, such as
recent usage.” EX1005, 30, 36-37 (emphasis added).

117. Carter also has several other disclosures that clearly show the
“assessing” of unused or under-utilized storage. For example, Carter discloses

“*determin[ing]* the *free* volume space *available* for allocation.” EX1005, 19:14-15. Carter also discloses “request[ing] the *total* available space information and us[ing] this information to quickly determine whether to proceed with the allocation processing.” *Id.*, 19:15-17. Carter also reserves specific memory for local use, and assesses and makes available other specific memory (non-private) for other network-connected users. E.g., EX1005, 28:19-29:20 (contrasting “private memory” that provides “local storage that can be kept inaccessible to the other nodes” and “virtual shared memory” that “map[s] to devices that provide physical storage”). This approach parallels the ‘827 patent. EX1001, 12:26-31 (“The capability allocation may also be a system-wide (i.e., coarse) allocation . . . The capabilities allocation may also be sub-system specific (i.e., fine) allocation, such as allocation of particular subsystem capabilities to particular affiliations.”).

118. Both assessing and allocating are very old, well-known and basic functions and would not require significant description, a fact consistent with the ‘827 patent specifications utter lack of description for both functions. A POSITA would have appreciated that from the earliest computer systems – either standalone or distributed – the system would assess unused or under-utilized storage to determine what disk space was free and thus available to store new file information, i.e., no system would allow a file to be written without such consideration because the new file information might otherwise over-write (or

clobber) some legitimate other file information. Free space would be allocated, thereby reserving the disk space for such a file, i.e., so some later write operation would not over-write (or clobber) this legitimate file information. These are well known basic functions that any POSITA would fairly appreciate and immediately envisage from Carter's teachings about "determining" and "distributing" teachings, as well as its repeated, explicitly disclosures of allocating available memory.

1(c) "representing with the corresponding software-based NAS component that the selected distributed devices respectively comprise NAS devices having an available amount of storage resources related to the unused or under-utilized storage resources for the selected distributed devices"

119. Carter teaches this limitation. As the Board found in the prior IPR, Carter's "shared memory subsystem" is a software-based NAS component on each node that cooperates with the "shared memory subsystems" on other nodes to make unused or under-utilized physical data storage resources available to network-connected user devices. IPR2017-00439, Paper 50 (EX1020) at 37; see also EX1005, Figs. 10, 1. Further, these "shared memory subsystems" allow each node to represent, or mimic, a dedicated NAS device by making its unused or under-utilized storage available to other nodes. IPR2017-00439, Paper 50 (EX1020) at 36 (citing EX1005, 9:18-19, 9:26-27, 10:6-26, 10:30-11:2, Fig. 1); *see also, id.*, Fig. 8. For the following reasons, I agree with the Board's analysis.

120. The '827 patent defines an “NAS device” as ‘broadly refer[ing] to a device that makes data storage resources available to network-connected user devices” and a “dedicated NAS device” as a device whose primary operational purpose is for providing NAS services.’” EX1001, 2:31-34. Further, the '827 patent further discloses “an integrated system” referred to as a “NAS device fabric.” See EX1001, 42:36-43:5. Like the '827 patent, Carter discloses NAS devices and a similarly integrated NAS device system.

121. In particular, the step of “representing” as recited in the claim is performed by the shared memory subsystem (*i.e.*, the “software-based NAS component”) either individually or in aggregate. First, as explained above with reference to Figs. 10 and 11, by communicating over the network with other nodes in the Carter system, the shared memory subsystem makes data storage resources available to network-connected user devices, thus providing the functionality of a NAS device, as explained above.

122. Further, in providing access to data storage resources, the shared memory subsystem allows each node to *appear* to be (*i.e.*, “represent” or “mimic”) a dedicated NAS device, whose primary operational purpose is for providing NAS services, by making their unused or under-utilized storage resources available. EX1005, 10:6-26, Fig. 1. With reference to Figure 1, Carter discloses that “[e]ach of the nodes 12a-12d can access the addressable shared memory space 20 through

the shared memory subsystems 34a-34d.” EX1005, 10:6-26, 10:30–11:2, Fig. 1.

Carter explains that “[t]his allows the interconnected nodes 12a-12d to employ the addressable shared memory space 20 as a space for storing and retrieving data.”

Id., 10:5–6, Fig. 1; *see also, id.*, 9:18-19.

123. The Board also found that the shared memory subsystem (i.e., claimed “corresponding software-based NAS component”) “represent[s]...that the selected distributed devices respectively comprise NAS devices having an available amount of storage” through shared memory space 20. IPR2017-00439, Paper 50 (EX1020) at 36 (citing EX1005, 10:30-11:2, Fig. 1). Specifically, Carter explains that each node transparently provides a disk-like interface to external clients and user applications, while concealing the underlying details of the shared storage space behind a layer of abstraction. Carter states:

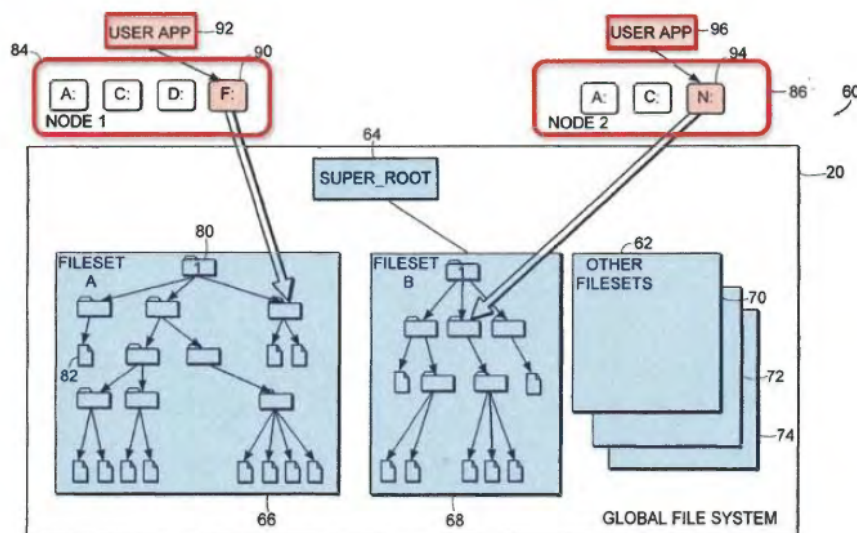
“The shared memory subsystem 240 depicted in FIG. 7 can encapsulate the memory management operations of the network node 212 to provide a virtual shared memory that can span across each node that connects into the network 254. Accordingly, each local node 212 views the network as a set of nodes that are each connected to a large shared computer memory.” EX1005, 31.

124. As this passage explains, because the shared memory subsystem hides the details of (or “encapsulates”) memory management from applications running on the local node, the local node “views” the shared storage space provided by

Carter as a single large storage device, rather than the aggregated storage resources of an entire network of participating computers.

125. For example, as explained above with reference to Figs. 10 and 11, the shared memory subsystem receives global addresses (from, *e.g.*, address consumer 408a, which may be an “application program, file system, hardware device or any other such element that requests access to the virtual memory”) and translates them into physical locations by traversing a distributed hierarchy of directory pages, ultimately identifying the node responsible for maintaining the physical memory location corresponding to the global address. EX1005, 39-41, Figs. 10, 11. After the address translation is complete, the shared memory subsystem, by communicating with the responsible node over the network, is able to read data from and/or write data to that physical memory, as directed by the requesting user. From the perspective of the address consumer (*e.g.*, a user application program), the translation of global addresses into physical locations happens behind the scenes, and thus the shared memory space appears to be a single, large hard drive.

126. Underscoring the transparency provided by the shared memory subsystem, the filesystem that overlays the virtual shared storage space appears, to software applications running on the nodes of the distributed storage system, as a “logical drive,” just like a standard network attached storage device.



Ex. 1140, Fig. 2 (annotated)

127. As shown in Fig. 2, above, user application 92 running on node 84 (outlined in red) sees fileset 66 of the global file system as the “F:” drive (shaded red box). EX1005, 13. Similarly, user application 96 running on node 86 (outlined in red) sees fileset 68 of the global file system as the “N:” drive (shaded red box). EX1005, Fig. 2.; *see also id.*, 13. Thus, even though the shared addressable memory of Carter is created by pooling together unused space from each of the distributed nodes in the system, this pooled storage appears to application programs as a dedicated network-attached storage (NAS) device.

128. Further, from the perspective of a software application running on one of the nodes of the Carter system, this NAS device has “an available amount of storage resources related to the unused or under-utilized storage resources for the selected distributed devices,” as required by the claim. Specifically, Carter explains “the total volume size” that is provided to the user is equal to the “*current*

aggregate storage capacity in the addressable shared memory space 20.” EX1005, 16:1-3 (emphasis added). In other words, using the shared memory subsystem, the shared storage space of Carter is represented as a NAS device that has a capacity equal to the total aggregate capacity of all the nodes participating in the system.

129. Carter discloses multiple aggregate instances. For example, if an analysis were to come out with only one NAS device being represented, it wouldn’t satisfy this claim language. However, multiple NAS devices, both at the individual level, aggregates because there is more than one Carter instance, e.g., in the file system example where there are two NAS devices said multiple NAS.” The ‘827 patent is similar, stating that “a more complex implementation is to provide a storage fabric across the Intranet or Internet where hundreds or thousands of devices ***work cooperatively as NAS devices***.” EX1001, 42:66-43:3 (emphasis added); *see also id.*, 43:59-44:23.

130. Additionally, Carter’s Figure 11 illustrates how the shared memory subsystem represents the multiple commercially available PC nodes (406a and 406b) as software-based NAS devices that make unused or under-utilized storage available. EX1005, 41:12-21, Fig. 11; *see also id.*, 41:22-42:3, 42:66-43:3. Further, as shown in Figure 11 below, “the “data associated with the directory pages are distributively [sic] stored across the two local memories” and “can move between different local memories and . . . between volatile and persistent storage”

in response to “data requests by memory users like application programs, or by operation of the migration controller.” EX1005, at 41:29–42:3, Fig. 11.

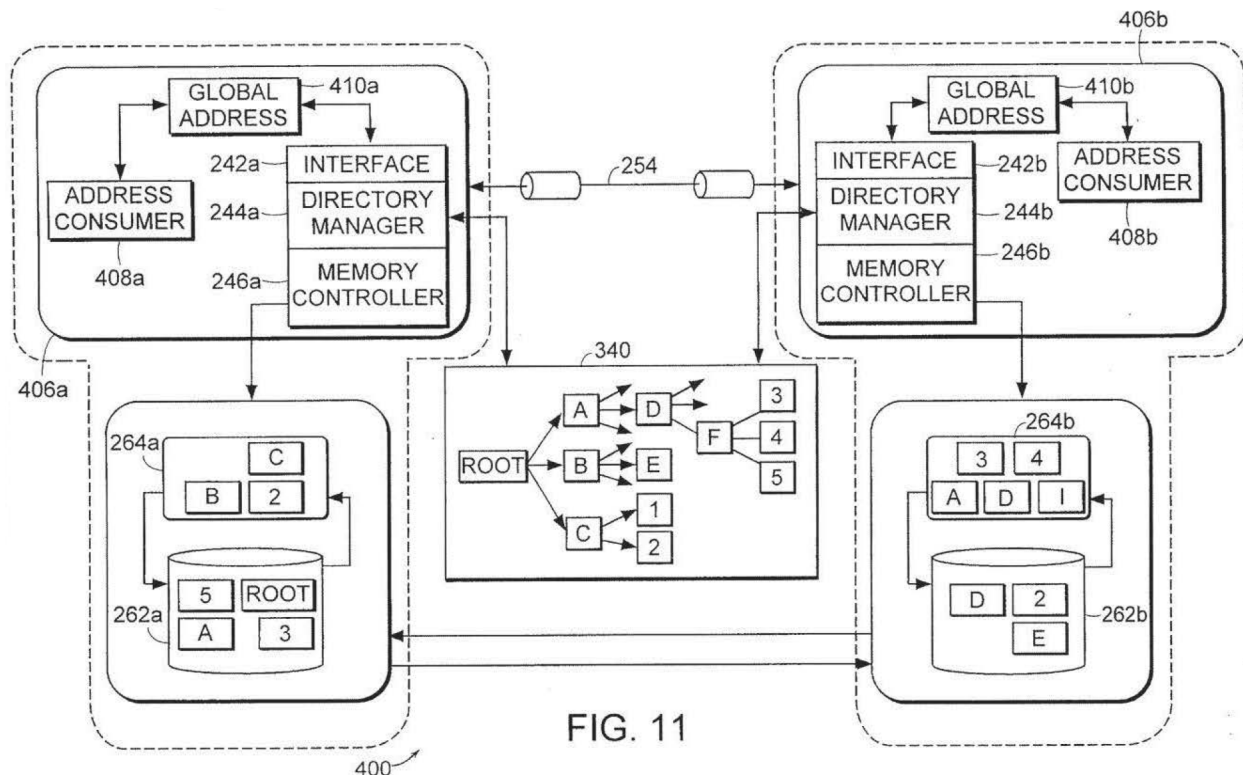


FIG. 11

1(d) “processing one or more data storage or access workloads for the distributed processing system by accessing data from or storing data to at least a portion of the available amount of storage resources to provide NAS service to a client device coupled to a network”

131. Carter teaches this limitation. In particular, storage or access workloads are processed by the “shared memory subsystem,” which is a component of the client agent taught by Carter. *See* EX1005, Fig. 2. The shared memory subsystems running on the distributed nodes communicate with each other over the network, and cooperate to manage the transfer and storage of data among the various physical storage devices that make up the distributed system.

132. The shared memory subsystem is responsible for processing storage workloads generated by the distributed devices. Specifically, Carter explains that “[e]ach node 212a-212c *through the shared memory subsystem* 220 can access each page 224a-224d stored in the virtual shared memory 222. EX1005, 29. “Each page 224a-224d represents a unique entry of computer data stored within the virtual shared memory 222” and thus, using the shared memory subsystem, each page 224a- 224d can be accessed or modified by each one of the nodes 212a-212c.” *Id.* Each node can also “store additional pages of data within the virtual shared memory 222,” and “[e]ach newly stored page of data can be accessible to each of the other nodes 212a-212c.” *Id.* All of these storage and access capabilities are provided by the shared memory subsystem, which is a component of the software package described by Carter (*i.e.*, the “client agent”) Thus, the shared memory subsystem is responsible for “processing one or more data storage or access workloads for the distributed processing system by accessing data from or storing data to at least a portion of the available amount of storage resources,” as the claim requires.

133. Further, the storage and access capabilities of the shared memory subsystem are used by the “data control program,” another component of Carter’s software package, to “provide NAS service to a client device coupled to a network,” as required by the claim. Specifically, the data control program uses the

low-level storage services provided by the shared memory subsystem to maintain a “structured storage system, such as a file system, that is adapted to maintain a structured store of data,” which is provided to “external *clients*.” EX1005, 10, 35 (emphasis added).

1(e) “enabling at least one of the selected distributed devices to function as a location distributed device to store location information associated with data stored by the selected distributed devices through use of the respective client agents for the particular distributed device”

134. Carter teaches this limitation. The shared memory subsystem enables every node in the network to “function as a location distributed device” that stores “location information associated with data stored . . . through use of the respective client agents” (e.g., as illustrated by the hierarchical page directory structure 340 shown in Figs. 10 and 11). EX1005, 39-41, Figs. 10-11.

135. The virtual-to-physical address mapping functionality taught by Carter and implemented by the shared memory subsystem meets this claim limitation. For example, Carter explains that the shared memory space is logically divided into sections of fixed size (e.g., “4k bytes, of the virtual memory space”), each of which is associated with a globally unique identifier, also known as a global address signal. *See* EX1005, 4. Carter further explains that any node (*i.e.*, a requesting node) connected to the system can use the global address signal corresponding to a particular block (what Carter refers to as a “4k page”) to access the data stored in that block or write data to that block for later use. EX1005, 31-

32. This global address signal is a virtual address because it is “decoupled from (i.e., unrelated to) the physical and virtual address spaces of the underlying computer hardware.” EX1005, 4. As explained above with reference to Figs. 10 and 11, a global address signal is translated into a physical address (e.g., the address of the physical storage such as RAM or a hard disk) by a memory subsystem using the shared memory space that is organized in a hierarchical directory structure. The directory structure (e.g., directory structure 340) includes a root page, a set of first level subordinate pages, and a set of second-level subordinate pages. The root page and the subordinate pages are distributed throughout the shared memory space, with each directory page being located on a “responsible node.” The directory pages shown in Figs. 10 and 11 are thus pieces of “location information” that are associated with the data stored in the shared memory by the various distributed devices, where each piece of location information is stored on one of the distributed devices (*i.e.*, each directory page is stored on a particular “owner node” that is tracked by a “responsible node,” as explained above). EX1005, 41, Figs. 10-11.

136. Further, Carter teaches that the ability to “store location information associated with data stored by the selected distributed devices” is provided “through use of the respective client agent[]” as the claim requires. Indeed, the “shared memory subsystem” and “data control program”—which are components of

the larger “client agent”—run on each distributed device in the network and enable each distributed device to function as a location distributed device by using the shared memory space and directory structure discussed above. Specifically, each node includes a “data control program,” the purpose of which is to “operate as a structured storage system, such as a file system, that is adapted to maintain a structured store of data and to employ the shared memory system as an addressable memory device that can store a structured store of data.” EX1005, 10. The “data control program” on each node interfaces with the “shared memory subsystem” which is able to “access and store data within the addressable shared memory space 20.” *Id.* Thus, as Carter explains, the data control program and the shared memory subsystem are “cooperating elements”: the data control program provides a transparent interface to user applications running on the node, making the shared storage space appear to be a dedicated hard drive that is attached to the local network. *See* EX1005, 10:8-28; *see also, id.*, 39:12-17, 41:12-30, Figs. 8, 10-11.

137. Additionally, as Carter further explains, the data control program implements a filesystem in which file metadata is distributed across the nodes in the network. This file metadata includes various types of information about the file contents, including its location, and therefore constitutes “location information.” In particular, the shared memory space contains both file data itself and also the metadata describing the structure and location of the files in the shared memory

space formed by pooling the nodes' collective storage resources. See *id.*, 12 (“[i]n contrast to known physical file systems which map a file organization onto disk blocks, the file system 60 *manages the mapping* of a directory and file structure onto a distributed *addressable shared memory system* 20” (emphasis added)). Thus, location information pertaining to the files in the filesystem is distributed across the nodes in the network.

138. Thus, Carter teaches “enabling at least one of the selected distributed devices to function as a location distributed device to store location information associated with data stored by the selected distributed devices,” as required by the claim.

iii. Claim 13

139. Claim 13 is a system claim that parallels the limitations of method claim 1. As described below, Carter teaches every limitation of claim 13 of the ‘827 patent for similar reasons to that discussed above in connection with claim 1.

13(pre) “A system comprising ... “

140. This is a preamble term and does not limit the claim. Nevertheless, Carter teaches systems. See, e.g., EX1005, Abstract (“A computer system employs a globally addressable storage environment that allows a plurality of networked computers to access data by addressing even when the data is stored on a persistent storage device such as a computer hard disk and other traditionally non-addressable data storage devices.” (emphasis added)); see also *id.*, 4 (“a further

object of this invention is to provide computer network systems ...”), 27, claims 1-20, each of which is directed to a “computer **system**” (emphasis added).

13(a) “a plurality of distributed devices configured to be coupled to a network, wherein the distributed devices include respective client agents configured to process respective portions of a workload for the distributed processing system”

141. This limitation is satisfied for the reasons given above in the context of claim 1.

13(b) “the respective client agents including respective instances of a software-based network attached storage (NAS) component wherein the NAS component is configured to: assess unused storage resources of the distributed devices”

142. This limitation is satisfied for the reasons given above in the context of claim 1.

13(c) “wherein the NAS component is configured to: ... allocate respective available amount of unused storage resources in selected distributed devices of the plurality of distributed devices; represent that the selected distributed devices comprise NAS devices having the respective available amounts of storage resources”

143. This limitation is satisfied for the reasons given above in the context of claim 1.

144. Additionally, the Board in the prior IPR determined that “allocate” as used in Carter has the same meaning as in the ’827 patent. IPR2017-00439, Paper 50 (EX1020) at 39-41. And Carter discloses this “allocating” limitation. *Id.* For the following reasons, I agree with the Board.

145. Carter discloses the “allocating” limitation—namely, it allocates the storage it has assessed as unused or under-utilized and thus available for other devices on the network to use. For example, Carter discloses a policy element 290 that assesses used, unused, and allocated storage space “to determine the availability of resources, such as . . . hard-disk space, available RAM and other such resources” and “dynamically distribut[ing] [these] available resources” to other devices on the network. *Id.* EX1005, 36:27-37:2; *see also id.*, 9:18-21.

146. Additionally, Carter further discloses “reserving something much larger than requested memory space for a file.” *See* EX1005, 19:14-17. Carter discloses several other instances of allocating. For example, Carter discloses a “distributed directory manager . . . for allocating a portion of the addressable memory space” and that “data streams are mapped onto pages allocated form the addressable shared memory space 20 for storage.” And Carter discloses allocating the first available free space on a page (EX1005, 16:21-23), data streams mapped onto allocated pages (*id.*, 17:17-20), and when a file is deleted, all shared memory space allocated to the files in the fileset is discarded (*id.*, 15:9-11). “Allocate” has the same meaning when used in Carter as it has in the ‘827 patent.

13(d) “wherein the NAS component is configured to: ... represent that the selected distributed devices comprise respective NAS devices having the respective available amounts of storage resources”

147. 1. This limitation is satisfied for the reasons given above in the context of claim 1.

13(e) “wherein the NAS component is configured to: ... process workloads associated with data storage and access by accessing data from and storing data into portions of the storage resources in the selected distributed devices to provide NAS service to a client device”

148. This limitation is satisfied for the reasons given above in the context of claim 1.

13(f) “wherein the respective client agents are configured to enable at least one of the selected distributed devices to function as a location distributed device to store location information for data stored by the selected distributed devices”

149. This limitation is satisfied for the reasons given above in the context of claim 1.

iv. Claims 2 and 14

150. Claim 2 depends on claim 1 and recites “further comprising enabling at least one of the selected distributed devices *to function as a stand-alone dedicated NAS device* through use of the respective client agents for the particular distributed device.” Claim 14 is of similar scope but is a system claim. Carter discloses a “NAS device” for the reasons discussed regarding limitation 1(d). As discussed below, Claims 2 and 14 are both obvious in light of Carter.

151. The Board in the prior IPR found that Carter causes individual nodes to *separately appear* to network-connected devices as a dedicated NAS devices because “Carter uses software that causes individual nodes (e.g., commercially

available PCs) to represent, or mimic, dedicated NAS devices” and “each node may be responsible for storing a particular element or elements of the structured store of data.” IPR2017-00439 FWD (EX1020) at 36, citing EX1005, 10:6-26, Fig. 1. The Board also noted that “Carter’s Figure 11 shows multiple commercially available PC nodes (406a and 406b) each mimicking NAS devices by making storage available.” *Id.*, citing EX1005, 41:12–21. Moreover, as discussed above, the shared memory subsystem enabling this functionality is part of a software package on each node that is the “client agent” that provides the required NAS device functionality.¹¹ EX1005, 39–42, Figs. 7, 10, 11. I agree with the Board’s findings. Thus, as the Board found in the previous IPR, Carter discloses a client agent that enables a node to function as a stand-alone dedicated NAS device (“a distributed device that separately appears to network connected user devices as a dedicated NAS device”), as required by claims 2 and 14.

¹¹ As I discuss in limitation 1(a) above, Carter’s software package is a “client agent” under either of the co-pending litigation constructions (“a program that manages the workloads and processes of a distributed processing system” or “a software program configured to run on each distributed device to provide what appears to client devices as dedicated NAS functionality”).

152. Additional aspects of Carter also disclose this limitation. Figure 2 shows “[t]wo network nodes 84 and 86 [] accessing the addressable shared memory space 20 . . . via the logical drives 90 and 94,” “F:” and “N:” respectively. EX1005, 13:8-14, Fig. 2. Each “logical drive” is a DOS device that connects to the fileset directories, 66 and 68 respectively, via Installable File System drivers associated with file system 60. *Id.* Importantly, user “application programs 92 and 96 executing on the nodes interact with the data control program . . . and cause the data control programs in the nodes to access the logical drives 90 and 94.” *Id.* Given the disclosure of Carter, it would have been obvious to a POSITA that data for the “F:” and “N:” drives are wholly distinct logical drives, and that an obvious implementation provides for the data for each of these drives to be hosted on separate *physical* devices that operate completely independent of one another without knowledge of the other drive or the set of memory pages associated with the other drive. For instance, a POSITA, in pursuit of the performance advantages associated with data isolation, would have understood that it would be beneficial, using the policy controller (EX1005, 36:29-37:2, Fig. 8 (290)), to implement the “F:” and “N:” drives as physically separate devices so that the storage access utilized by one client node would be isolated and to prevent any impact on the performance of storage access by another client node.

153. Additionally, these “N:” and “F:” drives would separately appear to the application programs as dedicated NAS devices whose primary operational purpose is to provide NAS service. Indeed, each drive is separately addressed by application programs by reference to the respective drive name “F:” or “N:” as each drive appears to application programs as a device whose primary operational purpose is providing NAS services. For example, as discussed above, “during a memory access operation, the memory subsystem [] passes to the responsible node . . . the address of the page being accessed” and “the shared memory system of that [responsible] node can identify a node that stores a copy of the page being accessed, including the owner node” and “the data read from the physical memory of the owner node can be passed via the network to . . . the requesting node for use by that node.” *Id.*, 40, 41. Fig 11. Thus, each of the logical drives “F:” and “N:” have one or more physical nodes that respond to requests for their data (e.g. *responsible nodes* and/or *owner nodes*). Consequently, these physical nodes separately appear to the respective application programs as dedicated NAS device(s), are separately addressed by application programs by reference to the respective drive name “F:” or “N:”, and such that each drive appears to application programs as a device whose primary operational purpose is providing NAS services. *Id.*

154. Moreover, the ‘827 patent itself states that it is a “*relatively simple implementation* [] to have the devices simply *mimic the functionality of stand-alone NAS devices*.” EX1001, 42:66–43:1 (emphases added). A POSITA would consider a “relatively simple implementation” an obvious variant of Carter even if the disclosure were not so explicit.

155. As discussed in Section VI.B., I understand that Patent Owner’s co-pending litigation construction of the term “stand-alone dedicated NAS devices” is “a dedicated NAS device that contains its own storage, processing, connectivity, and management resources.” For the reasons discussed above and below, the Carter-disclosed stand-alone dedicated NAS device is clearly a device that contains its own storage, processing, connectivity, and management resources. It is also a dedicated NAS device (“a device whose operational purpose is for providing NAS services”) for at least the reasons discussed above and in element 1(c).

156. I note that the ‘827 patent describes an optional feature of a stand-alone dedicated NAS device in that it may act without knowledge of other NAS devices connected to the network. Ex 1001, 3:17-21; *see also id.*, 43:47-52 (“In this first implementation, the NAS devices 108A, 108B...108C run client agent programs that configure the devices to be capable of mimicking dedicated stand-alone NAS devices” in which “[e]ach of these software-based NAS devices ... knows nothing of other software-based NAS or other NAS devices.”) Also, the

‘827 equates “knowing *nothing* of other software-based NAS devices” with at least the lack of data location libraries 2106A and 2106B, which contain “distributed data location information.” EX1001, 43:30-33, 4:56-58. A POSITA would understand that lacking any means with which to *interpret* data location information—even if stored locally, satisfies “knowing nothing about other NAS devices,” to the extent it is a requirement of a “stand-alone dedicated NAS device.”

157. Carter also discloses stand-alone dedicated NAS devices embodying these features. For example, Figure 6 discloses additional standalone dedicated NAS devices such as network memory device 226 illustrated in Figure 6 that can be used by client devices to store or access data. EX1005, 29:21-30:2, Fig. 6. As shown in Figure 6, network memory device 226 is separately connected to the network and contains its own storage “network disk,” which are resources that are independently managed by network memory device 226. *Id.*, EX1005, 29:22-30:2. Fig. 6.

158. Network memory device 226 clearly functions as a dedicated NAS device, since it is identified as a *network* memory device, contains a storage device (disk) with pages labeled similarly to pages in other nodes (e.g. 236a-236c), and no disclosed functionality other than the ability to connect with other devices in the system 230. Further, network memory device 226 is clearly different from nodes 236a-236c in that the only software component listed is a “network memory”

rather than the “memory subsystem” components 232a-232c of the other nodes.

While the network memory is not further described, the “memory subsystems”

232a-232c are further expanded in Fig 7 as containing “DSM Directory

Manager[s]” 244 which is required to interpret and process directory entries. *See,*

e.g., EX1005, 30:27-33:1 (“The directory manager 244 maintains a directory

structure that can operate on a global address received from the interface 242 and

identify, for 30 that address, a node on the network that is responsible for

maintaining the page associated with that address of the shared memory space.”)

Because of its lack of a “memory subsystem” (*e.g.* 232a-232c), the network

memory device 226 acts without the knowledge of other NAS devices connected to

the network and does not have access to the hierarchical structure of directory

pages (*i.e.*, “location information”) that is stored in the shared memory system, and

thus, further lacks data location libraries. EX1005, 5, 38-41, Figs. 10, 11.

Therefore, it would presumably be unable to interpret or process directory entries

and thus satisfy the limitation of “*functioning as a stand-alone dedicated NAS*

device.”

159. As another example, Carter discloses that not all nodes would have to store directory information, and thus these nodes function as stand-alone dedicated NAS devices without data location libraries. Specifically, Carter discloses data location information in the form of hierarchically linked directory pages. *See*

Section VII.B.ii.f., *supra*; see also EX1005, 39:8-41:11, Figs. 9-10. And “[d]uring a memory access operation, the memory subsystem 270 passes to the responsible node indicated in the leaf entry 356 the address of the page being accessed,” and then, “[t]he shared memory subsystem of that node can identify a node that stores a copy of the page being accessed, including the owner node.” *Id.*, 40:13-18.

Further, “the directory structure 340 provides ***location information*** for pages stored in the distributed address space.” *Id.*, 40:9-10 (emphasis added). Thus, the data ***location information*** is independent of the physical location of the data (stored, *e.g.*, on an owner node).

160. “Accordingly, [the directory structure] tracks ownership and responsibility for data” “provid[ing] a ***level of indirection*** between the directory and the physical location of the data.” *Id.*, 40:13-15 (emphasis added). The directory information is further “stored in the distributed memory space just as the other pages to which the directory pages provide access.” (*Id.*, 38:15-16). Consequently, one of ordinary skill would clearly understand that the directory information could be stored anywhere in the system and, further, *not all nodes would have to store directory information*. Nodes which did not store any directory information would meet the limitation of “function[ing] as stand-alone dedicated NAS devices,” since they would have no knowledge of other NAS devices. *Whether or not any particular system had such nodes would be a configuration*

choice. Consequently, Carter discloses such “stand-alone dedicated NAS devices.”

161. Carter further discloses this aspect of “stand-alone dedicated NAS devices” through the notion of “owner nodes” which provide direct access to the data of a shared page without referencing (or needing to know about) other NAS devices. Since the information from owner nodes is provided over the network, they function as stand-alone NAS devices with respect to the data for which they are owners. Carter explains:

162. “... The owner [] is the network node that has control over the memory storage element on which the data of the associated page is stored. The memory controller 246 of the owner can access, through the OS 216 of that node or through any interface, the memory of the owner to access the data of the page that is physically stored on that owner node.” EX1005, 33:15-21.

163. Carter further discloses that owner nodes act as NAS devices:

164. “The owner node receives the memory request across network 254 and through network module 252 that passes the memory request to the interface 242 of that owner node. The interface 242 couples to the memory controller 246 and can pass the memory request to the local memory controller of that owner node for operating the local storage elements, such as the disk or RAM elements, to perform the requested memory operation.

165. Once the owner node has performed the requested memory operation, such as reading a page of data, the memory subsystem 240 of the owner node can then transfer the page of data, or a copy of the page of data, via the network 254 to the node that originally requested access to that portion of the shared memory.” EX1005, 33:25-34:4.

166. Consequently, while nodes in Carter *could* serve a dual purpose of storing *both* directory information *and* acting as owner nodes for data (i.e. shared pages), a POSITA would have understood that Carter offers a clear option to arrange the placement of pages (both directory pages and shared pages) such that some nodes hold *only* shared pages, not directory pages. Such “shared page only” nodes would then “*function as stand-alone dedicated NAS devices.*”

167. In fact, Carter provides a mechanism for performing such data placement by disclosing the “migration controller” and “policy controller” which “perform hierarchical data migration and load balancing, such that data migrates primarily between computers of the same group, and passes to other groups in hierarchical order.” EX1005, 31:4-7. Carter explains that “[r]esource distribution is similarly managed.” *Id.* The policy controller, as described, gives significant flexibility for choosing when and where to distribute data. EX1005, 36:29-37:2 (“The policy controller can employ any of the suitable heuristics to direct the

elements, such as the paging controller, disk directory manager, and other elements to *dynamically distribute the available resources.*”)

168. A POSITA would understand that such a segregation of data would offer the possibility of specialization, namely the inclusion of NAS devices which did not need to interpret and act upon directory information and would thus offer a leaner, higher performing software stack.

169. A POSITA would further understand the performance advantages of simplifying the software stack of a NAS device to couple a disk as directly as possible to the network. Consequently, a POSITA would arrange to place shared data on devices functioning as stand-alone dedicated NAS device using the migration and/or policy controllers.

v. Claims 9 and 21, which claims 10-12 and 22-24 depend upon

170. Claim 9 depends from claim 1, and further requires “managing storage resources for the selected distributed devices with a storage priority control that facilitates use of the available amount of storage resources for the selected distributed devices.” EX1001, claim 9. Claim 21 depends from claim 13, and further requires “comprising a storage priority control configured to facilitate use of the respective available amount of unused storage resources for the selected distributed devices.” These limitations are substantively identical.

171. Carter teaches these additional limitations. For example, as explained above, the shared memory subsystems can include logic to assess the “*availability of resources*, such as printer capabilities, *hard-disk space*, available RAM and other such resources.” EX1005, 36-37 (emphasis added). The shared memory subsystem may also include a “policy controller [that] can employ any of the suitable heuristics to direct the elements, such as the paging controller, disk directory manager, and other elements to dynamically *distribute the available resources*.” EX1005, 36-37 (emphasis added).

172. Carter also discloses that the disclosed system may provide “a distributed shared memory that includes persistent storage for portions of the distributed memory.” EX1005, 30, Fig. 6. Specifically, the disclosed system “includes a memory subsystem, such as subsystem 232a, that interfaces to a persistent memory device, depicted as the disk 236a” and “subsystem 232a can operate the persistent memory device to provide persistent storage for portions of the shared memory space.” Accordingly, Carter explains, “the subsystem 232a can provide integrated control of persistent devices and electronic memory to allow the distributed shared memory space to span across both types of storage devices, and to allow portions of the distributed shared memory to move between persistent and electronic memory depending on predetermined conditions, such as recent usage.” EX1005, 31.

173. Carter also explains that “the memory subsystems 232a-232c can include a hierarchy manager that provides hierarchical control for the distribution of data. This includes controlling the migration controller, and policy controller ... to perform hierarchical data migration and *load balancing*.” EX1005, 31 (emphasis added); see *also id.* (“*Resource distribution* is similarly managed.” (emphasis added)).

vi. Claim 10

174. Claim 10 recites “further comprising accepting selection of the storage priority control, by the client device, that comprises a parameter.” EX1001, claim 10. Carter renders obvious claim 10 based on the same rationale as claim 9. Having disclosed storage priority control, a POSITA would understand that there would necessarily be one or more rules or parameters associated with the priority that would need to be set or selected.

175. For example, Carter explains that each node includes a data control program and a shared memory subsystem. EX1005, 10. As discussed above for claim 9, Carter explains that “the subsystem 232a can provide integrated control of persistent devices and electronic memory to allow the distributed shared memory space to span across both types of storage devices, and to allow portions of the distributed shared memory to move between persistent and electronic memory depending on predetermined conditions, such as recent usage.” EX1005, 31. And

the subsystem 232 a “control[s] ... hierarchical data migration and load balancing, such that data migrates primarily between computers of the same group, and passes to other groups in hierarchical order.” *Id.* Meanwhile, the data control program provides an “interfac[e] [for] a shared memory subsystem 34a” to “access and store data within the addressable shared memory space 20.” EX1005, 10, Fig. 1.

176. The identification of which “portions of the distributed shared memory to move,” whether data should be migrated “between computers of the same group,” whether data should be “pass[ed] to other groups in hierarchical order,” and how to “load balanc[e],” are each examples of storage priority control parameters, as they identify information that is needed to execute the relevant storage priority control. For example, the parameter, which “portions of the distributed shared memory to move,” identifies what memory to move from persistent to electronic memory and vice versa, and Carter provides “recent usage” as an example of one such parameter. If we look at Figure 2, in this context, the users of the shared memory space—*e.g.*, application programs 92 and 96 running on the network nodes—might want to prioritize more recently used data for storage in electronic memory as it would be easier and faster to access said data from there. To accomplish this, the application program 92s running on node 84—which includes data control program 32a and shared memory subsystem 34a from Figure 1—would set this parameter at the interface provided by the data control

program 32a, which accepts the selection of the storage control parameter and implements the storage priority controls using the lower-level capabilities of the shared memory subsystem 34a. Coincidentally, the '827 patent specifically identifies “[u]sage of the files, for example, least-recently-used (LRU), most-recently-used (MRU), etc. cache schemes” as an example storage priority scheme. Similarly, with respect to whether the data should be passed to computers of the same group (as opposed to computers of different groups), whether the data should be passed between to other groups in hierarchical order (as opposed to computers, for example, file size), and how to load balance (*e.g.*, whether to balance load equally across all devices or take into account geographical location or node utilization), each of these parameters dictate how storage is to managed and prioritized, allowing the system to execute the relevant storage priority control.

177. Moreover, a POSITA would have understood, and found it obvious, that the client device, which may have a portion of its storage resources allocated to provide NAS services and is the target of the “storage priority control,” would have to accept the parameter defining the relevant storage priority control when executing the storage priority control.

vii. Claim 11

178. Claim 11 recites “further comprising prioritizing one or more of storage or deletion of data using the storage priority control that comprises storage priority level schemes.” EX1001, claim 11. Carter renders obvious claim 11.

179. As discussed above, Carter discloses “storage priority control” that controls “what portions of the distributed shared memory to move” and controls “hierarchical data migration and load balancing.” EX1005, 30-31. “[M]ov[ing],” “migrat[ing],” and “load balancing” data involves removing data from its original physical location and storing data at a new physical location. Each of these processes disclose both “prioritizing storage [and] deletion of data....”

180. A POSITA would have understood that “electronic memory” (which is used interchangeably with “volatile” or “RAM” memory in Carter) would typically be more limited in quantity but faster than “persistent memory” (e.g. disk). Further, since electronic memory would not be persistent, it would typically be used to hold temporary (cached) copies of data as a way of enhancing performance. A POSITA would have understood that clients or applications received the limited caching resources would clearly be decided by a storage priority scheme and lead to a tradeoff between performance and the location of data, the number of copies of data, and the degree of persistence of data. Choosing to place a copy of one application’s data into the electronic memory of a node

would clearly involve the deletion of data from a different application to make room for this new data. Here, the data placed into the electronic memory would be prioritized over the data placed into the persistent memory and thus certain applications would have higher priority than others. On the other hand, moving one application's data from the electric memory of a node to the persistent memory would clearly involve the deletion of the data from the electronic memory to make room for new data. In this case, the data deleted from the electronic memory and placed into the persistent memory would be de-prioritized over the data that will eventually take its place in the electronic memory which is being prioritized and thus certain applications would have higher priority than others.

181. Carter further discloses utilizing the electronic memory and/or excess persistent memory for a distributed cache: “commonly used pages are copied and distributed, and rarely used pages are shuffled off to disk. Similarly, directory pages will migrate to those nodes that access them most, providing a degree of self-organization that reduces network traffic.” EX1005, 41:8-11; *see also id.*, 13:3-4 (disclosing automatically and dynamically migrating data to account for varying network usage and traffic patterns). Here, a POSITA would have understood that the “rarely used pages are shuffled off to disk” because disk is slower, allowing less used data to be deleted from electronic memory and freeing this precious resource to be utilized for more popular data. Further, a POSITA

would understand that moving (migrating) directory pages to nodes that access them most would provide a huge performance boost because it would permit these pages to be accessed without network traffic. The combination of moving and copying data to different nodes would also help with “load balancing”, because it would reduce network traffic by placing data close to where it is being used. This copying, distributing, and deleting renders obvious prioritization of storage or deletion of data based on how commonly the pages are used, i.e., according to a priority level scheme. For instance, the most frequently used data would be prioritized (giving it greater use of electronic memory and/or extra copies in persistent memory) over infrequently used data (which may only reside in persistent memory).

182. Carter also discloses another common virtual memory pattern, recognizable by a POSITA, in which persistent memory (disk) is utilized to back up the shared memory state of an application, providing an addressable shared memory space that appears larger than the total amount of electronic (volatile) memory in the system. *Id.*, 36:8-15 (“In one embodiment, a paging element can operate the RAM and disk directory managers to remap portions of the addressable memory space between one of the volatile memories and one of the persistent memories. In the shared memory system, this allows the paging element to remap pages from the volatile memory of one node to a disk memory of another node...

This allows for improved load balancing, by removing data from RAM memory, and storing it in the disk devices...”). A POSITA would have understood that the “paging element” would be operating according to a storage priority control that prioritized electronic memory over persistent memory for performance reasons: by removing infrequently accessed data from RAM memory (and placing it into slower persistent memory to preserve its contents), it would allow more frequently accessed data to be placed into that memory.

183. This disclosure of Carter would render claim 11 obvious to a POSITA.

viii. Claim 12

184. Claim 12 recites “wherein the managing storage resources further comprises marking directly data or files.” EX1001, claim 12.

185. Carter renders obvious claim 12. Carter describes a local RAM cache 276 that “provides storage for memory pages and their attributes.” EX1005, 37:16. In one embodiment, Carter discloses that “the local RAM cache 276 provides a global address index for accessing the cached pages of the distributed memory and the attributes based on that page.” *Id.*, 37:16-18. Carter explains that “[t]he local ram cache 276 provides the index by storing in memory a list of each global address cached in the local RAM.” *Id.*, 37:19-20. Carter further discloses that “[o]ptionally, with each listed global address, the index can further provide

attributed information including a version tag in the index that is representative of the version of the data, a dirty bit representative of whether the RAM cached data is a copy of the data held on disk, or whether the RAM cached data has been modified but not yet flushed to disk, a volatile bit to indicate if the page is backed by backing store in persistent memory, and *other such attribute information useful for managing the coherency of the stored data.*” EX1005, 37:16-26. A POSITA would have understood from this disclosure at the time of invention that the “storage priority control” and “storage priority schemes” discussed regarding claims 9-11 above would use this attribute information as marks made directly to the data or files representing that the same should be moved, migrated, load balanced, or distributed.

186. For example, a POSITA would have understood from these disclosures in Carter that it “would be useful for managing the coherency of the stored data” to mark each “commonly used page[]” with Carter’s disclosed attribute information, such as a “dirty bit,” to indicate that these pages have been “copied and distributed” because they are so frequently accessed. EX1005, 41:8-9; 37:16-26. For the same reason, a POSITA would have understood that attribute information would be used to mark “rarely used pages” that should be “shuffled off to disk.” EX1005, 41:9. As another example, a POSITA would have understood that Carter’s attribute information would be used to mark each “directory page” by

identifying the “nodes that access them most” so the “directory pages will migrate to those nodes.” EX1005, 41:9-10. As another example, Carter’s attribute information would be used to mark each data file that is sent to persistent storage based on “recent usage.” EX1005, 30:27-28. In each of these examples, a POSITA would have understood that Carter’s attribute information would be used together with the storage priority control to better manage the coherency of the stored data as taught by Carter. EX1005, 37:21-26. Thus, these disclosures of Carter would render claim 12 obvious to a POSITA.

ix. Claims 22

187. Claim 22 recites “[t]he system of claim 21, wherein the storage priority control comprises a parameter selectable by the client device.” EX1001, claim 22. Carter renders obvious claim 22 based on the same rationale as claim 21. Because Carter discloses storage priority control, a POSITA would understand that there would necessarily be one or more rules or parameters associated with the priority that would need to be set or selected.

188. Where a portion of a client device is providing NAS service, and, as discussed above regarding Claims 9-12, the client devices’ storage memory subsystem is deciding whether and how to “move,” “migrat[e],” or “load balan[ce],” the data (See EX1005, 30-31), the application program running on the client device “selects” the storage priority control parameter at the interface

provided by the data control program 32a, which accepts the selection of the storage control parameter and implements the storage priority controls using the lower-level capabilities of the shared memory subsystem 34a. EX1005, 10, Figs. 1-2; EX1006. For example, when a client device's storage memory subsystem is determining what portions of the distributed shared memory to move from electronic storage to persistent storage, the client system must "select" whether to make the decision based on "recent usage" or some other parameter (e.g., whether the portion contains the most recent version of the data, whether the data is currently backed up in the persistent storage, or "whether the RAM cached data has been modified but not yet flushed to disk," see EX1005, 37:16-26). D

x. Claims 23-24

189. Claims 23-24 recite similar limitations to claim 11-12, except in the form of system claims. Claim 23 recites "wherein the storage priority control comprises storage priority level schemes that prioritize one or more of storage or deletion of data." Claim 24 further recites "wherein the storage priority control comprises a priority marking directly given to data or files." Carter renders obvious claims 23-24 under the same rationale as claims 11-12, respectively.

C. Ground 2: Claims 2 and 14 are Obvious over OceanStore

i. Overview of OceanStore

190. OceanStore is a system described in a paper entitled "OceanStore: An Architecture for Global-Scale Persistent Storage" that appeared in the Proceedings

of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000) in November 2000. EX1006. Although the title of this conference (*i.e.*, ASPLOS) might appear to cater to a fairly narrow crowd of researchers, quite the opposite is true. This conference is targeted rather broadly to hardware/software tradeoffs that can span everything from chip design (*e.g.*, of new processors), to multimedia algorithms, to the interaction of distributed databases running on clusters, to cache-coherence protocols on multiprocessors, to new computer languages for expressing parallel computation. Despite its breadth of topics, ASPLOS is considered a “top-tier” conference with a relatively low acceptance rate (easily less than 20% of papers are accepted). When my research group and I wanted to publish the first OceanStore paper, ASPLOS seemed like natural fit since it would face a dynamic, “multi-cultural” audience.

191. In the year prior to publication of the ASPLOS paper, I submitted a “Career” research proposal about (OceanStore) to the National Science Foundation (NSF). This proposal ultimately won the Presidential Early Career Proposal for Scientists and Engineers (PECASE), a prestigious award for early faculty.

192. In 2018, the original OceanStore paper received an award as “Most Influential Paper” by the ASPLOS organization, as a testament to its lasting impact and influence.

193. I have been informed that OceanStore is prior art to the ‘827 patent under 35 U.S.C. § 102(a) for the ‘827 patent’s proper priority date of April 13, 2001. I have been further informed that OceanStore was not disclosed to the Patent Office during prosecution of the ‘827 patent.

194. As the title of the paper suggests, OceanStore concerns “an architecture for global-scale persistent storage.” EX1006, Title. Similarly to Condor, Chien, and Plank, “OceanStore hope[d] to take advantage of [the] excess of storage space [e.g., in “desktop workstations”].” EX1006, 191.

195. At the time that I proposed OceanStore, the world was rapidly transferring its photographs, documents, and other records to digital form while at the same time discovering new ways to produce digital information.¹² So much of this precious information was stored in ways that could be easily destroyed and was not easily accessible. In fact, people were being remarkably naive in how they were storing digital information. The challenge in this new world was how to preserve all of this information while still making it rapidly accessible from anywhere in the world. My students and I wanted to make it virtually impossible to

¹² In fact, one study in the early 2000s had estimated that the world had produced 1.5 exabyte (that’s a 1 with 18 zeros after it) of digital data in a year.

destroy information-taking inspiration from the hieroglyphs on Egyptian pyramids (although planning to use intelligent replication mechanisms instead).

196. Helping this vision was the fact that network connectivity had reached an all-time high, pushing the network into places of the world that it had never been before. By spreading copies of information geographically (even on separate continents), we could make it virtually impossible to destroy. At the same time, the size of individual disk drives was doubling every 18 months. Since much of this storage was unused, we sought to scavenge this storage to help with preserving information and providing increased performance through caching.

197. A final inspiration for me was the success of the electric grid and telecommunication companies, both of which were successful examples of cooperative utilities in which customers pay one company while utilizing resources owned by multiple companies. The utility model applied to storage seemed very appropriate to the type of global storage that OceanStore envisioned.

198. As a result, OceanStore was designed to span the globe and to offer “uniform and highly available access to information” through widely dispersed servers (a.k.a. nodes, pool devices, caches). EX1006, Abstract, 190. It envisioned providing service to 10 billion users, with millions of entities providing storage. EX1006, 191,198 (“OceanStore will consist of millions of servers with varying connectivity, disk capacity, and computational power.”). Consumers would pay a

monthly fee for storage services, for example, while storage providers could buy and sell capacity amongst themselves. EX1006, 190 (“We envision a cooperative utility in which consumers pay a monthly fee in exchange for access to persistent storage.”). Though a user would pay her fee to a “utility provider,” the user could actually consume storage resources from many different providers, *i.e.*, the owners of the entities actually providing the storage services. EX1006, 190 (“Each user would pay their fee to one particular ‘utility provider’, although they could consume storage and bandwidth resources from many different providers; providers would buy and sell capacity amongst themselves to make up the difference.”).

199. Figure 1 of OceanStore is reproduced below and shows that user devices could come in many forms, including desktop computers, PDAs and phones. EX1006, 191. As shown, the aggregate collection of servers in the world forms an “ocean” of data resources, in which data quickly “flows” to where it is needed, and individual caches could be thought of as comprising “pools” of data. EX1006, 191.

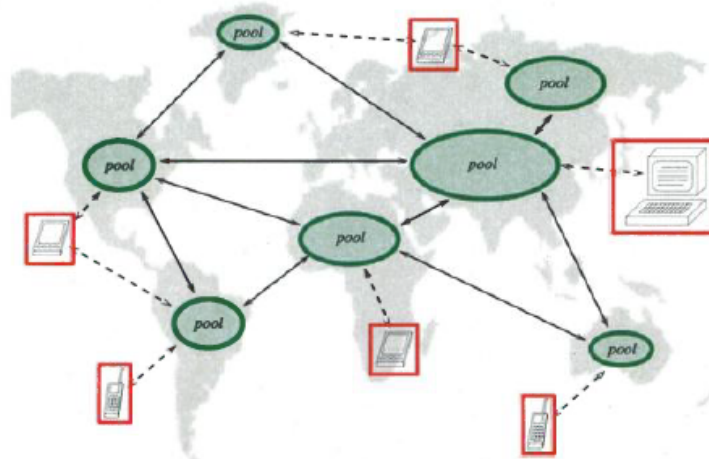


Figure 1: The OceanStore system. The core of the system is composed of a multitude of highly connected “pools”, among which data is allowed to “flow” freely. Clients connect to one or more pools, perhaps intermittently.

OceanStore, Fig. 1
(Ex. 1003, Fig. 1 (annotated).)

200. Because OceanStore expected to serve a variety of client types and a variety of applications, it provides support through “facades” to the standard OceanStore application programming interface (API). EX1006, 197. OceanStore discloses that a “facade is an interface to the [OceanStore] API that provides a traditional, familiar interface.” EX1006, 197 (“For example, a transaction facade would provide an abstraction atop the OceanStore API...”). OceanStore provided several “legacy facades that implement common APIs, including a Unix file system, a transactional database, and a gateway to the World Wide Web.” EX1006, 198.

201. OceanStore’s fundamental unit of information was a “persistent object,” which could contain not only data but also “pointers” to other objects or

names of other objects, thus representing directories or local name spaces.

EX1006, 191-192 (“Certain OceanStore objects act as directories, ... we allow directories to contain pointers to other directories.”). These objects were composed of “floating replicas,” the term “floating” indicating that “[a] given replica is independent of the server on which it resides at any one time” and indeed could move among servers, for example, to be positioned closer to the user to improve performance. EX1006, 191; *see also* at 193 (discussing “location-independent addressing”), 198 (“Replica management adjusts the number and location itself.”) Floating replicas were assigned a “globally unique identifier (GUID)” to identify the object and which were used to access objects. (EX1006, 191-192).

202. Every addressable entity in OceanStore is identified by one or more GUIDs, and supports “location-independent addressing” through a two-phase routing process, which “tackles the problem of data location head-on.” EX1006, 193 (“In order to perform this routing process, the OceanStore networking layer consults a distributed, fault-tolerant data structure that explicitly tracks the location of all objects.”) The routing mechanism is a two-tiered approach, featuring a “fast, probabilistic algorithm,” particularly useful if the object is locally available, “backed up by a slower, reliable hierarchical method.” EX1006, 193. In short, a

user would connect to a nearby OceanStore server,¹³ or pool device, as depicted in Figure 1 via dashed lines. That user could then read or update objects using a GUID to identify the relevant object being requested or updated. That object however could, in theory, be any place in the network, and need not be (though preferentially is) on the server or pool device that the user connected to. As a result, OceanStore provided data location algorithms to locate the relevant object.

203. For example, when a client needed to access data, the client sent a message to a node specifying the object it wished to access via its GUID.

EX1006, 94. Each node, based on its routing table, routed the message, until it finally reached the node that actually stored the requested object. *Id.* After the node that had the requested object was identified, messages could be directly sent to that node without having to route through the other nodes. *Id.*

204. To step back for a moment, OceanStore had to deal with two conflicting goals: preservation and performance. Said in another way: (1) information needed to be spread widely in order to preserve it-intentionally spreading it over geographic distances and (2) information in active use needed to be as close as possible to increase performance through caching. As a separate, but

¹³ For example, as explained further below a desktop workstation with excess storage space could be made to act as a storage server, serving storage resources to user devices, through the use of OceanStore software on the workstation.

important additional goal, users of information should be able to access their information from anywhere at any time while retaining performance. They may start in (say) Boston working for a while, then fly onto San Francisco, where they may continue working on the same documents. Ideally, documents would follow them from Boston to San Francisco for highest performance.

205. It is within this context that the two-tiered data location mechanism should be considered. To optimize for performance, we needed to find local copies of information quickly, with as few network hops as possible. To allow for geographic distribution of information, we needed a way to find information wherever it might reside-anywhere in the world. This geographic location algorithm would need to be scalable enough to locate information for any of the assumed 10 billion users. Consequently, we chose two separate algorithms for data location-one to find information if it is close by and another to locate information if it not found by the first algorithm. In particular, the second algorithm provides a more efficient mechanism for locating information when the user requests data that is many network hops away (*e.g.*, when the user has travels to a new geographic location).

206. With reference to the first location algorithm, it is “fully distributed.” If a query for a data object (identified by a GUID) cannot be satisfied by a server connected to the user device, that server uses “local location information” to route

the query to a neighbor server believed likely to have that object. EX1006, 196. That neighbor then performs analogously, by determining if it has the object and returning it if so, and by forwarding the request to another node if not. EX1006, 196-197. This process is depicted in Figure 2, which further explains that it is a modified version of the known Bloom filter algorithm.

207. To explain this figure in more detail, I would like to start by describing Bloom filters. Bloom filters were introduced in 1970 by Burton Bloom¹⁴⁸ and have been utilized for many tasks in computer science. Bloom filters provide a mechanism for *summarizing* the presence of items in a local database as a single long string of bits (each such long string of bits is called a “Bloom filter”). A user of a Bloom filter must choose two values: (1) the length of the Bloom filter in bits (e.g. 2048) and (2) a number n representing the number of bits set in the filter for each key (e.g. $n=3$).

208. A Bloom filter is constructed for a set of keys (such as GUIDs or database keywords) as follows: The Bloom filter starts with all of its bits clear (i.e. equal to zero). Then, for each key in the set, we execute a multi-valued hash function that chooses n bits in the Bloom filter to change from zero to one. The important point about the hash function is that the set of bits produced for each key

¹⁴ See EX1013.

is (1) apparently random, but (2) the same each time the same key is sent through the hash function. Each time that we add a new key to the Bloom filter we set more bits (some of which might have already been set).¹⁵

209. To use a specific example, one might imagine that the Bloom filter was 100 bits in length and that $n = 3$. In that example, the hash function would produce 3 different bits to set for each key (e.g. a given key might set bits 3, 37, and 87). Each key would produce a different (but deterministic) set of bits to set. When every key has been examined, the resulting string of 100 bits would be the Bloom filter. This string of bits is much smaller (in total size) than the original group of keys.

210. Once constructed, a Bloom filter is very easy to query. To see if a key *might* be in the database, one simply computes the hash function over the target key to generate the set of n bits, then checks to see if all n of those bits are actually set in the Bloom filter. *If all of the bits are set, then the key might be in the database . If any of the n bits are not set, then the key is definitely not in the*

¹⁵ In the limit, the Bloom filter would consist of all ones, after which it would clearly contain no useful information about the keys. An ideal point to target is one in which half of the bits are one (1). This represents the best tradeoff between length of the Bloom filter and accuracy of the retrieval.

database. The important point to note here is that a Bloom filter can generate a false-positive answer, since it is possible that all n bits are set, but they are set because two different key set those bits. Consequently, once one gets a positive answer from a query, one must actually check the data base to see if the key is there (a slower process).¹⁶ We say that the result of a query to the Bloom filter is probabilistically correct and will be less correct as the Bloom filter represents more and more keys.

211. Figure 2 can now be explained in more detail. The circles represent *nodes* in the system that contain documents (objects). The lines connecting them represent neighbor link for routing; nodes are connected if they are “near” one another in the network. Each node summarizes its local objects by constructing a Bloom filter from the GUIDs of its documents. In the figure, these *local Bloom filters* are represented by rounded boxes. Further, nodes pass their Bloom filters to their neighbors to generate the “routing information.” These *neighbor filters* shown as square (unrounded) boxes. Each node keeps one such neighbor filter for each immediate neighbor. For instance, node n_2 has 2 neighbors (n_3 and n_4) and keeps

¹⁶ The probability of a false positive can be reduced by increasing the size (number of bits) in the Bloom filter.

neighbor filters for each of them (see *4a* and *4b*). One can see that, *neighbor filter 4b* on m matches the *local filter* on n_3 .

212. If this simple technique were the only aspect of Figure 2, then a node could only make conclusions about whether or not a document was in its immediate neighbors. Instead, nodes combine and propagate filters of their neighbors to other neighbors. Consequently, node n_1 has a *neighbor filter* (labeled 2) with 2 components.¹⁷ The first (11100) represents documents at one hop and matches n_2 's *local filter*. The second (11009) represents documents at two hops and represents a combination of both n_3 's and n_4 's filters (combining 11008 and 00011 yields 11009, when the result has a bit set whenever a bit is set in either of the input filters). This process can be continued to provide information about nodes that are three or more hops away (in which case the neighbor filter would have more than two components). Of course, the combination of filters in this fashion tends to increase the number of ones in the result, thereby increasing the chance of a false-positive and *attenuating* (or making less) the accuracy of query results for

¹⁷ In this simplified graph, node n_2 has no nodes that are two hops away, so only has a neighbor filter with one component. Obviously this figure is simplified as much as possible for exposition.

higher hop counts. As a consequence, the OceanStore paper calls these “attenuated Bloom filters.” EX1006, p 193.

213. With respect to the ‘827 patent, these multilevel neighbor filters are location information that is utilized to forward queries from one node to another when data is not found successfully on the first node.

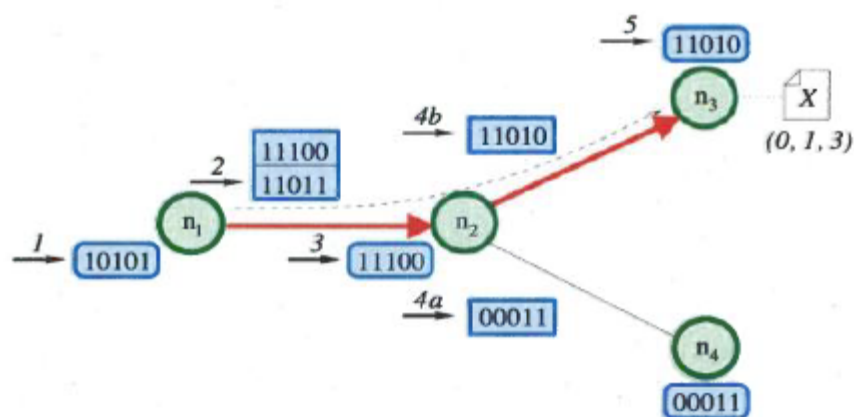


Figure 2: The probabilistic query process. The replica at n_1 is looking for object X , whose GUID hashes to bits 0, 1, and 3. (1) The local Bloom filter for n_1 (rounded box) shows that it does not have the object, but (2) its neighbor filter (unrounded box) for n_2 indicates that n_2 might be an intermediate node en route to the object. The query moves to n_2 , (3) whose Bloom filter indicates that it does not have the document locally, (4a) that its neighbor n_4 doesn't have it either, but (4b) that its neighbor n_3 might. The query is forwarded to n_3 , (5) which verifies that it has the object.

OceanStore, Fig. 2

214. Returning to the query shown in Figure 2, we start at node n_1 looking for document X (whose GUID produces bits 0, 1, and 3 when sent through the hash). When the query hits node n_1 , the local filter is examined and it is determined that X is not available locally (here, only bit 0 is set but not 1 or 3 - counting from the left). Then, the neighbor filter in the direction of n_2 is examined (step “2”) and

it is determined that X is definitely not one hops away (since 11100 is missing bit 3), but *might* be two hops away (since 11009 has all the right bits set). Thus, the query is directed to n_2 . At n_2 , the local filter is quickly queried and it is determined that X is not on the local node. Next, the neighbor filters are examined and it is seen that X is not on n_4 (00011 missing bits 0 and 1) but might be on node n_3 (11008 has bits 0, 1, and 3 set). So, the query is sent to n_3 where it is satisfied.

215. This local query algorithm is extremely fast (since hashing and checking bits is fast) and is able to find documents that reside within a small number of hops without excess network hops. Unfortunately, it is of limited scope: typical network graphs branch with distance, so each hop combines more and more Bloom filters together in the neighbor filter, setting more bits and increasing the false-positive rate precipitously. Further, propagating neighbor filters over long distances is expensive.

216. Consequently, the second data local algorithm operated with different goals: (1) it needed to find data that was an arbitrary number of hops away, (2) it needed to scale to all the world's data and (3) ideally it should still maintain some locality and not travel too far out of the way to find objects.

217. Its specific routing mechanism was different than the focal Bloom-filter-based algorithm and instead based off the known Plaxton algorithm for

locating replicas.¹⁸ Nodes connected with one another in a routing mesh that allowed routing one digit at a time to find a responsible node that contained location information for the requested GUID. The locality properties mentioned in the previous paragraph were achieved by arranging that each hop was geometrically longer (in network distance) and that location information was sprinkled around nodes close to objects to make sure that objects could be located in relatively few hops. Note that these properties come from the original Plaxton paper that was followed in making this algorithm work.

218. A specific example of a query is sketched in Figure 3 EX1006, p.194.

¹⁸ See EX1018.

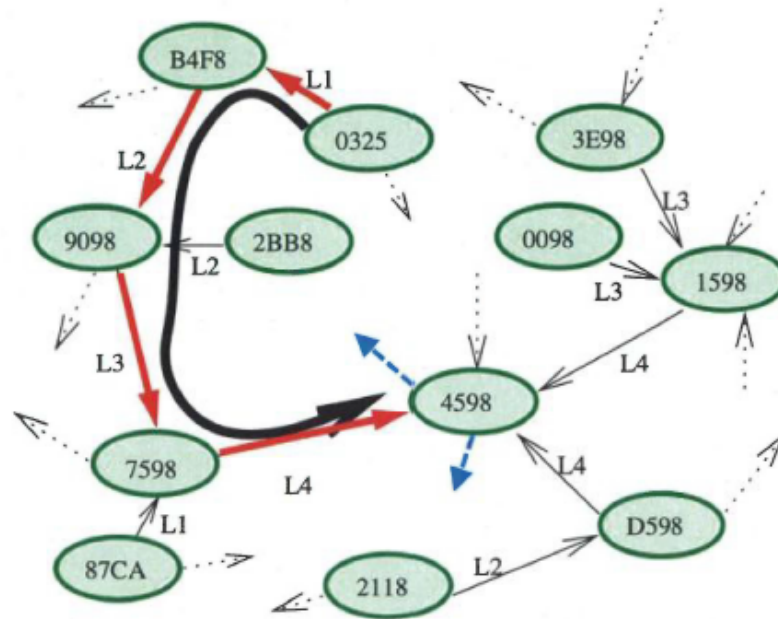


Figure 3: A portion of the global mesh, rooted at node 4598. Paths from any node to the root of any tree can be traversed by resolving the root's ID one digit at a time; the bold arrow shows a route from node 0325 to node 4598. Data location uses this structure. Note that most object searches do not travel all the way to the root (see text).

OceanStore, Fig. 3

219. As shown in this figure, the heavy arrow represents a query starting at a node whose name ends in “0325” and looking for a GUID that ends in “4598”.¹⁹ At each hop, the local node is queried to see if it contains *either* the requested data *or* location information to a node that has the data. If neither of these situations are true, then the query proceeds to the next hop that matches the requested GUID in more digits. Consequently, the first hop goes from 0325 → B4F8, which gains an

¹⁹ The number of digits in a real system is much larger, namely 10 hexadecimal digits, although the root is typically reached in less than 10 hops.

“8” in the last position; the next hop goes from B4F8 → 9098, which has “98” in the last position, etc. The last hop (to “4598”) matches the requested item in the most digits and is guaranteed (through other mechanism) to have location information that points to nodes that might host the desired object. This last hop is called the “root node” for that GUID and is the ultimate responsible node to track the location of the object (and copies of the object) represented by that GUID. The root would not store the object but would contain information pointing to the precise node which did hold the object See discussion on “data location” at bottom of EX1006, p.194.

ii. Claim 1

220. Claim 2 depends on claim 1. EX1001, Claim 2.

221. As described below, OceanStore renders obvious claim 1 of the ‘827 patent. I am informed by counsel that claims 1 and 13, which claims 2 and 14 depend upon, were invalidated as obvious over OceanStore in IPR2017-00374.

I(pre) “A computer-implemented method comprising ...”

222. I have been informed that this is a preamble term and does not limit the claim. Nevertheless, OceanStore teaches a “computer-implemented method.”

223. As the Board explained, OceanStore is a utility infrastructure designed for computers that are geographically dispersed and allows for continuous access to persistent information stored on remotely located storage devices. IPR2017-

00374, Paper 53 (EX1019) at 21 (citing EX1006, Abstract.) For the following reasons, I agree with the Board's explanation.

224. For example, Figure 1 of OceanStore teaches software that runs on servers (or "pools") in the network depicted in Fig. 1, reproduced above. In particular, it teaches a system "composed of a multitude of highly connected 'pools,'" comprised of individual caches of data on the aggregate collection of servers in the world. EX1006, 191. Further, "OceanStore is a utility infrastructure designed to span the globe [and is] comprised of untrusted servers." EX1006, Abstract.

225. Additionally, OceanStore discloses support of legacy computer coding languages by providing familiar computer interfaces such as the Unix file system interface. EX1006, 191.

1(a) "configuring a distributed processing system of a plurality of distributed devices coupled to a network, wherein the plurality of distributed devices include respective client agents configured to process respective portions of a workload for the distributed processing system"

226. OceanStore teaches this limitation. For example, OceanStore describes a widely-distributed network of devices including desktop workstations and servers with excess storage space to provide a "utility infrastructure designed to span the globe." EX1006, Abstract, 190-191, Fig. 1. The devices in OceanStore, known as "servers" or "nodes" each provide individual caches to the OceanStore network (i.e., the "ocean") and OceanStore clients access data by connecting to

one or more “pools” of nodes. *Id.* One embodiment disclosed in OceanStore indicates that the pools are also connected to each other, which allows data to flow freely between the pools. *Id.*

227. A person of skill in the art would have understood that the various server functionalities discussed in OceanStore (*e.g.*, assessing excess storage capacity, making that excess capacity available for use, handling of read and update requests for objects, routing of requests if needed) would be implemented by software that would be understood as an “agent” running on that networked, distributed device consistent with the way “client agents” are described in the ’827 patent. EX1006, 197-198; *see also, e.g.*, EX1001, 33:19-21, 41:54-57, 43:26-30.

228. Specifically, as I explained above, Figures 15 and 20 identify what a client agent should contain, including “communication interface” 1508 to communicate over the network, “core agent” 1502 to provide execution environment for the project and user interface components, “project component” 1504 that includes “task module 1512 [that] operates on top of the core agent module 1502 to provide processing of each work unit 1514,” and “user interface component” 1510. EX1001, 33:19-21, 41:54-57, Figs. 15, 20.

229. And as discussed in Section 5 of the OceanStore paper, the system is “written in Java with a state machine-based request model for fast I/O” (EX1006, 200). A person of skill in the art would appreciate that Java runs in a virtual

machine on top of the native operating system and connects to the network. The Java virtual machine can be directed to load and execute Java code (such as OceanStore); consequently, the entirety of the OceanStore code base would be a plugin to the Java virtual machine. Further, the Java virtual machine typically has an interface to users on the local machine to control and monitor execution. As discussed above, a POSITA would interpret this environment (the Java virtual machine) running the OceanStore code as a “client agent” as described in the ‘827 specification (including a “user interface” with users, even though that section argues that the user interface is optional).

230. The OceanStore application running on a distributed device will implement a Pool of data storage, as discussed above, to store and cache information. Further, it will implement the two-tiered data location algorithm to assist in the location of data objects. By participating in the OceanStore storage network, the distributed device will process storage workloads for remote users using the OceanStore application, which implements the OceanStore agent functionality and processes both read and write requests. EX1006, 197-198 (“The base [OceanStore] API provides full access to OceanStore functionality in terms of sessions” where each “sessions is a sequence of reads and write to potentially different objects . . .”). The OceanStore application is dedicated to providing NAS functionality to client devices. Accordingly, the OceanStore

agent is a “client agent” under either of the co-pending litigation constructions (“a program that manages the workloads and processes of a distributed processing system” or “a software program configured to run on each distributed device to provide what appears to client devices as dedicated NAS functionality”).

1(b) “wherein the respective client agents for particular distributed devices of the plurality of distributed devices have corresponding software-based network attached storage (NAS) components configured to assess unused or under-utilized storage resources in selected distributed devices of the plurality of distributed devices”

231. OceanStore teaches this limitation. A person of skill in the art would appreciate that the various server functionality discussed in OceanStore, including assessing excess storage capacity and making that excess capacity available for use by other networked devices, would be implemented by software on the networked, distributed devices. Further, a POSITA would appreciate that an OceanStore software-based component would assess the unused storage resources of the device (e.g., desktop workstation with excess storage space) when making this excess storage available for use by other devices on the network, as depicted by Figure 1. *See* EX1006, Fig. 1, 190, 191 (“In a time when desktop workstations routinely ship with tens of gigabytes of spinning storage, the management of data is far more expensive than the media to store it on. OceanStore hopes to take advantage of this excess of storage space to make the management of data seamless and carefree.”). In OceanStore, “[e]ach user ... could consume storage

and bandwidth resources from many different providers,” each configured with OceanStore software. EX1006, 190.

232. Additionally, as discussed above, the ’827 patent never mentions “assess” or describes any particular way in which the assessment is performed. And as the applicant explained during prosecution, the software must assess the unused storage in order to configure devices to mimic stand-alone NAS devices that use unused storage capabilities. *See* EX1002, 358 (“[I]t would have been obvious to one of ordinary skill in the art that the client agent programs must assess unused or under utilized capabilities for it to be able to configure the devices . . . using their spare and/or unused capabilities”).

233. Said another way, a person of skill in the art would also appreciate that portion of the device software that assesses and manages the use of the device’s excess storage capacity would correspond to the recited “software-based network attached storage (NAS) components.” This is so, because (a) that functionality is provided by software on, for example, the “desktop workstations” with the excess storage and (b) like the ’827 patent, that software makes the unused storage “available to network-connected user devices” (e.g., the user devices depicted in OceanStore Figure 1) consistent with the patent’s notion of a NAS device and (c) the fact that OceanStore is written in Java and runs in a Java virtual

machine meets the software architecture of a NAS component running as one of the processes in the “client agent.”

1(c) “representing with the corresponding software- based NAS component that the selected distributed devices respectively comprise NAS devices having an available amount of storage resources related to the unused or under-utilized storage resources for the selected distributed devices”

234. OceanStore teaches this limitation. For example, see above analysis for element 1[b]. In addition, the Ocean Store devices clearly represent, or mimic, real NAS devices. As described above, desktop workstations with excess capacity may be made to appear as, or mimic, network attached storage devices, to the various user devices including workstations, PDAs, and phones, for example, depicted in Figure 1.

235. As discussed in Ground 1, the ’827 patent defines a “NAS device” as a device that *makes data storage resources available* to network-connected user devices” and defines a “dedicated NAS device” as a device whose primary operational purpose is for *providing NAS services*. See EX1001, 2:31–34. In short, the OceanStore software makes the distributed devices in the “pools” and forming the “ocean” (including but not limited to “desktop workstations” with excess capacity) appear to be, or mimic, NAS devices having an available amount of storage related to the unused, or excess, storage for those distributed devices. EX1006, 191-199. It does so, by making available and providing those excess data

storage resources to other network connected devices, and by not appearing as mere workstations acting in isolation without providing any storage resources to other users. EX1006, 198. Specifically, the Board found, “[i]n OceanStore, multiple storage resources are provided by an aggregation of “pools” of storage, each of which can cache or serve data to the user devices” and “each pool device makes its storage available for use by others as depicted in Figures 2 and 3.” IPR2017-00374, Paper 53 (EX1019) at 33 (citing EX1006, 191-193). I agree with the Board’s analysis.

236. Thus, a person of skill in the art would appreciate that the OceanStore software causes the each pool devices to appear, function, and act as “a device that makes data storage resources available to network-connected user devices” as depicted in Figure 1 and described throughout the document. *See also* EX1001, 2:32-35. That is, the OceanStore software provided the functionality for the pool devices, such as desktop workstations with excess storage, to instead appear, function and act as storage servers on the network, providing storage resources to user devices connected to the network.

237. As discussed regarding limitation 1[b] above, as the applicant explained during prosecution, the OceanStore software must assess the unused storage in order to configure devices to mimic stand-alone NAS devices that use unused storage capabilities. *See* EX1002, 895; *see also supra*, ¶ 40. As explained

above, OceanStore provides at least similar or more disclosure than the '827 patent for this operation. And these unused storage resources are the resources that are made available by each of Oceanstore's pools-i.e., NAS devices—to the distributed devices.

238. Additionally, like the '827 patent, OceanStore discloses “facades” to its underlying client user that provide “a traditional, familiar interface,” including transactional database and Unix file system interfaces EX1006, 197-8; *compare* EX1001, 2:22-31 (“NAS devices often process only input/output (I/O) requests by supporting popular file sharing protocols such as NFS (UNIX) and SMB (DOS/Windows).”).

239. Further, OceanStore discloses potential applications for the utility service, including, email. EX1006, 192. OceanStore's introspection allows a user's email to migrate closer to his client, reducing the round-trip time to fetch messages from a remote server. *Id.* OceanStore also allow for users to operate on locally cached email even when disconnected from the network. *Id.* Further, OceanStore also allows a user to choose several directories as “roots” and secure those directories through external methods. *Id.*, 192, 199. A POSITA would have understood all of this to mean that OceanStore client devices see and utilize more than a single NAS device.

1(d) “processing one or more data storage or access workloads for the distributed processing system by accessing data from or storing data to

at least a portion of the available amount of storage resources to provide NAS service to a client device coupled to a network, and”

240. OceanStore teaches this limitation as discussed above in connection with element 1[b]. For example, as depicted in Figure 1 of OceanStore, various user devices (what the claim refers to as a “client device”), such as workstations, PDAs, and phones, may access OceanStore pools, caches or servers via network connections.

241. These user devices read and write objects to the devices contributing “pools” using OceanStore software. EX1006, 197-198 (“[The OceanStore] API provides full access to OceanStore functionality in terms of sessions ... A session is a sequence of reads and writes to potentially different objects Applications with more basic requirements are supported through facades to the standard API.... For example, a transaction facade would provide an abstraction atop the OceanStore API so that the developer could access the system in terms of traditional transactions ... OceanStore provides a number of legacy facades that implement common APIs, including the Unix file system, a transactional database, and a gateway to the World Wide Web. These interfaces exist as libraries or ‘plugins’ to the existing browsers or operating systems.”).

242. A POSITA would have understood that the read and write requests of these sessions are directed to the excess storage space of, for example, a desktop workstation that is part of Oceanstore’s “pool” of storage—i.e., the session read

and write requests are directed “to at least a portion of the available storage resources.” Moreover, these OceanStore sessions “provides NAS service to a client device” coupled to the OceanStore network as depicted in Figure 1. EX1006, 190 (Abstract); *see also id.*, 197-198 (API provides full access to functionality of sessions). And, as discussed above, “OceanStore provides a number of legacy facades that implement common APIs, including the Unix file system, a transactional database, and a gateway to the World Wide Web.” *Id.*

1(e) “enabling at least one of the selected distributed devices to function as a location distributed device to store location information associated with data stored by the selected distributed devices through use of the respective client agents for the particular distributed device”

243. The ‘827 patent states that the data location information on the distributed devices provide information “about where desired data is stored” across the distributed devices, and that this information can be distributed such that a distributed device contains the information, or a portion thereof, and that the information “can be organized as desired.” (See EX1001, 44: 1-9.) OceanStore teaches this limitation in at least three ways, since location information is spread throughout the OceanStore network.

244. First, the local, probabilistic location information described in OceanStore as an “attenuated Bloom filter” is “fully distributed.” EX1006, 193. Each node determines if it has the requested data, and if it does not, it routes the request to another node using locally stored routing information (the attenuated

Bloom filters). EX1006, 193 (“[I]f a query cannot be satisfied by a server, *local information is used to route the query* to a likely neighbor.” (emphasis added.)). Thus, in OceanStore the distributed devices acting as NAS devices can both hold data in available storage, and hold distributed location information to forward a request if the query cannot be satisfied. EX1006, 193 (“An attenuated Bloom filter is stored for each directed edge in the network.”); *compare with* EX1001, 44:9-12 (Some of the distributed devices providing software-based NAS services can provide “both storage and data location functionality, thereby acting both to provide data storage services and distributed data location services.”).

245. Second, OceanStore’s global “wide-scale distributed data location” algorithm teaches this limitation for similar reasons as the local algorithm. EX1006, 194. Although the internal details between the two data location algorithms differ, the global algorithm operates such that a node determines if it has the requested data, and if not, uses locally-stored routing information to forward the request to another node.

246. Third, stored objects themselves can include pointers to other versions of the object and thus are “location information associated with the data.” EX1006, 192 (“Certain OceanStore objects act as directories ... we allow directories to contain pointers to other directories.”); *see also id.*, 193 (“Every addressable entity in the OceanStore (e.g. floating replica, archival fragment, or client) is identified

by one or more GUIDs.... The role of the OceanStore routing layer is to route messages directly to the closest node that matches the predicate and has the desired GUID.”). A person of ordinary skill would appreciate these pointers include “global unique identifiers” (GUIDs) to read-only archival versions of objects, and GUIDs are information used to locate other versions of the object.

247. Accordingly, OceanStore anticipates claim 1.

iii. Claim 13

248. Claim 13, in short, is a system claim paralleling method claim 1. As described below, OceanStore teaches every limitation of claim 13 of the ‘827 patent for similar reasons to that discussed above in connection with claim 1.

13(pre) “A system comprising...”

249. Ocean Store is a system. *See* above discussion of limitation 1(pre).

13(a) “a plurality of distributed devices configured to be coupled to a network, wherein the distributed devices include respective client agents configured to process respective portions of a workload for the distributed processing system,”

250. Claim element 13[a] is satisfied for the same reasons as element 1[a].

13(b) “the respective client agents including respective instances of a software-based network attached storage (NAS) component wherein the NAS component is configured to: assess unused storage resources of the distributed devices”

251. Claim element 13[b] is satisfied for the same reasons as element 1[b].

13(c) “wherein the NAS component is configured to: ... allocate respective available amount of unused storage resources in selected distributed devices of the plurality of distributed devices; represent that

the selected distributed devices comprise NAS devices having the respective available amounts of storage resources”

252. Claim element 13[c] is satisfied for the same reasons as element 1[c].

Desktop workstations with excess storage and using OceanStore software would allocate the excess storage for use by OceanStore users (*i.e.*, making it available for other devices to use) but would reserve other workstation storage for purely local use, *e.g.*, to store computer programs, local operating system and the like.

253. As discussed above for element 1[c], the OceanStore software makes the distributed devices in the ‘pools’ and forming the ‘ocean’ (including but not limited to ‘desktop workstations’ with excess capacity) appear to be, or mimic, NAS devices having an available amount of storage related to the unused, or excess, storage for those distributed devices. It does so, by making available and providing those excess data storage resources to other network connected devices, and by not appearing as mere workstations acting in isolation without providing any storage resources to other users. EX1006, 198.

254. Therefore, a POSITA would have understood that the OceanStore software causes the pool devices to appear as “a device that makes data storage resources available to network-connected user devices’ as depicted in Figure 1 and described throughout the document.” EX1001, 2:32-35. Moreover, a POSITA would have understood that you need to allocate data in order to represent that you have an available amount of unused or under-utilized storage.

255. Additionally, during IPR2017-00374, Patent Owner’s expert, Dr. Shenoy, never contested that “allocating” available storage for particular uses was well-known and entirely conventional, and further, Dr. Shenoy did not identify “allocating” as a distinctive or inventive aspect of the claims. IPR2017-00374, EX1053, 38:6-15, *id.* 83:4-18. And the Board agreed, stating that the purpose of OceanStore is to provide storage space in servers around the world, allowing data to be “cached anywhere” (EX1006, 190 (Abstract)), and credited my opinion that a POSITA “would clearly understand that you need to allocate data in order to represent that you have an available amount of unused or under-utilized storage.” IPR2017-00374, Paper 53 (EX1019) at 34-37.

13(d) “wherein the NAS component is configured to: ... represent that the selected distributed devices comprise NAS devices having the respective available amounts of storage resources;”

256. Claim element 13[d] is satisfied for the same reasons as element 1[c].

13(e) “wherein the NAS component is configured to: ... process workloads associated with data storage and access by accessing data from and storing data into portions of the storage resources in the selected distributed devices to provide NAS service to a client device; and”

257. Claim element 13[e] is satisfied for the same reasons as element 1[d].

In addition, a POSITA would clearly understand that you need to allocate data storage resources in order to represent that you have an available amount of unused or under-utilized storage; the process of allocating data resources involves reserving (marking) it so that it will not be utilized by other consumers of space.

13(f) “wherein the respective client agents are configured to enable at least one of the selected distributed devices to function as a location distributed device to store location information for data stored by the selected distributed devices”

258. Claim element 13[f] is satisfied for the same reasons as element 1[e].

iv. Claims 2 and 14

259. As previous noted, claim 2 depends on claim 1 and recites “further comprising enabling at least one of the selected distributed devices *to function as a stand-alone dedicated NAS device* through use of the respective client agents for the particular distributed device.” Claim 14 is of similar scope but is a system claim.

260. The Board in the prior IPRs found that “[i]n OceanStore, multiple storage resources are provided by an aggregation of “pools” of storage, *each of which* can cache or serve data to the user devices.” IPR2017-00374, Paper 53 (EX1019) at 33 (emphasis in original), citing EX1006, 191. The Board also noted that “*each pool device* makes its storage available for use by others as depicted in Figures 2 and 3....” *Id.*, citing EX1006, 192, 193. Moreover, OceanStore discloses “facades” to its API, that provide “a traditional, familiar interface,” including transactional database and Unix file system interfaces. EX1006, 197-198. Since each OceanStore server device satisfies requests for data over the network directly to clients, they “**function as dedicated NAS devices.**” Further, “OceanStore allows a user’s email to migrate closer to his client and use locally cached email,

thereby indicating that the OceanStore clients see more than a single NAS device.”

Id., citing EX1006, 192, 199.

261. A POSITA would appreciate the OceanStore software causes the pool devices to appear as “a device that makes data storage resources available to network-connected user devices.” Indeed, as the Board found regarding claim 1, the OceanStore software running on each node would be considered the “client agent” that provides the required NAS device functionality.²⁰ *Id.*, 22-23, citing EX1006, 191. Therefore, the OceanStore software enables a node to function as a stand-alone dedicated NAS device (“a distributed device that separately appears to network connected user devices as a dedicated NAS device”).

262. As discussed in Section VI.B., I understand that Patent Owner’s co-pending litigation construction of the term “stand-alone dedicated NAS devices” is “a dedicated NAS device that contains its own storage, processing, connectivity, and management resources.” For the reasons discussed above and below, the

²⁰ As I discuss in limitation 1(a) above, the OceanStore agent is a “client agent” under either of the co-pending litigation constructions (“a program that manages the workloads and processes of a distributed processing system” or “a software program configured to run on each distributed device to provide what appears to client devices as dedicated NAS functionality”).

OceanStore-disclosed stand-alone dedicated NAS device is clearly a device that contains its own storage, processing, connectivity, and management resources. It is also a dedicated NAS device (“a device whose operational purpose is for providing NAS services”) for at least the reasons discussed above and in element 1(c).

263. I note that the ‘827 patent describes an optional feature of a stand-alone dedicated NAS device in that it may act without knowledge of other NAS devices connected to the network. EX1001, 3:17-21; *see also id.*, 43:47-52 (“In this first implementation, the NAS devices 108A, 108B...108C run client agent programs that configure the devices to be capable of mimicking dedicated stand-alone NAS devices” in which “[e]ach of these software-based NAS devices ... knows nothing of other software-based NAS or other NAS devices.”) Also, the ‘827 equates “knowing *nothing* of other software-based NAS devices” with at least the lack of data location libraries 2106A and 2106B, which contain “distributed data location information.” EX1001, 43:30-33, 4:56-58. A POSITA would understand that lacking any means with which to *interpret* data location information—even if stored locally, satisfies “knowing nothing about other NAS devices,” to the extent it is a requirement of a “stand-alone dedicated NAS device.”

264. Additional aspects of Oceanstore also meets this limitation of disclosing devices which “function as dedicated standalone NAS devices.” The

Oceanstore System Architecture is partitioned into a variety of components, including at least (1) storage services (also called “Pool servers”) and (2) data-centric DOLR networking (for Data Location and Routing). A “pool server” contains its own “storage, processing, connectivity and management resources.” But it is the *networking* component which has knowledge and ability to locate data (acting as a data location service):

265. “Entities in the OceanStore are free to reside on *any* of the OceanStore servers. This freedom provides maximum flexibility in selecting policies for replication, availability, caching, and migration. Unfortunately, it also complicates the process of locating and interacting with these entities. Rather than restricting the placement of data to aid in the location process, OceanStore tackles the problem of data location head-on. The paradigm is that of query routing, in which the network takes an active role in routing messages to objects.”

EX1006, 193 (Section 4.3).

266. The networking components of the OceanStore system comprise a distributed data location infrastructure *on top* of the standard IP (“Internet Protocol”) infrastructure. Consider EX1006, 193 (Section 4.3.1): (“Messages begin by routing from node to node along the distributed data structure until a destination is discovered. At that point, they route directly to the destination. It is

important to note that the OceanStore routing layer does not supplant IP routing, but rather provides additional functionality on top of IP.”)

267. Consequently, the OceanStore storage servers do *not* have knowledge of other storage servers (NAS devices) merely of the local data that they are serving to the network. If a particular storage server does not have a requested piece of data, they can only respond with a NAK (or “Negative Acknowledge”, *i.e.* “do not have requested data”). It is up to the network routing and location component to find a NAS device with the queried data. Consider, for instance, the probabilistic query process, discussed earlier and reprised as Figure 2 below (EX1006, 193, Fig. 2) that shows that it is the network routing component which has knowledge of data location information, not the servers storing replicas (here shown as circles with node IDs within):

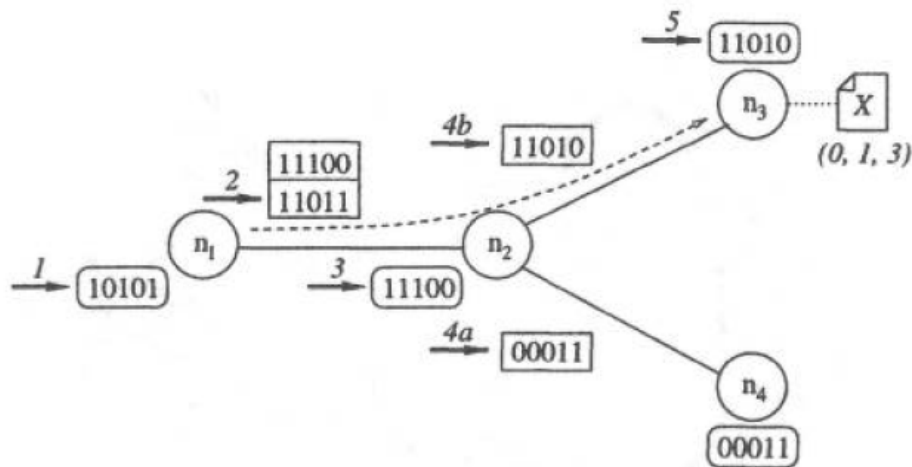


Figure 2: The probabilistic query process. The replica at n_1 is looking for object X , whose GUID hashes to bits 0, 1, and 3. (1) The local Bloom filter for n_1 (rounded box) shows that it does not have the object, but (2) its neighbor filter (unrounded box) for n_2 indicates that n_2 might be an intermediate node en route to the object. The query moves to n_2 , (3) whose Bloom filter indicates that it does not have the document locally, (4a) that its neighbor n_4 doesn't have it either, but (4b) that its neighbor n_3 might. The query is forwarded to n_3 , (5) which verifies that it has the object.

268. Note further that the Global Algorithm (EX1006, 194, Section 4.3.3 and Fig. 3) is also disclosed as a distributed data routing and location infrastructure independent of the storage servers.

269. Separating the data location service from the pools is more efficient because “the combination of location and routing permits communication with ‘the closest’ entity, rather than an entity that the client might have heard of in the past.” EX1006, 193. In fact, OceanStore implements a “two-tiered approach” for query routing that first uses a “probabilistic algorithm” to “route to entities rapidly if they are in the local vicinity” and if this attempt fails, a “hierarchical data structure” is

used to “locate[] entities that cannot be found locally.” *Id.* OceanStore’s probabilistic algorithm is implemented as an “attenuated Bloom filter” that uses “local information . . . to route the query to a likely neighbor. *Id.* And OceanStore’s hierarchical data structure is implemented as “a variation on Plaxton’s et al.’s randomized hierarchical distributed data structure, which embeds multiple random trees in the network.” *Id.* at 194. “[E]very server in the system is assigned a random (and unique nodeID) and these “node-IDs are then used to construct a mesh of neighbor links, as shown in Figure 3” where “each link is labeled with a level number that denotes the stage of routing that uses the link.” *Id.*; Fig. 3. These “links form a series of random embedded trees [as shown in Fig. 3], with each node as the root of one of these trees.” *Id.* “As a result, the neighbor links can be used to route from anywhere to a given node, simply by resolving the node’s address one link at a time.” *Id.*

270. As disclosed, the OceanStore implementation involves a series of software components that interact with one another to form the OceanStore system. Consider: “The system is written in Java with a state machine-based request model for fast I/O [22].” Ex 1006, 199 (Section 5). Further, the component-based architecture is explicitly disclosed for the probabilistic routing mechanism, making it clear that data location operates independently of data serving. *See Id.* (“A prototype for the probabilistic data location component has been implemented and

verified.”). A POSITA would understand that a consequence of the component-based architecture and the fact that the OceanStore system is a *distributed system* operating over IP is that any *particular physical node* would implement one or more components – not necessarily all of them. *In particular, it becomes a configuration choice as to whether a physical pool storage server (NAS device) with an OceanStore storage server component would also host a network routing and location component.*

271. A POSITA would understand that OceanStore’s nodes represent devices coupled to the network, each having its own storage, processing, connectivity, and management resources, and would further understand the performance advantages of specialization: optimizing physical components by restricting their functionality. In particular, a physical node that performed only NAS services without data location services would “*function as a stand-alone dedicated NAS device*”.

272. OceanStore also explicitly discloses untrusted servers which meet this limitation, e.g.: “Airports or small cafes could install servers on their premises to give customers better performance; in return they would get a small dividend for their participation in the global utility.” EX1006. 190-191. Since they would be owned by small organization, such servers would be optimized to serve data to

clients as efficiently as possible and without being overburdened by constant participation in the data location infrastructure.

273. Moreover, the ‘827 patent itself states that it is a “*relatively simple implementation* [] to have the devices simply *mimic the functionality of stand-alone NAS devices*.” EX1001, 42:66–43:1 (emphases added). A POSITA would consider a “relatively simple implementation” an obvious variant of OceanStore even if the disclosure were not so explicit. This analysis applies equally to claim 14.

D. Motivation to Combine Carter and Gibson

274. A POSITA would have been motivated to combine Carter with Gibson. First, both Carter and Gibson relate to the same filed of endeavor, i.e., a network system that shares resources available at distributed devices on the network. EX1005, Abstract (“The globally addressable storage environment allows data to be accessed and shared by and among the various computers on the plurality of networks.”); EX1007 (“Users are increasingly using distributed file systems to access data across local area networks” and “[n]etwork-attached storage provides high bandwidth by directly attaching storage to the network, avoiding file server store-and-forward operations and allowing data transfers to be striped over storage and switched-network links.”).

275. Second, both Carter and Gibson are directed to solving problems related to the efficiency, scalability, and data sharing capacity of distributed networks. See, e.g., EX1005, 4:2-11 (disclosing improving “network performance and productivity,” “fault tolerance,” and capacity “to communicate and share data in a transparent and dynamic manner,” of distributed networks as objects of the invention); EX1007, 272 (disclosing providing users of distributed file system architectures “with efficient, scalable, high-bandwidth access to stored data” as objects of the invention). A POSITA would therefore have sought out various available solutions, including those described in Carter and Gibson, for improving the efficiency, scalability, and data sharing capacity of distributed networks.

276. Carter solves the problem in part by creating and managing a virtual storage space that aggregates the unused storage resources of many storage devices (e.g., RAM and hard disk) connected to the network, including traditionally non-addressable data storage devices (e.g., desktop computers). See, e.g., EX1005, Abstract, 4:15-17. Carter also discloses additional network-attached storage (NAS) devices beyond the aggregated virtual storage space—such as network memory device 226 illustrated in Fig. 6—that can also be used by client devices to store or access data. EX1005, 29:21-30:2; Fig. 6. Gibson, expands on the use of NAS devices by describing various examples of standalone dedicated NAS devices such as “Network-attached Secure Disks (NASD)” that “provide[] high bandwidth

by directly attaching storage to the network” and “reduce[] the number of client-storage interactions that must be relayed through the file manager.” EX1007, 272, 275, Fig. 3. For example, Gibson’s NASDs allow “common, data-intensive operations, such as reads and writes, [to] go straight to the disk...while policy decisions are made in the file manager.” *Id.* at 275. This improves the scalability of networks by providing more efficient access to a greater volume of storage while significantly reducing file manager costs per client. EX1007, 281 (Section 6.2), 282 (Section 7) (“Network attached storage...can substantially increase distributed file system scalability while simultaneously enabling striped storage to satisfy the bursty, high-bandwidth demands of the increasingly high performance clients populating local area networks.”)

277. Third, it would have been obvious to a POSITA at the time of the ‘827 patent’s alleged invention to combine the teachings of Carter with Gibson. A POSITA would have appreciated that Gibson’s NASDs—i.e., stand-alone dedicated NAS devices (*see* Section VII.E.ii below)—would have been easily incorporated into Carter by directly attaching the NASDs to Carter’s network similar to Carter’s memory device 226 illustrated in Figure 6. This combination would have yielded the predictable result of expanding the scalability of Carter’s networks by providing more efficient access to a greater volume of storage through Gibson’s NASDs operating in parallel with Carter’s aggregated shared memory

space. And a POSITA would have had a reasonable expectation of success as this combination reflects nothing more than a predictable combination of known elements that would have been obvious to try, required nothing more than ordinary skill, and would have worked in an expected way.

E. Ground 3: Claims 2 and 14 are Obvious over Carter in view of Gibson

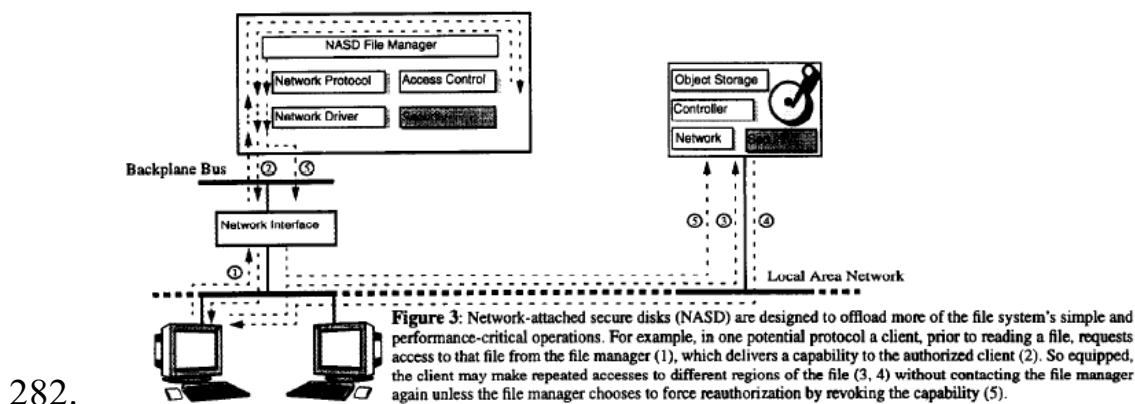
i. Overview of Gibson

278. Gibson is a publication and prior art to the ‘827 patent under 35 U.S.C. § 102(b) for the ‘827 patent’s proper priority date of April 13, 2001. It published in the Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics ‘97), Seattle, Washington, June 15-18, 1997. EX1007. It was not considered by the Patent Office during prosecution of the ‘827 patent. EX1001.

279. Gibson is titled “File Server Scaling with Network-Attached Secure Disks,” and discloses providing direct data transfer between storage and client using network-attached storage devices to improve scalability for existing distributed file systems and bandwidth for new parallel and distributed file systems. EX1007, 1 (Abstract, 272). Gibson states that it’s “principal contribution” is to demonstrate the potential of network-attached storage devices.” EX1007, 1 (Section I, Introduction, 272).

280. Gibson discloses “[n]etwork-attached storage provides high bandwidth by directly attaching storage to the network, avoiding file server store-and-forward operations and allowing data transfers to be striped over storage and switched-network links.” *Id.* More specifically, Gibson describes “two network-attached storage architectures-(1) Networked SCSI disks (NetSCSI) are network-attached storage devices with minimal changes from the familiar SCSI interface, while (2) Network-Attached Secure Disks (NASD) are drives that support independent client access to drive object services.” EX1007, 1 (Abstract, 272).

281. Section 3.4 of Gibson details the operation of network-attached secure disks, which are stand-alone devices, as shown below in Figure 3. EX1007, 4-5 (Section 3.4; 275-276).



283. EX1007, 5 (Fig. 3); see also *id.*, 5-8 (Section 3.5 “Summary”; Sections 4-4.4 “Analysis of a File System Workload”, 276-279). Gibson explains that network-attached secure disks offload “[c]ommon, data intensive operations,

such as reads and writes” from the file manager. EX1007, 275. Gibson also provides an example operation, stating that “[b]efore issuing its first read of a file, the client authenticates itself with the file manager and requests access to the file.” *Id.*, 276. But once access is granted, the client may access the NASD drive containing the file directly until the client’s access is revoked. *Id.*

284. Among other things, Gibson highlights the performance advantages of separating the organizational elements (e.g. file system management) from the raw NAS data access, with the NASD option of Fig. 3 showing the highest performance. It concludes, “Network-attached storage, by enabling direct transfers between client and storage, can substantially increase distributed file system scalability while simultaneously enabling striped storage to satisfy the bursty, high-bandwidth demands of the increasingly high-performance clients populating local area networks.” *Id.*, 11 (Section 7, “Conclusion and Future Directions”).

ii. Claims 2 and 14

285. Again, claim 2 depends on claim 1 and recites “further comprising enabling at least one of the selected distributed devices *to function as a stand-alone dedicated NAS device* through use of the respective client agents for the particular distributed device.” Claim 14 is of similar scope but is a system claim. Gibson discloses or renders obvious the limitation of claims 2 and 14.

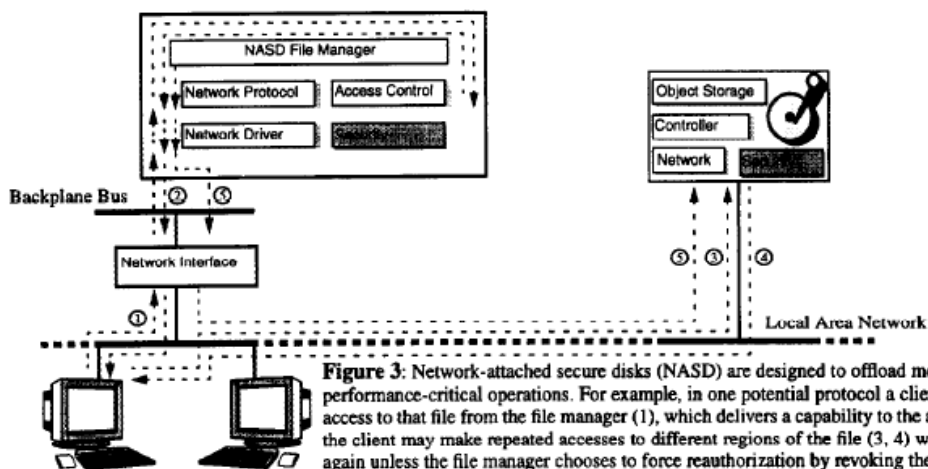
286. Gibson states that it's "principal contribution" is to "demonstrate the potential of network-attached storage devices." EX1007, 272. Gibson discloses NAS devices that meet the limitation of claim 2. "Network-attached storage provides high bandwidth by directly attaching storage to the network, avoiding file server store-and-forward operations and allowing data transfers to be striped over storage and switched-network links." *Id.*

287. Gibson describes "two network-attached storage architectures-(1) Networked SCSI disks (NetSCSI) are network-attached storage devices with minimal changes from the familiar SCSI interface, while (2) Network-Attached Secure Disks (NASD) are drives that support independent client access to drive object services." EX1007, 1 (Abstract, 272).

288. Section 3.4 of Gibson details the operation of network-attached secure disks, which are "stand-alone" devices, as shown in Figure 3, at least because each contains its own storage, processing, connectivity, and management resources. EX1007, 4-5 (Section 3.4; 275-276); *see also id.*, 5-8 (Section 3.5 "Summary"; Sections 4-4.4 "Analysis of a File System Workload", 276-279). As shown below in Figure 3, each of Gibson's NASDs is a "stand-alone dedicated NAS device" as each NASD makes data storage resources available to client devices over the local area network, the primary purpose of each NASD is to provide NAS service, and each NASD "separately appear[s] to network-connected user devices as a

dedicated NAS device.” EX1007, 275-276; Fig. 3 (reproduced below).

Accordingly, Gibson’s NASD is a “stand-alone dedicated NAS device” under either of the co-pending litigation constructions (“a dedicated NAS device that contains its own storage, processing, connectivity, and management resources” or “a distributed device that separately appears to network connected user devices as a dedicated NAS device”).



289. Moreover, Gibson discloses that each NASD functions as a stand-alone dedicated NAS device accessed directly by the client (i.e., recited “client agent”) by first consulting the “file manager”—similar to the use of the “responsible node” of Carter. For example, Gibson discloses that “[b]efore issuing its first read of a file, the client authenticates itself with the file manager and requests access to the file.” EX1007, 276. After access is granted, the client may directly access the NASD drive containing the file until the client’s access is revoked. EX1007, 276.

290. Subsequent investigation in the paper clearly details the performance advantages of providing NAS devices which are optimized to directly serve RAW data blocks (without interpretation) over the network. *Id.* 8 (Section 5 “Analytical Model”) and 8-11 (Section 6, “Replay Experiment”).

291. As discussed in Section VII.D, a POSITA would have been motivated to combine Gibson’s NASDs with Carter as both relate to the same field of endeavor and both disclose the implementation of NAS devices to improve efficiency, scalability and data sharing capacity of distributed networks. Moreover, a POSITA would have been motivated to expand Carter’s network-attached storage (NAS) devices (*e.g.*, network memory device 226 illustrated in Fig. 6) by implementing Gibson’s NASDs to work in parallel with Carter’s aggregated virtual storage space. Gibsons’ NASDs would attach directly to Carter’s network similar to memory device 226 shown in Fig. 6 and the NASDs would be managed by Carter’s “client agent” software package, which would be expanded to include Gibson’s file manager functionality. EX005, Fig. 6; EX1007, Fig. 3 (illustrating NASD access over LAN and describing file manager). A POSITA would have recognized that this combined system would realize the benefits of improving network efficiency, scalability, and data sharing capacity as taught by each of Carter and Gibson in a single combined system.

292. Moreover, combining the teachings of these references would have involved nothing more than the use of known features to yield predictable results. Indeed, the ‘827 patent itself states that it is a “relatively simple implementation [] to have the devices simply mimic the functionality of stand-alone NAS devices.”

EX1001, 42:66–43:1. The ‘827 patent states that:

The NAS (Network Attached Storage) market is currently one of the fastest growing segments of the overall storage market. NAS devices typically take the form of stand-alone devices or systems that contain their own storage, processing, connectivity and management resources. Through these internal capabilities, the NAS device becomes a storage resource that connects to any network enabling many devices on the network to share data via the NAS device.

A NAS device is a term that is often used to refer to a specialized file server that connects to the network to provide storage capacity to network-connected users. NAS devices typically contain a slimmed-down (micro-kernel) operating system and file system. NAS devices often process only input/output (I/O) requests by supporting popular file sharing protocols such as NFS (UNIX) and SMB (DOS/Windows). NAS devices also typically use traditional LAN protocols such as Ethernet and TCP/IP to communicate over the network. Some general-purpose computers using full-blown operating systems such as Windows or UNIX are often labeled as NAS products because they can provide storage that is attached to a network. However, a pure NAS device is one built from scratch as a dedicated file I/O device. As used herein, the term “NAS device” broadly refers to a device that makes data storage resources available to network-connected user devices. A “dedicated NAS device” is a device whose primary operational purpose is for providing NAS services.

A related term is the storage area network (SAN). “SAN” is a term that is often used to refer to a back-end network connecting storage devices to servers via peripheral channels such as SCSI, SSA, ESCON and Fibre Channel. By using a SAN, the data storage traffic

is off-loaded from the primary user network, which includes the user machines and the server machines. The servers then utilize the SAN to move and store data in storage devices connected to the storage network. SAN environments are becoming increasingly popular for enterprise storage solutions.

With respect to lower cost devices that are currently marketed as dedicated NAS devices, many have common personal computer (PC) specifications such as the following: Pentium III processor, 40 GB HD and a 10/100 Ethernet connection, along with other components. Considering the specifications of an average corporate desktop computer, the capabilities of these desktop computers often exceed the capabilities computers being sold as dedicated NAS devices, and these desktop computers often sit idle for significant amounts of time each day.

EX1001, 2:10-55.

293. A POSITA thus would have been motivated to combine Gibson's NASDs with Carter's disclosed system, and would have had a reasonable expectation of success.

VIII. AVAILABILITY FOR CROSS-EXAMINATION

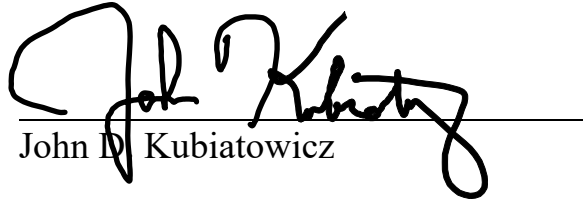
294. In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

IX. RIGHT TO SUPPLEMENT

295. I reserve the right to supplement my opinions in the future to respond to any arguments that the Patent Owner raises and to take into account new information as it becomes available to me.

I declare 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 full 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 12, 2025



John D. Kubiak