UNITED STATES DISTRICT COURT
DISTRICT OF NEW HAMPSHIRE

MANGOSOFT, INC. -and- MANGOSOFT CORPORATION, Plaintiffs, v. ORACLE CORPORATION, Defendant.	Case No. C 02-545 - JM **PLAINTIFF MANGOSOFT, INC. AND MANGOSOFT CORPORATION'S OPENING CLAIM CONSTRUCTION BRIEF**

Plaintiffs, Mangosoft, Inc. and Mangosoft Corporation (together "Mangosoft") respectfully submit this brief in support of their proposed claim construction in this patent case. For the reasons set forth below, Mangosoft requests that the Court adopt the meaning of the disputed claim terms proposed by Mangosoft.

I. THE PATENTS-IN-SUIT

Mangosoft alleges that defendant, Oracle Corporation ("Oracle"), infringes numerous claims of Mangosoft's US. Patent Nos. 6,148,377 ("the '377 patent") and 5,918,229 ("the '229 patent"). The Abstract of the '377 patent describes generally the subject matter of these patents:

> Distributed shared memory systems and processes that can connect into each node of a computer network to encapsulate the memory management operations of the connected nodes and to provide thereby an abstraction of a shared virtual memory that can span across each node of the network and that optionally span across each memory device connected to the computer network. Accordingly, each node on the network having the distributed shared memory system of the invention can access the shared memory.

'377 patent at Abstract. In other words, the asserted patents relate to networking a plurality of computers and configuring the computers so that memory can be shared among the plurality of computers through a shared memory system.

The patent claims asserted against Oracle are directed to computer systems consisting of groups of computers interconnected by a network connection. These systems are commonly known as a "cluster" or a "computer cluster." Each of the computers, or "nodes," of the cluster is responsible for managing its own memory resources, which includes Random Access Memory (RAM) and disk memory, and in accordance with the invention, making the data stored in its memory resources available to the other nodes of the cluster. As described in the patents, this function is accomplished using the concept of a memory space that is shared among the nodes of the cluster. The patents also describe a subsystem that manages the shared memory space and the relationship between the shared memory space and memory resources of the nodes in the cluster.

The memory resources of each node of a cluster typically include RAM and disk memory. One advantage of RAM is that it usually provides higher speed access to data than disk memory. However, RAM is generally understood to be volatile memory, i.e., it does not have the ability to store data when power is removed. Disk memory is generally understood to be persistent, i.e., it maintains a store of data after power is removed. But disk memory usually provides much slower access to data than does RAM. In prior systems, files and data are typically stored on disk memory and accessed by multiple users. In order to make access more efficient, a copy of frequently accessed data can also temporarily be held, or buffered, in RAM. When a user seeks to access a piece of data from a node, the node initially looks to see if the piece of data is buffered in its own RAM. If the data is buffered in its own RAM, the node will provide the data to the user on that node from the faster RAM memory. If the piece of data is not buffered in its own RAM, the node will access the piece of data from disk memory to which it has access.

In contrast, according to the'229 and '377 patents, if the piece of data is not buffered in the node's own RAM (the "local" RAM), the node looks to see if the data is buffered in another node's RAM (the "remote" RAM). If the piece of data is buffered in the remote node's RAM, then the local node will request that the data be sent from the remote node to the local node. The local node will access the data from disk memory only if the data is not buffered on any node's RAM.

III. "LOCAL … MEMORY DEVICE"

The claims of the '377 patent and the '229 patent use the terms "local volatile memory device" and "local persistent memory device" as follows in the exemplary claim elements:

> a local volatile memory device coupled to said computer and having volatile storage for data signals
>
> and
>
> a local persistent memory device coupled to said computer and having persistent storage for data signals.

'377 patent at 16:l-4.

Mangosoft proposes that these terms be construed to mean "a memory device that loses data when power is disconnected from the device, a portion or the whole of which can be contributed to the shared addressable memory space by a particular node" and "a memory device that does not lose data when power is disconnected from the device, a portion or the whole of which can be contributed to the shared addressable memory space by a particular node," respectively. Oracle, on the other hand, proposes that the term "local . . . memory device" be construed to mean "a memory device directly attached only to one node." As set forth below, Mangosoft's construction flows directly from the plain language of the claims, as confirmed by the specification.

The language of claim 1 of the ‘377 patent states plainly: “a local volatile memory device **coupled** to said computer and having volatile storage for data signals.” ‘377 patent at claim 1 (emphasis added). The term “coupled” is not ambiguous. Thus, there is no need to construe this term, let alone to construe it, as Oracle proposes, to add limitations found nowhere in the claims or the patent. Indeed, Oracle’s contention that the term “local ... memory device” should be construed to limit the invention to volatile and persistent memory devices that are **directly attached** to a single computer or node is inconsistent with the plain language of the claims.

The term “coupled” has been held to mean only “to link or join.” *See Level One Communications, Inc. v. SEEQ Technology, Inc.*, 987 F. Supp. 1191, 1196 n. 1 (N.D. Cal. 1997) (citing the Random House College Dictionary, “‘[c]ouple’ should be given its ordinary meaning of to link or join”). As the Federal Circuit has also held, “the unmodified term ‘coupled’ generically describes a connection, and does not require a mechanical or physical coupling.” *Johnson Worldwide Associates, Inc. v. Zebco Corp.*, 175 F.3d 985, 992 (Fed. Cir. 1999). Nothing in these definitions suggests the “direct attachment” requirement that Oracle now proposes.

Moreover, the specifications and file histories confirm that “local … memory devices” need only be “coupled” to the node and not direct attached. For example, Figure 2 of the ‘377 patent and Figure 6 of the ‘229 patent disclose a persistent network memory device, 26 and 226, respectively, that is not directly attached only to one node. With respect to Figures 2 and 6, the specifications state:

> The memory subsystems 32a-32c [and 232a-232c] can be software modules that act as distributors to map portions of the addressable

> memory space across the depicted memory devices. The memory subsystems further track the data stored in the local memory of each node 12 [and 212] and further operate network connections with network 38 [and 238] for transferring data between the nodes 12a-12c [and 212a-212c]. In this way, the memory subsystems 32a-32c [and 232a-232c] access and control each memory element on the network 38 [and 238] to perform memory access operations that are transparent to the operating system 16 [and 216].

‘377 patent at 7:24-34; ‘229 patent at 19:52-62. Figures 2 and 6 also show that the network memory device, 26 and 226, includes portions of the shared addressable memory space Cm, Cp, Ct, mapped thereon the same way the other local memory devices 36a-36b and 326a-326c show the addressable memory space mapped thereon, consistent with the portions of the specifications cited above.

In addition, the ‘229 patent provides an example in which the local persistent memory is not directly attached only to one node:

> For example, a portion of the addressable shared memory space 20 can be assigned or mapped to one or more hard disk drives that are on the network or associated with one or more of the network nodes 12a-12d as local hard disk storage for those particular nodes.

‘229 patent at 6:45-50. The above example demonstrates that a single hard disk drive, on the network or associated with one or more of the nodes, is considered “local” to its associated nodes, thus supporting Mangosoft’s proposed construction.

IV. CONCLUSION

Because, as demonstrated above, Mangosoft’s proposed construction of the term “local … memory device” is most closely aligned with the plain meaning of the claim terms and the intrinsic record, Mangosoft’s proposed construction should be adopted.

UNITED STATES DISTRICT COURT
DISTRICT OF NEW HAMPSHIRE

MANGOSOFT, INC.
-and-
MANGOSOFT CORPORATION,

Plaintiffs,

v.

ORACLE CORPORATION,

Defendant.

Case No. C 02-545 - JM

DEFENDANT ORACLE CORPORATION'S CLAIM CONSTRUCTION BRIEF

INTRODUCTION

Plaintiffs Mangosoft, Inc. and Mangosoft Corporation (collectively "Mangosoft") have accused an Oracle database software product called Real Application Clusters ("RAC") of infringing a patent related generally to shared memory in computer networks. The patents at issue are U.S. Patent No. 6,148,377 ("the '377 patent"), and its continuation-in-part, U.S. Patent No. 5,918,229 ("the '229 patent"). Consistent with the process adopted by the Court and set forth in the Pretrial Scheduling Order, the parties have identified the terms "local volatile memory device coupled to the node and providing volatile storage" and "local persistent memory device coupled to the node and providing persistent storage" for construction.

THE PATENTS-IN-SUIT

The '377 and '229 patents are directed generally to computer systems that allow multiple computers to share memory resources. Shared memory systems were well known in the art in 1996. For example, the Background of the Invention section of the '377 patent discusses prior art "client/server" systems, in which "client" computers access data stored on a central shared disk or "server." See '337 patent at 1:9-29. As noted in the patent, each client computer typically includes private main memory (RAM) and a hard disk. See id. at 1:13-17. As the patent explains, one of the limitations of these prior art client/server systems was that all shared data was stored and managed on a central server, and thus such systems failed to exploit the physical memory devices located on the client computers (e.g., each computer's RAM and disk), and were subject to complete system failure if the server system crashed. See id. at 1:35-63.

The system claimed in the patents-in-suit attempts to overcome these limitations by providing a shared memory space that is distributed across the computers on the network. As the above passage notes, the shared virtual memory space of the '377 patent spans the local physical memory devices of the network. See '377 patent at Figs. 1 and 2. The claimed system purports to enable the virtual shared memory space to span the physical memory devices on multiple nodes through the use of global address signals that are shared by all the nodes on the network. See id. at 2:29-44. According to the patent, in this way each computer on the network can access data stored in the physical memory of other nodes as if the data were stored in the computer's local memory:

> The invention provides systems that can create and manage a virtual memory space that can be shared by each computer on a network and can span the storage space of each memory device connected to the network. Accordingly, all data stored on the network can be stored within the virtual memory space and the actual physical location of the data can be in any of the memory devices connected to the network.

'377 patent at 2:21-29; see also id. at 2:45-63 ("The system provides physical memory storage for each portion of the virtual memory space in use by mapping each such portion to a physical memory device, such as a RAM memory or a hard-drive…. This 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.").

Consistent with the object of the claimed invention, by distributing the physical storage for the claimed virtual shared memory space across multiple computers on the network instead of relying on a central server system for data storage and management, the invention purports to exploit the memory resources (e.g., the RAM and hard disk storage) of these computers and avoid the risk of a complete system failure caused by central server crash. The

‘229 patent additionally describes providing distributed control and persistent storage for a “structured store of data” (discussed further below), including files, database records, and web pages, and using “web servers.”

“LOCAL … MEMORY DEVICE”

Representative claim 1 of the ‘229 patent and claim 1 of the ‘377 patent claim a “local volatile memory device” and a “local persistent memory device.” See ‘377 patent at claim 1 (“… a local volatile memory device coupled to said computer …” and “… a local persistent memory device coupled to said computer …”); ‘229 patent at claim 1 (“… a local volatile memory device coupled to the node …” and “… a local persistent memory device coupled to the node …”). There is no dispute as to the distinction between persistent and volatile memory. Instead, the parties’ dispute is focused on whether the claim language “local … memory device” would have been understood by one of ordinary skill in the art in 1996 to mean “a memory device directly attached only to one node” (Oracle’s proposal) or “a memory device …, a portion or the whole of which can be contributed to the shared addressable memory space by a particular node” (Mangosoft’s proposal). As set forth below, Oracle’s proposed construction is the construction most closely tied to the claim language, specification and file history of the patents-in-suit.

As the claims of the patents-in-suit make clear, the terms “said computer” and “the node” in the claims at issue refer to a single one of the disclosed “plurality of computers” and “plurality of nodes” on the network. See ‘229 patent at claim 1 (“a plurality of nodes inter-connected by a network, each of said plurality of nodes… including … a local volatile memory device coupled to the node…, a local persistent memory device coupled to the node…”) (emphasis added). Likewise, the term “local” in each of the dependent claims in the ‘377 patent also refers to the volatile or persistent memory device of a single one of the plurality of computers on the claimed network. See ‘377 patent at claims 2-7 and 9. Accordingly, one skilled in the art would understand from the language of the claims that each of the disclosed volatile and persistent memory devices is directly coupled to a single computer among the plurality of computers on the network. It is the requirement that the memory is attached to only a single node that makes it "local." Indeed, if a "local . . . memory device" were merely any memory coupled to a node, as Mangosoft proposes, the term "local" would be redundant of the remainder of the claims' requirement of a "memory device coupled to said computer."

The plain language of the claims also requires that the "local … memory device" be directly attached to the node, and not merely linked to the node in some other (unspecified) way as Mangosoft would have it. While the term "coupled" has different meanings in different contexts, the term has a well-known meaning in the context of computer systems. Specifically, the term is used to describe a device that is attached directly (i.e., without any intervening device or connection) to a computer. See McGraw Hill Dictionary of Scientific and Technical Terms (5th ed. 1994) at 1159-60 (defining local device as "peripheral equipment that is linked directly to a computer or other supporting equipment without an intervening communications channel") (emphasis added). Thus, Oracle's proposed construction gives the term "local … memory device" the meaning it would have to those of skill in the art, while Mangosoft's proposed construction apparently seeks to broaden the term well beyond this meaning and to cover any connection (direct or indirect) between the memory device and computer.

The use of the term "local memory device" in the specification of the '229 patent is entirely consistent with its usage in the claims, and confirms that each "local" memory device is directly coupled to a single node on the network. For example, the Summary of the Invention in the '377 patent makes clear that each "local" memory device is coupled to a single one of the plurality of computers. See '377 patent at 3:10-14 ("…local persistent memory devices that each couple to a respective one of the plural computers."); 3:28-29 ("…local persistent memory devices of the plural computers"); 3:45-47 ("…local persistent memory device of a first computer … local persistent memory device of a second computer"); 3:56-62 ("…the volatile memory devices can be comprised of a plurality of local volatile memory devices each coupled to a respective one of the plural computers, and the persistent memory devices each coupled to a respective one of the plural computers.") (emphasis added); see also '229 patent at 15:55-16:40 (providing substantially identical disclosure). Likewise, the rest of the specifications of both patents are consistent with the above disclosure in using the term "local" to refer to memory devices that are directly coupled to a single node. See '377 patent at 7:1-8 (describing "two local memory devices, the RAM 34 and the disk 36" coupled to each of the three nodes depicted in Figure 2); 7:18-24 ("…local memories of each of the nodes…"); 7:28 ("…local memory of each node…"); 9:66-10:8 ("…the local memory of the requesting node…'); 10:9-24 ("The local memory storage, both volatile and persistent, of the requesting node..."); 10:42-48; 12:9-14; 14:23-41; 15:8-11; Fig. 1; Fig. 2; Fig. 3; Fig. 4; Fig. 7; see also '229 patent at 6:45-50 (disclosing local hard disk storage on nodes 12*a*-12*d* of Figure 1 as "local hard disk storage of those particular nodes"); 6:66-7:16 (describing use a of a data control program to allow a network node to access a shared memory subsystem as if it were "a local memory device such as a local hard disk"). In short, the specifications of the patents-in-suit make clear—over and over again—that the claimed "local … memory device" is memory directly attached to a single node, nothing more and nothing less.

Given the consistent disclosure of the claims and specification, it is no surprise that the applicants' statements to the Patent Office during prosecution of the '377 patent also confirm that the term "local" in the claims describes volatile and persistent memory devices that are directly coupled to a single node. In an October 15, 1998 amendment following the final rejection of all pending claims, the applicants distinguished their claimed invention from prior art cited by the Patent Office based in part on their disclosure of local volatile and persistent memory devices on each node in the network:

> Applicants submit that none of the relied-upon references teaches or suggests local volatile memory devices (e.g., RAM associated with each networked computer) or persistent memory devices (e.g., hard disks associated with each networked computer)…

'377 File History at p.143 (emphasis added).

CONCLUSION

As set forth above, Oracle's proposed constructions for the disputed "local … memory device" terms are founded in the proper support: the ordinary meaning of the terms as used in the language of the claims and the context provided by the specification and the file history. The claims should be construed accordingly.

UNITED STATES DISTRICT COURT
DISTRICT OF NEW HAMPSHIRE

MANGOSOFT, INC. -and- MANGOSOFT CORPORATION, Plaintiffs, v. ORACLE CORPORATION, Defendant.	Case No. C 02-545 - JM **MANGOSOFT'S REPLY CLAIM CONSTRUCTION BRIEF**

As explained in Mangosoft's Opening Brief, Mangosoft's proposed construction of the "local … memory device" limitations flows directly from the plain language of the claims, as confirmed by the specification. Oracle's proposed construction, on the other hand, attempts to add two new limitations to the claims, presumably to support an argument that Oracle does not infringe. First, Oracle substitutes the phrase "directly attached" in place of the term "coupled" in the claims. Not only is this rewriting of the claim is both legally impermissible, *see Resonate, Inc. v. Alteon Websystems, Inc.*, 338 F.3d 1360, 1365 (Fed. Cir. 2003), but it is factually unsupported. Second, Oracle attempts to characterize "local ... memory devices" as those memory devices that are not "shared." This construction, however, is wholly inconsistent with the specifications. Because they contort the plain meaning of the claim terms, Oracle's attempts to rewrite and add these additional limitations into the claims should be rejected.

II. "LOCAL … MEMORY DEVICE"

A. The Claims Do Not Require That Local Memory Be Directly Attached To A Node

Oracle's contention that the term "local ... memory device" should be construed to limit the invention to volatile and persistent memory devices that are **directly attached** to a single computer or node is inconsistent with the plain language of the claims. As explained in

Mangosoft's Opening Brief, the language of claim 1 of the '377 patent states plainly: "a local volatile memory device **coupled** to said computer and having volatile storage for data signals." '377 patent at claim 1 (emphasis added). There is no reason to construe the term "coupled"—let alone to add two additional limitations found nowhere in the claims or the patent—and Oracle has offered none.

At the outset, while Oracle has purported to offer dictionary support for its proposal that the claims require direct attachment between a memory device and a node, Oracle's dictionary support is limited to a single definition of "local device" in the context computer peripherals. Even assuming that that definition of "local device" is applicable here, it is irrelevant to the present dispute. The claims specifically state that the "local … memory device" need only be "coupled" to the node. Mangosoft's Opening Brief established that the term "coupled" does not require direct attachment. Oracle does not meaningfully dispute that.

Likewise, Oracle does not meaningfully dispute that the specifications and file histories of the patents-in-suit also do not support limiting local volatile and persistent memory devices to "directly attached" memory devices even were it legally permissible to add such language to the claims. Indeed, while Oracle points to specification examples of "local … memory devices" linked to nodes, Oracle has offered no explanation for the examples (identified in Mangosoft's Opening Brief) of memory devices that are **not** directly attached.

B. There Is No Requirement That Local Memory Is Not Shared

In addition to urging the substitution of the phrase "directly attached" for the term "coupled" in the claim, Oracle further contends that that the term "local" is used to distinguish "local . . . memory devices" from "shared" memory devices, i.e., that "local . . . memory devices" are linked to only a single node and not shared. This construction is inconsistent with the language of the claims and the specifications. The term “local” is used merely to distinguish “local” memory devices from “remote” memory devices.” In fact, the specification refers to non-shared memory as “private memory,” **not** merely any “local” memory. *See* ‘377 patent at 6:20-31 (“The private memory provides the node with local storage that can be kept inaccessible to the other nodes on the network.”). This statement from the specification confirms the conclusion that “local” means something other than not shared.

Consistent with this, the very portions of the file history relied upon by Oracle support Mangosoft’s claim construction that the described local memory devices are merely associated with each node or computer and have portions of the shared-addressable memory mapped thereon:

> Applicants submit that none of the relied-upon references teaches or suggests **local** volatile memory devices (e.g., RAM **associated with** each networked computer) or persistent memory devices (e.g., hard disks **associated with** each networked computer)…

‘377 File History at p.143 (emphasis added). In other words, local memory is merely memory **associated with** a node, not connected to a single node to the exclusion of sharing with other nodes.

IV. CONCLUSION

Because, as demonstrated above, Oracle’s proposed construction of the term “local memory device” ignores the plain meaning of the claim terms, and attempts to read into the claims additional limitations that are unsupported in the intrinsic record, Oracle’s proposed constructions should be rejected.

UNITED STATES DISTRICT COURT
DISTRICT OF NEW HAMPSHIRE

MANGOSOFT, INC. -and- MANGOSOFT CORPORATION, Plaintiffs, v. ORACLE CORPORATION, Defendant.	Case No. C 02-545 - JM **JOINT CLAIM CONSTRUCTION STATEMENT**

Claim Language in Dispute	Plaintiff's Proposed Construction and Supporting Intrinsic Evidence	Defendant's Proposed Construction and Supporting Intrinsic Evidence	Consequence of Construction
"**local** volatile **memory device** coupled to the node and providing volatile storage" "**local** persistent **memory device** coupled to the node and providing persistent storage"	"a memory device … a portion or the whole of which can be contributed to the shared addressable memory space by a particular node" **Supporting evidence:** Specification: '377 patent at 6:20-31, 7:24-34, Figs. 2, 6, and claim 1 '229 patent at 6:45-50, 19:52-62, and claim 1 File History '377 File History at p.143	"a memory device directly attached only to one node" **Supporting evidence:** Specification: '377 patent at 3:10-14, 3:28-29, 3:45-47, 3:56-62, 7:1-8, 7:18-24, 7:28 , 9:66-10:8, 10:9-24, 10:42-48, 12:9-14, 14:23-41, 15:8-11, Fig. 1-4 and 7, and claims 1, 2-7 and 9 '229 patent at 6:45-50, 6:66-7:16, 15:55-16:40, and claim 1 File History '377 File History at p.143	Defendant contends that under its construction summary judgment of noninfringement is proper.

Respectfully submitted,

Attorneys for Plaintiffs
MANGOSOFT, INC. and MANGOSOFT CORPORATION

Attorneys for Defendant
ORACLE CORPORATION

US005918229A

United States Patent [19]

Davis et al.

[11] **Patent Number: 5,918,229**

[45] **Date of Patent: Jun. 29, 1999**

[54] **STRUCTURED DATA STORAGE USING GLOBALLY ADDRESSABLE MEMORY**

[75] Inventors: **Scott H. Davis**, Groton, Mass.; **John B. Carter**, Salt Lake City, Utah; **Steven J. Frank**, Hopkinton, Mass.; **Hsin H. Lee**; **Daniel J. Dietterich**, both of Acton, Mass.

[73] Assignee: **Mangosoft Corporation**, Westborough, Mass.

[21] Appl. No.: **08/827,534**

[22] Filed: **Mar. 28, 1997**

Related U.S. Application Data

[63] Continuation-in-part of application No. 08/754,481, Nov. 22, 1996.

[51] **Int. Cl.**^{6} **G06F 17/30**

[52] **U.S. Cl.** **707/10**; 707/200; 707/205; 395/200.48; 705/26; 711/147

[58] **Field of Search** 707/10, 201, 202, 707/203, 204, 205, 206; 711/203, 147, 148; 395/200.48; 705/26

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,096,567	6/1978	Millard et al.	364/200
4,426,688	1/1984	Moxley	365/200
4,710,926	12/1987	Brown et al.	371/9
4,868,738	9/1989	Kish et al.	364/200
4,934,764	6/1990	Leitermann et al.	312/111
5,055,999	10/1991	Frank et al.	711/163
5,077,736	12/1991	Dunphy, Jr. et al.	371/10.1
5,117,350	5/1992	Parrish et al.	395/425
5,119,481	6/1992	Frank et al.	395/280
5,226,039	7/1993	Frank et al.	370/405

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 288 636 A2	11/1988	European Pat. Off. .
0 661 651 A1	7/1995	European Pat. Off. .
2 257 273	1/1993	United Kingdom .
WO 95/02307	1/1995	WIPO .
WO 95/14279	5/1995	WIPO .
WO 95/22111	8/1995	WIPO .
WO 95/25306	9/1995	WIPO .
WO 96/23268	8/1996	WIPO .
WO 96/30847	10/1996	WIPO .

OTHER PUBLICATIONS

"Coarse and Fine Grain Objects in a Distributed Persistent Store,", Henskens et al., Object Orientation in Operating Systems, May 1993, IEEE, pp. 116–123.

"A Model for Persistent Shared Memory Addressing in Distributed Systems,", Amaral et al., Object Orientation in Operating Systems, Sep. 1992, IEEE, pp. 2–11.

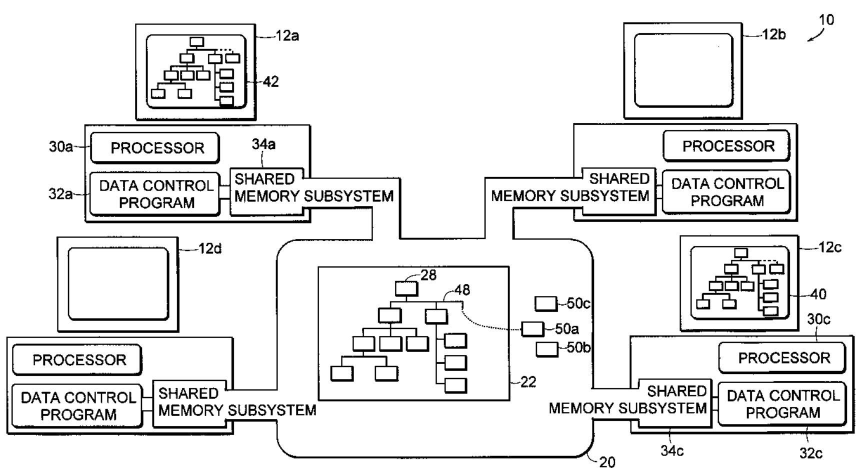
(List continued on next page.)

Primary Examiner—Paul R. Lintz
Assistant Examiner—Hosain T. Alam
Attorney, Agent, or Firm—Testa, Hurwitz & Thibeault, LLP

[57] **ABSTRACT**

In a network of computer nodes, a structured storage system interfaces to a globally addressable memory system that provides persistent storage of data. The globally addressable memory system may be a distributed shared memory (DSM) system. A control program resident on each network node can direct the memory system to map file and directory data into the shared memory space. The memory system can include functionality to share data, coherently replicate data, and create log-based transaction data to allow for recovery. In one embodiment, the memory system provides memory device services to the data control program. These services can include read, write, allocate, flush, or any other similar or additional service suitable for providing low level control of a memory storage device. The data control program employs these memory system services to allocate and access portions of the shared memory space for creating and manipulating a structured store of data such as a file system, a database system, or a Web page system for storing, retrieving, and delivering objects such as files, database records or information, and Web pages.

37 Claims, 11 Drawing Sheets



U.S. PATENT DOCUMENTS

5,245,563	9/1993	Hauck, Jr.	364/748.05
5,247,673	9/1993	Costa et al.	395/650
5,251,308	10/1993	Frank et al.	711/163
5,274,789	12/1993	Costa et al.	711/206
5,282,201	1/1994	Frank et al.	370/403
5,297,265	3/1994	Frank et al.	711/202
5,313,647	5/1994	Kaufman et al.	395/672
5,335,325	8/1994	Frank et al.	711/163
5,341,483	8/1994	Frank et al.	711/206
5,390,316	2/1995	Cramer et al.	395/425
5,390,326	2/1995	Shah	395/575
5,394,555	2/1995	Hunter et al.	711/148
5,408,649	4/1995	Beshears et al.	395/575
5,423,037	6/1995	Hvasshovd	395/600
5,452,447	9/1995	Nelson et al.	707/205
5,490,272	2/1996	Mathis et al.	395/650
5,493,728	2/1996	Solton et al.	711/113
5,513,314	4/1996	Kandasamy et al.	395/182.04
5,519,855	5/1996	Neeman et al.	395/600
5,522,045	5/1996	Sandberg	395/200
5,551,035	8/1996	Arnold et al.	395/650
5,560,027	9/1996	Watson et al.	395/800
5,560,029	9/1996	Papadopoulos	395/800
5,588,147	12/1996	Neeman et al.	395/601
5,592,625	1/1997	Sandberg	395/200
5,613,079	3/1997	Debique et al.	395/468
5,617,537	4/1997	Yamada et al.	395/200
5,649,194	7/1997	Miller et al.	707/200
5,675,787	10/1997	Miller et al.	707/104
5,687,308	11/1997	Jardine et al.	395/182.02
5,689,700	11/1997	Miller et al.	707/10
5,701,462	12/1997	Whitney et al.	395/610
5,727,150	3/1998	Laudon et al.	395/200
5,768,510	6/1998	Gish	395/200.33
5,781,537	7/1998	Ramaswami et al.	370/254
5,805,785	9/1998	Dias et al.	395/182.02
5,805,786	9/1998	Badovinatz et al.	395/182.02
5,812,533	9/1998	Cox et al.	370/259
5,812,751	9/1998	Ekrot et al.	395/182.02

OTHER PUBLICATIONS

"Towards a Resilient Shared Memory Concept for Distributed Persistent Object Systems,", Irlenbusch et al, Proceedings of teh 28th Annual Hawaii Intl. Conference on System Sciences, Jan. 1995, pp. 675–684, IEEE.

"COOL: System Support for Distributed Programming," Lea te al., Communications of teh ACM, vol. 36, No. 9, pp. 37–46, Sep. 1993.

"Support for Collaborative Design: Agents and Emergence, Edmonds et al.," Communications of the ACM, vol. 37, No. 7, pp. 41–47, Jul. 1994.

"Software Agents," Genesereth et al., Communications of the ACM, vol. 37, No. 7, pp. 48–53, Jul. 1994.

European Patent Office, European Search Report, Application No. EP 97 30 9471, mailed on Apr. 23, 1998, 3 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US 97/21457, mailed on Apr. 23, 1998, 4 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US 97/21458, mailed on Apr. 23, 1998, 3 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US 97/21459, mailed on Apr. 23, 1998, 4 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US 97/21460, mailed on Apr. 23, 1998, 5 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US 97/21466, mailed on Apr. 23, 1998, 3 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US 97/21733, mailed on Apr. 23, 1998, 3 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US 97/21734, mailed on Apr. 23, 1998, 4 pages.

Anderson, T., et al. "Serverless Network File Systems," *Operating Systems Review* (*SIGOPS*), vol. 29, No. 5, Dec. 1, 1995, pp. 109–126.

Carter, J., et al. "Network Multicomputing Using Recoverable Distributed Shared Memory," Proceedings of the Spring Computer Society International Conference (COMPCON), San Francisco, Feb. 22–26, 1993, pp. 519–527.

Huber, J., et al. "PPFS: A High Performance Portable Parallel File System," Proceedings of the International Conference on Supercomputing, Barcelona, Jul. 3–7, 1995, pp. 385–394.

"Java Dynamic Class Loader," *IBM Technical Disclosure Bulletin,* vol. 39, No. 11, Nov. 1996, pp. 107–108.

Lee, E., et al. "Petal: Distributed Virtual Disks," 7th International Conference on Architectural Support for Programming Languages and Operation Systems, Cambridge, MA, Oct. 1–5, 1996, pp. 84–92.

Neal, D. "The Harvest Object Cache in New Zealand," *Computer Networks and ISDN Systems,* vol. 11, No. 28, May 1996, pp. 1415–1430.

Raghavan, G. "A Domain Model of WWW Browsers," Proceedings of Southeastcon, Bringing Together Education, Science and Technology, Tampa, Apr. 11–14, 1996, pp. 436–439.

Ng, T. "The Design and Implementation of a Reliable Distributed Operating System—ROSE," Proceedings of the Symposium on Reliable Distributed Systems, Huntsville, Oct. 9–11, 1990, pp. 2–11.

Yeung, D., et al. "MGS" A Multigrain Shared Memory System, *Computer Architecture News,* vol. 24, No. 2, May 1, 1996, pp. 44–55.

Anderson et al. "Scheduler Activations: Effective Kernel Support for the User–Level Management of Parallelism," *Operating Systems Review* (*SIGOPS*), vol. 25, No. 1, Jan. 1, 1991, pp. 95–109.

Barton–Davis et al. "Adding Scheduler Activations to Mach 3.0," Proceedings of the USENIX Mach III Symposium, Santa FE, NM, USA, Apr. 19–21, 1993, pp. 119–136.

Koppe C. "Sleeping Threads: A Kernel Mechanism for Support of Efficient User Level Threads," Proceedings of Seventh IASTED–ISMM International Conference on Parallel and Distributed Computing and Systems, Washington, DC, USA, Oct. 19–21, 1995, pp. 11–15.

Dryden, P. "Users throwing cache at networks delays" *Computerworld,* Oct. 7, 1996, http://ww.computerworld.com/search/AT–html/9610/961007SL41cacheb.html, originally printed on May 22, 1997, reprinted on Nov. 18, 1998.

Enterprise Networking Web Page, http://www.lancache.com/enad1.html, printed on May 20, 1997.

Measurement Techniques Inc. Web Page, http://www.lancache.com/pr1_4.html, printed on May 20, 1997.

Measurement Techniques Inc. Web Page, http://www.lancache.com/slc.html, printed on May 20, 1997.

Network Appliance, Inc. Web Page, http://www.netapp.com/news/level3b/news_rel_970421.html, printed on May 22, 1997.

Network Appliance, Inc. Web Page, http://www.netapp.com/products/level3/netcache/datasheet.html, printed on May 22, 1997.

Network Appliance, Inc. Web Page, http://www.netapp.com/products/level 3/webfiler.html, printed on May 22, 1997.

Peak Technologies, Inc. Web Page, http://www.peak–media.com/PeakJet/quick.html, printed on May 20, 1997.

Peak Technologies, Inc. Web Page, http://www.peakJet/PeakJet.html, printed May 20, 1997.

Relay Web Page, http://www.relay.com/product/gold.html, printed on May 22, 1997.

Relay Web Page, http://www.relay.com/product/otg.html, printed on May 22, 1997.

Sequel Technology Web Page, http://www.sequeltech.com/product/netPIM/prodinfo.html, printed on May 22, 1997.

Stac Inc. Web Page, http://www.stac.com/replica/pr–legato-agreement.html, printed on May 22, 1997.

Wilson, Andrew W., "Organization and Statistical Simulation of Hierarchical Multiprocessors," UMI Dissertation Information Service (1985).

Li, Kai, "Shared Virtual Memory on Loosely Coupled Multiprocessors," Yale University, Department of Computer Science, (Sep., 1986).

Wilson, Andrew W., "Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors," *ACM,* pp. 244–252 (1987).

Carter, J.B. et al., "Optimistic Implementation of Bulk Data Transfer Protocols," *In Proceedings of the 1989 Signetrics Conference,* pp. 61–69 (May, 1989).

Carter, J.B. et als., "Distributed Operating Systems Based on a Protected Global Virtual Address Space," In the Proceedings of the Third Workshop on Workstation Operating Systems (WWOS) (May, 1992).

Carter, J.B. et als., "FLEX: A Tool for Building Efficient and Flexible Systems," In the Proceedings of the Fourth Workshop on Workstation Operating Systems ((WWOS) (Oct., 1993).

Carter, John B., "Design of the Munin Distributed Shared memory System," To appear in the special issue of the *Journal of Parallel and Distributed Computing* on distributed shared memory (1995).

Carter, J.B. et als., "Techniques for Reducing Consistency–Related Communication in Distributed Shared Memory Systems," *ACM Transactions on Computer Systems,* pp. 205–243, vol. 13, No. 3 (Aug., 1995).

Carter, J.B. et als., "Reducing Consistency Traffic and Cache Misses in the Avalanche Multiprocessor," University of Utah technical report.

Web page of Tricord Systems, Inc., http://www.tricord.com/2/10–3–96.html, printed on May 22, 1997.

Web page of Network Appliance, Inc., http://www.netapp.com/products/level3/nfs.html, printed on May 22, 1997.

Web page of Network Appliance, Inc., http://www.netapp.com/products/level3/ontap.html, printed on May 22, 1997.

Web page of Network Appliance, Inc., http://www.netapp.com/products/level3/windows.html, printed on May 22, 1997.

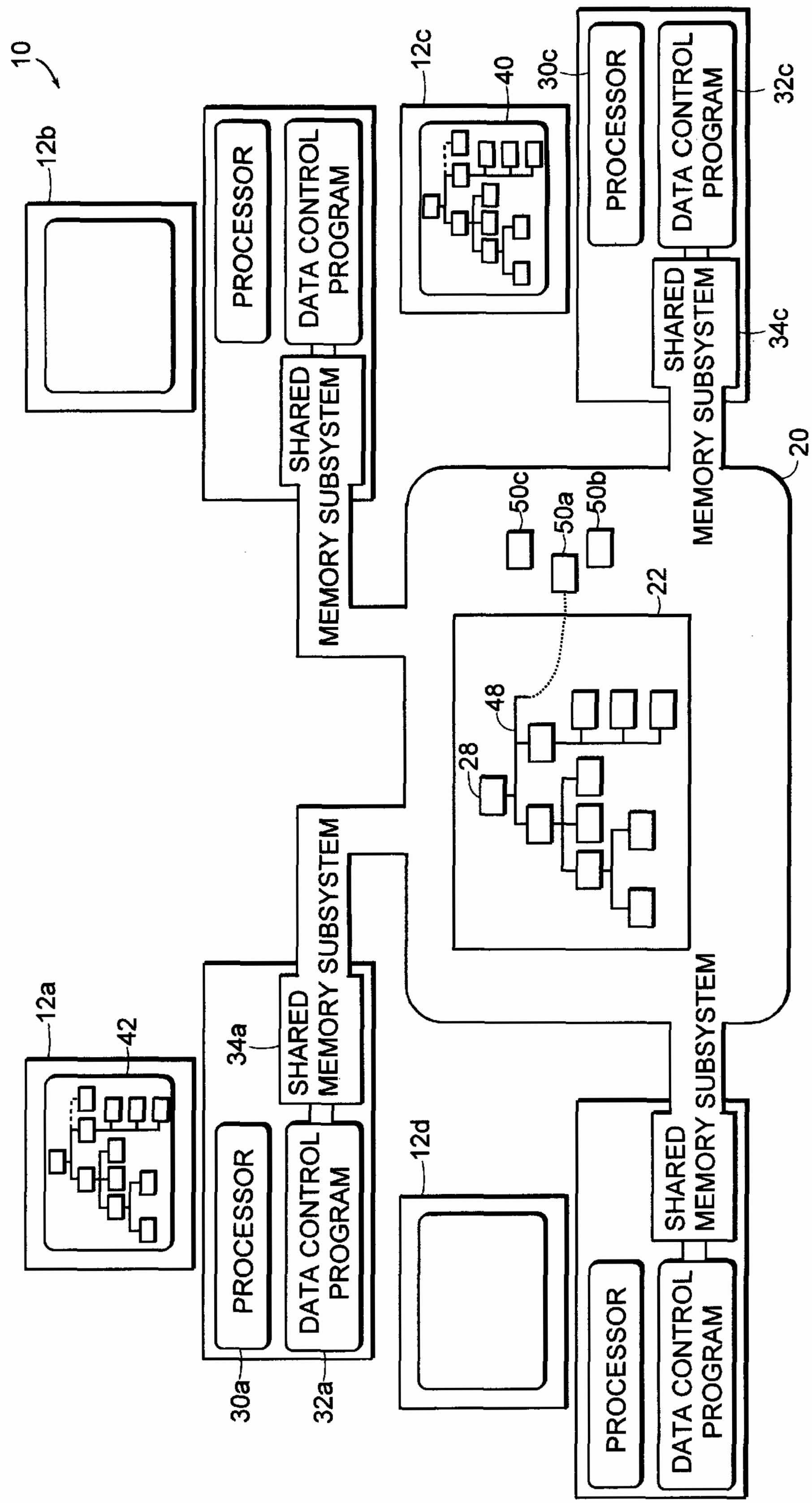

10
12a
42
12b
12c
40
12d
30a
32a
34a
30c
32c
34c
PROCESSOR
DATA CONTROL PROGRAM
SHARED MEMORY SUBSYSTEM
PROCESSOR
DATA CONTROL PROGRAM
SHARED MEMORY SUBSYSTEM
PROCESSOR
DATA CONTROL PROGRAM
SHARED MEMORY SUBSYSTEM
PROCESSOR
DATA CONTROL PROGRAM
SHARED MEMORY SUBSYSTEM
20
22
28
48
50a
50b
50c

FIG. 1

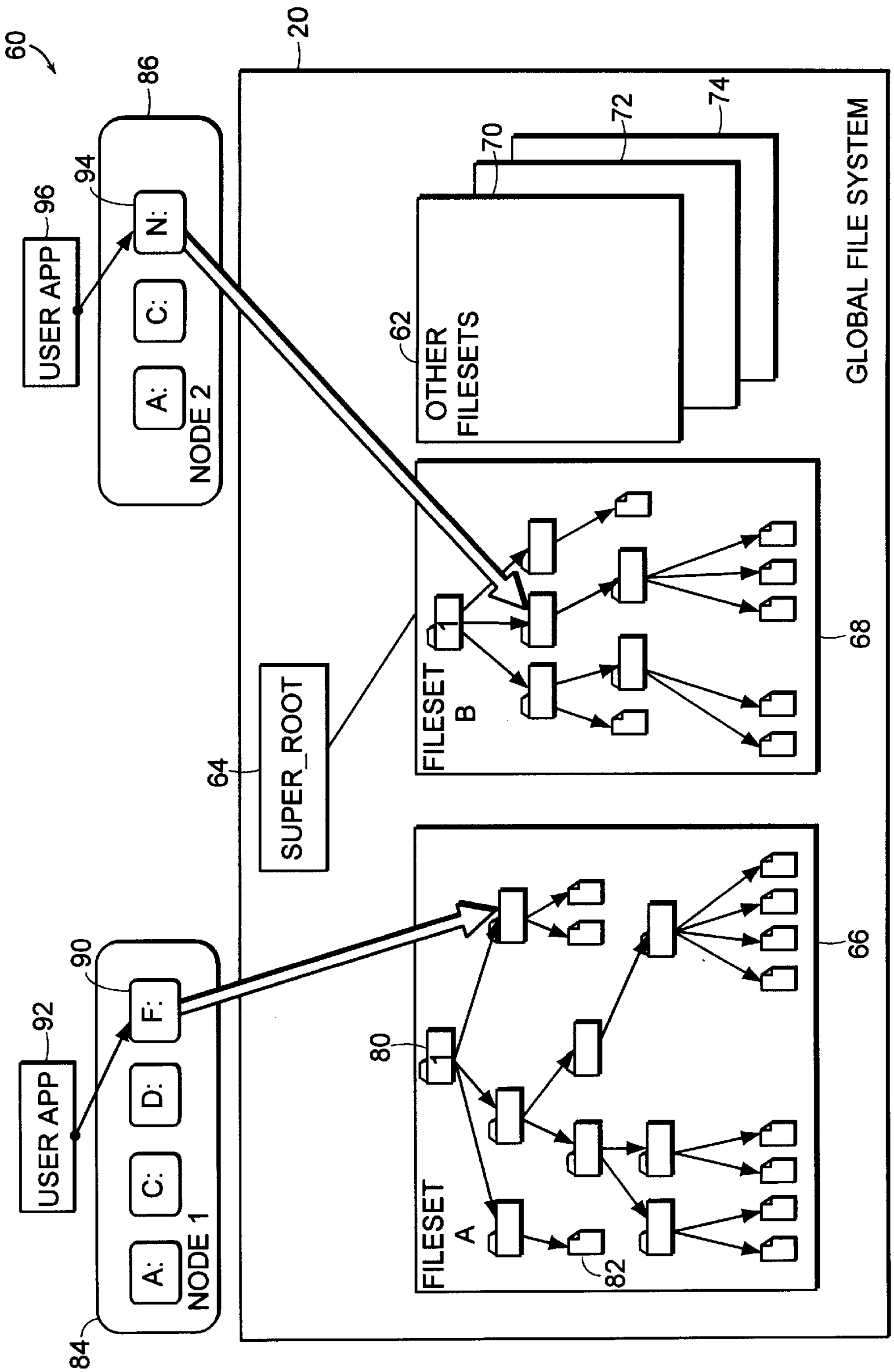

60
20
86
94
96
USER APP
N:
C:
A:
NODE 2
62
OTHER FILESETS
70
72
74
GLOBAL FILE SYSTEM
FILESET B
68
SUPER_ROOT
64
92
USER APP
90
F:
D:
C:
A:
NODE 1
84
80
FILESET A
82
66

FIG. 2

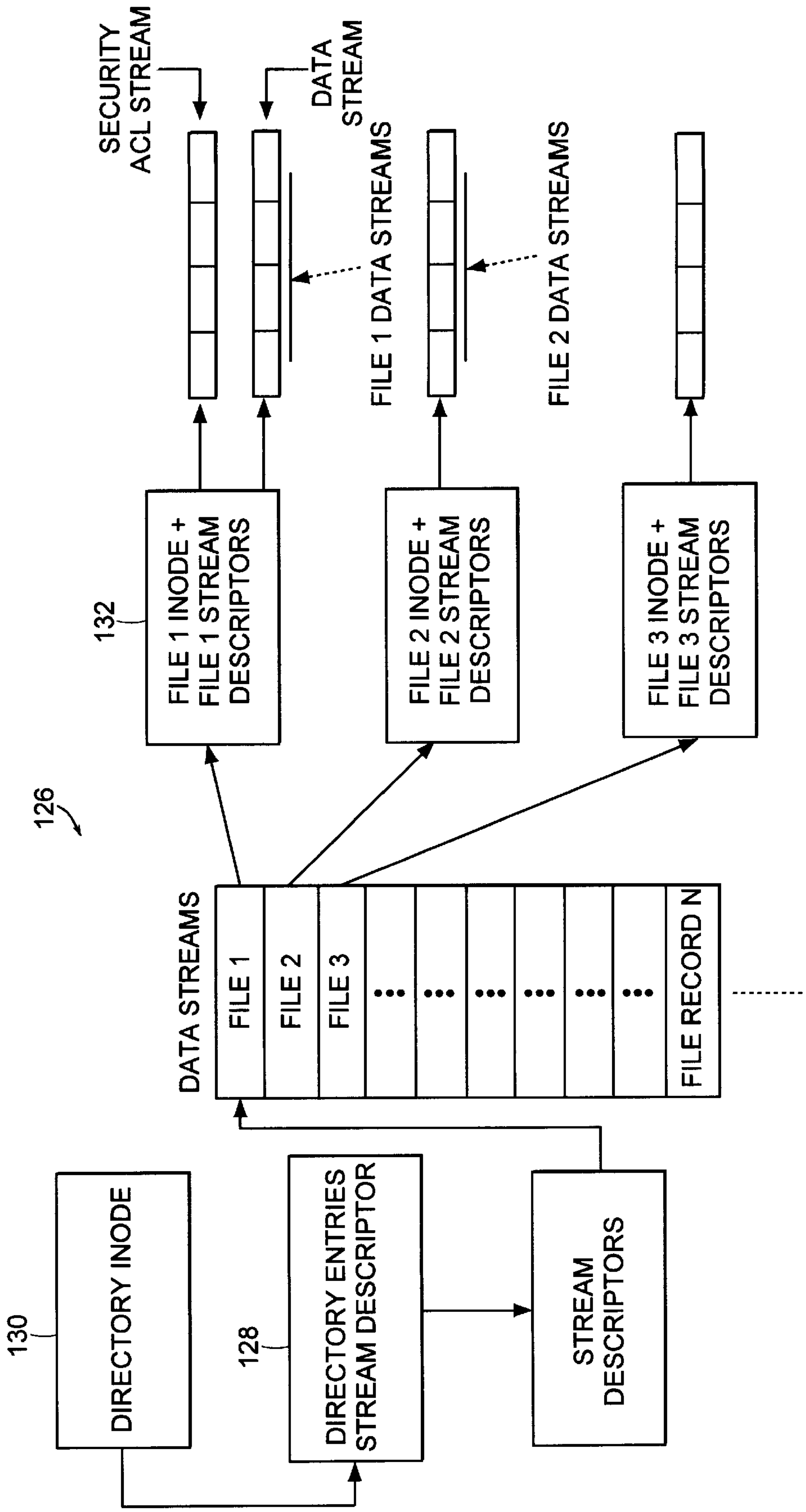
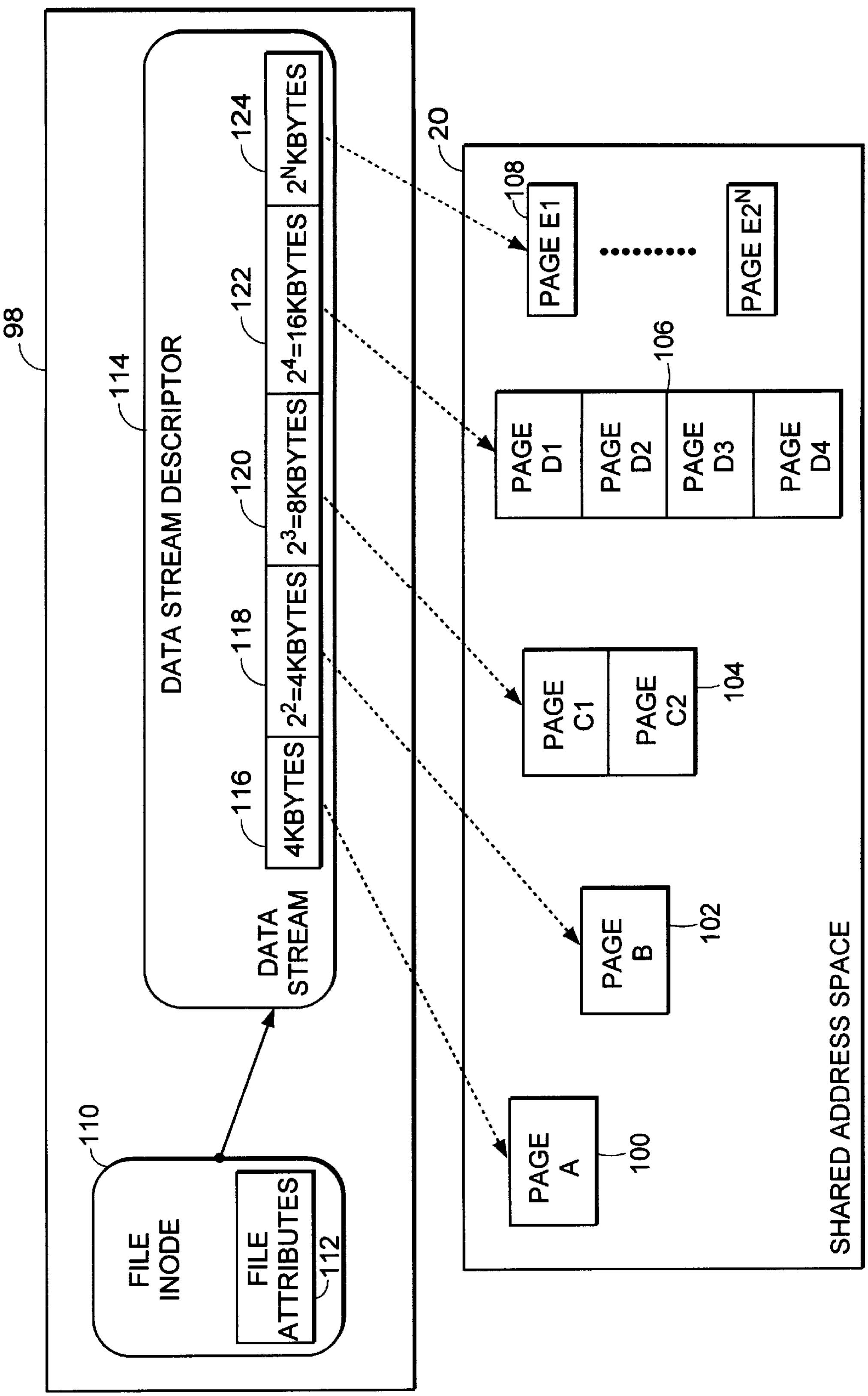

126
130
DIRECTORY INODE
128
DIRECTORY ENTRIES STREAM DESCRIPTOR
STREAM DESCRIPTORS
DATA STREAMS
FILE 1
FILE 2
FILE 3
FILE RECORD N
132
FILE 1 INODE + FILE 1 STREAM DESCRIPTORS
FILE 2 INODE + FILE 2 STREAM DESCRIPTORS
FILE 3 INODE + FILE 3 STREAM DESCRIPTORS
SECURITY ACL STREAM
DATA STREAM
FILE 1 DATA STREAMS
FILE 2 DATA STREAMS

FIG. 3



98
FILE INODE
110
FILE ATTRIBUTES
112
DATA STREAM
DATA STREAM DESCRIPTOR
114
4KBYTES
116
2^2=4KBYTES
118
2^3=8KBYTES
120
2^4=16KBYTES
122
2^NKBYTES
124
20
PAGE A
100
PAGE B
102
PAGE C1
PAGE C2
104
PAGE D1
PAGE D2
PAGE D3
PAGE D4
106
PAGE E1
108
PAGE E2^N
SHARED ADDRESS SPACE

FIG. 4

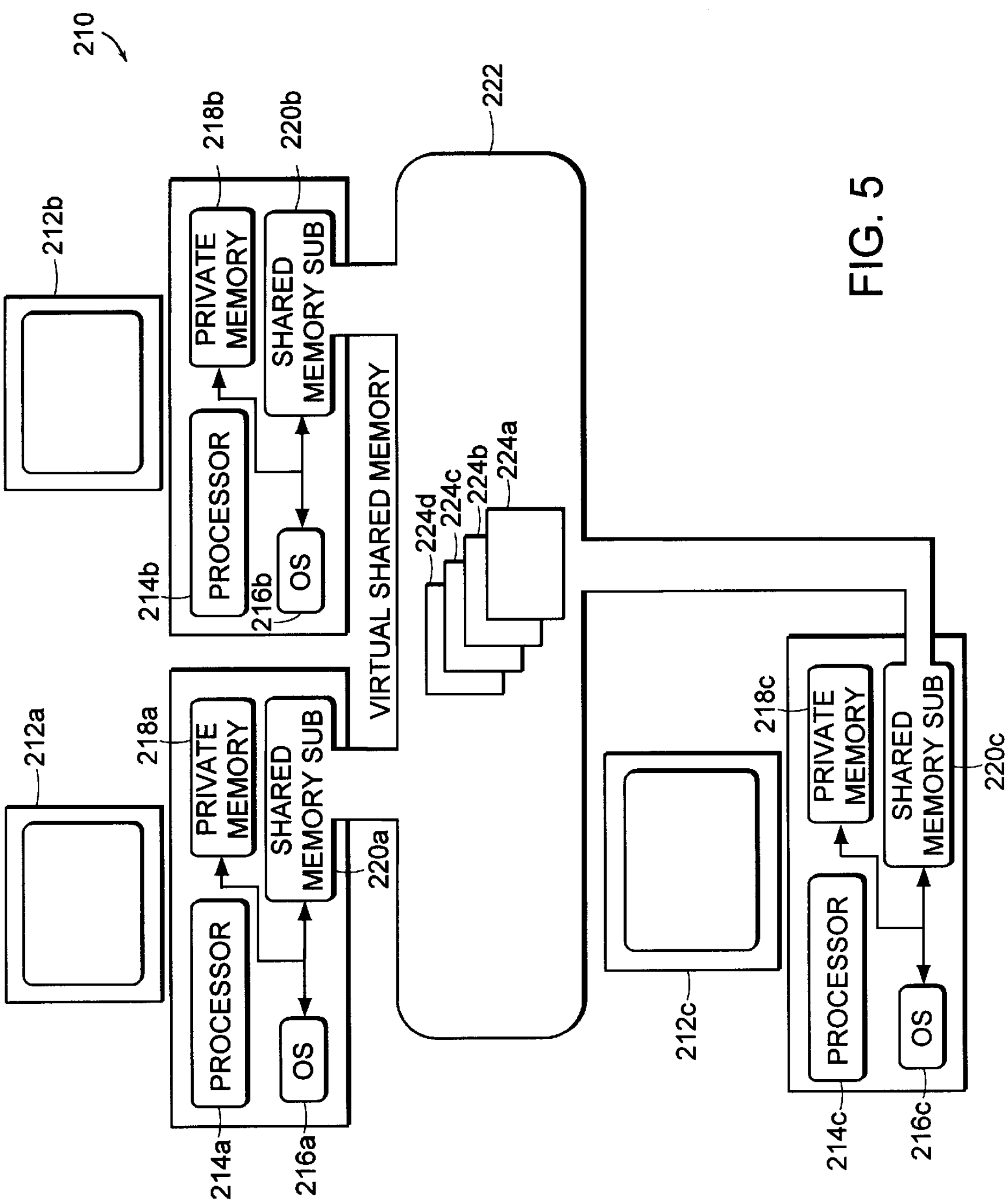

210
212a
212b
212c
PROCESSOR
214a
214b
214c
OS
216a
216b
216c
PRIVATE MEMORY
218a
218b
218c
SHARED MEMORY SUB
220a
220b
220c
VIRTUAL SHARED MEMORY
222
224a
224b
224c
224d

FIG. 5

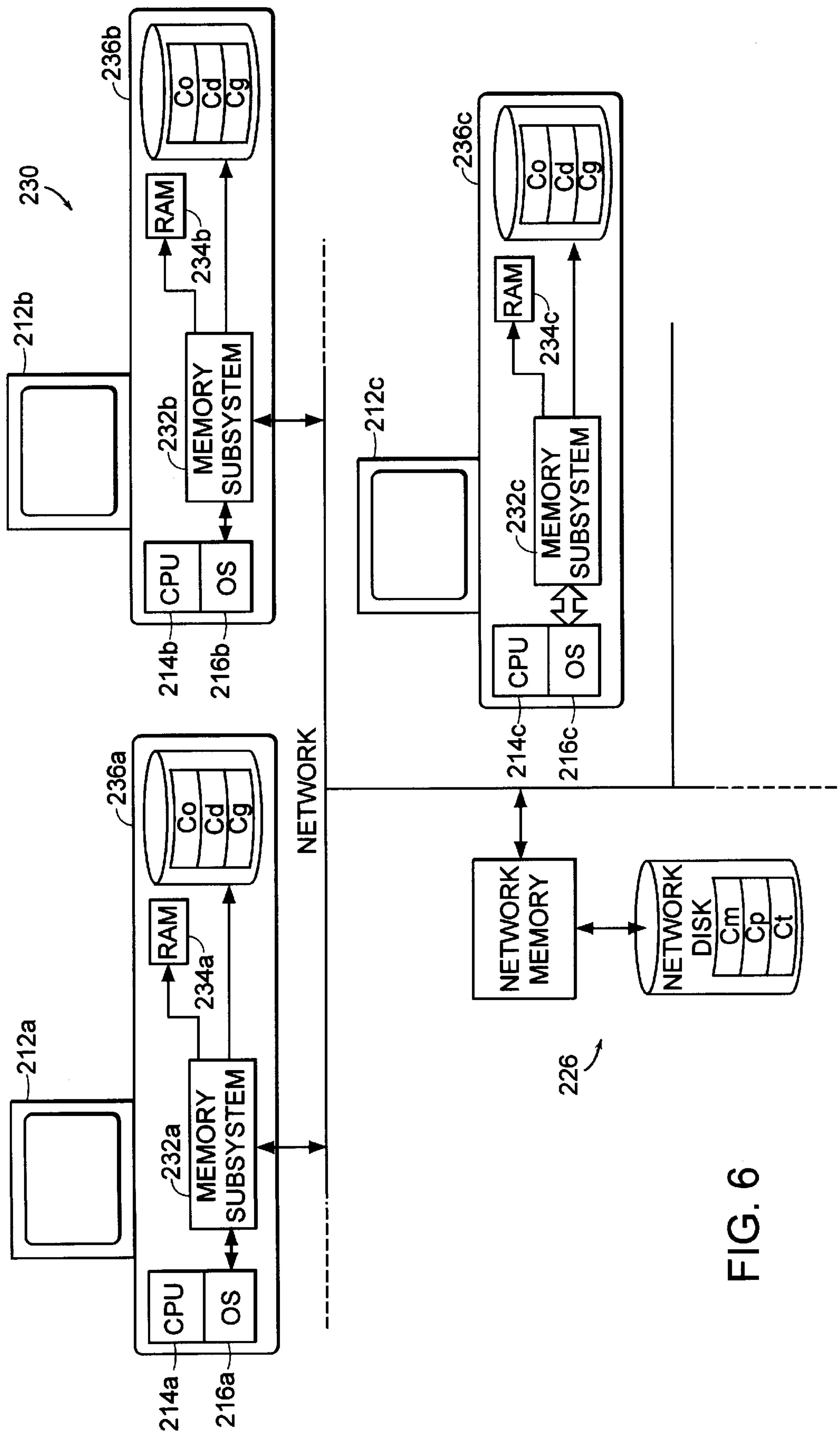

230
212a
212b
212c
214a
214b
214c
216a
216b
216c
232a
232b
232c
MEMORY SUBSYSTEM
MEMORY SUBSYSTEM
MEMORY SUBSYSTEM
234a
234b
234c
RAM
RAM
RAM
236a
236b
236c
Co
Cd
Cg
CPU
OS
NETWORK
NETWORK MEMORY
NETWORK DISK
Cm
Cp
Ct
226

FIG. 6

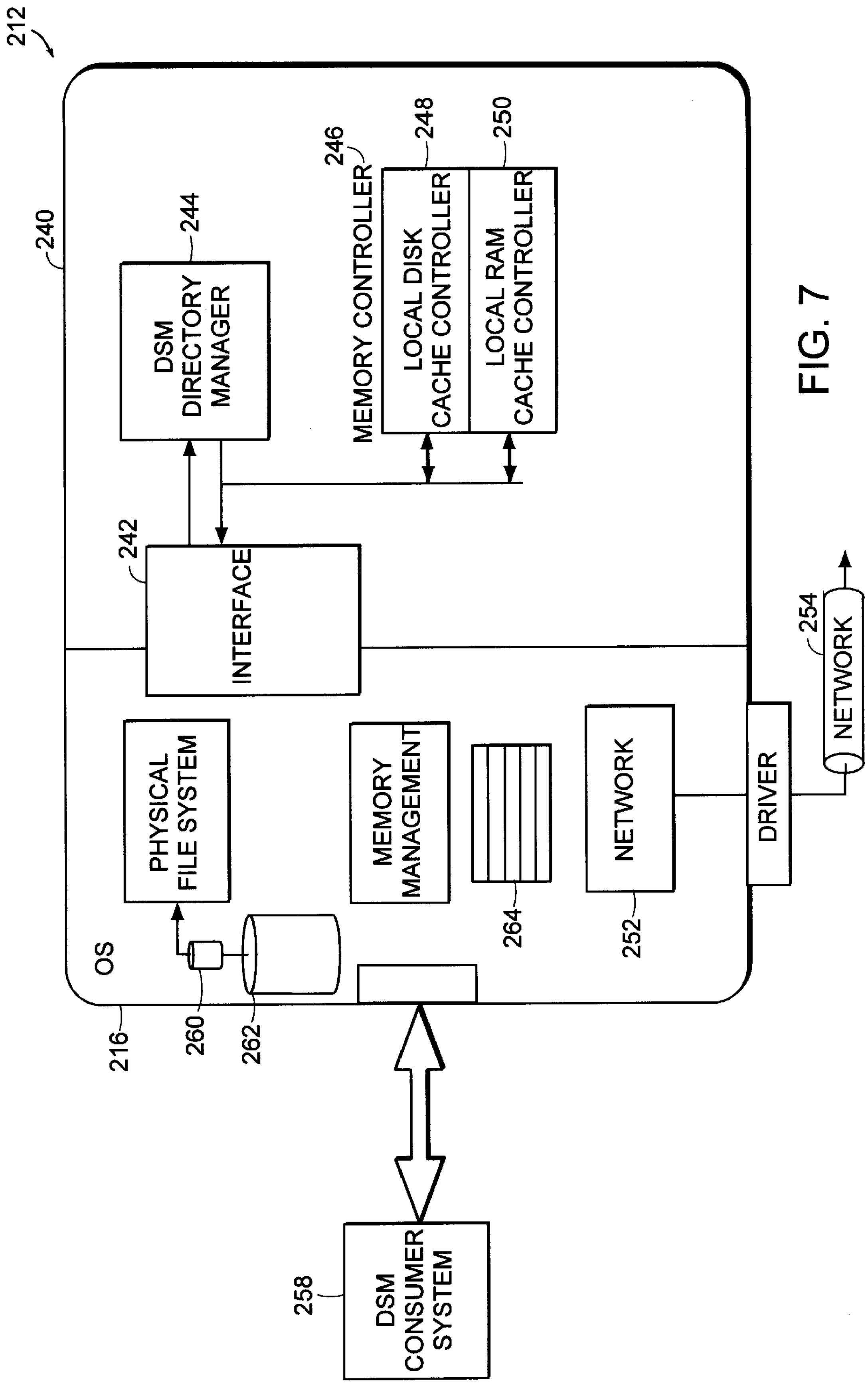

212
240
OS
DSM DIRECTORY MANAGER
244
242
INTERFACE
MEMORY CONTROLLER
246
LOCAL DISK CACHE CONTROLLER
248
LOCAL RAM CACHE CONTROLLER
250
PHYSICAL FILE SYSTEM
MEMORY MANAGEMENT
264
NETWORK
252
DRIVER
NETWORK
254
216
260
262
DSM CONSUMER SYSTEM
258

FIG. 7

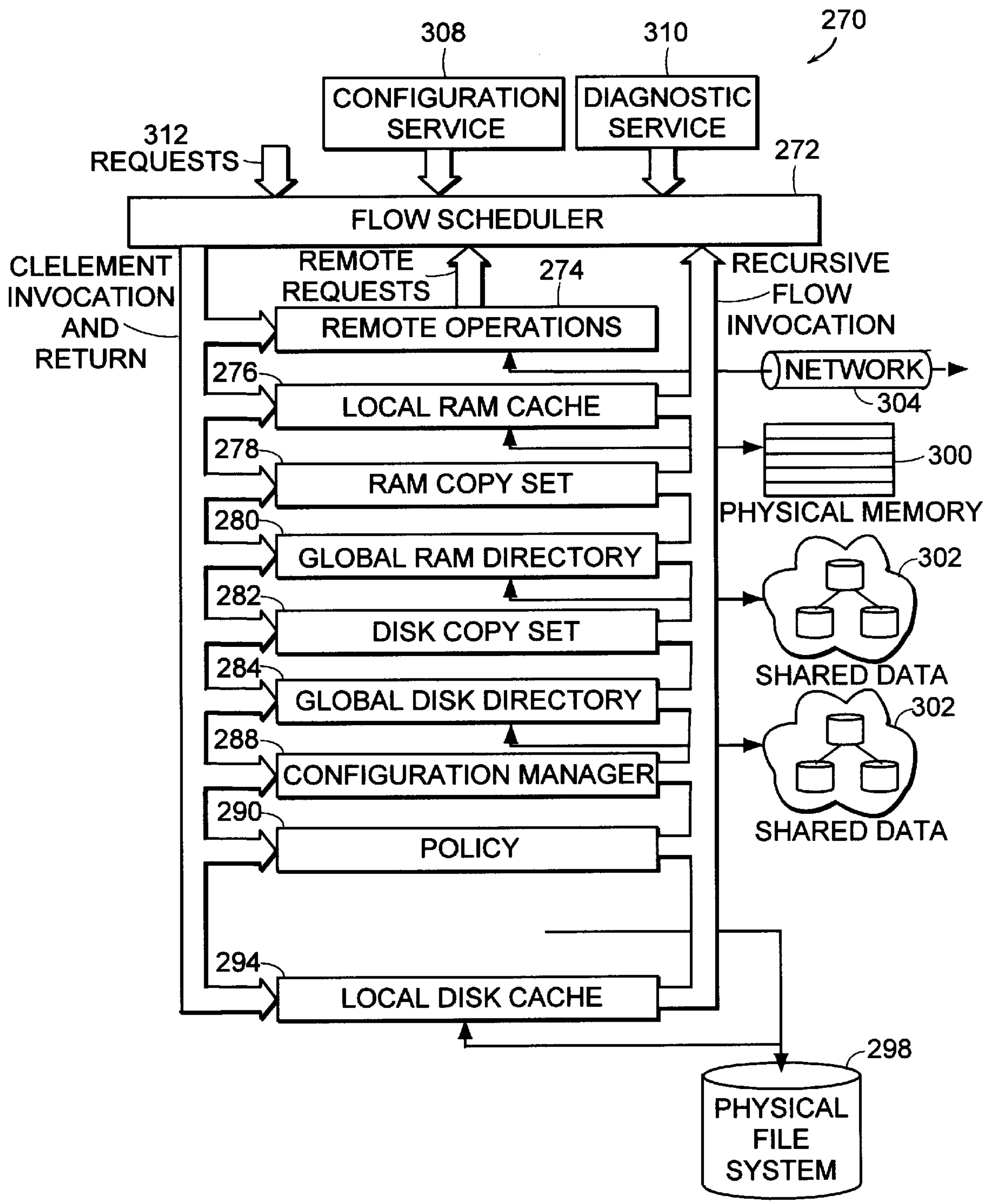

270
308
310
CONFIGURATION SERVICE
DIAGNOSTIC SERVICE
312
REQUESTS
272
FLOW SCHEDULER
CLELEMENT INVOCATION AND RETURN
REMOTE REQUESTS
274
RECURSIVE FLOW INVOCATION
REMOTE OPERATIONS
NETWORK
304
276
LOCAL RAM CACHE
300
278
RAM COPY SET
PHYSICAL MEMORY
280
GLOBAL RAM DIRECTORY
302
282
DISK COPY SET
284
GLOBAL DISK DIRECTORY
SHARED DATA
302
288
CONFIGURATION MANAGER
290
POLICY
SHARED DATA
294
LOCAL DISK CACHE
298
PHYSICAL FILE SYSTEM

FIG. 8

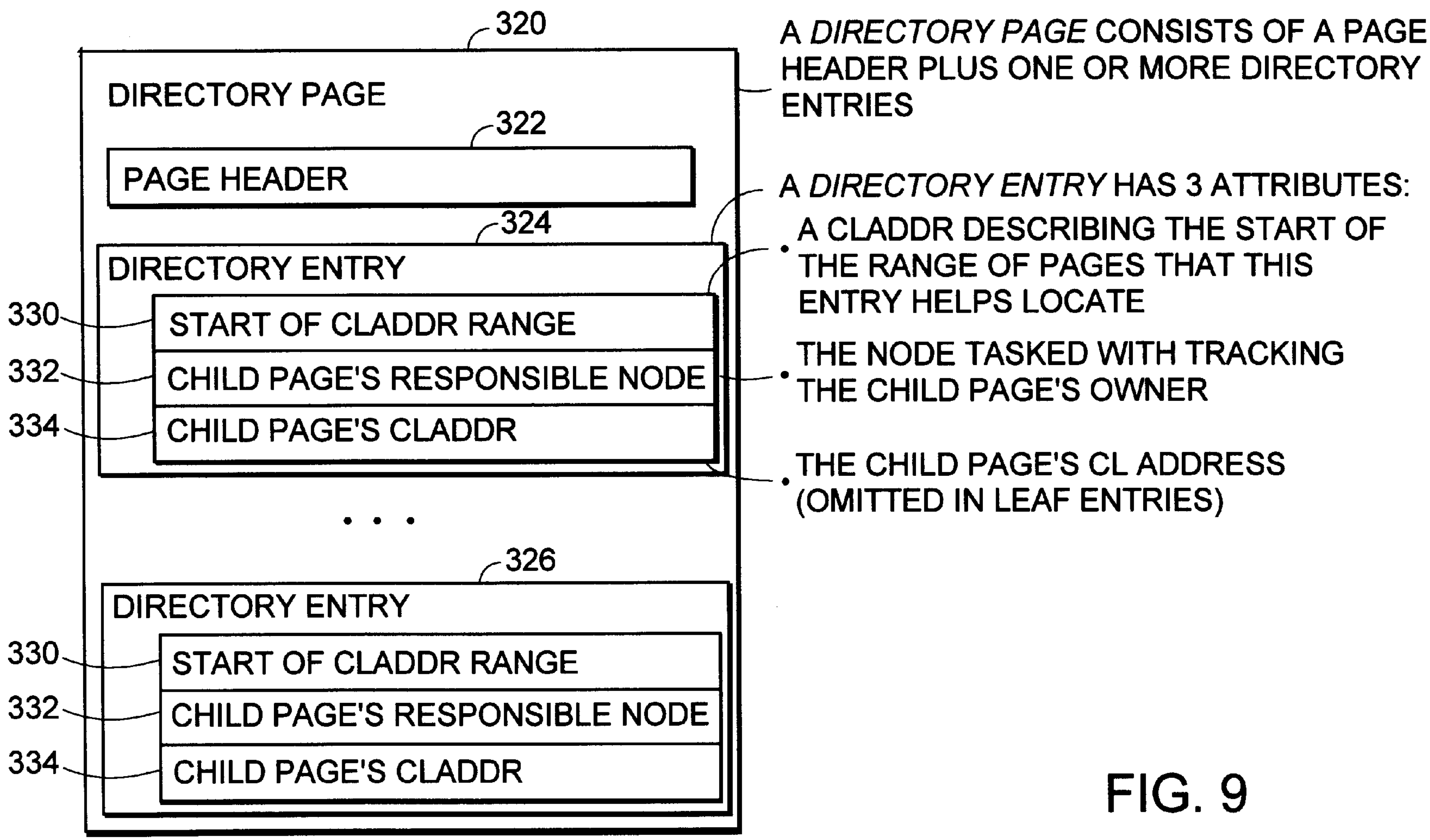

320
DIRECTORY PAGE
322
PAGE HEADER
324
DIRECTORY ENTRY
330
START OF CLADDR RANGE
332
CHILD PAGE'S RESPONSIBLE NODE
334
CHILD PAGE'S CLADDR
. . .
326
DIRECTORY ENTRY
330
START OF CLADDR RANGE
332
CHILD PAGE'S RESPONSIBLE NODE
334
CHILD PAGE'S CLADDR
A DIRECTORY PAGE CONSISTS OF A PAGE HEADER PLUS ONE OR MORE DIRECTORY ENTRIES
A DIRECTORY ENTRY HAS 3 ATTRIBUTES:
• A CLADDR DESCRIBING THE START OF THE RANGE OF PAGES THAT THIS ENTRY HELPS LOCATE
• THE NODE TASKED WITH TRACKING THE CHILD PAGE'S OWNER
• THE CHILD PAGE'S CL ADDRESS (OMITTED IN LEAF ENTRIES)

FIG. 9

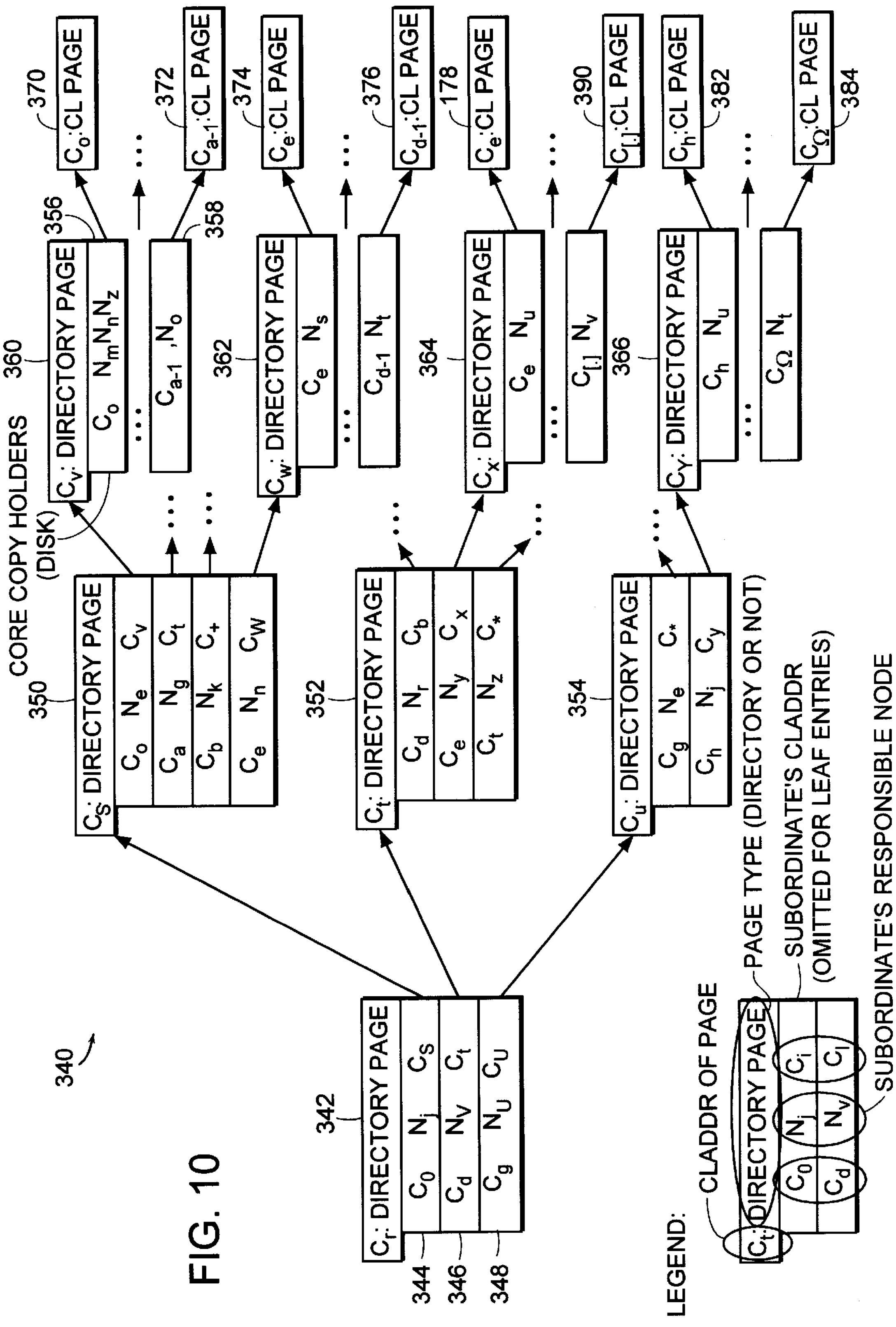

340
342
344
346
348
350
352
354
356
358
360
362
364
366
370
372
374
376
178
390
382
384
CORE COPY HOLDERS (DISK)
DIRECTORY PAGE
CL PAGE
LEGEND:
CLADDR OF PAGE
PAGE TYPE (DIRECTORY OR NOT)
SUBORDINATE'S CLADDR (OMITTED FOR LEAF ENTRIES)
SUBORDINATE'S RESPONSIBLE NODE

FIG. 10

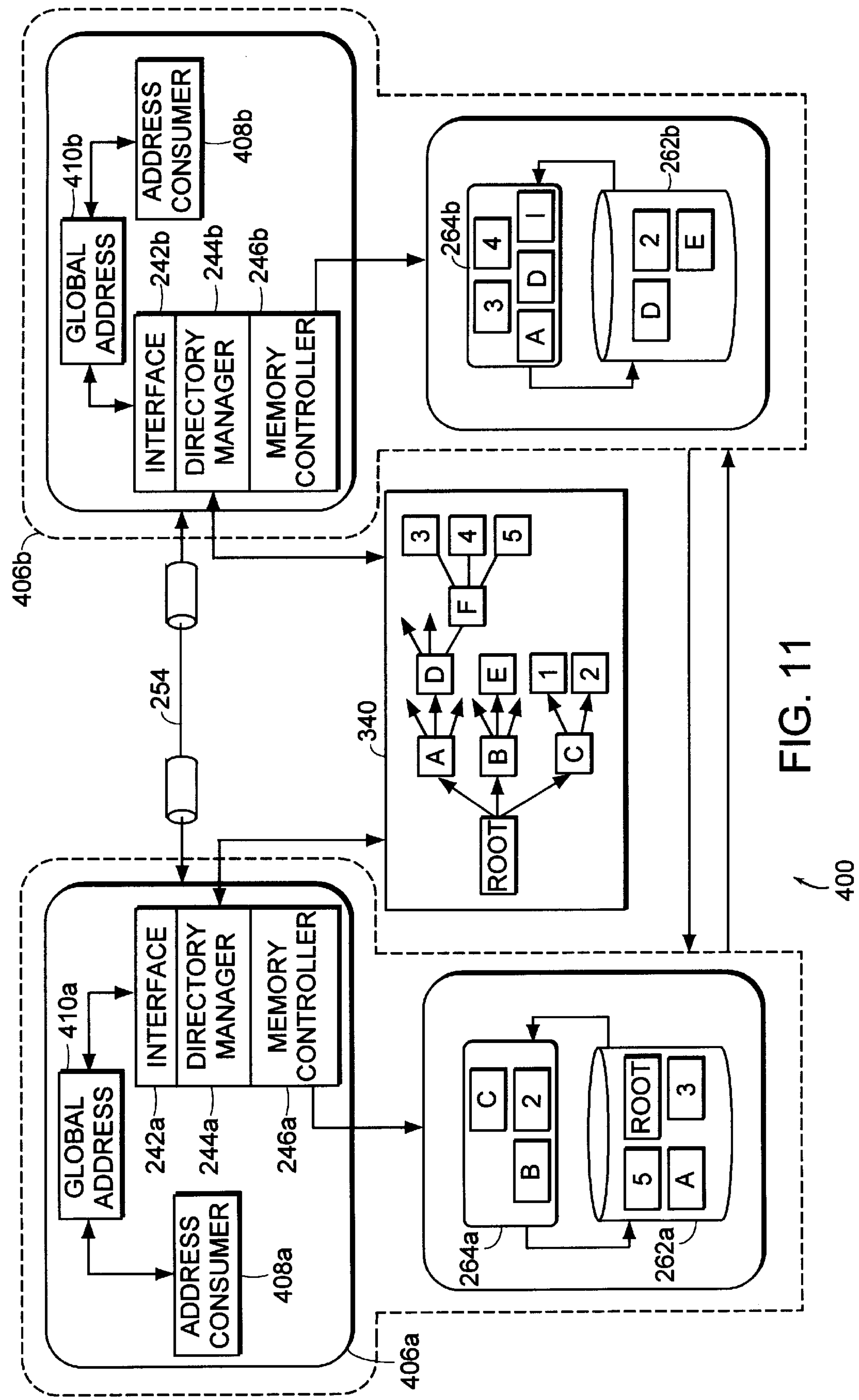

GLOBAL ADDRESS
410b
ADDRESS CONSUMER
408b
INTERFACE
DIRECTORY MANAGER
MEMORY CONTROLLER
242b
244b
246b
264b
262b
3
4
A
D
I
D
2
E
406b
254
340
ROOT
A
B
C
D
E
F
1
2
3
4
5
400
GLOBAL ADDRESS
410a
ADDRESS CONSUMER
408a
INTERFACE
DIRECTORY MANAGER
MEMORY CONTROLLER
242a
244a
246a
264a
262a
C
2
B
ROOT
3
5
A
406a

FIG. 11

STRUCTURED DATA STORAGE USING GLOBALLY ADDRESSABLE MEMORY

CROSS-REFERENCE TO RELATED APPLICATION

This application is a continuation-in-part of co-pending U.S. patent application Ser. No. 08/754,481 filed Nov. 22, 1996 now pending, which is incorporated herein by reference in its entirety and which is owned by the assignee of this application.

TECHNICAL FIELD

The present invention relates in general to structured storage systems (e.g., file systems, database systems, and systems for storing, sharing, and delivering data objects, JAVA applets, and Web pages). More specifically, the invention relates to systems and methods that maintain a structured store of data, preferably within a distributed, addressable, shared memory space.

BACKGROUND INFORMATION

Computer based structured storage systems, such as computer file systems and database systems, have been remarkably successful at providing users with quick and facile access to enormous amounts of data. The importance of these structured storage systems in today's commerce is difficult to exaggerate. For example, structured storage systems have allowed businesses to generate and maintain enormous stores of persistent data that the company can modify and update over the course of years. For many companies, this persistent data is a valuable capital asset that is employed each day to perform the company's core operations. The data can be, for example, computer files (e.g., source code, wordprocessing documents, etc.), database records and information (e.g., information on employees, customers, and/or products), and/or Web pages.

A typical computer based structured storage system includes a central server, such as a file system server or a database system server, that provides centralized control over the structured store of data. The structured store of data is the information that is being maintained by the system, such as the information in the files and the directories of the file system or within the rows and columns of the tables of the database system. The central server provides system services to a plurality of interconnected network client nodes, and each of the client nodes employs the central server to access and manipulate the structured store of data. Accordingly, the central server provides a nucleus for the structured storage system and maintains central control over the system and the data stored therein.

Although such server based systems have worked generally well, problems arise from relying on centralized control of the structured data store. For example, the operation of the structured storage system is dependent upon the proper functioning of the central server. Any failure of the server to maintain proper operation, such as a power failure, hardware failure, or other such system failure, will disable the entire structured storage system and prevent users from accessing the data store. Additionally, a flood of client service requests issued from the individual network nodes can overload the server process and slow down or crash the system. Accordingly, reliance on centralized control of the structured storage system can result in slow operation during periods of heavy use, as well as result in system failures due to overloading the central server.

An additional problem with a client-server network system is that it provides a static operating environment that is set for optimal performance at a certain level of network activity. Consequently, the network fails to exploit available resources to improve system performance. In particular, as the system activity rises above or drops below the expected level of network activity, the static operating environment lacks any ability to reconfigure dynamically the allocation of network resources to one providing better performance for the present level of activity.

Technology has been developed to improve the reliability and operation of these centralized structured storage network systems. This technology has been mostly directed to the development of reliable database and file systems, and has generally involved one of two methods: (1) static mapping of the data to one or more servers, or (2) storing the data in a globally shared data repository, such as a shared disk.

Systems using the first method distribute portions of the structured store of persistent data statically across a plurality of servers. Each of the servers maintains a portion of the structured store of data, as well as optionally maintaining an associated portion of a directory structure that describes the portions of the data stored within that particular server. These systems guard against a loss of data by distributing the storage of data statically across a plurality of servers such that the failure of any one server will result in a loss of only a portion of the overall data. Other developments in clustered database technology provide for replicating portions of the structured store of data, and storing the replicated portions statically across a plurality of servers. Accordingly, these systems go further in guarding against the loss of data by providing static redundancy within the structured storage system. However, although known clustered database technology can provide more fault tolerant operation in that it guards against data loss, the known systems still rely on static allocation of the data across various servers. Since data is not dynamically allocated between servers: (1) system resources are not allocated based on system usage which results in under utilization of those resources; (2) scaleable performance is limited because new servers must be provided whenever the dataset grows or whenever one particular server cannot service requests made to its portion of the dataset; and (3) such static allocation still requires at least one of servers storing the information to survive in order to preserve the data.

Systems using the second method store the structured data in central data repository, such as a shared disk. Each node in the system continually updates the central data repository with its portion of the structured store. For example, in a database system, each node exports tables it is currently using to the data store. While this method exports the problems of load balancing to the central data repository, it suffers from two main drawbacks. First, throughput is lowered because of increased overhead associated with ensuring coherency of the centralized data store. Second, locking is inefficient because entire pages are locked when a node accesses any portion of a page. As a result, nodes may experience contention for memory even when no true conflict exists.

SUMMARY OF THE INVENTION

It is an object of the invention to provide improved storage systems for maintaining a structured store of data.

It is a further object of the invention to provide structured storage systems that are more reliable, provide greater fault tolerant operation, and have the ability to dynamically move data in response to network activity levels and access patterns in order to optimize performance and minimize node access times.

It is yet another object of the invention to provide structured storage systems that provide distributed control over a structured store of persistent data, where the data can include, for example, computer files, database records and information, or Web pages.

It is still a further object of the invention to provide distributed control to a plurality of different types of structured storage systems, such as file systems, database systems, and systems that store, share, and deliver Web pages to requesting nodes and/or requesting networks.

Further objects of the invention will, in part, be described and, in part, be apparent to those of ordinary skill from the following description and the accompanying drawings.

The invention can be understood as structured storage systems, and related methods, that employ a globally addressable unstructured memory system to maintain a structured store of persistent data within a shared memory space. Optionally, a shared memory system can be employed, such as a distributed shared memory system (DSM) that distributes the storage of data across some or all of the memory devices connected to a network. Memory devices that may be connected to the network include hard disk drives, tape drives, floppy disk drive, CD-ROM drives, optical disk drives, random access memory chips, or read-only memory chips.

The structured storage system can be a computer program that interfaces to a DSM to operate the DSM as a memory device that provides persistent storage of data. The structured storage system control program can direct the DSM to map file and directory data into the shared memory space. The DSM can include functionality to share data and coherently replicate data. In one embodiment, the DSM provides memory device services to the data control program. These services can include read, write, allocate, flush, or any other similar or additional service suitable for providing low level control of a memory storage device. The data control program employs these DSM services to allocate and access portions of the shared memory space for creating and manipulating a structured store of persistent data.

In one aspect, the invention relates to a method, and related system, for providing distributed control over a structured store of data. The method involves providing a plurality of nodes inter-connected by a network, and storing on each the node an instance of a data control program for manipulating the structured store of data to provide multiple, distributed instances of the data control program. The method also involves interfacing each the instance of the data control program to a shared memory system that provides addressable persistent storage of data, and operating each the instance of the data control program to employ the shared memory system as a memory device having the structured store of data contained therein, whereby the shared memory system coordinates access to the structured store of data to provide distributed control over the structured store of data.

Embodiments of this aspect of the invention include interfacing each the instance of the data control program to a DSM that provides distributed storage across the interconnected nodes and that provides persistent storage of data. The interface step can further include directing the data control program to provide a stream of data to be stored in the structured store of data and directing the data control program to operate the shared memory system as a single-node memory device.

Other embodiments of this aspect of the invention include operating the shared memory system to replicate stored data coherently to provide a redundant store of data, and storing the coherently replicated data within different storage devices of the network to provide fault tolerant operation. Also included is coordinating shared access to data within the structured store by locking objects stored within a shared memory space, and generating a lock object data structure having information representative of a lock status on portions of the shared memory space and storing the lock object within the shared memory space to provide a shared system lock. Objects can be locked by directing the shared memory to generate locks on portions of the shared memory space. Also, the data control program can compress data to be stored in the structured store of data.

Still other embodiments according to this aspect of the invention include embodiments in which the structured store of data comprises a file system, a database system, a Web page system, or generally any object storing, retrieving, manipulating, and supplying the system. For the file system embodiment, the data control program comprises a file control program for manipulating the file system whereby the shared memory system controls access to the file system to provide a shared file system. For the database system embodiment, the data control program comprises a database control program for manipulating the database system, whereby the shared memory system controls access to the database system to provide a shared database system. For the Web page system embodiment, the data control program comprises a Web page control program for manipulating the Web page system, whereby the shared memory system controls access to the Web page system to provide a shared Web page system. For any of these particular embodiments, the shared system uses a directory and operates the shared memory system to maintain the directory within a shared memory space, and the directory is organized as a plurality of sets stored within the shared memory space. Also, for an object (e.g., file, database record, Web page, etc.) stored within the shared system, a descriptor is generated that has storage for a identifier being representative of a portion of a shared memory space, and contiguous portions of the shared memory space can be allocated, each represented by a respective identifier, to provide reduced bookkeeping information for the respective file and to optimize access to physical storage for the file.

The foregoing and other objects, aspects, features, and advantages of the invention will become more apparent from the following description and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, like reference characters generally refer to the same parts throughout the different views. Also, the drawings are not necessarily to scale, emphasis instead generally being placed upon illustrating the principles of the invention.

FIG. **1** is a conceptual block diagram of a distributed addressable shared memory structured data storage system according to the invention.

FIG. **2** is a diagram of one possible embodiment of the system of FIG. **1**, namely a distributed addressable shared memory file system providing storage for computer files such as source code files, wordprocessing documents files, etc.

FIG. **3** is a graphical representation of the organization of directory entries and associated file descriptors (also known as "Inodes"), suitable for use with the file system of FIG. **2**.

FIG. **4** is a diagram of an Inode suitable for use with the file system of FIG. **2**.

FIG. **5** illustrates a distributed shared memory computer network.

FIG. **6** is a functional block diagram that illustrates in more detail one distributed shared memory computer network of the type shown in FIG. **5**.

FIG. **7** illustrates in more detail a shared memory subsystem suitable for practice with the network illustrated in FIG. **6**.

FIG. **8** is a functional block diagram of one shared memory subsystem according to the invention.

FIG. **9** illustrates a directory page that can be provided by a shared memory subsystem of the type depicted in FIG. **8**.

FIG. **10** illustrates a directory that can be distributed within a shared memory and formed of directory pages of the type illustrated in FIG. **9**.

FIG. **11** illustrates in functional block diagram form a system that employs a directory according to FIG. **10** for tracking portions of a distributed shared memory.

DESCRIPTION

A network system **10** according to the invention includes a plurality of network nodes that access a memory space storing a structured store of data, such as a structured file system or a database. Each of the nodes includes at least a data control program which accesses and manages the structured store of data. 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. In such an embodiment, the data control program can access a desired portion of the structured store using a globally unique identifier. The underlying system would translate the identifier into one or more commands for accessing the desired data, including network transfer commands. In another embodiment, 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.

The system **10** can be a file system, a database system, a Web server, an object repository system, or any other structured storage system that maintains an organized set of data. As used herein, the term "Web server" means any processor which transmits data objects (such as Active X objects), applications (such as JAVA applets), or files (such as HTNL files), to a requestor via Web protocols (e.g., http or ftp). In one disclosed embodiment, the system **10** is a file system that maintains various computer files. However, this is just one embodiment of the invention that is provided for illustrative purposes. The invention can be employed to provide any one of a plurality of structured storage systems (e.g., database system, Web page system, Intranet, etc.). The invention is not to be limited to the file system or other particular embodiments described herein.

Referring to FIG. **1**, a network system **10** according to the invention includes a plurality of network nodes **12***a*–**12***d* and an addressable shared memory space **20** that has a portion **22** for storing a structured store of data **28**. Each of the nodes **12***a*–**12***d* can include several sub-elements. For example, node **12***a* includes a processor **30***a*, a data control program **32***a*, and a shared memory subsystem **34***a*. In the disclosed embodiment, two of the nodes, **12***a* and **12***c*, include monitors that provide displays **40** and **42** graphically depicting the structured store of data **28** within the addressable shared memory space **20**. The addressable shared memory space **20** interconnects each of the network nodes **12***a*–**12***d* and provides each node **12***a*–**12***d* with access to the structured store of data **28** contained within the addressable shared memory space **20**.

A system **10** according to the invention can provide, among other things, each network node **12***a*–**12***d* with shared control over the structured store of data **28** and, therefore, the system **10** can distribute control of the data store across the nodes of the network. To this end, each node of the system **10**, such as node **12***a*, includes a data control program **32***a* that interfaces to a shared memory subsystem **34***a*. The data control program **32***a* can 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. At the direction of the data control program **32***a*, the shared memory subsystem **34***a* can access and store data within the addressable shared memory space **20**. These cooperating elements provide a structured storage system that has a distributed architecture and thereby achieves greater fault tolerance, reliability, and flexibility than known structured storage systems that rely on centralized control and centralized servers. Accordingly, the invention can provide computer networks with distributively controlled and readily scaled file systems, database systems, Web page systems, object repositories, data caching systems, or any other structured storage system.

Still referring to FIG. **1**, the system **10** of the invention maintains within the addressable shared memory space **20** a structured store of data **28**. Each of the nodes **12***a*–**12***d* can access the addressable shared memory space **20** through the shared memory subsystems **34***a*–**34***d*. Each of the shared memory subsystems **34***a*–**34***d* provides its node with access to the addressable shared memory space **20**. The shared memory subsystems **34***a*–**34***d* coordinate each of the respective node's memory access operations to provide access to the desired data and maintain data coherency within the addressable shared memory space **20**. This allows the interconnected nodes **12***a*–**12***d* to employ the addressable shared memory space **20** as a space for storing and retrieving data. At least a portion of the addressable shared memory space **20** is supported by a physical memory system that provides persistent storage of data. For example, a portion of the addressable shared memory space **20** can be assigned or mapped to one or more hard disk drives that are on the network or associated with one or more of the network nodes **12***a*–**12***d* as local hard disk storage for those particular nodes. Accordingly, FIG. **1** illustrates that systems of the invention have shared memory subsystems providing the network nodes with access to an addressable shared memory space, wherein at least a portion of that space is assigned to at least a portion of one or more of the persistent storage memory devices (e.g., hard disks) to allow the nodes addressably to store and retrieve data to and from the one or more persistent storage memory devices. A preferred embodiment of such an addressable shared memory space is described in the commonly-owned U.S. patent application Ser. No. 08/754,481 filed Nov. 22, 1996, and incorporated by reference above.

Therefore, one realization of the present invention is that each of the nodes **12***a*–**12***d* can employ its respective shared memory subsystem as a memory device that provides persistent data storage.

Each of the data control programs **32***a*–**32***d* is a software module that couples to the respective shared memory subsystem 34a–34d in a way that operates similarly to an interface between a conventional data storage program and a local memory device. For example, the data control program 32a can stream data to, and collect data from, the shared memory subsystem 34a. Because the shared memory subsystems coordinate the memory accesses to the addressable shared memory space 20, each of the data control programs is relieved from having to manage and coordinate its activities with the other data control programs on the network or from having to manage and coordinate its activities with one or more central servers. Accordingly, each of the data control programs 32a–32d can be a peer incarnation (i.e., an instance) residing on a different one of the network nodes 12a–12d and can treat the respective shared memory subsystem 34a–34d as a local memory device such as a local hard disk.

One or more of the data control programs 32a–32d can provide a graphical user interface 42 that graphically depicts the structured store of data 28 contained within the addressable shared memory space 20. The graphical user interface 42 allows a user at a node, for example at node 12a, to insert data objects graphically within the structured store of data 28. To this end, the data control program 32a can generate a set of commands that will present a stream of data to the shared memory subsystem 34a and the shared memory subsystem 34a will employ the data stream to store an object within the structured store of data 28. Similarly, the other shared memory subsystems 34b–34d can provide information to their respective nodes that is indicative of this change to the structured store of data 28. Accordingly, as shown depicted in FIG. 1 for node 12c only for simplicity, that node (which includes a graphical user interface 40) reflects the change to the structured store of data 28 affected by the data control program 32a of the node 12a. In particular, the graphical user interface 40 of the node 12c can depict to a user that an object is being placed within the structured store of data 28. For example, the addressable shared memory space 20 also contains the data objects 50a–50c which can be placed within the structured data store 28 to become part of that structured data store. As illustrated, a system user at node 12a can direct object 50a to be inserted at a set location within the data store 28. The data control program 32a then directs the shared memory subsystem 34a to place the object 50a within the data store 28 at the proper location. Moreover, the shared memory subsystem 34c on node 12c detects the change within the data store 28 and reflects that change within the graphical user interface 40.

Referring now to FIG. 2, a structured file system 60 is a particular embodiment according to the invention 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 when in fact it spans all network nodes coupled to the addressable shared memory space 20.

The file system 60 of FIG. 2 differs from known physical and distributed file systems in a variety of ways. In contrast to known physical file systems which map a file organization onto disk blocks, the file system 60 according to the invention manages the mapping of a directory and file structure onto a distributed addressable shared memory system 20 which has at least a portion of its addressable space mapped or assigned to at least a portion of one or more persistent storage devices (e.g., hard disks) on the network. Unlike known distributed file systems, the file system 60 of the invention employs peer nodes, each of which have an incarnation or instance of the same data control program. Also, unlike known file systems generally, the file system 60 of the invention: maintains data coherence among network nodes; automatically replicates data for redundancy and fault tolerance; automatically and dynamically migrates data to account for varying network usage and traffic patterns; and provides a variety of other advantages and advances, some of which are disclosed in the commonly-owned U.S. patent application Ser. No. 08/754,481 filed Nov. 22, 1996, and incorporated by reference above.

Still referring to FIG. 2, the file system 60 resides in part within the addressable shared memory space 20, and includes a structured store of data 62, a super root 64, file sets 66–74, directory entry 80, and file or document 82. Two network nodes 84 and 86 are shown accessing the addressable shared memory space 20 (in the manner described previously with reference to FIG. 1) via the logical drives 90 and 94. Application programs 92 and 96 executing on the nodes interact with the data control programs (not shown in FIG. 2 but shown in FIG. 1 as 32a–32d) and cause the data control programs in the nodes to access the logical drives 90 and 94. In the disclosed embodiment, the logical drives are DOS devices that "connect to" the fileset directories via Installable File System drivers associated with the file system 60.

The file system 60 supports one global file system per addressable shared memory space 20 shared by all of the network nodes. This global file system is organized into one or more independent collections of files, depicted as the filesets 66–74. A fileset can be thought as logically equivalent to a traditional file system partition. It is a collection of files organized hierarchically as a directory tree structure rooted in a root directory. The non-leaf nodes in the tree are the directories 80, and the leaves in the tree are regular files 82 or empty directories. Sub-directory trees within a fileset can overlap by linking a file to multiple directories.

A benefit of breaking up the file system 60 into filesets 66–74 is that it provides more flexible file system management for users of the system 60. As the file system 60 grows into very large sizes (e.g., hundreds of nodes with thousands of gigabits of storage), it is desirable to have the files organized into groups of management entities such that management actions can be independently applied to individual groups without affecting the operation of the others.

The filesets in the addressable shared memory space 20 are described and enumerated in a common structure, the root 64 of which provides the starting point to locate the filesets in the addressable shared memory space 20. The root 64 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. When a node is accessing a fileset for the first time, it first looks up the root 64 to determine the identifier associated with the fileset, e.g., the shared memory address used to access the fileset. Once it has determined the identifier, the node can access the root directory of the fileset. From the root directory, it then can traverse the entire fileset directory tree to locate the desired file. Filesets used by the file system 60 are described in greater detail below under the heading "Fileset."

Referring to FIG. 3, in the disclosed embodiment of the file system 60 according to the invention, a directory 126 (such as the directory 80 of FIG. 2) is accessed by starting at a directory Inode or descriptor 128 containing an address that points to a directory entries stream descriptor 130. This descriptor 130 is a pointer to a block of data containing directory entries for files File 1 through File 3. The directory entry for File 1 has a number of entries; one of the entries is a string containing the name of the file and another entry is the address of the Inodes and stream descriptors **132**. The stream descriptors for File **1** are used to locate and retrieve the various 4 kilobyte pages in the addressable shared memory space **20** that constitute File **1**. Other files are retrieved and constructed from the addressable shared memory space **20** in the same fashion. The directories used by the file system **60** are described in greater detail below under the heading "Directory."

In the embodiment of the file system **60** disclosed in FIG. **4**, a file **98** (such as the file **82** of FIG. **2**) is represented by one or more shared pages of data **100**, **102**, **104**, **106**, and **108** in the addressable shared memory space **20**. Each file **98** has a file Inode or descriptor **110** that includes various file attributes **112**. The file descriptor **110** contains an address that points to a data stream descriptor **114**, and the data stream itself includes one or more addresses **116**, **118**, **120**, **122**, and **124** that point to particular pages in the identifiable shared memory space **20**. In the disclosed embodiment, a page is the atomic unit in the addressable shared memory space **20**, and it contains up to 4 kilobytes of data. Even if the entire 4 kbytes is not needed, an entire page is used. This is illustrated by the page **108** that only contains about 2 kbytes of data. The files used by the file system **60** are described in greater detail below under the heading "Files."

FILESET

The filesets are the basic unit for the file system **60**. Each fileset is identified with a name having up to 255 characters. The file system **60** exports a set of fileset level operations that allow an administrator to manage the filesets through the following type of actions.

Fileset Creation

This operation creates a new fileset. The fileset is initially created with one file, the empty root directory. A default fileset is created automatically at the initialization of the addressable shared memory space **20**.

Fileset Deletion

This operation deletes a fileset. All files in the fileset are removed, and all shared memory space allocated to the files in the fileset is discarded and the backing physical storage freed for new storage. The file system **60** will only allow deletion of a fileset until there are no open handles to file data stream in the fileset. In order to ready a fileset for deletion, the fileset must be "shutdown" by putting it off-line.

Fileset Enumeration

This operation enumerates a specific fileset, or all the filesets, in the addressable shared memory space **20**.

Fileset Control

This operation performs fileset level control routines such as setting fileset attributes.

Mount Export Control

Directory are attached to local devices, i.e. "mounted" using parameters stored in the Windows NT registry, or some other similar central storage area for such information. When first started up, the data control program **60** accesses the central storage and determines which filesets should be mounted. The data control program creates a file object representing each fileset identified by the entries in the central storage. In some embodiments an API may be provided which allows the data control program **60** to dynamically mount and unmount filesets by making appropriate API calls.

The users of the file system **60** are not aware of the shared memory "logical volume," but rather view each fileset as a volume (or partition in the sense of a traditional physical file system). The Win32 GetVolumeInformation is used to get information on the fileset (more precisely, on the logical device on which the fileset is attached to). Because all the filesets share the same pool of the storage in the addressable shared memory space **20**, the total volume size returned to the user for each fileset is the current aggregate storage capacity in the addressable shared memory space **20**. The same approach is taken for the total free space information, and the aggregate value of the addressable shared memory space **20** is returned for each fileset.

DIRECTORY

Directory entry scanning is one of the most frequently performed operations by user applications. It is also may be the most visible operation in terms of performance. Consequently, much attention is directed to making the directory scan efficient and the WindowsNT Files System (NTFS) duplicates sufficient file Inode information in the directory entry such that a read directory operation can be satisfied by scanning and reading the directory entries without going out to read the information from the file Inodes. The problem with this scheme is that the doubly stored file metadata, such as the file time stamps and file size, can be updated quite frequently, making the metadata update more expensive. However, this overhead is considered acceptable in face of the performance gained in directory scan operations.

The file system **60** adopts the same philosophy of providing efficient directory scanning by duplicating file Inode information in directory entries. Each directory entry contains sufficient information to satisfy the Win32 query file information requests. The file Inode is stored with the file stream descriptors on a separate page. The Inode is located via a pointer in the directory entry.

The file system's directory entries are stored in the directory file's directory entry data stream. To maximize space utilization, each directory entry is allocated on the first available free space in a page that can hold the entire entry. The length of the entry varies depending on the length of the file's primary name. The following information is part of the directory entry: creation time; change time; last write time; last accessed time; pointers to stream descriptor; pointer to parent directory Inode; MS-DOS type file attributes; and MS-DOS style file name (8.3 naming convention). For average file name lengths, a page contains up to about 30 entries. All the file information in the directory entry is also contained in the file Inode, except for the file primary name and MS-DOS file name. The file primary names and associated short names are only stored in the directory entries. This makes the Inode size fixed.

When a file information is modified (except for file names), the Inode is updated in the context of the update transaction and therefore always contains the most up-to-date information. The associated directory entry change is lazily flushed to reduce the cost of double updating. This means the Inode updates are either flushed or recoverable, but not the corresponding directory entry updates. If the directory entry gets out of synch with the Inode (when the Inode change is successfully flushed but not the directory change), the entry is updated the next time the Inode is updated. In order to facilitate synchronization of directory updates, the directory entries (Inodes) can not span multiple pages. FIG. **3** illustrates the organization of directory entries and associated Inodes.

FILES

A file of the file system **60** comprises streams of data and the file system metadata to describe the file. Files are described in the file system **60** by objects called Inodes. The Inode is a data structure that stores the file metadata. It represents the file in the file system **60**.

A data stream is a logically contiguous stream of bytes. It can be the data stored by applications or the internal information stored by the file system **60**. The data streams are mapped onto pages allocated from the addressable shared memory space **20** for storage. The file system **60** segments a data stream into a sequence of 4 kilobyte segments, each segment corresponding to a page. The file system **60** maintains two pieces of size information per data stream: the number of bytes in the data stream; and the allocation size in number of pages. The byte-stream to segment/page mapping information is part of the file metadata and is stored in a structure called data stream descriptor. See FIG. **4**.

Users' requests for data are specified in terms of range of bytes and the position of the starting byte measured by its offset from the beginning of the data stream, byte position zero. The file system **60** maps the offset into the page containing the starting byte and the intra-page offset from the beginning of the page.

Every file of the file system **60** has at least two data streams: the default data stream; and the Access Control List (ACL) stream. Each file may optionally have other data streams. The ACL stream is used to store the security Access Control Lists set on the file. Each data stream is individually named so that the user can create or open access to a specific data stream. The name of the default data stream is assumed to be the primary name of the file. To access a data stream, the user of the file system **60** must first open a file handle to the desired data stream by name. If the file name is used then the handle to the default data stream is opened. This open file handle represents the data stream in all the file system services that operates on the data stream.

The file system **60** exports a set of services to operate at the file level. The input to the services are the file object handle (Inode) or the data stream object handle, and the operation specific parameters, including the desired portions of the data stream in byte positions.

Open files are represented by data stream objects (or just file objects). Users access files using these file objects, identified to the users through file handles. A file handle is a 32-bit entity representing an instance of an open file stream. For example, WindowsNT creates the file object and returns a file handle to the users in response to the user request for file creation or file open. The file system **60** initializes a pointer to a file control block. Multiple file objects point to the same file control block and each file control block maintains separate stream objects for each open context. Externally, the file handle is opaque to the users. Multiple opens can be issued against the same file. When the user closes a file, the file object and the associated file handle is removed.

The file system **60** maps file streams into sequences of segments which become progressively larger; each segment corresponds to one or more pages. The file system **60** attempts to reserve contiguous pages for data streams but only allocates real backing storage on an as needed basis, usually as a result of a file extension requested by writing beyond the data stream allocation size. When a file extension request is received, the file system **60** rounds the extension size in number of bytes up to a multiple of 4 kilobytes to make it an integer number of pages, and requests pages for actual allocation. The number of 4 kilobyte pages allocated by the file system depends on the number of file extension requests made. The file system **60** allocate one 4 kilobyte page for the first extension request, two 4 kilobyte pages for the second request, four 4 kilobyte pages for the third extension request, and so on. The newly allocated pages are zero filled. By reserving contiguous pages, the file system **60** can reduce the amount of bookkeeping information on the byte offset to page mapping. The file system **60** reserves (sometimes much) larger than requested memory space for a file, and substantiates the storage by allocating backing storage page by page.

Four kilobyte allocation segments are chosen to reduce the unused storage space and yet provide a reasonable allocation size for usual file extensions. Since allocation is an expensive operation (most likely involving distributed operations), smaller allocation size is not efficient. Larger allocation size would lead to inefficient space utilization, or additional complexity to manage unused space. A 4 kilobyte segment also maps naturally to a page, simplifying the data stream segment to page mapping. Although an analogy could be made with the NTFS's allocation policy of 4 kilobyte clusters (segment) size for large disks to speed up allocation and reduce fragmentation, such analogy is not completely valid because the actual on-disk allocation segment size depends greatly on the local disk size and the physical file systems.

Similar to the NTFS, which controls the allocation of each disk partition and therefore can quickly determine the free volume space available for allocation, the file system **60** requests the total available space information and uses this information to quickly determine whether to proceed with the allocation processing. If the total available space is less than the required allocation size, the request is denied immediately. Otherwise, the file system **60** will proceed to allocate the pages to satisfy the request. The fact that the file system **60** can proceed with the allocation does not guarantee that the allocation will succeed, because the actual total available space may change constantly.

The file system **60** takes advantage of the page level replication capability of the underlying distributed addressable shared memory system **20** disclosed in the U.S. patent application incorporated by reference above. Page level replication allows the system to provide file replication. The data streams of a replicated file are backed by pages, which are themselves replicated. In this way, data streams are replicated automatically without intervention of the file system **60**. The extra space consumed by the multiple replicas is not reflected in the file (data stream) sizes. The stream allocation size still reports the total allocation size in pages required for one replica. The pages backing temporary files, however, are not replicated.

FILE ACCESS AND RESOURCE SHARING - LOCKING

The shared memory provides the distribution mechanism for resource sharing among peer nodes running the file system **60** software. Each instance of the file system **60** on each network node views the shared memory resources (i.e., pages) as being shared with other local or remote threads. The file system **60** needs a way to implement high level, file system locks to provide consistent resource sharing. Any concurrency control structure can be used to implement locks, such as lock objects or semaphores. In database applications, locking may also be achieved by implementing concurrency control structures associated with database indices or keys. In file system applications access to files or directories may be controlled. Another example of file system locks is Byte Range Locking, which provides the users the ability to coordinate shared access to files. A byte range lock is a lock set on a range of bytes of a file. Coordinated shared access to a file can be accomplished by taking locks on the desired byte ranges. In general, the high level file system lock works in the following fashion: (a) a file system resource is to be shared by each file system **60** instance, and the access to the resource is coordinated by a locking protocol using a lock object data structure that represents the high level lock to coordinate the shared resource, and it is the value of the data structure that represents the current state of the lock; (b) to access the resource, the instance at each node must be able to look at the state (or value) of the lock data structure, and if it is "free," modify it so that it becomes "busy," but if it is "busy," then it has to wait to become "free," and there could be intermediate states between "free" and "busy" (i.e., more than two lock states), but in any event, in this byte range locking example, a lock is a description of a certain byte range being shared/exclusively locked by some thread of the file system **60**, and a conflicting new byte range lock request that falls in or overlaps the already locked byte range will be denied or the requester may block (depending on how the request was made); and (c) access to or modification of the lock data structure by each node's instance needs to be serialized so that it in turn can then be used to coordinate high level resource sharing.

The locking features and capabilities of the shared memory engine described in the U.S. patent application Ser. No. 08/754,481, incorporated by reference above, allow the file system **60** to coordinate access to pages. The engine can also be used to coordinate access to resources, but in the case of complex high level resource locking such as Byte Range Locking, using the engine's locking features and capabilities directly to provide locks may be too costly for the following reasons: (a) each byte range lock would require a page representing the lock, and since the number of byte range locks can be large, the cost in terms of page consumption may be too high; and (b) the engine locks only provide two lock states (i.e., shared and exclusive), and high level file system locks may require more lock states.

The file system **60** of the invention implements the file system locking using the engine locking as a primitive to provide serialization to access and update the lock data structures. To read a lock structure, the file system **60** takes a shared lock on the data structure's page using the engine locking features and capabilities before it reads the page to prevent the data structure being modified. To modify the lock structure, it sets a exclusive lock on the page. The page lock is taken and released as soon as the lock structure value is read or modified.

With the serialization provided by the page locking and the page invalidation notification, the file system **60** implements the high level locks in the following way: (a) to take a file system lock (FS lock), the file system **60** sets a shared lock on the FS lock page and reads the page and then examines the lock structure; (b) if the lock structure indicates the resource is unlocked or locked in compatible lock mode, then the file system **60** requests to exclusively lock the page, and this guarantees only one file system **60** node instance can modify the lock data structure, and if the request succeeds then the file system **60** write maps the lock page and then changes the lock structure to set the lock and unlocks the page and sets page access to none; and (c) if the resource is locked in incompatible lock mode, the file system **60** unlocks the page but retains the page read mapped, and it then puts itself (the current thread) in a queue and waits for a system event notifying that the lock value has changed, and when the lock value does change then the file system **60** thread gets notified and repeats the step (a) above. The file system **60** implements the notification using a signal primitive. The file system **60** threads waiting for a lock are blocked on a system event. When the page containing the lock changes, a signal is sent to each blocked file system **60** thread. Each blocked file system **60** threads then wakes up and repeats step (a). FS locks are stored in volatile pages.

FILE ACCESS AND RESOURCE SHARING - BYTE RANGE LOCKING

Byte Range Locking is a file system locking service exported to the users through the Win32 LockFile() and LockFileEx() API. It allows simultaneous access to different non-overlapping regions of a file data stream by multiple users. To access the data stream, the user locks the region (byte range) of the file to gain exclusive or shared read access to the region.

The file system **60** supports byte range locking for each individual data stream of the file. The following Win32-style byte range locking behavior is supported: (a) locking a region of a file is used to acquire shared or exclusive access to the specified region of the file, and the file system **60** will track byte range locks by file handle, therefore file handles provide a way to identify uniquely the owner of the lock; (b) locking a region that goes beyond the current end-of-file position is not an error; (c) locking a portion of a file for exclusive access denies all other processes both read and write access to the specified region of the file, and locking a portion of a file for shared access denies all other processes write access to the specified region of the file but allows other processes to read the locked region, and this means that the file system **60** must check byte range locks set on the data stream not only for lock requests but for every read or write access; (d) if an exclusive lock is requested for a region that is already locked either shared or exclusively by other threads, the request blocks or fails immediately depending on the calling option specified.; and (e) locks may not overlap an existing locked region of the file.

For each byte range lock, the file system **60** creates a byte range lock record to represent the lock. The record contains the following information: (a) byte range; (b) lock mode (shared or exclusive); (c) process identification; and (d) a Win32 lock key value.

The file system **60** regards the file byte ranges as resources with controlled access. For each byte range lock record, the file system **60** creates a file system lock (as discussed above) to coordinate the access to the byte range "resource." A compatible byte range lock request (share lock) translates into taking read lock on the file system lock associated with the byte range record. An exclusive byte range lock request is mapped to taking write lock on the file system lock.

Using the file system locking mechanism discussed above, lock requests waiting on the page containing the desired byte range will be notified when the page content changes.

Addressable Shared Memory Space

Having described the invention and various embodiments thereof in some detail, a more detailed description is now provided of the addressable shared memory space that is disclosed in the commonly-owned U.S. patent application Ser. No. 08/754,481 filed Nov. 22, 1996, and incorporated by reference above. All of the information provided below is contained in that patent application.

The addressable shared memory system disclosed in the U.S. patent application incorporated by reference is an "engine" that can create and manage a virtual memory space that can be shared by each computer on a network and can span the storage space of each memory device connected to the network. Accordingly, all data stored on the network can be stored within the virtual memory space and the actual physical location of the data can be in any of the memory devices connected to the network.

More specifically, 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. The global address signal can be decoupled from, i.e. unrelated to, the physical and identifier spaces of the underlying computer hardware, to provide support for a memory space large enough to span each volatile and persistent memory device connected to the system. For example, systems of the invention can operate on 32-bit computers, but can employ global address signals that can be 128 bits wide. Accordingly, the virtual memory space spans 2^{128} bytes, which is much larger than the 2^{32} address space supported by the underlying computer hardware. Such an address space can be large enough to provide a separate address for every byte of data storage on the network, including all RAM, disk and tape storage.

For such a large virtual memory space, typically only a small portion is storing data at any time. Accordingly, the system includes a directory manager that tracks those portions of the virtual memory space that are in use. The system provides physical memory storage for each portion of the virtual memory space in use by mapping each such portion to a physical memory device, such as a RAM memory or a hard-drive. Optionally, the mapping includes a level of indirection that facilitates data migration, fault-tolerant operation, and load balancing.

By allowing each computer to monitor and track which portions of the virtual memory space are in use, each computer can share the memory space. This 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.

In one aspect, the invention of the above-identified, incorporated-by-reference U.S. patent application can be understood to include computer systems having a addressable shared memory space. The systems can comprise a data network that carries data signals representative of computer readable information a persistent memory device that couples to the data network and that provides persistent data storage, and plural computers that each have an interface that couples to the data network, for accessing the data network to exchange data signals therewith. Moreover, each of the computers can include a shared memory subsystem for mapping a portion of the addressable memory space to a portion of the persistent storage to provide addressable persistent storage for data signals.

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. To this same end, the system can also include a distributor for mapping portions of the addressable memory space across the plurality of local persistent memory devices and a disk directory manager for tracking the mapped portions of the addressable memory space to provide information representative of the local persistent memory device that stores that portion of the addressable memory space mapped thereon.

The systems can also include a cache system for operating one of the local persistent memory devices as a cache memory for cache storing data signals associated with recently accessed portions of the addressable memory space. Further the system can include a migration controller for selectively moving portions of the addressable memory space between the local persistent memory devices of the plural computers. The migration controller can determine and respond to data access patterns, resource demands or any other criteria or heuristic suitable for practice with the invention. Accordingly, the migration controller can balance the loads on the network, and move data to nodes from which it is commonly accessed. The cache controller can be a software program running on a host computer to provide a software managed RAM and disk cache. The RAM can be any volatile memory including SRAM, DRAM or any other volatile memory. The disk can be any persistent memory including any disk, RAID, tape or other device that provides persistent data storage.

The systems can also include a coherent replication controller for generating a copy, or select number of copies, of a portion of the addressable memory space maintained in the local persistent memory device of a first computer and for storing the copy in the local persistent memory device of a second computer. The coherent replication controller can maintain the coherency of the copies to provide coherent data replication.

The systems can also be understood to provide integrated control of data stored in volatile memory and in persistent memory. In such systems a volatile memory device has volatile storage for data signals, and the shared memory subsystem includes an element, typically a software module, for mapping a portion of the addressable memory space to a portion of the volatile storage. In these systems the volatile memory device can be comprised of a plurality of local volatile memory devices each coupled to a respective one of the plural computers, and the persistent memory device can be comprised of a plurality of local persistent memory devices each coupled to a respective one of the plural computers.

In these systems, a directory manager can track the mapped portions of the addressable memory space, and can include two sub-components; a disk directory manager for tracking portions of the addressable memory space mapped to the local persistent memory devices, and a RAM directory manager for tracking portions of the addressable memory space mapped to the local volatile memory devices. Optionally, a RAM cache system can operate one of the local volatile memory devices as a cache memory for cache storing data signals associated with recently accessed portions of the addressable memory space.

The systems can include additional elements including a paging element for remapping a portion of the addressable memory space between one of the local volatile memory devices and one of the local persistent memory devices; a policy controller for determining a resource available signal representative of storage available on each of the plural computers and, a paging element that remaps the portion of addressable memory space from a memory device of a first computer to a memory device of a second computer, responsive to the resource available signal; and a migration controller for moving portions of addressable memory space between the local volatile memory devices of the plural computers.

Optionally, the systems can include a hierarchy manager for organizing the plural computers into a set of hierarchical groups wherein each group includes at least one of the plural computers. Each the group can include a group memory manager for migrating portions of addressable memory space as a function of the hierarchical groups.

The system can maintain coherency between copied portions of the memory space by including a coherent replication controller for generating a coherent copy of a portion of addressable memory space.

The system can generate or receive global address signals. Accordingly the systems can include an address generator for generating a global address signal representative of a portion of addressable memory space. The address generator can include a spanning unit for generating global address signals as a function of a storage capacity associated with the persistent memory devices, to provide global address signals capable of logically addressing the storage capacity of the persistent memory devices.

In distributed systems, the directory manager can be a distributed directory manager for storing within the distributed memory space, a directory signal representative of a storage location of a portion of the addressable memory space. The distributed directory manager can include a directory page generator for allocating a portion of the addressable memory space and for storing therein an entry signal representative of a portion of the directory signal. The directory page generator optionally includes a range generator for generating a range signal representative of a portion of the addressable memory space, and for generating the entry signal responsive to the range signal, to provide an entry signal representative of a portion of the directory signal that corresponds to the portion of the addressable memory space. Moreover, the distributed directory manager can include a linking system for linking the directory pages to form a hierarchical data structure of the linked directory pages as well as a range linking system for linking the directory pages, as a function of the range signal, to form a hierarchical data structure of linked directory pages.

As the data stored by the system can be homeless, in that the data has no fixed physical home, but can migrate, as resources and other factors dictate, between the memory devices of the network, a computer system according to the invention can include a directory page generator that has a node selector for generating a responsible node signal representative of a select one of the plural computers having location information for a portion of the shared address space. This provides a level of indirection that decouples the directory from the physical storage location of the data. Accordingly, the directory needs only to identify the node, or other device, that tracks the physical location of the data. This way, each time data migrates between physical storage locations, the directory does not have to be updated, since the node tracking the location of the data has not changed and still provides the physical location information.

Accordingly, the system can include page generators that generate directory pages that carry information representative of a location monitor, such as a responsible computer node, that tracks a data storage location, to provide a directory structure for tracking homeless data. Moreover, the directory itself can be stored as pages within the virtual memory space. Therefore, the data storage location can store information representative of a directory page, to store the directory structure as pages of homeless data.

In another aspect, the invention of the above-identified, incorporated-by-reference U.S. patent application 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.

Turning now to the drawings related to the addressable shared memory system or engine of the above-identified, incorporated-by-reference U.S. patent application, FIG. **5** illustrates a computer network **10** that provides a shared memory that spans the memory space of each node of the depicted computer network **210**.

Specifically, FIG. **5** illustrates a computer network **210** that includes a plurality of nodes **212***a*–**212***c*, each having a CPU **214**, an operating system **216**, an optional private memory device **218**, and a shared memory subsystem **220**. As further depicted in by FIG. **5**, each node **212***a*–**212***c* connects via the shared memory subsystem **220** to a virtual shared memory **222**. As will be explained in greater detail hereinafter, by providing the shared memory subsystem **220** that allows the node **212***a*–**212***c* to access the virtual shared memory **222**, the computer network **210** enables network nodes **212***a*–**212***c* to communicate and share functionality using the same techniques employed by applications when communicating between applications running on the same machine. These techniques can employ object linking and embedding, dynamic link libraries, class registering, and other such techniques. Accordingly, the nodes **212** can employ the virtual shared memory **222** to exchange data and objects between application programs running on the different nodes **212** of the network **210**.

In the embodiment depicted in FIG. **5**, each node **212** can be a conventional computer system such as a commercially available IBM PC compatible computer system. The processor **214** can be any processor unit suitable for performing the data processing for that computer system. The operating system **216** can be any commercially available or proprietary operating system that includes, or can access, functions for accessing the local memory of the computer system and networking.

The private memory device **218** can be any computer memory device suitable for storing data signals representative of computer readable information. The private memory 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.

The shared memory subsystem **220** depicted in FIG. **5** is an embodiment of the invention that couples between the operating system **216** and the virtual shared memory **222** and forms an interface between the operating system **216** and the virtual shared memory to allow the operating system **216** to access the virtual shared memory **222**. The depicted shared memory subsystem **220** is a software module that operates as a stand-alone distributed shared memory engine. The depicted system is illustrative and other systems of the invention can be realized as shared memory subsystems that can be embedded into an application program, or be implemented as an embedded code of a hardware device. Other such applications can be practiced without departing from the scope of the invention.

The depicted virtual shared memory **222** illustrates a virtual shared memory that is accessible by each of the nodes **212***a*–**212***c* via the shared memory subsystem **220**. The virtual shared memory **222** can map to devices that provide physical storage for computer readable data, depicted in FIG. **5** as a plurality of pages **224***a*–**224***d*. In one embodiment, the pages form portions of the shared memory space and divide the address space of the shared memory into page addressable memory spaces. For example the address space can be paged into 4K byte sections. In other embodiments alternative granularity can be employed to manager the shared memory space. Each node **212***a*–**212***c* through the shared memory subsystem **220** can access each page **224***a*–**224***d* stored in the virtual shared memory **222**. Each page **224***a*–**224***d* represents a unique entry of computer data stored within the virtual shared memory **222**. Each page **224***a*–**224***d* is accessible to each one of the nodes **212***a*–**212***c*, 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 **212***a*–**212***c*. Accordingly, the virtual shared memory **222** provides a system for sharing and communicating data between each node **212** of the computer network **210**.

FIG. **6** illustrates in functional block diagram form a computer network **230** that has a distributed shared memory. In this embodiment, each node **212***a*–**212***c* has a memory subsystem **232** that connects between the operating system **216** and the two local memory devices, the RAM **234** and the disk **236**, and that further couples to a network **238** that couples to each of the depicted nodes **212***a*, **212***b* and **212***c* and to a network memory device **226**.

More particularly, FIG. **6** illustrates a distributed shared memory network **30** that includes a plurality of nodes **212***a*–**212***c*, each including a processing unit **214**, an operating system **216**, a memory subsystem **232**, a RAM **234**, and a disk **236**. FIG. **6** further depicts a computer network system **38** that connects between the nodes **212***a*–**212***c* and the network memory device **226**. The network **238** provides a network communication system across these elements.

The illustrated memory subsystems **232***a*–**232***c* that connect between the operating system **216***a*–**216***c*, the memory elements **234***a*–**234***c*, **236***a*–**236***c*, and the network **238**, encapsulate the local memories of each of the nodes to provide an abstraction of a shared virtual memory system that spans across each of the nodes **212***a*–**212***c* on the network **238**. The memory subsystems **232***a*–**232***c* can be software modules that act as distributors to map portions of the addressable memory space across the depicted memory devices. The memory subsystems further track the data stored in the local memory of each node **212** and further operate network connections with network **238** for transferring data between the nodes **212***a*–**212***c*. In this way, the memory subsystems **232***a*–**232***c* access and control each memory element on the network **238** to perform memory access operations that are transparent to the operating system **216**. Accordingly, the operating system **216** interfaces with the memory subsystem **232** as an interface to a global memory space that spans each node **212***a*–**212***c* on the network **238**.

FIG. **6** further depicts that the system **230** provides a distributed shared memory that includes persistent storage for portions of the distributed memory. In particular, the depicted embodiment includes a memory subsystem, such as subsystem **232***a*, that interfaces to a persistent memory device, depicted as the disk **236***a*. The subsystem **232***a* can operate the persistent memory device to provide persistent storage for portions of the distributed shared memory space. As illustrated, each persistent memory device **236** depicted in FIG. **6** has a portion of the addressable memory space mapped onto it. For example, device **236***a* has the portions of the addressable memory space, C_o, C_d, C_g, mapped onto it, and provides persistent storage for data signals stored in those ranges of addresses.

Accordingly, the subsystem **232***a* can provide integrated control of persistent storage 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.

In one optional embodiment, the nodes of the network are organized into a hierarchy of groups. In this embodiment, the memory subsystems **232***a*–**232***c* can include a hierarchy manager that provides hierarchical control for the distribution of data. This includes controlling the migration controller, and policy controller, which are discussed in detail below, 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.

FIG. **7** illustrates in more detail one shared memory subsystem **240** according to the invention. FIG. **7** depicts a shared memory subsystem **240**, that includes an interface **242**, a DSM directory manager **244**, a memory controller **246**, a local disk cache controller **248**, and a local RAM cache controller **250**. FIG. **7** further depicts the network **254**, an optional consumer of the DSM system, depicted as the service **258**, the operating system **216**, a disk driver **260**, a disk element **262** and a RAM element **264**.

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.

The depicted interface **242** provides an entry point for the local node to access the shared memory space of the computer network. The interface **242** can couple directly to the operating system **216**, to a distributed service utility such as the depicted DSM file system **258**, to a distributed user-level service utility, or alternatively to any combination thereof.

The depicted interface **242** provides an API that is a memory oriented API. Thus, the illustrated interface **242** can export a set of interfaces that provide low-level control of the distributed memory. As illustrated in FIG. **7**, the interface **242** exports the API to the operating system **216** or to the optional DSM service **258**. The operating system **216** or the service employs the interface **242** to request standard memory management techniques, such as reading and writing from portions of the memory space. These portions of the memory space can be the pages as described above which can be 4K byte portions of the shared memory space, or other units of memory, such as objects or segments. 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. The address generator can include a spanning module that generates address signals for a memory space that spans the storage capacity of the network.

Accordingly, in one embodiment, the interface **242** receives requests to manipulate pages of the shared memory space. To this end, the interface **242** can comprise a software module that includes a library of functions that can be called by services, the OS **216**, or other caller, or device. The function calls provide the OS **216** with an API of high level memory oriented services, such as read data, write data, and allocate memory. The implementation of the functions can include a set of calls to controls that operate the directory manager **244**, and the local memory controller **246**. Accordingly, the interface **242** can be a set of high level memory function calls to interface to the low-level functional elements of shared memory subsystem **240**.

FIG. **7** further depicts a DSM directory manager **244** that couples to the interface **242**. The interface **242** passes request signals that represent requests to implement memory operations such as allocating a portion of memory, locking a portion of memory, mapping a portion of memory, or some other such memory function. The directory manager **244** manages a directory that can include mappings than can span across each memory device connected to the network **238** depicted in FIG. **6**, including each RAM and disk element accessible by the network. The directory manager **244** stores a global directory structure that provides a map of the global address space. In one embodiment as will be explained in greater detail hereinafter, the directory manager **244** provides a global directory that maps between global address signals and responsible nodes on the network. A responsible node stores information regarding the location and attributes of data associated with a respective global address, and optionally stores a copy of that page's data. Consequently, the directory manager **244** tracks information for accessing any address location within the identifier space.

The control of the distributed shared memory can be coordinated by the directory manager **244** and the memory controller **246**. The directory manager **244** maintains a directory structure that can operate on a global address received from the interface **242** and identify, for that address, a node on the network that is responsible for maintaining the page associated with that address of the shared memory space. Once 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**. Accordingly, the depicted directory manager **244** is responsible for managing a directory structure that identifies for each page of the shared memory space a responsible node that tracks the physical location of the data stored in the respective page. Thus, the directory, rather than directly providing the location of the page, can optionally identify a responsible node, or other device, that tracks the location of the page. This indirection facilitates maintenance of the directory as pages migrate between nodes.

The memory controller **246** performs the low level memory access functions that physically store data within the memory elements connected to the network. In the depicted embodiment, the directory manager **244** of a first node can pass a memory access request through the interface **242**, to the network module of the OS **216**, and across the network **254** to a second node that the directory manager **244** identifies as the responsible node for the given address. The directory manager **244** can then query the responsible node to determine the attributes and the current owner node of the memory page that is associated with the respective global address. The owner of the respective page 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 node to access the data of the page that is physically stored on that owner node.

In particular, as depicted in FIG. **7**, the directory manager **244** couples to the network module **252** which couples to the network **254**. The directory manager can transmit to the network module **252** a command and associated data that directs the network interface **252** to pass a data signal to the owner node. 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.

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. The page of data is transferred via the network **254** to the network module **252** of the requesting node and the shared memory subsystem **240** operates the memory controller **246** to store in the local memory of the requesting node a copy of the accessed data.

Accordingly, in one embodiment of the invention, when a first node accesses a page of the shared memory space which is not stored locally on that node, the directory manager **244** identifies a node that has a copy of the data stored in that page and moves a copy of that data into the local memory of the requesting node. The local memory storage, both volatile and persistent, of the requesting node therefore becomes a cache for pages that have been requested by that local node. This embodiment is depicted FIG. **7** which depicts a memory controller that has a local disk cache controller **248** and a local RAM cache controller **250**. Both of these local cache controllers can provide to the operating system **216**, or other consumer pages of the shared memory space that are cache stored in the local memory of the node, including local persistent memory and local volatile memory.

The shared memory subsystem can include a coherent replication controller that maintains coherency between cached pages by employing a coherence through invalidation process, a coherence through migration process or other coherence process suitable for practice with the present invention. The coherent replication controller can automatically generate a copy of the data stored in each page and can store the copy in a memory device that is separate from the memory device of the original copy. This provides for fault tolerant operation, as the failure of any one memory device will not result in the loss of data. The coherent replication controller can be a software model that monitors all copies of pages kept in volatile memory and made available for writing. The controller can employ any of the coherency techniques named above, and can store tables of location information that identifies the location information for all generated copies.

FIG. **8** illustrates in greater detail one embodiment of a shared memory subsystem according to the invention. The shared memory subsystem **270** depicted in FIG. **8** includes a remote operations element **274**, a local RAM cache **276**, a RAM copyset **278**, a global RAM directory **280**, a disk copyset **282**, a global disk directory **284**, a configuration manager **288**, a policy element **290**, and a local disk cache **94**. FIG. **8** further depicts a network element **304**, a physical memory **300**, shared data element **302**, a physical file system **298**, which is part of the operating system **216**, a configuration service **308**, a diagnostic service **310**, and a memory access request **312**. The depicted subsystem **270** can be a computer program that couples to the physical memory, file system, and network system of the host node, or can be electrical circuit card assemblies that interface to the host node, or can be a combination of programs and circuit card assemblies.

The flow scheduler **272** depicted in FIG. **8** can orchestrate the controls provided by an API of the subsystem **270**. In one embodiment, the flow scheduler **272** can be 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. Alternatively, the flow scheduler **272** can be a program, such as an interpreter, that provides an execution environment and can map these op-codes into control flow programs called applets. The applets can be independent executable programs that employ both environment services, such as threading, synchronization, and buffer management, and the elements depicted in FIG. **8**. The API is capable of being called from both external clients, like a distributed shared memory file system, as well as recursively by the applets and the other elements **274–294** of the subsystem **270**. Each element can provide a level of encapsulation to the management of a particular resource or aspect of the system. To this end, each element can export an API consisting of functions to be employed by the applets. This structure is illustrated in FIG. **8**. Accordingly, the flow scheduler **272** can provide an environment to load and execute applets. The applets are dispatched by the flow scheduler **272** on a per op-code basis and can perform the control flow for sequential or parallel execution of an element to implement the op-code on the specified global address, such as a read or write operation. Optionally, the flow scheduler **272** can include an element to change dynamically the applet at run time as well as execute applets in parallel and in interpreted mode.

The depicted shared memory subsystem **270** includes a bifurcated directory manager that includes the global RAM directory **280** and the global disk directory **284**. The global RAM directory **280** is a directory manager that tracks information that can provide the location of pages that are stored in the volatile memory, typically RAM, of the network nodes. The global disk directory **284** is a global disk directory manager that manages a directory structure that tracks information that can provide the location of pages that are stored on persistent memory devices. Together, the global RAM directory **280** and the global disk directory **284** provide the shared memory subsystem **270** with integrated directory management for pages that are stored in persistent storage and volatile memory.

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. Accordingly, the RAM directory manager passes control of that page to the disk directory manager which can then treat the page as any other page of data. This allows for improved load balancing, by removing data from RAM memory, and storing it in the disk devices, under the control of the disk directory manager.

The local memory controller of the subsystem **270** is provided by the local RAM cache **276** and the local disk cache **294**. The local RAM cache **276** which couples to the physical memory **300** of the local node can access, as described above, the virtual memory space of the local node to access data that is physically stored within the RAM memory **300**. Similarly, the local disk cache **294** couples to the persistent storage device **298** and can access a physical location that maintains in the local persistent storage data of the distributed shared memory.

FIG. **8** also depicts a remote operations element **274** that couples between the network **304** and the flow scheduler **272**. The remote operations element **274** negotiates the transfer of data across the network **304** for moving portions of the data stored in the shared memory space between the nodes of the network. The remote operations element **274** can also request services from remote peers, i.e. invalidate to help maintain coherency or for other reasons.

FIG. **8** also depicts 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. 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.

FIG. **8** further depicts a memory subsystem **270** that includes a RAM copyset **278** and a disk copyset **282**. These copysets can manage copies of pages that are cached at a single node. The disk copyset **282** can maintain information on copies of pages that are stored in the local disk cache, which can be the local persistent memory. Similarly, the RAM copyset **278** can maintain information on copies of pages that are stored in the local RAM cache which can be the local RAM. These copysets encapsulate indexing and storage of copyset data that can be employed by applets or other executing code for purposes of maintaining the coherency of data stored in the shared memory space. The copyset elements can maintain copyset data that identifies the pages cached by the host node. Further, the copyset can identify the other nodes on the network that maintain a copy of that page, and can further identify for each page which of these nodes is the owner node, wherein the owner node can be a node which has write privileges to the page being accessed. The copysets themselves can be stored in pages of the distributed shared memory space.

The local RAM cache **276** provides storage for memory pages and their attributes. In one embodiment, 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. In this embodiment, the local ram cache **276** provides the index by storing in memory a list of each global address cached in the local RAM. With each listed global address, the index provides a pointer into a buffer memory and to the location of the page data. Optionally, with each listed global address, the index can further provide attribute information including a version tag 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.

In the embodiment depicted in FIG. **8**, the memory subsystem **270** provides the node access to the distributed memory space by the coordinated operation of the directory manager that includes the global RAM directory **280** and the global disk directory **284**, the cache controller that includes the local RAM cache and the local disk cache elements **276** and **294**, and the copyset elements which include the RAM copyset **278** and the disk copyset **282**.

The directory manager provides a directory structure that indexes the shared address space. Continuing with the example of a paged shared address space, the directory manager of the subsystem **270** allows the host node to access, by global addresses, pages of the shared memory space.

FIGS. **9** and **10** illustrate one example of a directory structure that provides access to the shared memory space. FIG. **9** depicts a directory page **320** that includes a page header **322**, directory entries **324** and **326**, wherein each directory entry includes a range field **330**, a responsible node field **332**, and an address field **334**. The directory pages can be generated by a directory page generator that can be a software module controlled by the directory manager. It will be understood that the directory manager can generate multiple directories, including one for the Global disk and one for the Global RAM directories. The depicted directory page **320** can be a page of the global address space, such as a 4K byte portion of the shared address space. Therefore, the directory page can be stored in the distributed shared memory space just as the other pages to which the directory pages provide access.

As further depicted in FIG. **9**, each directory page **120** includes a page header **322** that includes attribute information for that page header, which is typically metadata for the directory page, and further includes directory entries such as the depicted directory entries, **324** and **326**, which provide an index into a portion of the shared address space wherein that portion can be one or more pages, including all the pages of the distributed shared memory space. The depicted directory page **320** includes directory entries that index a selected range of global addresses of the shared memory space. To this end, the directory generator can include a range generator so that each directory entry can include a range field **330** that describes the start of a range of addresses that that entry locates.

Accordingly, each directory page **320** can include a plurality of directory entries, such as entries **324** and **326**, that can subdivide the address space into a subset of address ranges. For example, the depicted directory page **320** includes two directory entries **324** and **326**. The directory entries **324** and **326** can, for example, subdivide the address space into two sub-portions. In this example, the start address range of the directory entry **324** could be the base address of the address space, and the start address range of the directory entry **326** could be the address for the upper half of the memory space. Accordingly, the directory entry **324** provides an index for pages stored in the address space between the base address and up to the mid-point of the memory space and, in complement thereto, the directory entry **326** provides an index to pages stored in the address space that ranges from the mid-point of the address space to the highest address.

FIG. **9** further depicts a directory page **320** that includes, in each directory entry, a responsible node field **332** and the child page global address field **334**. These fields **332**, **334** provide further location information for the data stored in pages within the address range identified in field **330**.

FIG. **10** depicts a directory **340** formed from directory pages similar to those depicted in FIG. **9**. FIG. **10** depicts that the directory **340** includes directory pages **342**, **350–354**, and **360–366**. FIG. **10** further depicts that the directory **340** provides location information to the pages of the distributed shared memory space depicted in FIG. **10** as pages **370–384**.

The directory page **342** depicted in FIG. **10** acts like a root directory page and can be located at a static address that is known to each node coupled to the distributed address space. The root directory page **342** includes three directory entries **344**, **346**, and **348**. Each directory entry depicted in FIG. **10** has directory entries similar to those depicted in FIG. **9**. For example, directory entry **344** includes a variable Co which represents the address range field **330**, a variable Nj representative of the field **332**, and a variable Cs representative of the field **334**. The depicted root directory page **342** subdivides the address space into three ranges illustrated as an address range that extends between the address Co and Cd, a second address range that extends between the address Cd and Cg, and a third address range that extends between Cg and the highest memory location of the address space.

As further depicted in FIG. **10**, each directory entry **344**, **346**, and **348** points to a subordinate directory page, depicted as directory pages **350**, **352**, and **354**, each of which further subdivides the address range index by the associated directory entry of the root directory **342**. In FIG. **9**, this subdivision process continues as each of the directory pages **350**, **352**, and **354** each again have directory entries that locate subordinate directory pages including the depicted examples of directory pages **360**, **362**, **364**, and **366**.

The depicted example of directory pages **360**, **362**, **364**, and **366** are each leaf entries. The leaf entries contain directory entries such as the directory entries **356** and **358** of the leaf entry **360**, that store a range field **330** and the responsible node field **332**. These leaf entries identify an address and a responsible node for the page in the distributed memory space that is being accessed, such as the depicted pages **370–384**. For example, as depicted in FIG. **10**, the leaf entry **356** points to the page **370** that corresponds to the range field **330** of the leaf entry **356**, which for a leaf entry is the page being accessed. In this way, the directory structure **340** provides location information for pages stored in the distributed address space.

In the depicted embodiment of FIG. **10**, a node selector can select a responsible node for each page, as described above, so that the leaf entry **356** provides information of the address and responsible node of the page being located. Accordingly, this directory tracks ownership and responsibility for data, to provide a level of indirection between the directory and the physical location of the data. During 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. The shared memory subsystem of that node can identify a node that stores a copy of the page being accessed, including the owner node. This identification of a node having a copy can be performed by the RAM copyset or disk copyset of the responsible node. The node having a copy stored in its local physical memory, such as the owner node, can employ its local cache elements, including the local RAM cache and local disk cache to the identify from the global address signal a physical location of the data stored in the page being accessed. The cache element can employ the operating system of the owner node to access the memory device that maintains that physical location in order that the data stored in the page can be accessed. For a read-memory operation, or for other similar operations, the data read from the physical memory of the owner node can be passed via the network to the memory subsystem of the node requesting the read and subsequently stored into the virtual memory space of the requesting node for use by that node.

With reference again to FIG. **10**, it can be seen that the depicted directory structure **340** comprises a hierarchical structure. To this end, the directory structure **340** provides a structure that continually subdivides the memory space into smaller and smaller sections. Further, each section is represented by directory pages of the same structure, but indexes address spaces of different sizes. As pages are created or deleted, a linker inserts or deletes the pages from the directory. In one embodiment, the linker is a software module for linking data structures. The linker can operate responsive to the address ranges to provide the depicted hierarchical structure. Accordingly, the depicted directory **340** provides a scaleable directory for the shared address space. Moreover, the directory pages are stored in the distributed address space and maintained by the distributed shared memory system. A root for the directory can be stored in known locations to allow for bootstrap of the system. Consequently, 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.

FIG. **11** depicts the directory of FIG. **10** being employed by a system according to the invention. In particular FIG. **11** depicts a system **400** that includes two nodes, **406*a*** and **406*b***, a directory structure **340**, and a pair of local memories having volatile memory devices **264*a*** and **264*b***, and persistent memory devices **262*a*** and **262*b***. Depicted node **406*a*** includes an address consumer **408*a***, a global address **410*a***, and interface **242*a***, a directory manager **244*a*** and a memory controller **246*a***. Node **406*b*** has corresponding elements. The nodes are connected by the network **254**. The directory **340** has a root page, directory pages A–F, and pages **1–5**.

Each node **406*a*** and **406*b*** operates as discussed above. The depicted address consumers **408*a*** and **408*b*** can be an application program, file system, hardware device or any other such element that requests access to the virtual memory. In operation, the address consumers **408*a*** and **408*b*** 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. As addresses get generated, the respective directory managers **244*a*** and **244*b*** generate directory pages and store the pages in the directory structure **340**. As depicted, the directory structure **340** tracks the portions of the address space being employed by the system **400**, and physical storage for each page is provided within the local memories.

As shown in FIG. **11**, the data associated with the directory pages are distributively stored across the two local memories and duplicate copies can exist. As described above and now illustrated in FIG. **11**, the data can move between different local memories and also move, or page, between volatile and persistent storage. The data movement can be responsive to data requests made by memory users like application programs, or by operation of the migration controller described above. As also described above, the movement of data between different memory locations can occur without requiring changes to the directory **340**. This is achieved by providing a directory **340** that is decoupled from the physical location of the data by employing a pointer to a responsible node that tracks the data storage location. Accordingly, although the data storage location can change, the responsible node can remain constant, thereby avoiding any need to change the directory **340**.

Variations, modifications, and other implementations of what is described herein will occur to those of ordinary skill in the art without departing from the spirit and the scope of the invention as claimed. Accordingly, the invention is to be defined not by the preceding illustrative description but instead by the spirit and scope of the following claims.

What is claimed is:

1. A method for providing distributed control over a structured store of data, comprising:

providing a plurality of nodes inter-connected by a network, each of the plurality of nodes sharing a shared addressable memory space of a shared memory system and including (i) an interface for accessing the network, (ii) a local volatile memory device coupled to the node and providing volatile storage, (iii) a local persistent memory device coupled to the node and providing persistent storage, and (iv) a shared memory subsystem for mapping a portion of the shared addressable memory space to at least a portion of the persistent and volatile storage to provide thereby addressable persistent and volatile storage accessible by each of the plurality of nodes, the shared memory subsystem including (a) a distributor for mapping portions of the addressable memory space across the plurality of local persistent and volatile memory devices to distribute the addressable memory space across the plurality of local persistent and volatile memory devices, and (b) a disk directory manager for tracking the mapped portions of the addressable memory space to provide information representative of which of the local persistent and volatile memory devices has which of the portions of the addressable memory space mapped thereon;

storing on each node an instance of a data control program for manipulating the structured store of data to provide multiple, distributed instances of the data control program;

interfacing each instance of the data control program to the shared memory system; and

operating each instance of the data control program to employ the shared memory system as a memory device having the structured store of data contained therein, whereby the shared memory system coordinates access to the structured store of data to provide distributed control over the structured store of data.

2. The method of claim **1** wherein said interfacing step further includes:

directing the data control program to provide a stream of data to be stored in the structured store of data; and

directing the data control program to operate the shared memory system as a single-node memory device.

3. The method of claim **1** wherein the structured store of data comprises a file system, and wherein the data control program comprises a file control program for manipulating the file system, whereby the shared memory system controls access to the file system to provide a shared file system.

4. The method of claim **3** further comprising:

providing the shared file system with a file directory; and

operating the shared memory system to maintain the file directory within a shared memory space.

5. The method of claim **4** further comprising:

organizing the file directory as a plurality of logical file partitions stored within the shared memory space.

6. The method of claim **4** further comprising the step of coordinating shared access to data within the structured store by locking directories stored within a shared memory space.

7. The method of claim **3** further comprising:

generating, for a file stored within the shared file system, a file descriptor having storage for an identifier being representative of a portion of a shared memory space.

8. The method of claim **7** further comprising:

allocating contiguous portions of the shared memory space, each represented by a respective identifier, to provide reduced bookkeeping information for the file.

9. The method of claim **7** further comprising:

reserving contiguous segments of a storage device for storing data associated with the contiguous portions of the shared memory space for optimizing access to physical storage for the file.

10. The method of claim **1** wherein the structured store of data comprises a database system, and wherein the data control program comprises a database control program for manipulating the database system, whereby the shared memory system controls access to the database system to provide a shared database system.

11. The method of claim **10** further comprising:

providing the shared database system with a database directory and set of index structures; and

operating the shared memory system to maintain the database directory and set of index structures within a shared memory space.

12. The method of claim **11** further comprising:

organizing the database directory as a plurality of sets stored within the shared memory space.

13. The method of claim **10** further comprising the steps of:

associating concurrency control structures with portions of the database system;

storing the concurrency control structures in the shared memory space; and

coordinating shared access to the database system by locking concurrency control structures.

14. The method of claim **13** further comprising locking of database indices.

15. The method of claim **13** further comprising locking of database keys.

16. The method of claim **10** further comprising:

generating, for a database object stored within the shared database system, a database record descriptor having storage for an identifier being representative of a portion of a shared memory space.

17. The method of claim **16** further comprising:

allocating contiguous portions of the shared memory space, each represented by a respective identifier, to provide reduced bookkeeping information for the respective database record.

18. The method of claim **16** further comprising:

reserving contiguous segments of a storage device for storing data associated with the contiguous portions of the shared memory space for optimizing access to physical storage for the database record.

19. The method of claim **1** wherein the structured store of data comprises a Web server system, and wherein the data control program comprises a control program for manipulating the Web server system, and controlling access to the Web server system to provide a shared Web server system.

20. The method of claim **19** further comprising:

providing the shared Web server system with a directory mapping the files to their contents; and

operating the shared memory system to maintain the Web server directory within a shared memory space.

21. The method of claim **19** further comprising:

generating, for a file stored within the shared Web server system, a file descriptor having storage for an identifier being representative of a portion of a shared memory space.

22. The method of claim **21** further comprising:

allocating contiguous portions of the shared memory space, each represented by a respective identifier, to provide reduced bookkeeping information for the files.

23. The method of claim **21** further comprising:

reserving contiguous segments of a storage device for storing data associated with the contiguous portions of the shared memory space for optimizing access to physical storage for the files.

24. The method of claim **1** further comprising:

operating the shared memory system to replicate stored data coherently in order to provide a redundant store of data.

25. The method of claim **24** further comprising:

storing the coherently replicated data within different storage devices of the network to provide fault tolerant operation.

26. The method of claim **1** further comprising:

associating concurrency control structures with portions of the shared memory space;

storing the concurrency control structures in the shared memory space; and

coordinating shared access to data within the structured store by locking concurrency control structures.

27. The method of claim **26** further comprising:

generating a lock object data structure having information representative of a lock status on portions of the shared memory space; and

storing the lock object within the shared memory space to provide thereby a shared system lock.

28. The method of claim **26** wherein the locking step includes:

directing the shared memory to generate byte range locks representative of locks placed on portions of the shared memory space.

29. The method of claim **1** further comprising operating each instance of the data control program to employ the shared memory system as clustered structured storage, the memory system coordinating access to the clustered structured storage to provide distributed control over the clustered structured storage.

30. A method for providing distributed control over a structured store of data, comprising:

providing a plurality of nodes inter-connected by a network;

storing on each node an instance of a data control program for manipulating the structured store of data to provide multiple, distributed instances of the data control program, the structured store of data comprising a Web server system and the data control program comprising a control program for manipulating the Web server system and controlling access to the Web server system to provide a shared Web server system;

interfacing each instance of the data control program to a shared memory system that provides addressable persistent storage of data;

operating each instance of the data control program to employ the shared memory system as a memory device having the structured store of data contained therein, whereby the shared memory system coordinates access to the structured store of data to provide distributed control over the structured store of data;

providing the shared Web server system with a directory mapping files to their contents; and

operating the shared memory system to maintain the Web server directory within a shared memory space.

31. A method for providing distributed control over a structured store of data, comprising:

providing a plurality of nodes inter-connected by a network;

storing on each node an instance of a data control program for manipulating the structured store of data to provide multiple, distributed instances of the data control program, the structured store of data comprising a Web server system and the data control program comprising a control program for manipulating the Web server system and controlling access to the Web server system to provide a shared Web server system;

interfacing each instance of the data control program to a shared memory system that provides addressable persistent storage of data;

operating each instance of the data control program to employ the shared memory system as a memory device having the structured store of data contained therein, whereby the shared memory system coordinates access to the structured store of data to provide distributed control over the structured store of data; and

generating, for a file stored within the shared Web server system, a file descriptor having storage for an identifier being representative of a portion of a shared memory space.

32. The method of claim **31** further comprising:

allocating contiguous portions of the shared memory space, each represented by a respective identifier, to provide reduced bookkeeping information for the files.

33. The method of claim **31** further comprising:

reserving contiguous segments of a storage device for storing data associated with the contiguous portions of the shared memory space for optimizing access to physical storage for the files.

34. A method for providing distributed control over a structured store of data, comprising:

providing a plurality of nodes inter-connected by a network;

storing on each node an instance of a data control program for manipulating the structured store of data to provide multiple, distributed instances of the data control program, the structured store of data comprising a Web server system and the data control program comprising a control program for manipulating the Web server system and controlling access to the Web server system to provide a shared Web server system;

interfacing each instance of the data control program to a globally addressable unstructured storage system;

operating each instance of the data control program to employ the globally addressable unstructured storage system as a memory device containing structured storage to provide distributed control over the structured store of data;

providing the shared Web server system with a directory mapping files to their contents; and

operating the globally addressable unstructured storage system to maintain the Web server directory.

35. A method for providing distributed control over a structured store of data, comprising:

providing a plurality of nodes inter-connected by a network;

storing on each node an instance of a data control program for manipulating the structured store of data to provide multiple, distributed instances of the data control program, the structured store of data comprising a Web server system and the data control program comprising a control program for manipulating the Web server system and controlling access to the Web server system to provide a shared Web server system;

interfacing each instance of the data control program to a globally addressable unstructured storage system;

operating each instance of the data control program to employ the globally addressable unstructured storage system as a memory device containing structured storage to provide distributed control over the structured store of data; and

generating, for a file stored within the shared Web server system, a file descriptor having storage for an identifier being representative of a portion of the globally addressable unstructured storage system.

36. The method of claim **35** further comprising:

allocating contiguous portions of the globally addressable unstructured storage system, each represented by a respective identifier, to provide reduced bookkeeping information for the files.

37. The method of claim **35** further comprising:

reserving contiguous segments of a storage device for storing data associated with the contiguous portions of the globally addressable unstructured storage system for optimizing access to physical storage for the files.

* * * * *

US006148377A

United States Patent [19]

Carter et al.

[11] **Patent Number: 6,148,377**

[45] **Date of Patent: *Nov. 14, 2000**

[54] **SHARED MEMORY COMPUTER NETWORKS**

[75] Inventors: **John B. Carter**, Salt Lake City, Utah; **Scott H. Davis**, Groton, Mass.; **William Abraham**, Windham, N.H.; **Steven J. Frank**, Hopkinton, Mass.; **Thomas G. Hansen**, Leominster, Mass.; **Robert S. Phillips**, Brookfield, Mass.; **Daniel J. Dietterich**, Acton, Mass.; **David Porter**, Littleton, Mass.

[73] Assignee: **Mangosōft Corporation**, Westborough, Mass.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/754,481**

[22] Filed: **Nov. 22, 1996**

[51] **Int. Cl.**[7] **G06F 12/08**

[52] **U.S. Cl.** **711/147**; 711/100; 711/201; 711/202; 711/203

[58] **Field of Search** 711/203, 202, 711/201, 147, 100

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,096,567	6/1978	Millard et al.	364/200
4,426,688	1/1984	Moxley	365/200
4,710,926	12/1987	Brown et al.	371/9
4,868,738	9/1989	Kish et al.	364/200
4,934,764	6/1990	Leitermann et al. .	
5,055,999	10/1991	Frank et al. .	
5,077,736	12/1991	Dunphy, Jr. et al.	371/10.1
5,117,350	5/1992	Parrish et al.	395/428
5,119,481	6/1992	Frank et al. .	
5,226,039	7/1993	Frank et al. .	
5,245,563	9/1993	Hauck, Jr. .	
5,247,673	9/1993	Costa et al.	395/650
5,251,308	10/1993	Frank et al. .	
5,282,201	1/1994	Frank et al. .	
5,297,265	3/1994	Frank et al. .	
5,313,626	5/1994	Jones et al.	395/575
5,313,647	5/1994	Kaufman et al. .	
5,335,325	8/1994	Frank et al. .	
5,341,483	8/1994	Frank et al. .	
5,390,326	2/1995	Shah	395/575
5,423,037	6/1995	Hvasshovd	395/600
5,452,447	9/1995	Nelson et al. .	
5,493,728	2/1996	Solton et al. .	
5,513,314	4/1996	Kandasamy et al. .	
5,560,027	9/1996	Watson et al.	395/800
5,592,625	1/1997	Sandberg	395/200.08
5,906,658	5/1994	Raz	710/39

FOREIGN PATENT DOCUMENTS

0 288 636 A2	11/1988	European Pat. Off. .
0 661 651 A1	7/1995	European Pat. Off. .
2 257 273	1/1993	United Kingdom .
WO 95/02307	1/1995	WIPO .
WO 95/14279	5/1995	WIPO .
WO 95/22111	8/1995	WIPO .
WO 95/25306	9/1995	WIPO .
WO 96/23268	8/1996	WIPO .
WO 96/30847	10/1996	WIPO .

OTHER PUBLICATIONS

European Patent Office, European Search Report, Application No. EP 97 30 9471, mailed on Apr. 23, 1998, 3 pages.

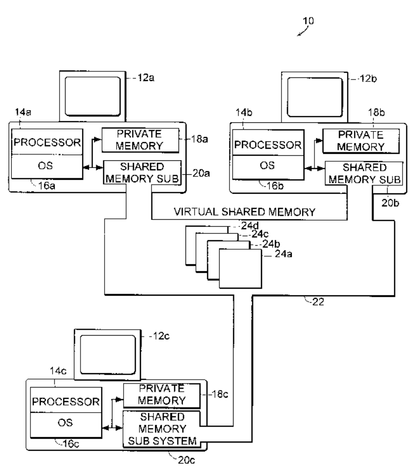
(List continued on next page.)

Primary Examiner—Tod R. Swann
Assistant Examiner—David Bryan Ransom
Attorney, Agent, or Firm—Proskauer Rose LLP

[57] **ABSTRACT**

Distributed shared memory systems and processes that can connect into each node of a computer network to encapsulate the memory management operations of the connected nodes and to provide thereby an abstraction of a shared virtual memory that can span across each node of the network and that optionally spans across each memory device connected to the computer network. Accordingly, each node on the network having the distributed shared memory system of the invention can access the shared memory.

23 Claims, 7 Drawing Sheets



OTHER PUBLICATIONS

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US97/21457, mailed on Apr. 23, 1998, 4 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US97/21458, mailed on Apr. 23, 1998, 3 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US97/21459, mailed on Apr. 23, 1998, 4 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US97/21460, mailed on Apr. 23, 1998, 5 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US97/21466, mailed on Apr. 23, 1998, 3 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US97/21733, mailed on Apr. 23, 1998, 3 pages.

Patent Cooperation Treaty, International Search Report, International Application No. PCT/US97/21734, mailed on Apr. 23, 1998, 4 pages.

Anderson, T., et al. "Serverless Network File Systems," *Operating Systems Review* (SIGOPS), vol. 29, No. 5, Dec. 1, 1995, pp. 109–126.

Carter, J., et al. "Network Multicomputing Using Recoverable Distributed Shared Memory," *Proceedings of the Spring Computer Society International Conference* (COMPCON), San Francisco, Feb. 22–26, 1993, pp. 519–527.

Huber, J., et al. "PPFS: A High Performance Portable Parallel File System," *Proceedings of the International Conference on Supercomputing,* Barcelona, Jul. 3–7, 1995, pp. 385–394.

"Java Dynamic Class Loader," *IBM Technical Disclosure Bulletin,* vol. 39, No. 11, Nov. 1996, pp. 107–108.

Lee, E., et al. "Petal: Distributed Virtual Disks," *7th International Conference on Architectural Support for Programming Languages and Operation Systems,* Cambridge, MA, Oct. 1–5, 1996, pp. 84–92.

Neal, D. "The Harvest Object Cache in New Zealand," *Computer Networks and ISDN Systems,* vol. 11, No. 28, May 1996, pp. 1415–1430.

Raghaven, G. "A Domain Model of WWW Browsers," *Proceedings of Southeastcon, Bringing Together Education, Science and Technology,* Tampa, Apr. 11–14, 1996, pp. 436–439.

Ng, T. "The Design and Implementation of a Reliable Distributed Operating System –ROSE," *Proceedings of the Symposium on Reliable Distributed Systems,* Huntsville, Oct. 9–11, 1990, pp. 2–11.

Yeung, D., et al. "MGS" A Multigran Shared Memory System, *Computer Architecture News,* vol. 24, No. 2, May 1, 1996, pp. 44–55.

Wilson, Andrew W., "Organization and Statistical Stimulation of Hierarchial Multiprocessors," UMI Dissertation Information Service (1985).

Li, Kai, "Shared Virtual Memory on Loosely Coupled Multiprocessors," Yale University , Department of Computer Science, (Sep. 1986).

Wilson, Andrew W., "Hierachical Cache/Bus Architecture for Shared Memory Multiprocessors," *ACM,* pp. 244–252 (1987).

Carter, J.B. et al., "Optimistic Implementation of Bulk Data Transfer Protocols," *In Proceedings of the 1989 Signetrics Conference,* pp. 61–69 (May, 1989).

Carter, J.B. et al., "Distributed Operating Systems Based on a Protected Global Virtual Address Space," In the Proceedings of the Third Workshop on Workstation Operating Systems (WWOS) (May, 1992).

Carter, J.B. et al., "FLEX: A Tool for Building Efficient and Flexible Systems," In the Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS) (Oct., 1993).

Carter, John B., "Design of the Munin Distributed Shared memory System," To appear in the special issue of the *Journal of Parallel and Distributed Computing* on distributed shared memory (1995).

Carter, J.B. et als., "Techniques for Reducing Consistency–Related Communication in Distributed Shared Memory Systems," *ACM Transactions on Computer Systems,* pp. 205–243, vol. 13, No. 3 (Aug., 1995).

Carter, J.B. et als., "Reducing Consistency Traffic and Cache Misses in the Avalanche Multiprocessor," University of Utah technical report.

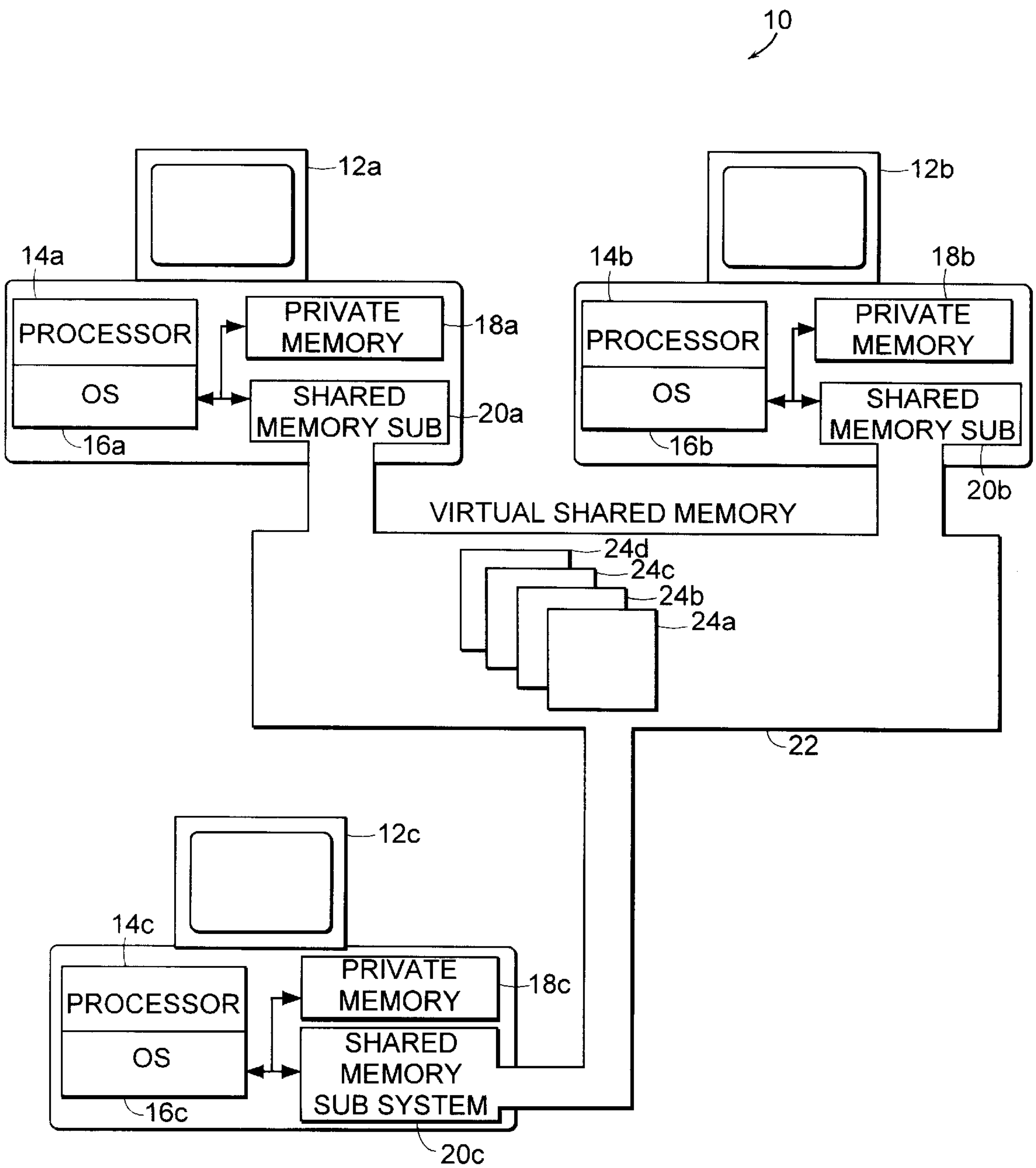

10
12a
14a
PROCESSOR
OS
PRIVATE MEMORY
18a
SHARED MEMORY SUB
20a
16a
12b
14b
18b
PROCESSOR
OS
PRIVATE MEMORY
SHARED MEMORY SUB
16b
20b
VIRTUAL SHARED MEMORY
24d
24c
24b
24a
22
12c
14c
PROCESSOR
OS
PRIVATE MEMORY
18c
SHARED MEMORY SUB SYSTEM
16c
20c

FIG. 1

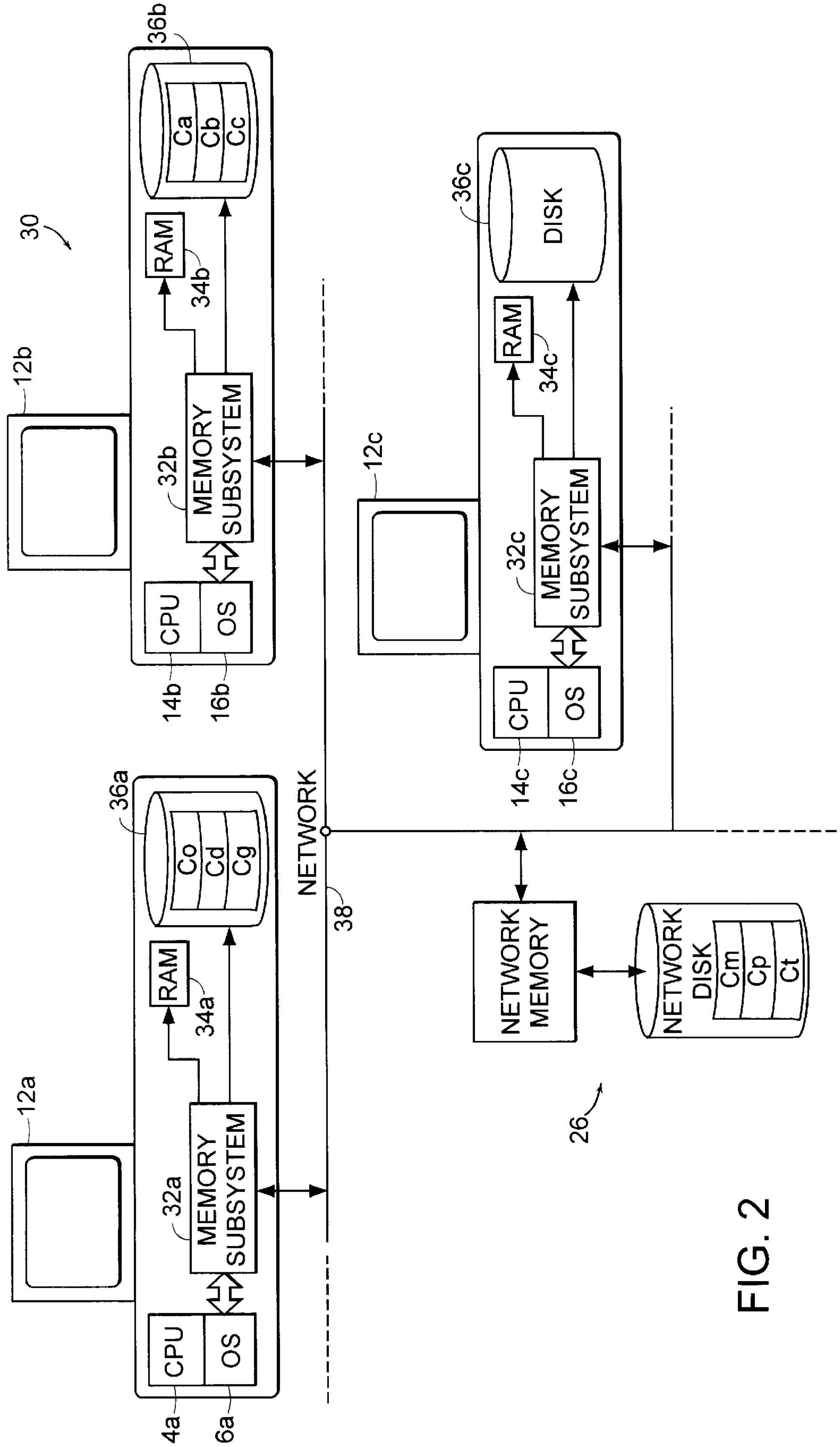

30
12a
12b
12c
CPU
OS
MEMORY SUBSYSTEM
RAM
DISK
NETWORK
NETWORK MEMORY
NETWORK DISK
Co
Cd
Cg
Ca
Cb
Cc
Cm
Cp
Ct
14a
16a
32a
34a
36a
14b
16b
32b
34b
36b
14c
16c
32c
34c
36c
38
26

FIG. 2

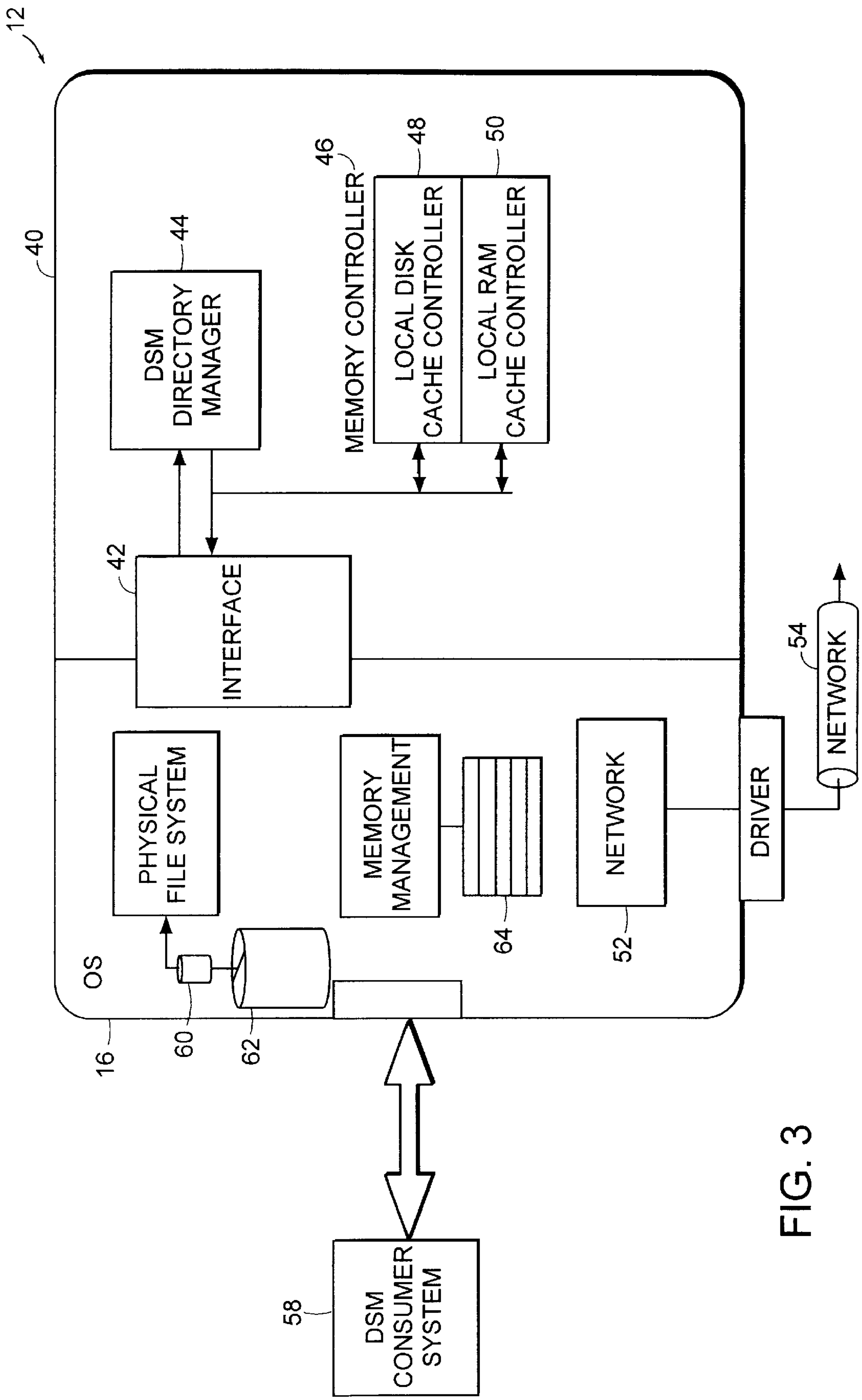
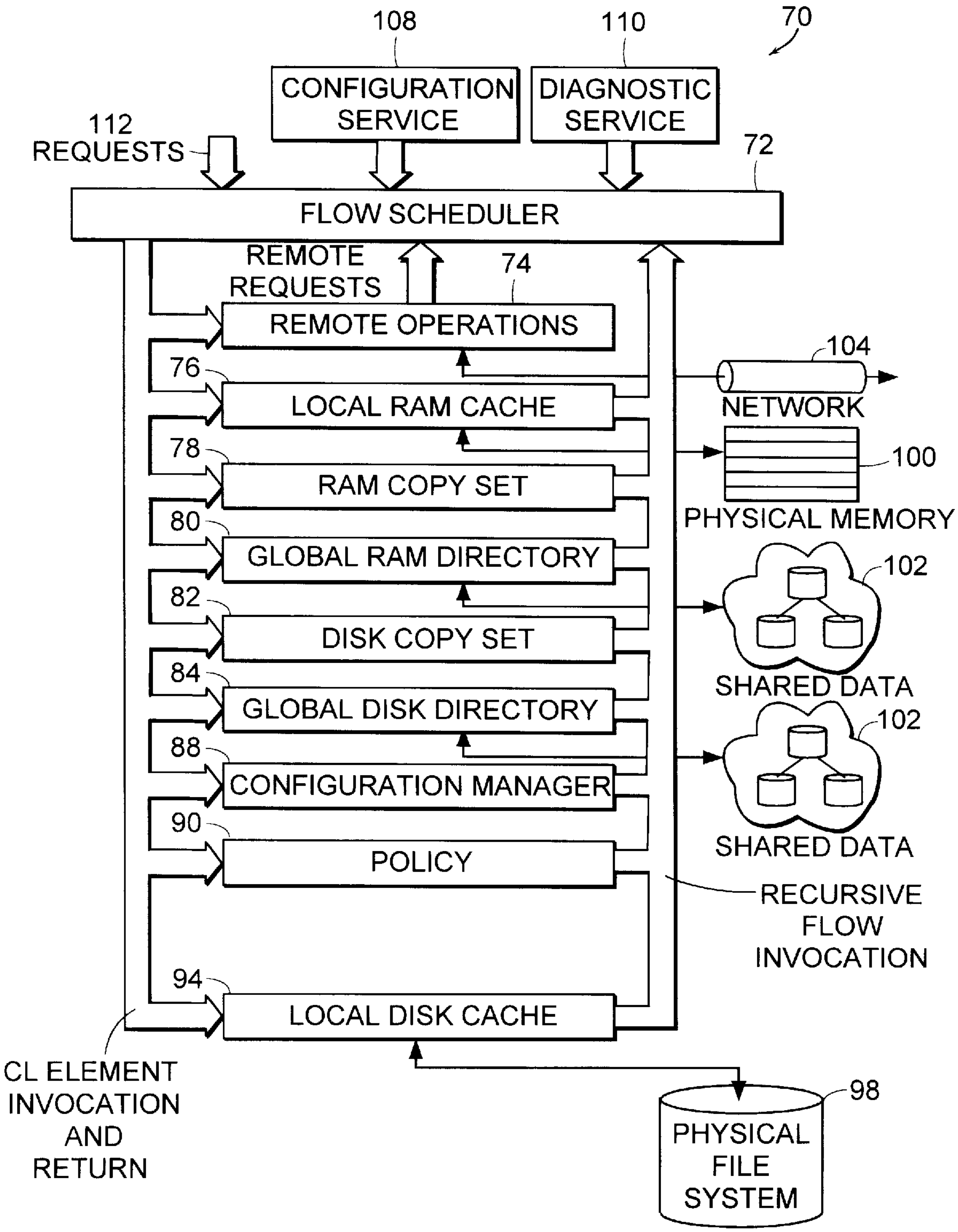

12
40
44
DSM DIRECTORY MANAGER
46
MEMORY CONTROLLER
48
LOCAL DISK CACHE CONTROLLER
50
LOCAL RAM CACHE CONTROLLER
42
INTERFACE
PHYSICAL FILE SYSTEM
MEMORY MANAGEMENT
NETWORK
DRIVER
54
NETWORK
OS
64
52
16
60
62
58
DSM CONSUMER SYSTEM

FIG. 3



108
110
70
CONFIGURATION SERVICE
DIAGNOSTIC SERVICE
112
REQUESTS
72
FLOW SCHEDULER
REMOTE REQUESTS
74
REMOTE OPERATIONS
104
76
LOCAL RAM CACHE
NETWORK
78
RAM COPY SET
100
PHYSICAL MEMORY
80
GLOBAL RAM DIRECTORY
102
82
DISK COPY SET
84
GLOBAL DISK DIRECTORY
SHARED DATA
102
88
CONFIGURATION MANAGER
90
POLICY
SHARED DATA
RECURSIVE FLOW INVOCATION
94
LOCAL DISK CACHE
CL ELEMENT INVOCATION AND RETURN
98
PHYSICAL FILE SYSTEM

FIG. 4

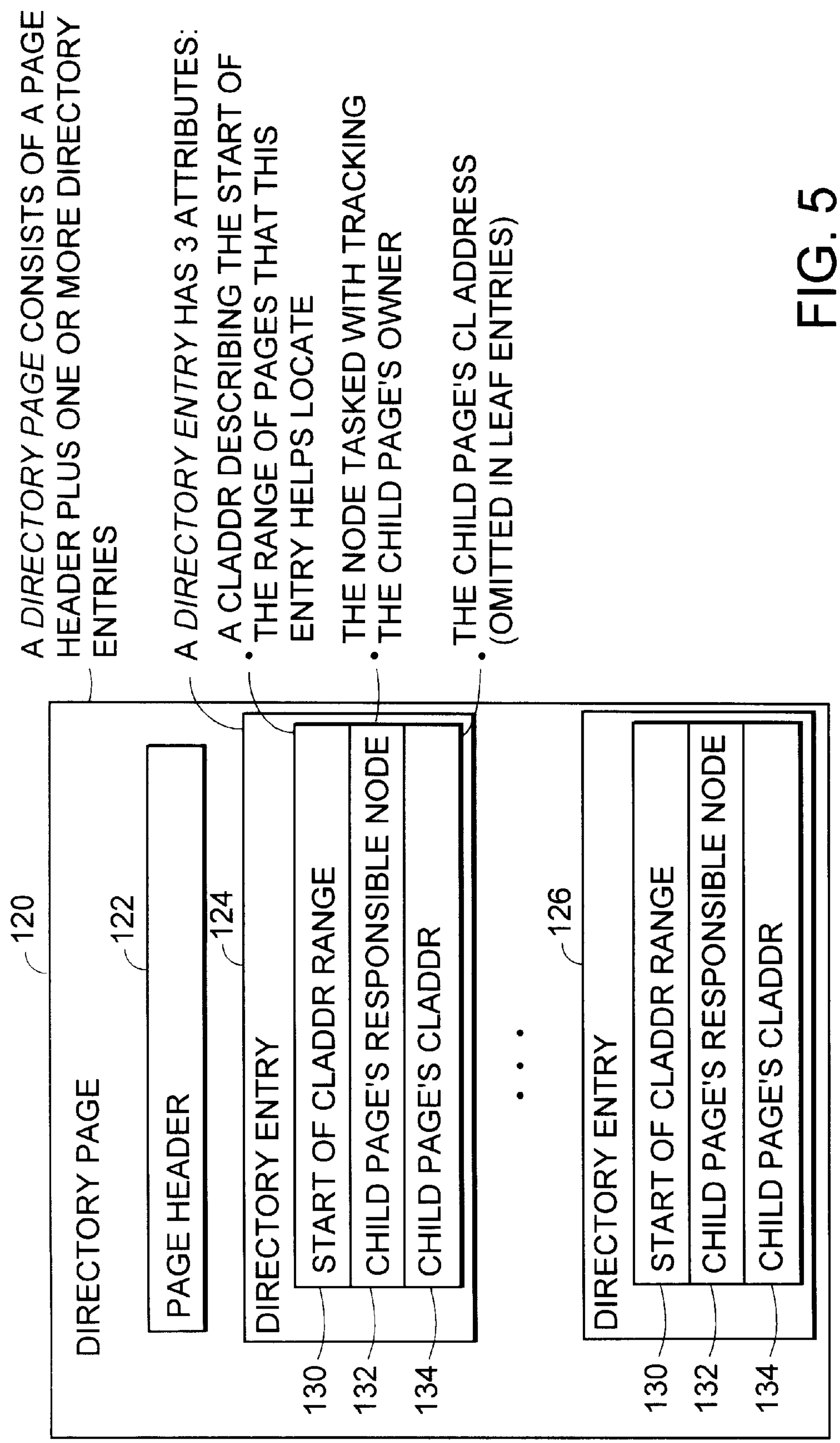
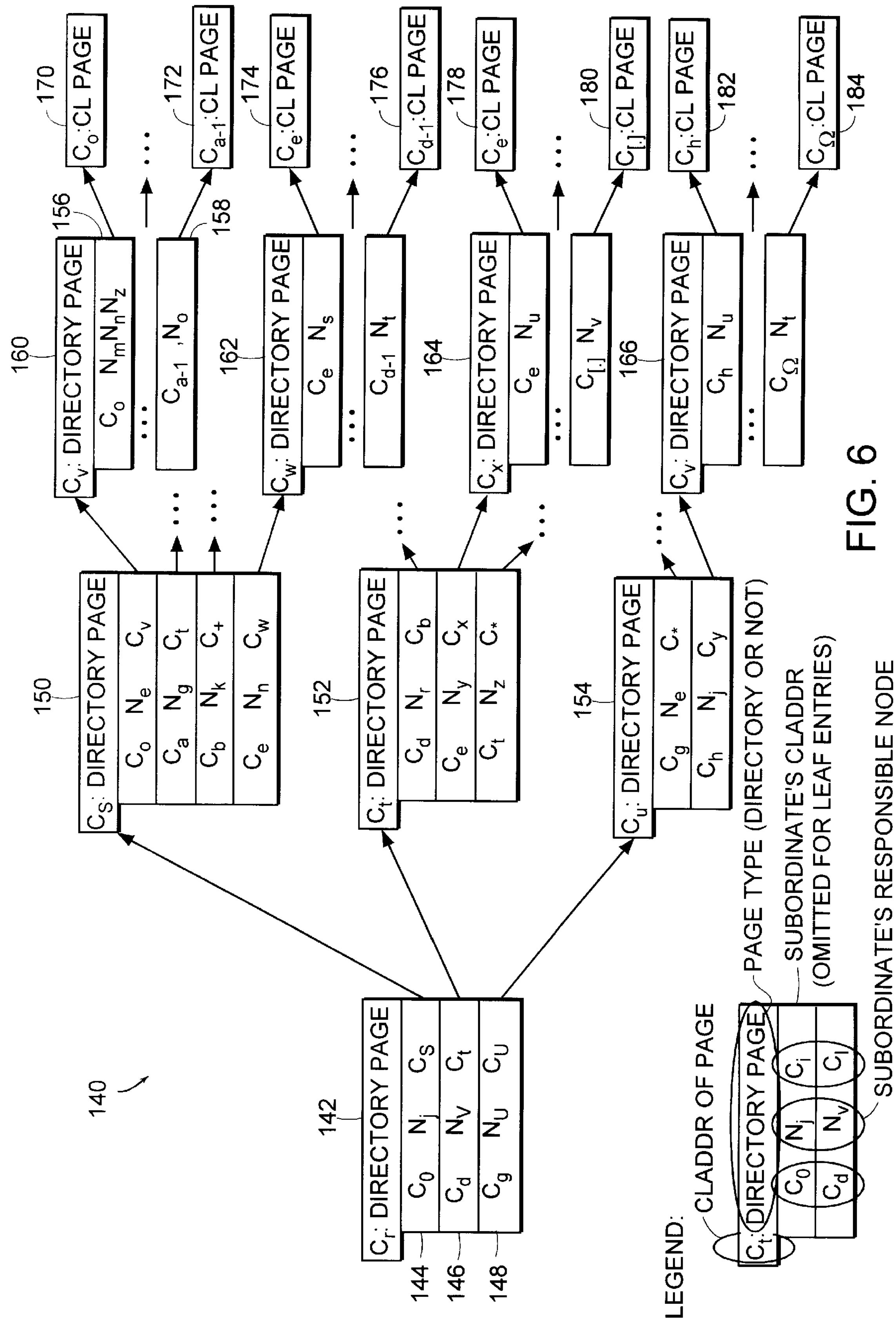

A *DIRECTORY PAGE* CONSISTS OF A PAGE HEADER PLUS ONE OR MORE DIRECTORY ENTRIES
A *DIRECTORY ENTRY* HAS 3 ATTRIBUTES:
• A CLADDR DESCRIBING THE START OF THE RANGE OF PAGES THAT THIS ENTRY HELPS LOCATE
• THE NODE TASKED WITH TRACKING THE CHILD PAGE'S OWNER
• THE CHILD PAGE'S CL ADDRESS (OMITTED IN LEAF ENTRIES)
120
DIRECTORY PAGE
122
PAGE HEADER
124
DIRECTORY ENTRY
START OF CLADDR RANGE
CHILD PAGE'S RESPONSIBLE NODE
CHILD PAGE'S CLADDR
130
132
134
126
DIRECTORY ENTRY
START OF CLADDR RANGE
CHILD PAGE'S RESPONSIBLE NODE
CHILD PAGE'S CLADDR
130
132
134

FIG. 5



140
142
144
146
148
150
152
154
156
158
160
162
164
166
170
172
174
176
178
180
182
184
C_r: DIRECTORY PAGE
C_S: DIRECTORY PAGE
C_t: DIRECTORY PAGE
C_u: DIRECTORY PAGE
C_v: DIRECTORY PAGE
C_w: DIRECTORY PAGE
C_x: DIRECTORY PAGE
C_y: DIRECTORY PAGE
C_o:CL PAGE
C_{a-1}:CL PAGE
C_e:CL PAGE
C_{d-1}:CL PAGE
$C_{[.]}$:CL PAGE
C_h:CL PAGE
C_Ω:CL PAGE
LEGEND:
CLADDR OF PAGE
PAGE TYPE (DIRECTORY OR NOT)
SUBORDINATE'S CLADDR
(OMITTED FOR LEAF ENTRIES)
SUBORDINATE'S RESPONSIBLE NODE

FIG. 6

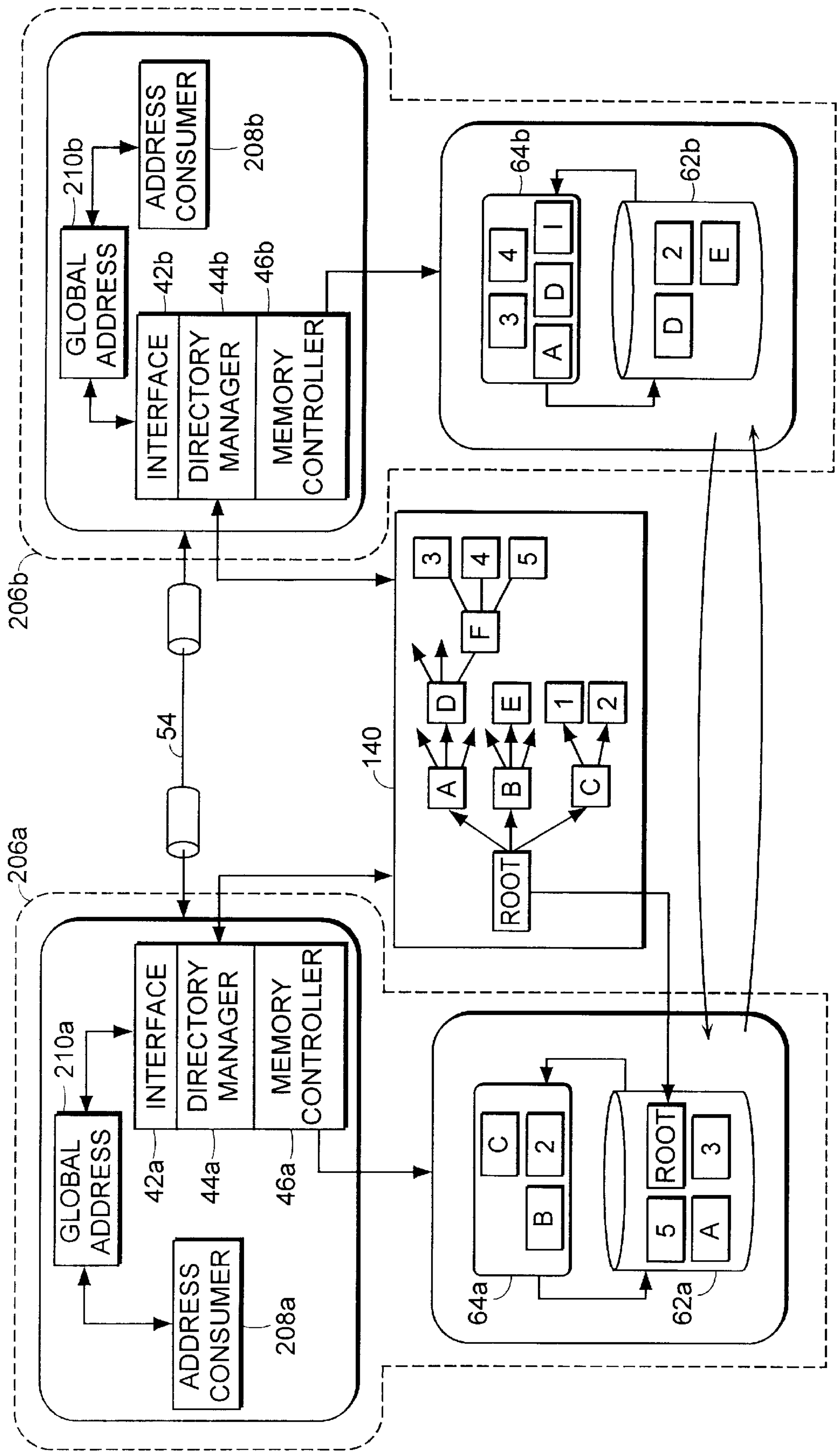

GLOBAL ADDRESS
ADDRESS CONSUMER
INTERFACE
DIRECTORY MANAGER
MEMORY CONTROLLER
210b
208b
42b
44b
46b
206b
64b
62b
3
4
A
D
I
D
2
E
54
140
ROOT
A
B
C
D
E
F
1
2
3
4
5
206a
GLOBAL ADDRESS
ADDRESS CONSUMER
INTERFACE
DIRECTORY MANAGER
MEMORY CONTROLLER
210a
208a
42a
44a
46a
C
2
B
ROOT
3
5
A
64a
62a

FIG. 7

SHARED MEMORY COMPUTER NETWORKS

FIELD OF THE INVENTION

The invention relates to computer systems, and more particularly, to computer networking systems and methods that provide shared memory systems and services.

BACKGROUND OF THE INVENTION

The conventional computer network includes a number of client computers connected together and further connected to a server computer that stores the data and the programs that client computers employ during network operation. This configuration is generally referred to as a client-server network. Typically, each client is a conventional computer system that includes a private main memory, typically a RAM memory, and a persistent storage, typically a hard disk. The server is usually an expensive high end machine that includes a high speed processor unit and a large memory, often having ten to one hundred times more storage than the individual client computers. The clients and server cooperate to share data and services among the different users, and to thereby make the individual computers appear as a unified distributed system.

To this end, the server acts as a central controller that provides through its large memory a central repository of network data, and that distributes services to the individual client computers, generally on an as-available basis. Typically, these services are provided by means of specialized software running on a high speed processor.

Although computer networks based on this client-server model have generally been successful at providing users with necessary computer services, as the user demands on computer systems have increased, the weaknesses in the client-server network are beginning to place limits on the services that can be provided.

An additional problem with the client-server network is that it provides a static operating environment that is set for optimal performance at a certain level of network activity. Consequently, the client-server network fails to exploit available resources to improve system performance. In particular, as the system activity rises above or drops below the expected level of network activity, the static operating environment lacks any ability to reconfigure dynamically the allocation of network resources to one providing better performance for the present level of activity.

Moreover, the client-server computer network requires that computer programs written to operate on the client-server network distribute themselves between clients and the server. This requires that the application programs implement a set of functions that divide the program between the clients and the server. This distribution of the application programs requires that the client-server application programs be quite complex. For example, a client-server computer program that shares data between different machines must include functionality that allows for the distribution of multiple copies of data files, the maintenance of coherency for the distributed copies, and other such low-level management services.

Further troubling is that the client-server network stores all important applications and data files in the memory of the server system. Consequently, the client-server network is subject to complete system failure each time the server system crashes.

For the above reasons, among others, the present client-server computer architecture fails to provide an adequate response to the increased demands of today's computer users.

Accordingly, it is an object of the invention to provide improved computer systems.

A further object of the invention is to provide computer network systems that have adaptable system configurations for dynamically exploiting distributed resources and thereby increasing network productivity.

It is a further object of the invention to provide computer network systems that eliminate the need for application programs to provide low-level memory management services across two or more distributed nodes.

It is still a further object of the invention to provide computer network systems that have improved fault tolerance and that are more readily scaleable for adding additional workstations as well as for the interconnection of two or more networks.

Further objects of the present invention shall be seen from the following description of certain illustrated embodiments.

SUMMARY OF THE INVENTION

The invention provides systems that can create and manage a virtual memory space that can be shared by each computer on a network and can span the storage space of each memory device connected to the network. Accordingly, all data stored on the network can be stored within the virtual memory space and the actual physical location of the data can be in any of the memory devices connected to the network.

More specifically, the system can create or receive, a global address signal that represents a portion, for example 4K bytes, of the virtual memory space. The global address signal can be decoupled from, i.e. unrelated to, the physical and virtual address spaces of the underlying computer hardware, to provide support for a memory space large enough to span each volatile and persistent memory device connected to the system. For example, systems of the invention can operate on 32-bit computers;, but can employ global address signals that can be 128 bits wide. Accordingly, the virtual memory space spans 2^{128} bytes, which is much larger than the 2^{32} address space supported by the underlying computer hardware. Such an address space can be large enough to provide a separate address for every byte of data storage on the network, including all RAM, disk and tape storage.

For such a large virual memory space, typically only a small portion is storing data at any time. Accordingly, the system includes a directory manager that tracks those portions of the virtual memory space that are in use. The system provides physical memory storage for each portion of the virtual memory space in use by mapping each such portion to a physical memory device, such as a RAM memory or a hard-drive. Optionally, the mapping includes a level of indirection that facilitates data migration, fault-tolerant operation, and load balancing.

By allowing each computer to monitor and track which portions of the virtual memory space are in use, each computer can share the memory space. This 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.

In one aspect, the invention can be understood to include computer systems having a shared addressable memory space. The systems can comprise a data network that carries data signals representative of computer readable information a persistent memory device that couples to the data network and that provides persistent data storage, and plural computers that each have an interface that couples to the data network, for accessing the data network to exchange data signals therewith. Moreover, each of the computers can include a shared memory subsystem for mapping a portion of the addressable memory space to a portion of the persistent storage to provide addressable persistent storage for data signals.

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. To this same end, the system can also include a distributor for mapping portions of the addressable memory space across the plurality of local persistent memory devices and a disk directory manager for tracking the mapped portions of said addressable memory space to provide information representative of the local persistent memory device that stores that portion of said addressable memory space mapped thereon.

The systems can also include a cache system for operating one of the local persistent memory devices as a cache memory for cache storing data signals associated with recently accessed portions of the addressable memory space. Further the system can include a migration controller for selectively moving portions of the addressable memory space between the local persistent memory devices of the plural computers. The migration controller can determine and respond to data access patterns, resource demands or any other criteria or heuristic suitable for practice with the invention. Accordingly, the migration controller can balance the loads on the network, and move data to nodes from which it is commonly accessed. The cache controller can be a software program running on a host computer to provide a software managed RAM and disk cache. The RAM can be any volatile memory including SRAM, DRAM or any other volatile memory. The disk can be any persistent memory including any disk, RAID, tape or other device that provides persistent data storage.

The systems can also include a coherent replication controller for generating a copy, or select number of copies, of a portion of the addressable memory space maintained in the local persistent memory device of a first computer and for storing the copy in the local persistent memory device of a second computer. The coherent replication controller can maintain the coherency of the copies to provide coherent data replication.

The systems can also be understood to provide integrated control of data stored in volatile memory and in persistent memory. In such systems a volatile memory device has volatile storage for data signals, and the shared memory subsystem includes an element, typically a software module, for mapping a portion of the addressable memory space to a portion of the volatile storage. In these systems the volatile memory device can be comprised of a plurality of local volatile memory devices each coupled to a respective one of the plural computers, and the persistent memory device can be comprised of a plurality of local persistent memory devices each coupled to a respective one of the plural computers.

In these systems, a directory manager can track the mapped portions of the addressable memory space, and can include two sub-components; a disk directory manager for tracking portions of the addressable memory space mapped to the local persistent memory devices, and a RAM directory manager for tracking portions of the addressable memory space mapped to the local volatile memory devices. Optionally, a RAM cache system can operate one of the local volatile memory devices as a cache memory for cache storing data signals associated with recently accessed portions of the addressable memory space.

The systems can include additional elements including a paging element for remapping a portion of the addressable memory space between one of the local volatile memory devices and one of the local persistent memory devices; a policy controller for determining a resource available signal representative of storage available on each of the plural computers and, a paging element that remaps the portion of addressable memory space from a memory device of a first computer to a memory device of a second computer, responsive to the resource available signal; and a migration controller for moving portions of addressable memory space between the local volatile memory devices of the plural computers.

Optionally, the systems can include a hierarchy manager for organizing the plural computers into a set of hierarchical groups wherein each group includes at least one of the plural computers. Each the group can include a group memory manager for migrating portions of addressable memory space as a function of the hierarchical groups.

The system can maintain coherency between copied portions of the memory space by including a coherent replication controller for generating a coherent copy of a portion of addressable memory space.

The system can generate or receive global address signals. Accordingly the systems can include an address generator for generating a global address signal representative of a portion of addressable memory space. The address generator can include a spanning unit for generating global address signals as a function of a storage capacity associated with the persistent memory devices, to provide global address signals capable of logically addressing the storage capacity of the persistent memory devices.

In distributed systems, the directory manager can be a distributed directory manager for storing within the distributed memory space, a directory signal representative of a storage location of a portion of the addressable memory space. The distributed directory manager can include a directory page generator for allocating a portion of the addressable memory space and for storing therein an entry signal representative of a portion of the directory signal. The directory page generator optionally includes a range generator for generating a range signal representative of a portion of the addressable memory space, and for generating the entry signal responsive i:o the range signal, to provide an entry signal representative of a portion of the directory signal that corresponds to the portion of the addressable memory space. Moreover, the distributed directory manager can include a linking system for linking the directory pages to form a hierarchical data structure of the linked directory pages as well as a range linking system for linking the directory pages, as a function of the range signal, to form a hierarchical data structure of linked directory pages.

As the data stored by the system can be homeless, in that the data has no fixed physical home, but can migrate, as resources and other factors dictate, between the memory devices of the network, a computer system according to the invention can include a directory page generator that has a node selector for generating a responsible node signal representative of a select one of the plural computers having location information for a portion of the shared address space. This provides a level of indirection that decouples the directory from the physical storage location of the data. Accordingly, the directory needs only to identify the node, or other device, that tracks the physical location of the data. This way, each time data migrates between physical storage locations, the directory does not have to be updated, since the node tracking the location of the data has not changed and still provides the physical location information.

Accordingly, the system can include page generators that generate directory pages that carry information representative of a location monitor, such as a responsible computer node, that tracks a data storage location, to provide a directory structure for tracking homeless data. Moreover, the directory itself can be stored as pages within the virtual memory space. Therefore, the data storage location can store information representative of a directory page, to store the directory structure as pages of homeless data.

In another aspect, the invention can be understood as methods for providing a computer system having a shared addressable 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.

BRIEF DESCRIPTION OF THE ILLUSTRATED EMBODIMENTS

FIG. **1** illustrates a distributed shared memory computer network according to the invention;

FIG. **2** is a functional block diagram that illustrates in more detail one distributed shared memory computer network of the type shown in FIG. **1**;

FIG. **3** illustrates in more detail a shared memory subsystem suitable for practice with the network illustrated in FIG. **2**;

FIG. **4** is a functional block diagram of one shared memory subsystem according to the invention;

FIG. **5** illustrates a directory page that can be provided by a shared memory subsystem of the type depicted in FIG. **4**;

FIG. **6** illustrates a directory that can be distributed within a shared memory and formed of directory pages of the type illustrated in FIG. **5**; and

FIG. **7** illustrates in functional block diagram form a system of the invention that employs a directory according to FIG. **6** for tracking portions of a distributed shared memory.

DETAILED DESCRIPTION OF THE ILLUSTRATED EMBODIMENTS

FIG. **1** illustrates a computer network **10** that provides a shared memory that spans the memory space of each node of the depicted computer network **10**.

Specifically, FIG. **1** illustrates a computer network **10** that includes a plurality of nodes **12***a*–**12***c*, each having a CPU **14**, an operating system **16**, an optional private memory device **18**, and a shared memory subsystem **20**. As further depicted in by FIG. **1**, each node **12***a*–**12***c* connects via the shared memory subsystem **20** to a virtual shared memory **22**. As will be explained in greater detail hereinafter, by providing the shared memory subsystem **20** that allows the node **12***a*–**12***c* to access the virtual shared memory **22**, the computer network **10** enables network nodes **12***a*–**12***c* to communicate and share functionality using the same techniques employed by applications when communicating between applications running on the same machine. These techniques can employ object linking and embedding, dynamic link libraries, class registering, and other such techniques. Accordingly, the nodes **12** can employ the virtual shared memory **22** to exchange data and objects between application programs running on the different nodes **12** of the network **10**.

In the embodiment depicted in FIG. **1**, each node **12** can be a conventional computer system such as a (commercially available IBM PC compatible computer system. The processor **14** can be any processor unit suitable for performing the data processing for that computer system. The operating system **16** can be any commercially available or proprietary operating system that includes, or can access, functions for accessing the local memory of the computer system and networking.

The private memory device **18** can be any computer memory device suitable for storing data signals representative of computer readable information. The private memory provides the node with local storage that can be kept inaccessible to the other nodes on the network. Typically the private memory device **18** includes a RAM, or a portion of a RAM memory, for temporarily storing data and application programs and for providing the processor **14** 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.

The shared memory subsystem **20** depicted in FIG. **1** is an embodiment of the invention that couples between the operating system **16** and the virtual shared memory **22** and forms an interface between the operating system **16** and the virtual shared memory to allow the operating system **16** to access the virtual shared memory **22**. The depicted shared memory subsystem **20** is a software module that operates as a stand-alone distributed shared memory engine. The depicted system is illustrative and other systems of the invention can be realized as shared memory subsystems that can be embedded into an application program, or be implemented as an embedded code of a hardware device. Other such applications can be practiced without departing from the scope of the invention.

The depicted virtual shared memory **22** illustrates a virtual shared memory that is accessible by each of the nodes **12***a*–**12***c* via the shared memory subsystem **20**. The virtual shared memory **22** can map to devices that provide physical storage for computer readable data, depicted in FIG. **1** as a plurality of pages **24***a*–**24***d*. In one embodiment, the pages form portions of the shared memory space and divide the address space of the shared memory into page addressable memory spaces. For example the address space can be paged into 4K byte sections. In other embodiments alternative granularity can be employed to manager the shared memory space. Each node **12***a*–**12***c* through the shared memory subsystem **20** can access each page **24***a*–**24***d* slored in the virtual shared memory **22**. Each page **24***a*–**24***d* represents a unique entry of computer data stored within the virtual shared memory **22**. Each page **24***a*–**24***d* is accessible to each one of the nodes **12***a*–**12***c*, and alternatively, each node can store additional pages of data within the virtual shared memory **22**. Each newly stored page of data can be accessible to each of the other nodes **12***a*–**12***c*. Accordingly, the virtual shared memory **22** provides a system for sharing and communicating data between each node **12** of the computer network **10**.

FIG. 2 illustrates in functional block diagram form a computer network **30** that has a distributed shared memory. In this embodiment, each node **12**a–**12**c has a memory subsystem **32** that connects between the operating system **16** and the two local memory devices, the RAM **34** and the disk **36**, and that further couples to a network **38** that couples to each of the depicted nodes**12**a,**12**b and **12**c and to a network memory device **26**.

More particularly, FI G. **2** illustrates a distributed shared memory network **30** that includes a plurality of nodes **12**a,**12**b and **12**c, each including a processing unit **14**, an operating system **16**, a memory subsystem **32**, a RAM **34**, and a disk **36**. FIG. **2** further depicts a computer network system **38** that connects between the nodes **12**a,**12**b and **12**c and the network memory device **26**. The network **38** provides a network communication system across these elements.

The illustrated memory subsystems **32**a–**32**c that connect between the operating system **16**a–**16**c, the memory elements **34**a–**34**c, **36**a–**36**c, and the network **38**, encapsulate the local memories of each of the nodes to provide an abstraction of a shared virtual memory system that spans across each of the nodes **12**a, **12**b and **12**c on the network **38**. The memory subsystems **32**a–**32**c can be software modules that act as distributors to map portions of the addressable memory space across the depicted memory devices. The memory subsystems further track the data stored in the local memory of each node **12** and further operate network connections with network **38** for transferring data between the nodes **12**a–**12**c. In this way, the memory subsystems **32**a–**32**c access and control each memory element on the network **38** to perform memory access operations that are transparent to the operating system **16**. Accordingly, the operating system **16** interfaces with the memory subsystem **32** as an interface to a global memory space that spans each node **12**a–**12**c on the network **38**.

FIG. **2** further depicts; that the system **30** provides a distributed shared memory that includes persistent storage for portions of the distributed memory. In particular, the depicted embodiment includes a memory subsystem, such as subsystem **32**a, that interfaces to a persistent memory device, depicted as the disk **36**a. The subsystem **32**a can operate the persistent memory device to provide persistent storage for portions of the distributed shared memory space. As illustrated, each persistent memory device **36** depicted in FIG. **2** has a portion of the addressable memory space mapped onto it. For example, device **36**a has the portions of the addressable memory space, C_o, C_d, C_g, mapped onto it, and provides persistent storage For data signals stored in those ranges of addresses.

Accordingly, the subsystem **32**a can provide integrated control of persistent storage 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.

In one optional embodiment, the nodes of the network are organized into a hierarchy of groups. In this embodiment, the memory subsystems **32**a–**32**c can include a hierarchy manager that provides hierarchical control for the distribution of data. This includes controlling the migration controller, and policy controller, which are discussed in detail below, 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.

FIG. **3** illustrates in more detail one shared memory subsystem **40** according to the invention. FIG. **3** depicts a shared memory subsystem **40**, that includes an interface **42**, a DSM directory manager **44**, a memory controller **46**, a local disk cache controller **48**, and a local RAM cache controller **50**. FIG. **3** further depicts the network **54**, an optional consumer of the DSM system, depicted as the service **58**, the operating system **16**, a disk driver **60**, a disk element **62** and a RAM element **64**.

The shared memory subsystem **40** depicted in FIG. **3** can encapsulate the memory management operations of the network node **12** to provide a virtual shared memory that can span across each node that connects into the network **54**. Accordingly, each local node **12** views the network as a set of nodes that are each connected to a large shared computer memory.

The depicted interface **42** provides an entry point for the local node to access the shared memory space of the computer network. The interface **42** can couple directly to the operating system **16**, to a distributed service utility such as the depicted DSM file system **58**, to a distributed user-level service utility, or alternatively to any combination thereof.

The depicted interface **42** provides an API that is a memory oriented API. Thus, the illustrated interface **42** can export a set of interfaces that provide low-level control of the distributed memory. As illustrated in FIG. **3**, the interface **42** exports the API to the operating system **16** or to the optional DSM service **58**. The operating system **16** or the service employs the interface **42** to request standard memory management techniques, such as reading and writing from portions of the memory space. These portions of the memory space can be the pages as described above which can be 4K byte portions of the shared memory space, or other units of memory, such as objects or segments. 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. The address generator can include a spanning module that generates address signals for a memory space that spans the storage capacity of the network.

Accordingly, in one embodiment, the interface **42** receives requests to manipulate pages of the shared memory space. To this end, the interface **42** can comprise a software module that includes a library of functions that can be called by services, the OS **16**, or other caller, or device. The function calls provide the OS **16** with an API of high level memory oriented services, such as read data, write data, and allocate memory. The implementation of the functions can include a set of calls to controls that operate the directory manager **44**, and the local memory controller **46**. Accordingly, the interface **42** can be a set of high level memory function calls to interface to the low-level functional elements of shared memory subsystem **40**.

FIG. **3** further depicts a DSM directory manager **44** that couples to the interface **42**. The interface **42** passes request signals that represent requests to implement memory operations such as allocating a portion of memory, locking a portion of memory, mapping a portion of memory, or some other such memory function. The directory manager **44** manages a directory that can include mappings than can span across each memory device connected to the network **38** depicted in FIG. **2**, including each RAM and disk element accessible by the network. The directory manager **44** stores a global directory structure that provides a map of the global address space. In one embodiment as will be explained in greater detail hereinafter, the directory manager **44** provides a global directory that maps between global address signals and responsible nodes on the network. A responsible node stores information regarding the location and attributes of data associated with a respective global address, and optionally stores a copy of that page's data. Consequently, the directory manager **44** tracks information for accessing any address location within the virtual address space.

The control of the distributed shared memory can be coordinated by the directory manager **44** and the memory controller **46**. The directory manager **44** maintains a directory structure that can operate on a global address received from the interface **42** and identify, for that address, a node on the network that is responsible for maintaining the page associated with that address of the shared memory space. Once the directory manager **44** identifies which node is responsible for maintaining a particular address, the directory manager **44** can identify a node that stores information for locating a copy of the page, and make the call to the memory controller **46** of that node and pass to that node's memory controller the memory request provided by the memory interface **42**. Accordingly, the depicted directory manager **44** is responsible for managing a directory structure that identifies for each page of the shared memory space a responsible node that tracks the physical location of the data stored in the respective page. Thus, the directory, rather than directly providing the location of the page, can optionally identify a responsible node, or other device, that tracks the location of the page. This indirection facilitates maintenance of the directory as pages migrate between nodes.

The memory controller **46** performs the low level memory access functions that physically store data within the memory elements connected to the network. In the depicted embodiment, the directory manager **44** of a first node can pass a memory access request through the interface **42**, to the network module of the OS **16**, and across the network **54** to a second node that the directory manager **44** identifies as the responsible node for the given address. The directory manager **44** can then query the responsible node to determine the attributes and the current owner node of the memory page that is associated with the respective global address. The owner of the respective page 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 **46** of the owner can access, through the OS **16** of that node or through any interface, the memory of the owner node to access the data of the page that is physically stored on that owner node.

In particular, as depicted in FIG. **3**, the directory manager **44** couples to the network module **52** which couples to the network **54**. The directory manager can transmit to the network module **52** a command and associated data that directs the network interface **52** to pass a data signal to the owner node. The owner node receives the memory request across network **54** and through network module **52** that passes the memory request to the interface **42** of that owner node. The interface **42** couples to the memory controller **46** 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.

Once the owner node has performed the requested memory operation, such as reading a page of data, the memory subsystem **40** of the owner node can then transfer the page of data, or a copy of the page of data, via the network **54** to the node that originally requested access to that portion of the shared memory. The page of data is transferred via the network **54** to the network module **52** of the requesting node and the shared memory subsystem **40** operates the memory controller **46** to store in the local memory of the requesting node a copy of the accessed data.

Accordingly, in one embodiment of the invention, when a first node accesses a page of the shared memory space which is not stored locally on that node, the directory manager **4** identifies a node that has a copy of the data stored in that page and moves a copy of that data into the local memory of the requesting node. The local memory storage, both volatile and persistent, of the requesting node therefore becomes a cache for pages that have been requested by that local node. This embodiment is depicted FIG. **3** which depicts a memory controller that has a local disk cache controller **48** and a local RAM cache controller **50**. Both of these local cache controllers can provide to the operating system **16**, or other consumer pages of the shared memory space that are cache stored in the local memory of the node, including local persistent memory and local volatile memory.

The shared memory subsystem can include a coherent replication controller that maintains coherency between cached pages by employing a coherence through invalidation process, a coherence through migration process or other coherence process suitable for practice with the present invention. The coherent replication controller can automatically generate a copy of the data stored in each page and can store the copy in a memory device that is separate from the memory device or the original copy. This provides for fault tolerant operation, as the failure of any one memory device will not result in the loss of data. The coherent replication controller can be a software model that monitors all copies of pages kept in volatile memory and made available for writing. The controller can employ any of the coherency techniques named above, and can store tables of location information that identifies the location information for all generated copies.

FIG. **4** illustrates in greater detail one embodiment of a shared memory subsystem according to the invention. The shared memory subsystem **70** depicted in FIG. **4** includes a remote operations element **74**, a local RAM cache **76**, a RAM copyset **78**, a global RAM directory **80**, a disk copyset **82**, a global disk directory **84**, a configuration manager **88**, a policy element **90**, and a local disk cache **94**. FIG. **4** further depicts a network element **104**, a physical memory **100**, shared data element **102**, a physical file system **98**, which is part of the operating system **16**, a configuration service **108**, a diagnostic service **110**, and a memory access request **112**. The depicted subsystem **70** can be a computer program that couples to the physical memory, file system, and network system of the host node, or can be electrical circuit card assemblies that interface to the host node, or can be a combination of programs and circuit card assemblies.

The flow scheduler **72** depicted in FIG. **4** can orchestrate the controls provided by an API of the subsystem **70**. In one embodiment, the flow scheduler **72** can be a state machine that monitors and responds to the requests **112** and remote requests through network **104** which can be instructions for memory operations and which can include signals representative of the global addresses being operated on. These memory operation requests **112** 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.

Alternatively, the flow scheduler **72** can be a program, such as an interpreter, that provides an execution environment and can map these op-codes into control flow programs called applets. The applets can be independent executable programs that employ both environment services, such as threading, synchronization, and buffer management, and the elements depicted in FIG. **4**. The API is capable of being called from both external clients, like a distributed shared memory file system, as well as recursively by the applets and the other elements **74–94** of the subsystem **70**. Each element can provide a level of encapsulation to the management of a particular resource or aspect of the system. To this end, each element can export an API consisting of functions to be employed by the applets. This structure is illustrated in FIG. **4**. Accordingly, the flow scheduler **72** can provide an environment to load and execute applets. The applets are dispatched by the flow scheduler **72** on a per op-code basis and can perform the control flow for sequential or parallel execution of an element to implement the op-code on the specified global address, such as a read or write operation. Optionally, the flow scheduler **72** can include an element to change dynamically the applet at run time as well as execute applets in parallel and in interpreted mode.

The depicted shared memory subsystem **70** includes a bifurcated directory manager that includes the global RAM directory **80** and the global disk directory **84**. The global RAM directory **80** is a directory manager that tracks information that can provide the location of pages that are stored in the volatile memory, typically RAM, of the network nodes. The global disk directory **84** is a global disk directory manager that manages a directory structure that tracks information that can provide the location of pages that are stored on persistent memory devices. Together, the global RAM directory **80** and the global disk directory **84** provide the shared memory subsystem **70** with integrated directory management for pages that are stored in persistent storage and volatile memory.

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. Accordingly, the RAM directory manager passes control of that page to the disk directory manager which can then treat the page as any other page of data. This allows for improved load balancing, by removing data from RAM memory, and storing it in the disk devices, under the control of the disk directory manager.

The local memory controller of the subsystem **70** is provided by the local RAM cache **76** and the local disk cache **94**. The local RAM cache **76** which couples to the physical memory **100** of the local node can access, as described above, the virtual memory space of the local node to access data that is physically stored within the RAM memory **100**. Similarly, the local disk cache **94** couples to the persistent storage device **98** and can access a physical location that maintains in the local persistent storage data of the distributed shared memory.

FIG. **4** also depicts t remote operations element **74** that couples between the network **104** and the flow scheduler **72**. The remote operations element **74** negotiates the transfer of data across the network **104** for moving portions of the data stored in the shared memory space between the nodes of the network. The remote operations element **74** can also request services from remote peers, i.e. invalidate to help maintain coherency or for other reasons.

FIG. **4** also depicts a policy element **90** 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. 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.

FIG. **4** further depicts a memory subsystem **70** that includes a RAM copyset **78** and a disk copyset **82**. These copysets can manage copies of pages that are cached at a single node. The disk copyset **82** can maintain information on copies of pages that are stored in the local disk cache, which can be the local persistent memory. Similarly the RAM copyset **78** can maintain information on copies of pages that are stored in the local RAM cache which can be the local RAM. These copysets encapsulate indexing and storage of copyset data that can be employed by applets or other executing code for purposes of maintaining the coherency of data stored in the shared memory space. The copyset elements can maintain copyset data that identifies the pages cached by the host node. Further, the copyset can identify the other nodes on the network that maintain a copy of that page, and can further identify for each page which of these nodes is the owner node, wherein the owner node can be a node which has write privileges to the page being accessed. The copysets themselves can be stored in pages of the distributed shared memory space.

The local RAM cache **76** provides storage for memory pages and their attributes. In one embodiment, the local RAM cache **76** provides a global address index for accessing the cached pages of the distributed memory and the attributes based on that page. In this embodiment, the local ram cache **76** provides the index by storing in memory a list of each global address cached in the local RAM. With each listed global address, the index provides a pointer into a buffer memory and to the location of the page data. Optionally, with each listed global address, the index can further provide attribute information including a version tag 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.

In the embodiment depicted in FIG. **4**, the memory subsystem **70** provides the node access to the distributed memory space by the coordinated operation of the directory manager that includes the global RAM directory **80** and the global disk directory **84**, the cache controller that includes the local RAM cache and the local disk cache elements **76** and **94**, and the copyset elements which include the RAM copyset **78** and the disk copyset **82**.

The directory manager provides a directory structure that indexes the shared address space. Continuing with the example of a paged shared address space, the directory manager of the subsystem **70** allows the host node to access, by global addresses, pages of the shared memory space.

FIGS. **5** and **6** illustrate one example of a directory structure that provides access to the shared memory space. FIG. **5** depicts a directory page **120** that includes a page header **122**, directory entries **124** and **126**, wherein each directory entry includes a range field **130**, a responsible node field **132**, and an address field **134**. The directory pages can be generated by a directory page generator that can be a software module controlled by the directory manager. It will be understood that the directory manager can generate multiple directories, including one for the Global disk and one for the Global RAM directories. The depicted directory page **120** can be a page of the global address space, such as a 4K byte portion of the shared address space. Therefore, the directory page can be stored in the distributed shared memory space just as the other pages to which the directory pages provide access.

As further depicted in FIG. **5**, each directory page **120** includes a page header **122** that includes attribute information for that page header, which is typically metadata for the directory page, and further includes directory entries such as the depicted directory entries, **124** and **126**, which provide an index into a portion of the shared address space wherein that portion can be one or more pages, including all the pages of the distributed shared memory space. The depicted directory page **120** includes directory entries that index a selected range of global addresses of the shared memory space. To this end, the directory generator can include a range generator so that each directory entry can include a range field **130** that describes the start o f a range of addresses that that entry locates.

Accordingly, each directory page **120** can include a plurality of directory entries, such as entries **124** and **126**, that can subdivide the address space into a subset of address ranges. For example, the depicted directory page **120** includes two directory entries **124** and **126**. The directory entries **124** and **126** can, for example, subdivide the address space into two sub-portions. In this example, the start address range of the directory entry **124** could be the base address of the address space, and the start address range of the directory entry **126** could be the address for the upper half of the memory space. Accordingly, the directory entry **124** provides an index for pages stored in the address space between the base address and up to the mid-point of the memory space and, in complement thereto, the directory entry **126** provides an index to pages stored in the address space that ranges from the mid-point of the address space to the highest address.

FIG. **5** further depicts a directory page **120** that includes, in each directory entry, a responsible node field **132** and the child page global address field **134**. These fields **132**, **134** provide further location information for the data stored in pages within the address range identified in field **130**.

FIG. **6** depicts a directory **140** formed from directory pages similar to those depicted in FIG. **5**. FIG. **6** depicts that the directory **140** includes directory pages **142**, **150–154**, and **160–166**. FIG. **6** further depicts that the directory **140** provides location information to the pages of the distributed shared memory space depicted in FIG. **6** as pages **170–184**.

The directory page **142** depicted in FIG. **6** acts like a root directory page and can be located at a static address that is known to each node coupled to the distributed address space. The root directory page **142** includes three directory entries **144**, **146**, and **148**. Each directory entry depicted in FIG. **6** has directory entries similar to those depicted in FIG. **5**. For example, directory entry **144** includes a variable Co whichrepresents the address range field **130**, a variable Nj representative of the field **132**, and a variable Cs representative of the field **134**. The depicted root directory page **142** subdivides the address space into three ranges illustrated as an address range that extends between the address Co and Cd, a second address range that extends between the address Cd and Cg, and a third address range that extends between Cg and the highest memory location of the address space.

As further depicted in FIG. **6**, each directory entry **144**, **146**, and **148** points to a subordinate directory page, depicted as directory pages **150**, **152**, and **154**, each of which further subdivides the address range index by the associated directory entry of the root d rectory **142**. In FIG. **6**, this subdivision process continues as each of the directory pages **150**, **152**, and **154** each again have directory entries that locate subordinate directory pages including the depicted examples of directory pages **160**, **162**, **164**, and **166**.

The depicted example of directory pages **160**, **162**, **164**, and **166** are each leaf entries. The leaf entries contain directory entries such as the directory entries **156** and **158** of the leaf entry **160**, that store a range field **130** and the responsible node field **132**. These leaf entries identify an address and a responsible node for the page in the distributed memory space that is being accessed, such as the depicted pages **170–184**. For example, as depicted in FIG. **6**, the leaf entry **156** points to the page **170** that corresponds to the range field **130** of the leaf entry **156**, which for a leaf entry is the page being accessed. In this way, the directory structure **140** provides location information for pages stored in the distributed address space.

In the depicted embodiment of FIG. **6**, a node selector can select a responsible node for each page, as described above, so that the leaf entry **156** provides information of the address and responsible node of the page being located. Accordingly, this directory tracks ownership and responsibility for data, to provide a level of indirection between the directory and the physical location of the data. During a memory access operation, the memory subsystem **70** passes to the responsible node indicated in the leaf entry **156** the address of the page being accessed. The shared memory subsystem of that node can identify a node that stores a copy of the page being accessed, including the owner node. This identification of a node having a copy can be performed by the RAM copyset or disk copyset of the responsible node. The node having a copy stored in its local physical memory, such as the owner node, can employ its local cache elements, including the local RAM cache and local disk cache to the identify from the global address signal a physical location of the data stored in the page being accessed. The cache element can employ the operating system of the owner node to access the memory device that maintains that physical location in order that the data stored in the page can be accessed. For a read-memory operation, or for other similar operations, the data read from the physical memory of the owner node can be passed via the network to the memory subsystem of the node requesting the read and subsequently stored into the virtual memory space of the requesting node for use by that node.

With reference again to FIG. **6**, it can be seen that the depicted directory structure **140** comprises a hierarchical structure. To this end, the directory structure **140** provides a structure that continually subdivides the memory space into smaller and smaller sections. Further, each section is represented by directory pages of the same structure, but indexes address spaces of different sizes. As pages are created or deleted, a linker inserts or deletes the pages from the directory. In one embodiment, the linker is a software module for linking data structures. The linker can operate responsive to the address ranges to provide the depicted hierarchical structure. Accordingly, the depicted directory **140** provides a scaleable directory for the shared address space. Moreover, the directory pages are stored in the distributed address space and maintained by the distributed shared memory system. A root for the directory can be stored in known locations to allow for bootstrap of the system.

Consequently, 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.

FIG. **7** depicts the directory of FIG. **6** being employed by a system according to the invention. In particular FIG. **7** depicts a system **200** that includes two nodes, **206***a* and **206***b*, a directory structure **140**, and a pair of local memories having volatile memory devices **64***a* and **64***b*,and persistent memory devices **62***a* and **62***b*. Depicted node **206***a* includes an address consumer **208***a*, a global address **210***a*, and interface **42***a*, a directory manager **44***a* and a memory controller **46***a*. Node **206***b* has corresponding elements. The nodes are connected by the network **54**. A directory **140** having a root page, directory pages A–F and pages **1–5** is further depicted.

Each node **206***a* and **206***b* operates as discussed above. The depicted address consumers **208***a* and **208***b* can be an application program, file system, hardware device or any other such element that requests access to the virtual memory. In operation, the address consumers **208***a* and **208***b* 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. As addresses get generated, the respective directory managers **44***a* and **44***b* generate directory pages and store the pages in the directory structure **140**. As depicted, the directory structure **140** tracks the portions of the address space being employed by the system **200**, and physical storage for each page is provided within the local memories.

As shown in FIG. **7**, the data associated with the directory pages are distributively stored across the two local memories and duplicate copies can exist. As described above and now illustrated in FIG. **7**, the data can move between different local memories and also move, or page, between volatile and persistent storage. The data movement can be responsive to data requests made by memory users like application programs, or by operation of the migration controller described above. As also described above, the movement of data between different memory locations can occur without requiring changes to the directory **140**. This is achieved by providing a directory **140** that is decoupled from the physical location of the data by employing a pointer to a responsible node that tracks the data storage location. Accordingly, although the data storage location can change, the responsible node can remain constant, thereby avoiding any need to change the directory **140**.

It will be understood to those of ordinary skill in the art that certain modification, additions, and subtractions can be made to the embodiments described above without departing from the spirit and scope of the invention. Accordingly, the invention described above is not to be limited to the illustrated embodiments and is to be understood by the claims set forth below.

What is claimed is:

1. A computer system having a shared addressable memory space, comprising

a data network for carrying data signals representative of computer readable information, and

a plurality of computers, each of said plurality of computers sharing the shared addressable memory space and including

an interface, coupled to said data network, for accessing said data network to exchange data signals therewith,

a local volatile memory device coupled to said computer and having volatile storage for data signals,

a local persistent memory device coupled to said computer and having persistent storage for data signals, and

a shared memory subsystem for mapping a portion of said shared addressable memory space to a portion or the whole of said persistent storage and said volatile storage to provide thereby addressable persistent and volatile storage for data signals accessible by each of the plural computers, said shared memory subsystem including

a distributor for mapping portions of said addressable memory space across said plurality of local persistent memory devices, to distribute said addressable memory space across said plurality of local persistent memory devices, and

a disk directory manager for tracking said mapped portions of said addressable memory space to provide information representative of which of said local persistent memory devices has which of said portions of said addressable memory space mapped thereon.

2. A computer system according to claim **1** further comprising

a cache system for operating one of said local persistent memory devices as a cache memory for cache storing data signals associated with recently accessed portions of said addressable memory space.

3. A computer system according to claim **1** further comprising

a migration controller for selectively moving portions of said addressable memory space between said local persistent memory devices of said plurality of computers.

4. A computer system according to claim **1** further comprising

a replication controller for generating a copy of a portion of said addressable memory space maintained in said local persistent memory device of a first one of said computers and for storing said copy in said local persistent memory device of a second one of said computers.

5. A computer system according to claim **1** further comprising

a RAM directory manager for tracking said mapped portions of said addressable memory space to provide information representative of which of said local volatile memory devices has which of said portions of said addressable memory space mapped thereon.

6. A computer system according to claim **1** further comprising

a RAM cache system for operating one of said local volatile memory devices as a cache memory for cache storing data signals associated with recently accessed portions of said addressable memory space.

7. A computer system according to claim **1** further comprising

a paging element for remapping a portion of said addressable memory space between one of said local volatile memory devices and one of said local persistent memory devices.

8. A computer system according to claim **7** further comprising

a policy controller for determining a resource available signal representative of storage available on each of said plural computers and, and wherein said paging element remaps said portion of addressable memory space from a memory device of a first computer to a memory device of a second computer, responsive to said resource available signal.

9. A computer system according to claim **1** further comprising

a migration controller for moving portions of addressable memory space between said local volatile memory devices of said plurality of computers.

10. A computer system according to claim **1** further comprising

a hierarchy manager for organizing said plurality of computers into a set of hierarchical groups wherein each group includes at least one of said plurality computers.

11. A computer system according to claim **10** wherein each said group includes

a group memory manager for migrating portions of addressable memory space as a function of said hierarchical groups.

12. A computer system according to claim **1** further comprising

a coherent replication controller for generating a coherent copy of a portion of addressable memory space.

13. A computer system according to claim **1** further comprising

an address generator for generating a global address signal representative of a portion of addressable memory space.

14. A computer system according to claim **13**, wherein said address generator includes a spanning unit for generating global address signals as a function of a storage capacity associated with said persistent memory devices, to provide global address signals capable of logically addressing said storage capacity of said persistent memory devices.

15. A computer system according to claim **1** further comprising

a distributed directory manager for storing within said distributed memory space, a directory signal representative of a storage location of a portion of said addressable memory space.

16. A computer system according to claim **15** wherein said distributed directory manager includes

a directory page generator for allocating a portion of said addressable memory space and for storing therein an entry signal representative of a portion of said directory signal.

17. A computer system according to claim **16** wherein said directory page generator includes

a range generator for generating a range signal representative of a portion of said addressable memory, space, and for generating said entry signal responsive to said range signal, to provide an entry signal representative of a portion of said directory signal that corresponds to said portion of said addressable memory space.

18. A computer system according to claim **17** wherein said distributed directory manager includes; a linking system for linking said directory pages to form a hierarchical data structure of said linked directory pages.

19. A computer system according to claim **17** wherein said distributed directory manager includes a range linking system for linking said directory pages, as a function of said range signal, to form a hierarchical data structure of linked directory pages.

20. A computer system according to claim **16** wherein said directory page generator includes a node selector for generating a responsible node signal representative of a select one of said plural computers having location information for a portion of said shared address space.

21. A computer system according to claim **1** further comprising,

a page generator for generating a directory page that carries information representative of a location monitor that tracks a data storage location, to provide a directory structure for tracking homeless data.

22. A computer system according to claim **21** wherein said data storage location stores information representative of a directory page, to store said directory structure as pages of homeless data.

23. A computer system according to claim **1** further comprising

a distributed directory manager for storing a directory signal representative of a portion of said addressable memory space.

* * * * *

#12/B
ONE
0/21/98

OIPE
OCT 15 1998
PATENT & TRADEMARK OFFICE

EXPRESS MAIL LABEL NO.: EM400950284US

PATENT

Atty. Docket No. CLC-001 (3834/3)

RECEIVED
98 OCT 19 PM 2:22
GROUP 2700

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANTS: Carter et al.

SERIAL NO.: 08/754,481 EXAMINER: David Ransom

FILED: November 22, 1996 GROUP NO.: 2752

TITLE: Shared Memory Computer Networks

RESPONSE UNDER 37 C.F.R. 1.116
EXPEDITED PROCEDURE
EXAMINING GROUP NO. 2752

Box AF
Assistant Commissioner for Patents
Washington, D.C. 20231

Okay to Enter
B
10-26-98

AMENDMENT AFTER FINAL REJECTION

In response to the Final Office Action mailed from the Patent Office on September 29, 1998, please amend this application as indicated below and consider the remarks that follow.

In the claims:

Please cancel claims 2, 3, 4, 8, 9, 10, 12, 13, 29, 30, and 33, without prejudice.

Please amend claims 1, 5, 6, 7, 11, 14, 15, 17, 18, 20, 23, and 31, and add new claim 34, as follows.

B1

1. (Twice Amended) A computer system having a shared addressable memory space, comprising

a data network for carrying data signals representative of computer readable information, and

[a persistent memory device, coupled to said data network, and having persistent storage for data signals,]

a plurality of computers, each of said plurality of computers sharing the shared addressable memory space and including

an interface, coupled to said data network, for accessing said data network to exchange data signals therewith,

a local volatile memory device coupled to said computer and having volatile storage for data signals,

a local persistent memory device coupled to said computer and having persistent storage for data signals, and

B1

a shared memory subsystem for mapping a portion of said shared addressable memory space to a portion or the whole of said persistent storage and said volatile storage to provide thereby addressable persistent and volatile storage for data signals accessible by each of the plural computers, said shared memory subsystem including

a distributor for mapping portions of said addressable memory space across said plurality of local persistent memory devices, to distribute said addressable memory space across said plurality of local persistent memory devices, and

a disk directory manager for tracking said mapped portions of said addressable memory space to provide information representative of which of said local persistent memory devices has which of said portions of said addressable memory space mapped thereon.

~~8.~~ 2. (Amended) A computer system according to claim [2] 1 further comprising

a cache system for operating one of said local persistent memory devices as a cache memory for cache storing data signals associated with recently accessed portions of said addressable memory space.

B2

~~6.~~ 3. (Amended) A computer system according to claim [2] 1 further comprising

a migration controller for selectively moving portions of said addressable memory space between said local persistent memory devices of said plurality of computers.

B2

4 ~~7.~~ (Amended) A computer system according to claim [2] 1 further comprising

a replication controller for generating a copy of a portion of said addressable memory space maintained in said local persistent memory device of a first one of said computers and for storing said copy in said local persistent memory device of a second one of said computers.

B3

5 ~~11.~~ (Amended) A computer system according to claim 1 [10] further comprising

a RAM directory manager for tracking said mapped portions of said addressable memory space to provide information representative of which of said local volatile memory devices has which of said portions of said addressable memory space mapped thereon.

6 ~~14.~~ (Amended) A computer system according to claim 1 [9] further comprising

a RAM cache system for operating one of said local volatile memory devices as a cache memory for cache storing data signals associated with recently accessed portions of said addressable memory space.

B4

7 ~~15.~~ (Amended) A computer system according to claim 1 [9] further comprising

a paging element for remapping a portion of said addressable memory space between one of said local volatile memory devices and one of said local persistent memory devices.

B5

9 ~~17.~~ (Amended) A computer system according to claim [9] 1 further comprising

a migration controller for moving portions of addressable memory space between said local volatile memory devices of said [plural] plurality of computers.

10
~~18~~. (Amended) A computer system according to claim [9] 1 further comprising

B5 a hierarchy manager for organizing said plurality of computers into a set of hierarchical groups wherein each group includes at least one of said plurality computers.

12
~~20~~. (Amended) A computer system according to claim [9] 1 further comprising

B6 a coherent replication controller for generating a coherent copy of a portion of addressable memory space.

16
~~23~~. (Amended) A computer system according to claim [3] 1 further comprising

B7 a distributed directory manager for storing within said distributed memory space, a directory signal representative of a storage location of a portion of said addressable memory space.

21
~~31~~. (Amended) A computer system according to claim [29] 1 further comprising,

B8 a page generator for generating a directory page that carries information representative of a location monitor that tracks a data storage location, to provide a directory structure for tracking homeless data.

23
~~34~~. A computer system according to claim 1 further comprising

B9 a distributed directory manager for storing a directory signal representative of a portion of said addressable memory space.--

4-B

REMARKS

Extension

Please consider this a conditional petition and request for any extension needed to have this communication entered and considered, and please use our Deposit Account No. 20-0531 for any required fees not enclosed.

Claims

Applicants have canceled independent claims 29 and 33, and dependent claims 2, 3, 4, 8, 9, 10, 12, 13, and 30 without prejudice. Applicants have amended independent claim 1, and dependent claims 5, 6, 7, 11, 14, 15, 17, 18, 20, 23, and 31. Applicants have added dependent claim 34. Independent claim 1, and dependent claims 5-7, 11, 14-28, 31, 32, and 34 are now pending in this application.

Applicants' representatives thank the Examiner for the telephone conferences on Oct. 6, 1998, and Oct. 13, 1998, during which the claims and some of the references of record were discussed.

Applicants submit that the claims as amended are in condition for allowance. In general, applicants have amended claim 1 to include the subject matter of claims 2, 3, 4, 8, and 9. Amended claim 1 recites "a local persistent memory device" and "a local volatile memory device" associated with each of the computers coupled to the network. Portions of the addressable memory space are mapped across these various local volatile and persistent memory devices. Applicants submit that none of the relied-upon references teaches or suggests local volatile memory devices (e.g., RAM associated with each networked computer) or persistent memory devices (e.g., hard disks associated with each networked computer), having portions of a shared addressable memory space mapped thereon.

Conclusion

In view of the foregoing, applicants respectfully request reconsideration, withdrawal of all objections and rejections, and allowance of all pending claims in due course.

Respectfully submitted,

Date: October 15, 1998

Ira V. Heffan
Attorney for Applicants
Reg. No. 41,059

Testa, Hurwitz & Thibeault, LLP
High Street Tower
125 High Street
Boston, MA 02110
Tel.: (617) 248-7176
Fax: (617) 248-7100
Email: heffan@tht.com

A667879

McGRAW-HILL DICTIONARY OF SCIENTIFIC AND TECHNICAL TERMS

Fifth Edition

Sybil P. Parker
Editor in Chief

McGraw-Hill, Inc.
New York San Francisco Washington, D.C.
Auckland Bogotá Caracas Lisbon London Madrid Mexico City Milan
Montreal New Delhi San Juan Singapore Sydney Tokyo Toronto

On the cover: Photomicrograph of crystals of vitamin B_1.
(Dennis Kunkel, University of Hawaii)

Included in this Dictionary are definitions which have been published previously in the following works: P. B. Jordain, *Condensed Computer Encyclopedia*, Copyright © 1969 by McGraw-Hill, Inc. All rights reserved. J. Markus, *Electronics and Nucleonics Dictionary*, 4th ed., Copyright © 1960, 1966, 1978 by McGraw-Hill, Inc. All rights reserved. J. Quick, *Artists' and Illustrators' Encyclopedia*, Copyright © 1969 by McGraw-Hill, Inc. All rights reserved. *Blakiston's Gould Medical Dictionary*, 3d ed., Copyright © 1956, 1972 by McGraw-Hill, Inc. All rights reserved. T. Baumeister and L. S. Marks, eds., *Standard Handbook for Mechanical Engineers*, 7th ed., Copyright © 1958, 1967 by McGraw-Hill, Inc. All rights reserved.

In addition, material has been drawn from the following references: R. E. Huschke, *Glossary of Meteorology*, American Meteorological Society, 1959; *U.S. Air Force Glossary of Standardized Terms*, AF Manual 11-1, vol. 1, 1972; *Communications-Electronics Terminology*, AF Manual 11-1, vol. 3, 1970; W. H. Allen, ed., *Dictionary of Technical Terms for Aerospace Use*, 1st ed., National Aeronautics and Space Administration, 1965; J. M. Gilliland, *Solar-Terrestrial Physics: A Glossary of Terms and Abbreviations*, Royal Aircraft Establishment Technical Report 67158, 1967; *Glossary of Air Traffic Control Terms*, Federal Aviation Agency; *A Glossary of Range Terminology, White Sands Missile Range, New Mexico*, National Bureau of Standards, AD 467-424; *A DOD Glossary of Mapping, Charting and Geodetic Terms*, 1st ed., Department of Defense, 1967; P. W. Thrush, comp. and ed., *A Dictionary of Mining, Mineral, and Related Terms*, Bureau of Mines, 1968; *Nuclear Terms: A Glossary*, 2d ed., Atomic Energy Commission; F. Casey, ed., *Compilation of Terms in Information Sciences Technology*, Federal Council for Science and Technology, 1970; *Glossary of Stinfo Terminology*, Office of Aerospace Research, U.S. Air Force, 1963; *Naval Dictionary of Electronic, Technical, and Imperative Terms*, Bureau of Naval Personnel, 1962; *ADP Glossary*, Department of the Navy, NAVSO P-3097.

McGRAW-HILL DICTIONARY OF SCIENTIFIC AND TECHNICAL TERMS, Fifth Edition

3 4 5 6 7 8 9 0 DOW/DOW 9 9 8 7 6 5

ISBN 0-07-042333-4

Library of Congress Cataloging-in-Publication Data

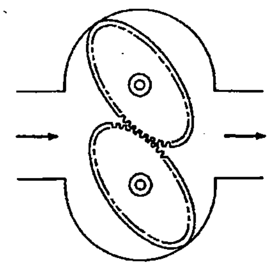
McGraw-Hill dictionary of scientific and technical terms /
Sybil P. Parker, editor in chief.—5th ed.
p. cm.
ISBN 0-07-042333-4
1. Science—Dictionaries. 2. Technology—Dictionaries.
I. Parker, Sybil P.
Q123.M34 1993
503—dc20 93-34772
CIP

INTERNATIONAL EDITION

When ordering this title, use ISBN 0-07-113584-7.

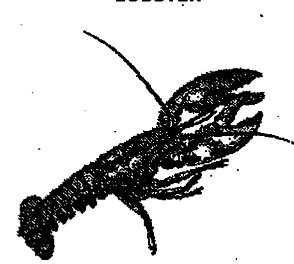
lobectomy [MED] Surgical removal of a lobe of an organ, particularly of a lung. { lō'bek·tə·mē }
lobed impeller meter [ENG] A type of positive displacement meter in which a fluid stream is separated into discrete quantities by rotating, meshing impellers driven by interlocking gears. { lōbd im'pel·ər ,mēd·ər }
lobefin fish [VERT ZOO] The common name for members composing the subclass Crossopterygii. { 'lōb,fin ,fish }
lobe-half-power width [ELECTROMAG] In a plane containing the direction of the maximum energy of a lobe, the angle between the two directions in that plane about the maximum in which the radiation intensity is one-half the maximum value of the lobe. { ¦lōb ¦haf ¦paů·ər ,width }
lobeline [PHARM] $C_{22}H_{27}NO_2$ A crystalline compound isolated from the herb and seeds of Indian tobacco (*Labelia inflata*); melting point is 130–131°C; soluble in hot alcohol, chloroform, and benzene; used in medicine as a respiratory stimulant. { 'lō·bə,lēn }
lobe penetration [ELECTROMAG] Penetration of the radar coverage of a station which is not limited by pulse repetition frequency, scope limitations, or the screening angle at the azimuth of penetration. { 'lōb ,pen·ə'trā·shən }
lobe switching *See* beam switching. { 'lōb ,swich·iŋ }
lobing [ELECTROMAG] Formation of maxima and minima at various angles of the vertical plane antenna pattern by the reflection of energy from the surface surrounding the radar antenna; these reflections reinforce the main beam at some angles and detract from it at other angles, producing fingers of energy. { 'lōb·iŋ }
loblolly pine [BOT] *Pinus taeda.* A hard yellow pine of the central and southeastern United States having a reddish-brown fissured bark, needles in groups of three, and a full bushy top. { 'läb,läl·ē 'pīn }
lobopodia [INV ZOO] Broad, thick pseudopodia. { ,lō·bə'päd·ē·ə }
Lobosia [INV ZOO] A subclass of the protozoan class Rhizopodea generally characterized by lobopodia. { lō'bō·sē·ə }
lobotomy [MED] An operative section of the fibers between the frontal lobes of the brain. Also known as leukotomy; prefrontal lobotomy. { lō'bäd·ə·mē }
Lobry de Bruyn–Ekenstein transformation [ORG CHEM] The change in which an aldose sugar treated with dilute alkali results in a mixture of an epimeric pair and 2-keto-hexose due to the production of enolic forms in the presence of hydroxyl ions, followed by a rearrangement. { lō¦brē·də¦brīn 'ā·kən,shtīn ,trans·fər,mā·shən }
lobster [INV ZOO] The common name for several bottom-dwelling decapod crustaceans making up the family Homaridae which are commercially important as a food item. { 'läb·stər }
lobular pneumonia *See* bronchopneumonia. { 'läb·yə·lər nů'mō·nyə }
lobule [BIOL] **1.** A small lobe. **2.** A division of a lobe. { 'läb·yül }
local action [ELEC] **1.** Internal losses of a battery caused by chemical reactions producing local currents between different parts of a plate. **2.** Quantitatively, the percentage loss per month in the capacity of a battery on open circuit, or the amount of current needed to keep the battery fully charged. [MET] Electrochemical corrosion resulting from the action of local cells. { 'lō·kəl 'ak·shən }
local algebra [MATH] An algebra A over a field F which is the sum of the radical of A and the subalgebra consisting of products of elements of F with the multiplicative identity of A. { ¦lō·kəl 'al·jə·brə }
local anesthetic [PHARM] A drug which induces loss of sensation only in the region to which it is applied. { 'lō·kəl ,an·əs'thed·ik }
local angular momentum [METEOROL] Angular momentum about an arbitrarily located vertical axis which is fixed with respect to the earth. { 'lō·kəl 'aŋ·gyə·lər mə'men·təm }
local apparent noon [ASTRON] Twelve o'clock local apparent time, or the instant the apparent sun is over the upper branch of the local meridian. { 'lō·kəl ə¦par·ənt ¦nün }
local apparent time [ASTRON] The arc of the celestial equator, or the angle at the celestial pole, between the lower branch of the local celestial meridian and the hour circle of the apparent or true sun, measured westward from the lower branch of the local celestial meridian through 24 hours. { 'lō·kəl ə¦par·ənt ¦tīm }
local-area network [COMPUT SCI] A communications network connecting various hardware devices together within a building by means of a continuous cable or an in-house voice-data telephone system. Also known as LAN. { 'lō·kəl¦er·ē·ə 'net,wərk }
local-area underwater navigation system [NAV] A system applied chiefly in the inspection and maintenance of offshore structures, in which time differences between the reception of acoustic signals from a chain of transmitters are used to determine the positions of divers and of crewed and uncrewed underwater vehicles. Abbreviated LAUNS. { 'lō·kəl ¦er·ē·ə ¦ən·dər¦wȯd·ər ,nav·ə'gā·shən ,sis·təm }
local arm *See* Orion arm. { 'lō·kəl 'ärm }
local attraction *See* local magnetic disturbance. { 'lō·kəl ə'trak·shən }
local base [MATH] For a point x in a topological space, a family of neighborhoods of x such that every neighborhood of x contains a member of the family. Also known as base for the neighborhood system. { ¦lō·kəl 'bās }
local base level *See* temporary base level. { 'lō·kəl ¦bās ¦lev·əl }
local battery [ELEC] Battery that actuates the telegraphic station recording instruments, as distinguished from the battery furnishing current to the line. { 'lō·kəl 'bad·ə·rē }
local-battery telephone set [ELECTR] Telephone set for which the transmitter current is supplied from a battery, or other current supply circuit, individual to the telephone set; the signaling current may be supplied from a local hand generator or from a centralized power source. { 'lō·kəl 'bad·ə·rē 'tel·ə,fōn ,set }
local buckling [MECH] Buckling of thin elements of a column section in a series of waves or wrinkles. { 'lō·kəl 'bək·liŋ }
local cable [COMMUN] Handmade cable form for terminations of circuits at the attendant's switchboard, at unit equipment, and other locations, where wiring is run inside the section or unit. { 'lō·kəl 'kā·bəl }
local cell [ELEC] A galvanic cell resulting from differences in potential between adjacent areas on the surface of a metal immersed in an electrolyte. { 'lō·kəl 'sel }
local central office [COMMUN] A telephone central office, which terminates subscriber lines and makes connections with other central offices, usually equipped to serve 10,000 main telephones of its immediate community. { 'lō·kəl ¦sen·trəl 'ȯf·əs }
local change [OCEANOGR] The time rate of change of a scalar quantity (such as temperature, salinity, pressure, or oxygen content) in a fixed locality. { 'lō·kəl 'chānj }
local circuit [COMMUN] Circuit to a main or auxiliary circuit which can be made available at any station or patched from point to point through one or more stations. { 'lō·kəl 'sər·kət }
local civil time [ASTRON] United States terminology during 1925–1952 for local mean time. { 'lō·kəl ¦siv·əl ¦tīm }
local cluster of stars *See* local star system. { 'lō·kəl ¦kləs·tər əv ¦stärz }
local coefficient [MATH] By using fiber bundles where the fiber is a group, one may generalize cohomology theory for spaces; one uses such bundles as the algebraic base for such a theory and calls the bundle a system of local coefficients. { 'lō·kəl ,kō·i'fish·ənt }
local coefficient of heat transfer [THERMO] The heat transfer coefficient at a particular point on a surface, equal to the amount of heat transferred to an infinitesimal area of the surface at the point by a fluid passing over it, divided by the product of this area and the difference between the temperatures of the surface and the fluid. { 'lō·kəl ,kō·i'fish·ənt əv 'hēt ,tranz·fər }
local control [COMMUN] System or method of radio-transmitter control whereby the control functions are performed directly at the transmitter. { 'lō·kəl kən'trōl }
local controller *See* first-level controller. { 'lō·kəl kən'trōl·ər }
local coordinate system [MATH] The coordinate system about a point which is induced when the global space is locally euclidean. { 'lō·kəl kō'ȯrd·ən·ət ,sis·təm }
local derivative [FL MECH] The rate of change of a quantity f with respect to time at a fixed point of a fluid, $\partial f/\partial t$; it is related to the individual derivative df/dt through the expression $\partial f/\partial t = df/dt - V \cdot \nabla f$, where f is a thermodynamic property $f(x,y,z,t)$ of the fluid, V the vector velocity of the fluid, and ∇ the del operator. { 'lō·kəl də'riv·əd·iv }
local device [COMPUT SCI] Peripheral equipment that is linked

LOBED IMPELLER METER



Meshing impellers in the lobed impeller meter. Arrows show direction of fluid flow.

LOBSTER



The lobster, showing the stout pincers on the front legs which are used to crush prey.

directly to a computer or other supporting equipment, without an intervening communications channel. { 'lō·kəl di'vīs }

local distortion [MATH] The absolute value of the derivative of an analytic function at a given point. { 'lō·kəl di'stȯr·shən }

local exchange *See* exchange. { 'lō·kəl iks'chānj }

local extra observation [METEOROL] An aviation weather observation taken at specified intervals, usually every 15 minutes, when there are impending aircraft operations and when weather conditions are below certain operational weather limits; the observation includes ceiling, sky condition, visibility, atmospheric phenomena, and pertinent remarks. { 'lō·kəl 'ek·strə ,äb·zər'vā·shən }

local first selector [COMMUN] The second portion of a line that connects to a calling line, through a line primary switch, to a local second selector and special service second selector, and returns a dial tone to the calling subscribers. { 'lō·kəl ¦fərst si¦lek·tər }

local forecast [METEOROL] Generally, any weather forecast of conditions over a relatively limited area, such as a city or airport. { 'lō·kəl 'fȯr,kast }

Local Group [ASTRON] A group of at least 20 known galaxies in the vicinity of the sun; the Andromeda Spiral is the largest of the group, and the Milky Way Galaxy is the second largest. { 'lō·kəl 'grüp }

local hidden-variable theory *See* hidden-variable theory of the second kind. { 'lō·kəl ¦hid·ən 'ver·ē·ə·bəl ,thē·ə·rē }

local hour angle [ASTRON] Angular distance west of the local celestial meridian. { 'lō·kəl 'aůr ,aŋ·gəl }

local immunity [IMMUNOL] Immunity localized in a specific tissue or region of the body. { 'lō·kəl i'myü·nəd·ē }

local inflow [HYD] The water that enters a stream between two stream-gaging stations. { 'lō·kəl 'in,flō }

local invariance [PHYS] The property of physical laws which remain unchanged under a specified set of symmetry transformations even when these transformations are chosen independently at every point of space and time. { 'lō·kəl in'ver·ē·əns }

locality [PHYS] The condition that two events at spatially separated locations are entirely independent of each other, provided that the time interval between the events is less than that required for a light signal to travel from one location to the other. { lō'kal·əd·ē }

localization [COMPUT SCI] Imposing some physical order upon a set of objects, so that a given object has a greater probability of being in some particular regions of space than in others. { ,lō·kə·lə'zā·shən }

localized state [QUANT MECH] A state of motion in which an electron may be found anywhere within a region of a material of linear extent smaller than that of the material. { 'lō·kə,līzd 'stāt }

localized vector [MECH] A vector whose line of application or point of application is prescribed, in addition to its direction. { 'lō·kə,līzd 'vek·tər }

localizer [NAV] A directional radio beacon to provide aircraft with signals for lateral guidance with respect to the runway centerline. { 'lō·kə,līz·ər }

local level [NAV] The plane normal to the local vertical. { 'lō·kəl 'lev·əl }

local line [COMMUN] A telephone line terminating at the local central office. { 'lō·kəl 'līn }

local loop *See* home loop. { 'lō·kəl 'lüp }

local lunar time [ASTRON] The arc of the celestial equator, or the angle at the celestial pole, between the lower branch of the local celestial meridian and the hour circle of the moon, measured westward from the lower branch of the local celestial meridian through 24 hours; local hour angle of the moon, expressed in time units, plus 12 hours; local lunar time at the Greenwich meridian is called Greenwich lunar time. { 'lō·kəl ¦lü·nər ¦tīm }

locally arcwise connected topological space [MATH] A topological space in which every point has an arcwise connected neighborhood, that is, an open set any two points of which can be joined by an arc. { 'lō·kə·lē 'ärk,wīz kə,nek·təd ¦täp·ə¦läj·ə·kəl ¦spās }

locally compact topological space [MATH] A topological space in which every point lies in a compact neighborhood. { 'lō·kə·lē kəm'pakt ¦täp·ə¦läj·ə·kəl ¦spās }

locally connected topological space [MATH] A topological space in which every point has a connected neighborhood. { 'lō·kə·lē kə'nek·təd ¦täp·ə¦läj·ə·kəl ¦spās }

locally convex space [MATH] A Hausdorff topological vector space E such that every neighborhood of any point x belonging to E contains a convex neighborhood of x. { 'lō·kə·lē 'kän,veks ,spās }

locally euclidean topological space [MATH] A topological space in which every point has a neighborhood which is homeomorphic to a euclidean space. { 'lō·kə·lē yü'klid·ē·ən ¦täp·ə¦läj·ə·kəl ¦spās }

locally finite family of sets [MATH] A family of subsets of a topological space such that each point of the topological space has a neighborhood that intersects only a finite number of these subsets. { ¦lō·kə·lē ¦fī,nīt 'fam·lē əv 'sets }

locally integrable function [MATH] A function is said to be locally integrable on an open set S in n-dimensional euclidean space if it is defined almost everywhere in S and has a finite integral on compact subsets S. { ¦lō·kə·lē ¦int·ə·grə·bəl 'fəŋk·shən }

locally one to one [MATH] A function is locally one to one if it is one to one in some neighborhood of each point. { 'lō·kə·lē 'wən tə 'wən }

locally trivial bundle [MATH] A bundle for which each point in the base has a neighborhood U whose inverse image under the projection map is isomorphic to a cartesian product of U with a space isomorphic to the fibers of the bundle. { ¦lō·kə·lē ¦triv·ē·əl 'bən·dəl }

local Mach number [AERO ENG] The Mach number of an isolated section of an airplane or its airframe. { 'lō·kəl 'mäk ,nəm·bər }

local magnetic disturbance [GEOPHYS] An anomaly of the magnetic field of the earth, extending over a relatively small area, due to local magnetic influences. Also known as local attraction. { 'lō·kəl mag'ned·ik di'stər·bəns }

local maximum [MATH] A local maximum of a function f is a value $f(c)$ of f where $f(x) \leq f(c)$ for all x in some neighborhood of c; if $f(c)$ is a local maximum, f is said to have a local maximum at c. { 'lō·kəl 'mak·sə·məm }

local mean noon [ASTRON] Twelve o'clock local mean time, or the instant the mean sun is over the upper branch of the local meridian; local mean noon at the Greenwich meridian is called Greenwich mean noon. { 'lō·kəl ¦mēn ¦nün }

local mean time [ASTRON] The arc of the celestial equator, or the angle at the celestial pole, between the lower branch of the local celestial meridian and the hour circle of the mean sun, measured westward from the lower branch of the local celestial meridian through 24 hours. { 'lō·kəl ¦mēn ¦tīm }

local meridian [ASTRON] The meridian through any particular position which serves as the reference for local time. { 'lō·kəl mə'rid·ē·ən }

local minimum [MATH] A local minimum of a function f is a value $f(c)$ of f where $f(x) \geq f(c)$ for all x in some neighborhood of c; if $f(c)$ is a local minimum, f is said to have a local minimum at c. { 'lō·kəl 'min·ə·məm }

local networking [CONT SYS] The system of communication linking together the components of a single robot. { 'lō·kəl 'net,wərk·iŋ }

local noon [ASTRON] Noon at the local meridian. { 'lō·kəl 'nün }

local oscillator [ELECTR] The oscillator in a superheterodyne receiver, whose output is mixed with the incoming modulated radio-frequency carrier signal in the mixer to give the frequency conversions needed to produce the intermediate-frequency signal. { 'lō·kəl 'äs·ə,lād·ər }

local-oscillator injection [ELECTR] Adjustment used to vary the magnitude of the local oscillator signal that is coupled into the mixer. { 'lō·kəl 'äs·ə,lād·ər in¦jek·shən }

local-oscillator radiation [ELECTR] Radiation of the fundamental or harmonics of the local oscillator of a superheterodyne receiver. { 'lō·kəl 'äs·ə,lād·ər ,rād·ē'ā·shən }

local peat [GEOL] Peat formed by groundwater. Also known as basin peat. { 'lō·kəl ,pēt }

local preheating [MET] The heating of a specific portion of a material or structure prior to the performance of a joining or fabrication process. { 'lō·kəl prē'hēd·iŋ }

local procurement [ORD] **1.** Procurement of supplies or equipment in the continental United States by other than a centralized purchasing office, such as purchase by an installation of supplies and equipment for use of that installation. **2.** Procurement of supplies or equipment for its own use in an area outside the United States by a United States military command located in that area. { 'lō·kəl prə'kyůr·mənt }