(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 February 2007 (01.02.2007)

PCT

(10) International Publication Number
WO 2007/014079 A1

(51) International Patent Classification:
H04L 29/08 (2006.01) *G06F 3/06* (2006.01)
H04L 29/14 (2006.01)

(21) International Application Number:
PCT/US2006/028494

(22) International Filing Date: 20 July 2006 (20.07.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/187,781 22 July 2005 (22.07.2005) US

(71) Applicant *(for all designated States except US)*: **NETWORK APPLIANCE, INC.** [—/US]; 495 East Java Drive, Sunnyvale, CA 94089 (US).

(72) Inventors: **ERASANI, Pranoop**; 525 E. Maude Ave, #22, Sunnyvale, CA 94085 (US). **DANIEL, Stephen**; 5 Fallen Oak Court, Durham, NC 27713 (US). **RICHART, Joseph**; 10520 Byrum Woods Drive, Raleigh, NC 27613 (US). **HAYNES, Thomas**; 9110 E 66th Street, Tulsa, OK 74133 (US).

(74) Agents: **BARBAS, Charles, J.** et al.; Cesari and McKenna, LLP, 88 Black Falcon Avenue, Boston, MA 02210 (US).

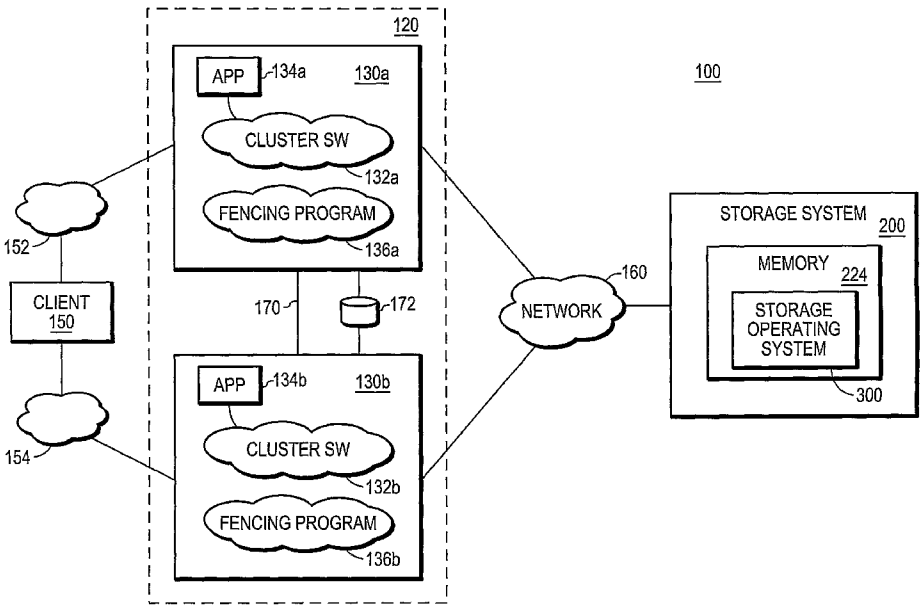
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: CLIENT FAILURE FENCING MECHANISM FOR FENCING NETWORK FILE SYSTEM DATA IN A HOST-CLUSTER ENVIRONMENT



(57) Abstract: A method and system performs a fencing technique in a host cluster storage environment. The fence program executes on each cluster member in the cluster, and the cluster is coupled to a storage system by a network. When a cluster member fails or cluster membership changes, the fence program is invoked and a host fencing API message is sent via the network to the storage system. The storage system in turn modifies export lists to restrict further access by the failed cluster node to otherwise fence the failed cluster node off from that storage system or from certain directories within that storage system.

WO 2007/014079 A1

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

CLIENT FAILURE FENCING MECHANISM FOR FENCING NETWORK FILE SYSTEM DATA IN A HOST-CLUSTER ENVIRONMENT

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to data storage systems, and more particularly, to failure fencing in networked data storage systems.

Background Information

A storage system is a computer that provides storage service relating to the organization of information on writable persistent storage devices, such as memories, tapes or disks. The storage system is commonly deployed within a storage area network (SAN) or a networked storage environment. When used within a networked environment, the storage system may be embodied as a storage system including an operating system that implements a file system to logically organize the information as a hierarchical structure of directories and files on, e.g. the disks. Each "on-disk" file may be implemented as a set of data structures, e.g., disk blocks, configured to store information, such as the actual data for the file. A directory, on the other hand, may be implemented as a specially formatted file in which information about other files and directories are stored.

In the client/server model, the client may comprise an application executing on a computer that "connects" to a storage system over a computer network, such as a point-to-point link, shared local area network, wide area network or virtual private network implemented over a public network, such as the Internet. NAS systems generally utilize file-based access protocols; therefore, each client may request the services of the storage system by issuing file system protocol messages (in the form of packets) to the file system over the network. By supporting a plurality of file system protocols, such as the conventional Common Internet File System (CIFS), the Network File System

(NFS) and the Direct Access File System (DAFS) protocols, the utility of the storage system may be enhanced for networking clients.

A SAN is a high-speed network that enables establishment of direct connections between a storage system and its storage devices. The SAN may thus be viewed as an extension to a storage bus and, as such, an operating system of the storage system (a storage operating system, as hereinafter defined) enables access to stored information using block-based access protocols over the "extended bus." In this context, the extended bus is typically embodied as Fibre Channel (FC) or Ethernet media (i.e., network) adapted to operate with block access protocols, such as Small Computer Systems Interface (SCSI) protocol encapsulation over FC or TCP/IP/Ethernet.

A SAN arrangement or deployment allows decoupling of storage from the storage system, such as an application server, and placing of that storage on a network. However, the SAN storage system typically manages specifically assigned storage resources. Although storage can be grouped (or pooled) into zones (e.g., through conventional logical unit number or "lun" zoning, masking and management techniques), the storage devices are still pre-assigned by a user that has administrative privileges, (e.g., a storage system administrator, as defined hereinafter) to the storage system.

Thus, the storage system, as used herein, may operate in any type of configuration including a NAS arrangement, a SAN arrangement, or a hybrid storage system that incorporates both NAS and SAN aspects of storage.

Access to disks by the storage system is governed by an associated "storage operating system," which generally refers to the computer-executable code operable on a storage system that manages data access, and may implement file system semantics. In this sense, the NetApp® Data ONTAP™ operating system available from Network Appliance, Inc., of Sunnyvale, California that implements the Write Anywhere File Layout (WAFL™) file system is an example of such a storage operating system implemented as a microkernel. The storage operating system can also be implemented as an application program operating over a general-purpose operating system, such as UNIX® or Windows NT®, or as a general-purpose operating system with configurable functionality, which is configured for storage applications as described herein.

In many high availability server environments, clients requesting services from applications whose data is stored on a storage system are typically served by coupled server nodes that are clustered into one or more groups. Examples of these node groups are Unix®-based host-clustering products. The nodes typically share access to the data stored on the storage system from a direct access storage/storage area network (DAS/SAN). Typically, there is a communication link configured to transport signals, such as a heartbeat, between nodes such that during normal operations, each node has notice that the other nodes are in operation.

In the case of a two-node cluster, for example, the absence of a heartbeat signal indicates to a node that there has been a failure of some kind. However, if both nodes are still in normal operating condition, the absent heartbeat signal may be the result of interconnect failure. In that case, the nodes are not in communication with one another and, typically, only one node should be allowed access to the shared storage system. In addition, a node that is not properly functioning may need to have its access to the data of the storage system restricted.

But, in a networked storage device, access to a storage system is typically through a conventional file system protocol, such as the network file system (NFS) protocol. Thus, any techniques that are used to restrict access to data with respect to a NAS device would need to incorporate the NFS protocol. Moreover, the NFS protocol does not support SCSI reservations, and thus prior techniques which relied on SCSI reservations would not be suitable for an environment in which access to the storage system is through NFS. Thus, a network accessed storage system does not fit into this traditionally disk-based host cluster model.

There remains a need, therefore, for a host cluster environment that includes failure fencing but that can support NFS data access from a networked clustered environment that is interfaced with the storage system.

There remains a further need for performing fencing operations without requiring a traditional SCSI-based reservation mechanism when a cluster does not predominantly share data from a directly attached disk, but instead functions in a networked storage environment.

In addition, there remains a need for a simple user interface adapted to perform fencing operations for the cluster, which can be easily downloaded into the host clustering framework.

SUMMARY OF THE INVENTION

The present invention overcomes the disadvantages of the prior art by providing a clustered storage environment that includes a fencing mechanism that supports a file system protocol, such as the network file system (NFS) protocol, as a shared data source in the clustered environment. More specifically, a plurality of nodes interconnected as a cluster is configured to utilize the storage services provided by an associated networked storage system. Each node in the cluster is an identically configured redundant node that may be utilized in the case of failover or for load balancing with respect to the other nodes in the cluster. The nodes are hereinafter referred to as a "cluster members." Each cluster member is supervised and controlled by cluster software executing on one or more processors in the cluster member. In accordance with illustrative embodiments of the invention, in addition to the cluster software, a novel fencing program resides on each cluster member and implements program instructions for fencing operations for the clustered storage environment.

The clusters are coupled with the associated storage system through an appropriate network such as a wide area network, a virtual private network implemented over a public network (Internet), or a shared local area network. For a networked environment, the clients are typically configured to access information stored on the storage system as directories and files. The cluster members typically communicate with the storage system over a network by exchanging discreet frames or packets of data according to predefined protocols, such as the NFS over Transmission Control Protocol/Internet Protocol (TCP/IP).

In accordance with illustrative embodiments of the invention, a change in cluster membership is detected by the cluster software and, in response, the fencing program is initiated. The fencing program generates host fencing application programming interface (API) messages that notify a NFS server of the storage system to instruct an export module to change access permission rights of one or more cluster members

affected by the change (i.e., the "target" cluster members) such that write access by the target cluster members is restricted from certain data in the form of files, directories, or the storage system itself. Details about the API by which the export module is instructed to change access permission rights are provided in commonly-owned United States Patent Application No. [112056-0237; P01–2252], of Haynes, et al., for AN API FOR FENCING CLUSTER HOSTS VIA EXPORT ACCESS RIGHTS, filed on even date herewith, which is hereby incorporated by reference as though fully set forth herein.

Cluster membership can change when there is a failure of one or more of the cluster members, or upon the addition of a new cluster member, or upon a failure of the communication link between cluster members, for example. However, a change in cluster membership and thus the need for fencing, and/or unfencing, is determined by the cluster software, which may in turn be controlled by a quorum device or other software arrangement controlling the cluster, all of which are hereinafter generally referred to as the cluster infrastructure.

More specifically, when a fencing operation is to be performed, the fencing program, either by an administrator through a user interface, or automatically, generates and transmits an API message via a predetermined protocol, such as the HyperText Transfer Protocol ("HTTP"), over the network to the storage system. In particular, if a member of the cluster is to be fenced off, one of the surviving cluster members transmits the API message using HTTP, which is received by a HTTP module of the storage system. The message is parsed and sent to an NFS subsystem of the storage system.

The NFS server has associated therewith certain export lists that provide rules relating to access to the NFS data served by the storage system. More specifically, an access rule provides the authentication level required for read access, read/write access and/or administrative (root) access by a client identified by a particular network address. For example, when processing an incoming data access request directed to the storage system, a search is performed through an ordered export list of rules in a rule set to find a match between the network address of the client and a specific rule in the export list. If no match is found, then the data access request is denied. If a match is found, then the matching rule indicates the permission rights that the particular client

has, *i.e.*, read-write access, or read-only access to the specified data container, which may be a file, a directory or a storage system.

According to illustrative embodiments of the present invention, the API message issued by the fencing program executing on the cluster member not affected by the change in cluster membership (i.e., the "surviving" cluster member) notifies the NFS server in the storage system that a modification must be made in one of the export lists such that a target cluster member, for example, cannot write to any of the disks in the storage system, thereby fencing off that member from that data. The fenced cluster member may continue to have read access so that when it comes back online, the fenced member can read relevant storage devices in the storage system to bring the node up to date with the current state of the data in the storage system. After the fencing is no longer needed i.e when the node becomes member of the cluster (either through administrative action or the cluster framework implementation), an "unfence" operation can be performed to allow access to the relevant data.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and further advantages of the invention may be better understood by reference to the following description in conjunction with the accompanying drawings, in which like reference numerals indicate identical or functionally similar elements:

Fig. 1 is a schematic block diagram of a storage environment, which includes an interconnected set of cluster members coupled by a network to a storage system in which illustrative embodiments of the present may be advantageously employed;

Fig. 2 is a schematic block diagram of the storage system that may be advantageously used in accordance with an embodiment of the present invention;

Fig. 3 is a schematic block diagram of a storage operating system that may be advantageously used in accordance with an embodiment of the present invention;

Fig. 4 is a schematic block diagram illustrating an NFS server, which can be advantageously employed in the storage system in accordance with an embodiment of the present invention;

Fig. 5 is a schematic block diagram illustrating the format of an exemplary export list in accordance with an embodiment of the present invention;

Fig. 6A is an exemplary block of code representing a configuration file for configuring aspects of the storage system in accordance with illustrative embodiments of the invention;

Fig. 6B is a chart setting forth various details for a host API message in accordance with illustrative embodiment of the invention;

Fig. 7 is a flow chart detailing the steps of a procedure for performing fencing operations in accordance with the present invention; and

Figs. 8A and 8B together form a flow chart detailing the steps of a procedure for configuration, validation and use of the fencing program in connection with the cluster software running on a cluster member in accordance with illustrative embodiments of the present invention.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

A. Cluster Environment

Fig. 1 is a schematic block diagram of a storage environment 100 that includes a cluster 120 having cluster members 130a and 130b, each of which is an identically configured redundant node that utilizes the storage services of an associated storage system 200. For purposes of clarity of illustration, the cluster 120 is depicted as a two-node cluster, however, the architecture of the environment 100 can vary from that shown while remaining within the scope of the present invention.

Cluster members 130a and 130b comprise various functional components that cooperate to provide data from storage devices of the storage system 200 to a client 150. The cluster member 130a includes a plurality of ports that couple the member to the client 150 over a computer network 152. Similarly, the cluster member 130b includes a plurality of ports that couple that member with the client 150 over a computer network 154. In addition, each cluster member 130, for example, has a second set of ports that connect the cluster member to the storage system 200 by way of a network 160. The cluster members 130a and 130b, in the illustrative example, communicate over the network 160 using a Transmission Control Protocol/Internet Protocol (TCP/IP). It should be understood that although networks 152, 154 and 160 are de-

picted in Fig. 1 as individual networks, these networks may in fact comprise a single network or any number of multiple networks, and the cluster members 130a and 130b can be interfaced with one or more of such networks in a variety of configurations while remaining within the scope of the present invention.

In addition to the ports which couple the cluster member 130a to the client 150 and to the network 160, the cluster member 130a also has a number of program modules executing thereon. For example, cluster software 132a performs overall configuration, supervision and control of the operation of the cluster member 130a. An application 134a running on the cluster member 130a communicates with the cluster software to perform the specific function of the application running on the cluster member 130a. This application 134a may be, for example, an Oracle® database application. In accordance with an illustrative embodiment of the invention, the cluster member 130a also includes a fencing program 136a, as described in further detail hereinafter.

Similarly, cluster member 130b includes cluster software 132b which is in communication with an application program 134b. A fencing program 136b, in accordance with the invention, executes on the cluster member 130b. The cluster members 130a and 130b are illustratively coupled by a cluster interconnect 170. In addition, each of the cluster members 130a and 130b may optionally be directly attached to a quorum disk 172 which provides quorum services for the cluster 120 such that if a cluster member detects the absence of a notification, such as a heartbeat, from another cluster member, that cluster member will attempt to assert a claim by, for example, asserting a SCSI-3 persistent reservation, on the quorum device in order to have continued access to the storage device, as described previously herein.

B. Storage System

Fig. 2 is a more detailed schematic block diagram of the storage system 200 that is connected to the network 160 in order to serve data with respect to the cluster 120. The storage system 200 comprises one or more processors 222, a memory 224, a network adapter 225 and a storage adapter 228 interconnected by a system bus 223. The network adapter 225 comprises a plurality of ports adapted to couple the storage system 200 to the cluster 120 by way of the network 160. The storage system 200 also includes a storage operating system 300 that includes a virtualization system including a file system to logically organize the information as a hierarchical structure of named directory, file and virtual disk (vdisk) storage objects on disk drives 230. An example of a storage system that may be advantageously used with the present invention is described in co-pending, commonly assigned United States Patent Application Publication No. 2004/0030668 A1, filed on August 9, 2002, entitled A MULTIPROTOCOL STORAGE APPLIANCE THAT PROVIDES INTEGRATED SUPPORT FOR FILE AND BLOCK ACCESS PROTOCOLS, which application is hereby incorporated by reference as though fully set forth herein.

In the illustrative embodiment, the memory 224 comprises storage locations that are addressable by the processor and adapters for storing software program code and data structures, the latter containing information passed between disk drives and the network during normal runtime operations. The processor and adapters may comprise processing elements and/or logic circuitry configured to execute the software code and manipulate the data structures. The storage operating systems 200, portions of which are typically resident in memory and executed by the processing elements, functionally organizes the storage system by, *inter alia*, invoking storage operations in support of the storage service implemented by the system. It will be apparent to those skilled in the art that other processing and memory means, including various computer readable media, may be used for storing and executing program instructions pertaining to the invention described herein.

Whereas clients of a NAS-based network environment have a storage viewpoint of files, the clients of a SAN-based network environment have a storage viewpoint of blocks or disks. To that end, the storage system 200 presents (exports) disks to SAN

clients through the creation of logical unit numbers (luns) or vdisk objects. A vdisk object (hereinafter "vdisk") is a special file type that is implemented by the virtualization system and translated into an emulated disk as viewed by the SAN clients. The storage system 200 thereafter makes these emulated disks accessible to SAN clients.

The network adapter 225 couples the storage system 200 to a cluster, such as the cluster 120, over point-to-point links, wide area networks, virtual private networks implemented over a public network (Internet) or a shared local area network, as illustrated by the arrow 265. For a NAS-based network environment, the clients are configured to access information stored on the storage system 200 as files. The cluster 120 communicates with the storage system over network 160 by exchanging discrete frames or packets of data according to pre-defined protocols, such as NFS over the TCP/IP protocol.

The cluster 120 may include general-purpose computers configured to execute applications over a variety of operating systems, including the UNIX® and Microsoft® Windows™ operating systems. Client systems generally utilize file-based access protocols when accessing information (in the form of files and directories) over a NAS-based network. Therefore, each client 150 may request the services of the storage system 200 by issuing file access protocol messages (in the form of packets) to the system over the network 160. For example, a cluster member 130a running the Windows operating system may communicate with the storage system 200 using the Common Internet File System (CIFS) protocol over TCP/IP. On the other hand, a cluster member 130b running the UNIX operating system may communicate with the storage system using either the Network File System (NFS) protocol over TCP/IP or the Direct Access File System (DAFS) protocol over a virtual interface (VI) transport in accordance with a remote DMA (RDMA) protocol over TCP/IP. It will be apparent to those skilled in the art that other clients running other types of operating systems may also communicate with the storage system using other file access protocols.

The storage adapter 228 cooperates with the storage operating system 300 executing on the storage system 200 to access information requested by the clients. The information may be stored in the disk drives 230 or other similar media adapted to store information. The storage adapter includes I/O interface circuitry that couples to the

disks over an I/O interconnect arrangement, such as a conventional high-performance, FC serial link topology. The information is retrieved by the storage adapter and, if necessary, processed by the processor 222 (or the adapter 228 itself) prior to being forwarded over the system bus 223 to the network adapters 225 where the information is formatted into packets or messages and returned to the clients.

Storage of information on the storage system 200 is preferably implemented as one or more storage volumes (e.g., VOL1-2 250) that comprise a cluster of physical storage disk drives 230, defining an overall logical arrangement of disk space. The disks within a volume are typically organized as one or more groups of Redundant Array of Independent (or *Inexpensive*) Disks (RAID). RAID implementations enhance the reliability/integrity of data storage through the writing of data "stripes" across a given number of physical disks in the RAID group, and the appropriate storing of redundant information with respect to the striped data. The redundant information enables recovery of data lost when a storage device fails.

Specifically, each volume 250 is constructed from an array of physical disk drives 230 that are organized as RAID groups 240, 242, and 244. The physical disks of each RAID group include those disks configured to store striped data (D) and those configured to store parity (P) for the data, in accordance with an illustrative RAID 4 level configuration. However, other RAID level configurations (e.g. RAID 5) are also contemplated. In the illustrative embodiment, a minimum of one parity disk and one data disk may be employed. However, a typical implementation may include three data and one parity disk per RAID group and at least one RAID group per volume.

To facilitate access to the disk drives 230, the storage operating system 300 implements a write-anywhere file system that cooperates with virtualization modules to provide a system that "virtualizes" the storage space provided by the disk drives 230. The file system logically organizes the information as a hierarchical structure of named directory and file objects (hereinafter "directories" and "files") on the disks. Each "on-disk" file may be implemented as set of disk blocks configured to store information, such as data, whereas the directory may be implemented as a specially formatted file in which names and links to other files and directories are stored. The virtualization system, which is described more fully with reference to Fig. 3, allows the file system to

further logically organize information as a hierarchical structure of named vdisks on the disks, thereby providing an integrated NAS and SAN appliance approach to storage by enabling file-based (NAS) access to the files and directories, while further enabling block-based (SAN) access to the vdisks on a file-based storage platform.

As noted, in the illustrative embodiment, the storage operating system is preferably the NetApp® Data ONTAP™ operating system that implements a Write Anywhere File Layout (WAFL®) file system. However, it is expressly contemplated that any appropriate storage operating system, including a write in-place file system, may be enhanced for use in accordance with the inventive principles described herein. As such, where the term "WAFL" is employed, it should be taken broadly to refer to any storage operating system that is otherwise adaptable to the teachings of this invention.

As used herein, the term "storage operating system" generally refers to the computer-executable code operable on a computer that manages data access and may, in the case of a storage system, implement data access semantics, such as the Data ONTAP storage operating system. The storage operating system can also be implemented as an application program operating over a general-purpose operating system, such as UNIX® or Windows NT®, or as a general-purpose operating system with configurable functionality, which is configured for storage applications as described herein.

In addition, it will be understood to those skilled in the art that the inventive technique described herein may apply to any type of special-purpose (e.g., storage serving appliance) or general-purpose computer, including a standalone computer or portion thereof, embodied as or including a storage system. Moreover, the teachings of this invention can be adapted to a variety of storage system architectures including, but not limited to, a network-attached storage environment, a storage area network and disk assembly directly-attached to a client or host computer. The term "storage system" should therefore be taken broadly to include such arrangements in addition to any subsystems configured to perform a storage function and associated with other equipment or system.

C. Storage Operating System

Fig. 3 is a schematic block diagram of a storage operating system 300 that may be advantageously used with embodiments of the present invention. The storage operating system 300 comprises a series of software layers organized to form an integrated network protocol stack or, more generally, a multi-protocol engine that provides data paths for clients to access information stored on the storage system 200 using block and file access protocols. The protocol stack includes a media access layer 310 of network drivers (e.g., gigabit Ethernet drivers) that interfaces to network protocol layers, such as the IP layer 312 and its supporting transport mechanisms, the TCP layer 314. A file system protocol layer provides multi-protocol file access and, to that end, includes support for the NFS protocol 320, the CIFS protocol 322 and the Hypertext Transfer Protocol (HTTP) protocol 324. As described further herein, the NFS protocol layer 320 is illustratively embodied as an NFS server.

An iSCSI driver layer 328 provides block protocol access over the TCP/IP network protocol layers, while a FC driver layer 330 operates with the network adapter to receive and transmit block access requests and responses to and from the storage system. The FC and iSCSI drivers provide FC-specific and iSCSI-specific access control to the luns (vdisks) and, thus, manage exports of vdisks to either iSCSI or FCP or, alternatively, to both iSCSI and FCP when accessing a single vdisk on the storage system. In addition, the storage operating system includes a disk storage layer 340 that implements a disk storage protocol, such as a RAID protocol, and a disk driver layer 350 that implements a disk access protocol such as, e.g., a SCSI protocol.

Bridging the disk software layers with the integrated network protocol stack layers is a virtualization system 370. The virtualization system 370 includes a file system 374 interacting with virtualization modules illustratively embodied as, e.g., vdisk module 376 and SCSI target module 378. It should be noted that the vdisk module 376, the file system 374 and SCSI target module 378 can be implemented in software, hardware, firmware, or a combination thereof. The vdisk module 376 communicates with the file system 374 to enable access by administrative interfaces in response to a storage system administrator issuing commands to the storage system 200. In essence, the vdisk module 376 manages SAN deployments by, among other things, implementing a comprehensive set of vdisk (lun) commands issued by the storage system administrator.

These vdisk commands are converted to primitive file system operations ("primitives") that interact with the file system 374 and the SCSI target module 378 to implement the vdisks.

The SCSI target module 378 initiates emulation of a disk or lun by providing a mapping procedure that translates luns into the special vdisk file types. The SCSI target module is illustratively disposed between the FC and iSCSI drivers 330, 328 and the file system 374 to thereby provide a translation layer of the virtualization system 370 between the SAN block (lun) space and the file system space, where luns are represented as vdisks 376. To that end, the SCSI target module 378 has a set of APIs that are based on the SCSI protocol and that enable a consistent interface to both the iSCSI and FC drivers 328, 330, respectively. By "disposing" SAN virtualization over the file system 374, the storage system 200 reverses the approaches taken by prior systems to thereby provide a single unified storage platform for essentially all storage access protocols.

The file system 374 provides volume management capabilities for use in block-based access to the information stored on the storage devices, such as disks. That is, in addition to providing file system semantics, such as naming of storage objects, the file system 374 provides functions normally associated with a volume manager. These functions include (i) aggregation of the disks, (ii) aggregation of storage bandwidth of the disks, and (iii) reliability guarantees, such as mirroring and/or parity (RAID), to thereby present one or more storage objects layered on the file system.

The file system 374 illustratively implements the WAFL® file system having an on-disk format representation that is block-based using, e.g., 4 kilobyte (KB) blocks and using inodes to describe files. The WAFL® file system uses files to store metadata describing the layout of its file system; these metadata files include, among others, an inode file. A file handle, i.e., an identifier that includes an inode number, is used to retrieve an inode from disk. A description of the structure of the file system, including on-disk inodes and the inode file, is provided in commonly owned U.S. Patent No. 5,819,292, titled METHOD FOR MAINTAINING CONSISTENT STATES OF A FILE SYSTEM AND FOR CREATING USER-ACCESSIBLE READ-ONLY COPIES

OF A FILE SYSTEM by David Hitz et al., issued October 6, 1998, which patent is hereby incorporated by reference as though fully set forth herein.

It should be understood that the teachings of this invention can be employed in a hybrid system that includes several types of different storage environments such as the particular storage environment 100 of Fig. 1. The invention can be used by a storage system administrator that deploys a system implementing and controlling a plurality of satellite storage environments that, in turn, deploy thousands of drives in multiple networks that are geographically dispersed. Thus, the term "storage system" as used herein, should, therefore, be taken broadly to include such arrangements.

D. Fencing Operations

In an illustrative embodiment of the present invention, the host-cluster environment includes a fencing mechanism that supports a file system protocol, such as NFS, as a shared data source in the clustered environment. More specifically, a plurality of nodes is interconnected as a cluster and is configured to utilize the storage services provided by an associated networked storage system. Each node in the cluster, referred to as a "cluster member," is supervised and controlled by cluster software executing on one or more processors in the cluster member. In accordance with illustrative embodiments of the invention, in addition to the cluster software, a fencing program resides on each cluster member and implements program instructions for fencing operations in the networked storage environment.

In accordance with illustrative embodiments of the invention, a change in cluster membership is detected by the cluster software and, in response, the fencing program is initiated. The fencing program generates host fencing API messages that notify the NFS server on the storage system to instruct an export module to change access permission rights of one or more cluster members affected by the change (i.e., the "target" cluster members) such that write access by the target cluster members is restricted from certain data in the form of files, directories, or the storage system itself. Cluster membership can change when there is a failure of one or more of the cluster members, or upon the addition of a new cluster member, or upon a failure of the communication link between cluster members, for example. The cluster member is fenced off from the

shared data or storage device to avoid any risk of spurious messages being sent by a failed or failing component. It is noted that for the purpose of clarity of illustration, the invention is described with reference to a two-node cluster embodiment. However, the invention is not limited to such a deployment and instead, the invention can be readily employed in a wide variety of clustered environments that include one or more clusters of greater than two nodes as well as multiple storage systems that may include one or more clusters of storage systems.

The fencing technique of the present invention on the host cluster side is illustratively implemented in a software program embodied as fencing program 136a on the cluster member 130a, for example. In alternate embodiments, the functionality of the fencing program 136 may be implemented as a module that is integrated with the clustering software 132 executing on each node. Referring to Figs. 1 through 3, the fencing program transmits a host API message to the storage system when a fencing operation is to be performed. The host API message is illustratively sent as a TCP/IP packet utilizing the HTTP protocol. The packet is received by the storage system at media access layer 310, which processes the packet prior to passing it to IP layer 312 of the protocol stack. The IP portion of the packet is stripped and interpreted, and the remainder of the packet is passed to TCP layer 314. The TCP layer 314, in turn, parses the packet and interprets it as an HTTP message and, thus, passes the remaining portion of the packet to HTTP layer 324. The HTTP layer then interprets the message and passes it to an Export Module within the NFS layer 320 (as described further with reference to Fig. 4) via a host fencing API interface 325.

Fig. 4 depicts a schematic software process arrangement 400 embodying the NFS layer 320 that includes an NFS server 402 that communicates with an export module 404 (residing in the memory 224 of the storage system 200.) The export module 404 contains NFS export lists 406, 408 and 410, which are also referred to herein as "NFS exports." The NFS export lists set forth permission rights regarding the read-write access and/or read-only access of cluster members (as identified by their respective IP addresses or hostnames) with respect to certain data in files and/or directories of the storage system. The restriction can also apply to an entire storage system.

The storage system, and in particular the Export Module 404, are initially configured to support the fencing program running on the cluster members. The details of the fencing procedure of the present invention on the storage system side are set forth in the above cited commonly-owned United States Patent Application No. [112056-0237; P01–2252], of Haynes, et al., for AN API FOR FENCING CLUSTER HOSTS VIA EXPORT ACCESS RIGHTS, filed on even date herewith, which has been incorporated by reference as though fully set forth herein.

Briefly, upon initial set up and configuration, export configuration module 420 specifies whether a cluster member has read-write access or read-only access with respect to particular directories or files. Read-only access is illustratively identified by an '**–ro=**' entry, which means that the cluster member identified after the **-ro=** has read only permission to files, directories, and/or storage systems specified. A cluster member that is granted read-write access, indicated by '**–rw=**' means the identified cluster member has read-write permission to the files, directories, and/or storage systems specified. In addition, administrative (root) access can be granted to cluster members if desired in a particular application of the invention.

An example of a set of export lists 500 is schematically illustrated in Fig. 5. The export list 502 is a list of cluster members that have either read-write permission, or read-only permission to File X. Though not shown in the schematic illustration of Fig. 5, the cluster members can be identified by any one of their respective IP addresses, hostnames netgroups and the like. Typically, a cluster member is addressed by either its IP address or host name. The export list 504 is a list of cluster members that have either read-write permission or read-only permission to Directory X, for example. For additional protection, a constraint may also be included whereby an IP address must appear on one of the lists or otherwise is considered a non-valid address and access is restricted completely for that particular IP address.

In order to implement a fencing operation procedure of the present invention, the cluster 120 may be initially configured such that certain members have read-write permission and other members have read-only permission to certain data. In an illustrative embodiment, however, all cluster members are granted read-write access and if a cluster member is subsequently fenced off, its access permission rights to given exports

are changed to read–only access. An initial configuration is illustratively provided by the fencing export module 420 (Fig. 4) on the storage system using a configuration file. The configuration file is illustratively written for a particular cluster member and specifies operations directed to individual exports and/or the storage system from which that cluster member may be fenced. The configuration file illustratively comprises up to 2 sections, which are named *storage systems* and *directories* respectively. If an entire storage system is identified, e.g., in the storage systems section, then all of the exports for that storage system are included in the fencing operation. A code block of a typical configuration file in accordance with an illustrative embodiment of the invention is illustrated in Fig. 6A.

For example, the first portion applies to the directories section having entries of the form (columns are separated by white spaces):

<storage system> <export> <username> <password>

The subsequent storage systems section has entries of the form (columns are separated by whitespaces):

<storage system> <username> <password>

Thus, in the example shown in Fig. 6A, the cluster member to which the configuration file code block 600 relates is fenced off from (i) the export /vol/vol0/home on the storage system x, (ii) the export /vol/vol0/home on the storage system y, and (iii) all exports on the storage system Z.

After the cluster member and/or the export lists are initially configured and the storage system is in normal operation, the export module 404 utilizes the configuration file to perform fencing operations to restrict access to certain data. The fencing operations are illustratively initiated by the fencing program 136a when generating and sending host API messages that notify the export module 404 that the export lists 406 through 410 are to be modified. Modification is performed in accordance with the fencing technique of the present invention to restrict write access to data under certain circumstances, such as a failure of one of the cluster members.

The fencing program supports at least four key operations, each of which can be specified by a different command line option. The four operations are illustratively Fence (-D option), UnFence (-A option), Validate (-V) and Read Exports (-R option). The cluster 120 (Fig. 1) uses the –D and –A options to achieve fencing and unfencing. In addition, the cluster member node may also use –V for out-of-band sanity/validation checks.

When cluster membership changes, a surviving cluster member asserts a claim on the quorum device 172 (Fig. 1). If that cluster member claims the quorum device first, it then continues to communicate with the storage system 200 over the network 160. In addition, the surviving cluster member initiates the fencing operation of the present invention. In accordance with the fencing program, a host API message 360 (Fig. 4) is generated by the fencing program 136a (Fig. 1) of the surviving cluster member and is sent to the storage system, as illustrated in Fig. 4.

Upon receipt of the host API message 360, the Server API 365 instructs the export module 404 to modify the export lists as specified in the API message 360. The details of the NFS server's Fencing API instructions are set forth in commonly-assigned, co-pending United States Patent Application [Attorney Docket No. 112056-0237- P01-2252] of Haynes, et al. for AN API FOR FENCING CLUSTER HOSTS VIA EXPORT ACCESS RIGHTS filed on even date herewith, which is presently incorporated herein by this reference in its entirety, and which sets forth further details about the fencing operations as implemented on the storage system side of the network.

In accordance with an illustrative embodiment of the present invention, the fencing program is illustratively implemented as a command line interface (CLI) command such as:

fence <operation> [<options>] <config> [<node list>]

The <operation> field identifies the operation to be performed. As noted, any of the above mentioned operations can be used. For example, -A denotes "allow", i.e., the –A operation commands the NFS server to allow or add the supplied list of nodes to the read-write portion of the export list. This effectively unfences the affected cluster member from those exports. The -D entry is used to deny or remove the supplied list of

nodes from the export lists so that those nodes are fenced from those exports. The -R entry denotes reading of the access controls and that the node list is not needed and is enforced. The -V entry denotes that the server should validate the configuration; in other words, the storage system and exports configuration should be validated.

The [<options>] field may be one of the following: -l indicates that NFS locks should be reclaimed. This is described in further detail in the above cited commonly-assigned, co-pending United States Patent Application [Attorney Docket No. 112056-0237- P01-2252]. The other options in accordance with an illustrative embodiment of the invention include: -u for a default username (root), -p for a default password (_none_), and –h which commands an associated user interface to "display this usage message."

The affected exports or storage system are identified in the <config> field. An –f entry indicates an entire storage system, meaning that all exports on the named storage system are subjected to the operation. Alternatively, a "config" file can be referenced and, in this case, all the exports and/or storage systems identified in the respective config file are subjected to the operation. The [<node list>] identifies the cluster members that are the subject of the operation. A user name and password are also specified. The above described usage information based upon the fencing program 136a of the present invention generates the host fencing API messages 360 is summarized in the chart 650 of Fig. 6B.

A specific example of a command line invocation of the fencing program in accordance with the invention may be of the form:

fence -D -f manor -u username -p fu_bar 10.1.1.56

This message specifies that the IP address 10.1.1.56 is fenced from all exports of the storage system called "manor." The username and password (fu_bar) used to connect to the storage system are also specified. As mentioned, the storage system is specified with the –f option.

Upon receiving the host fencing API message 360, the Server API 365 instructs the export module to modify the export lists 406, 408, and 410 such that specified cluster member(s) has an "-rw=" entry modified as to the identified exports or storage sys-

tem. Accordingly, the specified cluster member no longer has read-write access to the files, directories or volumes indicated in that segment of the host API message 360. In an illustrative embodiment of the invention, those cluster members are listed on the read-only "export list" and are re-configured with an '-ro' permission for the files specified until they are unfenced by an appropriate command.

Using this software code to generate the host fencing API messages 360, the fencing and unfencing operations are performed in accordance with the present invention. More specifically, the flow chart of Fig. 7 illustrates the procedure 700, which summarizes in broad terms, the steps of fencing technique of the present invention. As noted, the present invention relates to the fencing program 136a implemented on a cluster node 130a, for example (Fig. 1). Moreover, the above-cited commonly owned United States Patent Application Serial No. [112056-0237/P01-2252] filed on even date herewith, sets forth further details about the fencing operations as implemented on the storage system side of the network.

The procedure 700 begins with the start step 702 and continues to step 704 where the cluster software detects that the cluster membership in an established cluster has changed. For example, one of the clusters in the cluster 120 of Fig. 1 may have failed, or alternatively, a new node could be added as a cluster member to the cluster 120, or the communications over interconnect 170 between the clusters may have failed. In any of these circumstances, the cluster software detects that cluster membership has changed, or an administrator has notified the system of the change in cluster membership, in accordance with step 704.

In step 706, a surviving cluster member, such as the cluster member 130a, initiates the fence program 136a of the present invention. In step 708, the surviving cluster member transmits a host fencing API message over the network 160 to the storage system 200, where it is received by the storage system 200 in step 710 and passed to the NFS server. As a result, the Server API Module 365 (Fig. 4) instructs the export module 404 to modify the export lists 406 through 410 to fence off the identified former cluster member 130b, for example, as illustrated in step 712. Once the failed cluster member has been fenced off, operations continue with the surviving cluster member 130a and the procedure 700 completes in accordance with step 714.

The fencing operations on the cluster side of the storage environment are illustrated in further detail in the flow chart of Figs. 8A and 8B which together form a flow chart of a procedure 800. The procedure 800 starts with step 802 and continues to step 804, where the fence program is (down)loaded into each cluster member controlled by the fencing/unfencing procedure of the present invention. As part of this step 804, a user name and password required to access the storage system is noted for access from the cluster member.

In step 806, the storage system and the export configuration are validated and, in step 808, the cluster member attempts to connect to the specified storage system. Authentication with the specified user name and password is performed in step 810. In step 812, the fence program checks for the presence of "-rw" and "-ro" wild cards. If any such wild cards are detected, an error message is issued, and the procedure ends.

If no such wild cards are detected, then the procedure continues to step 814 where a determination is made as to whether the export paths specified in the configuration file for that particular storage system are indeed correct for the corresponding storage system. At this point, the fence program has been downloaded from a storage system provider's website or via an other suitable medium, as will be understood by those skilled in the art and the various validations and authentications have occurred and the program is ready for execution. A relatively weaker form of validation is performed when fence (-D) and unfence (-A) operations are executed on a cluster member. In this embodiment, it is confirmed whether the appropriate error messages are issued, but the operation continues to proceed. In contrast, a –V operation is specifically executed to validate the exports configuration, before using the shared data sources in this environment.

In step 816, a change in cluster membership is detected when there is a loss, addition or communication failure between cluster members. In that case, the cluster member software establishes a new quorum via the quorum disk 172 (Fig. 1) with a group of new cluster members in step 818. One of the surviving cluster members generates a host fencing API message 360 in accordance with the fence program in step 820 and transmits that message to the storage system over the network in step 822. Illustratively, the API message is sent in the form described herein.

In response to the host fencing API message 360, the export lists are modified accordingly to either fence (i.e., to restrict read-write access to one or more former cluster members) or unfence (i.e., to reinstate or allow read-write access to a previously fenced node). Once the fencing or unfencing operation has been performed, the fenced off cluster member can no longer write to the files and/or directories served by the storage system and does not share data with any of the new cluster members. A previously fenced cluster member waits to read the disk to determine status and to be brought to the same state as the other cluster members prior to continuing operations thereafter as shown in the step 824. Once the fencing operation is performed, the procedure 800 returns to wait until a further change is detected by loss, addition or other reason that a fence or unfence operation is to occur.

It should be understood that the present invention provides a unique mechanism and cluster side tool used to support a file system protocol, such as NFS, as a shared data source with Unix®-based host clustering products. The present invention is advantageous in that NFS data access over a network can still be afforded fencing techniques without the need for using SCSI-based reservations to perform fencing from the exported file systems of a storage system.

Thus, the invention provides a simplified user interface for fencing cluster members, which is easily portable across all Unix®-based host platforms. In addition, the invention can be implemented and used over TCP with insured reliability. The invention also provides a means to fence cluster members and enables the use of NFS in a shared collaborative clustering environment. It should be noted that while the present invention has been written in terms of files and directories, the present invention may be utilized to fence/unfence any form of networked data containers associated with a storage system. It should be further noted that the system of the present invention provides a simple and complete user interface that can be plugged into a host cluster framework which can accommodate different types of shared data containers. Furthermore, the system and method of the present invention supports NFS as a shared data source in a high-availability environment that includes one or more storage system clusters and one or more host clusters having end-to-end availability in mission-critical deployments having 24 x 7 availability.

The foregoing has been a detailed description of the invention. Various modifications and additions can be made without departing from the spirit and scope of the invention. Furthermore, it is expressly contemplated that the various processes, layers, modules and utilities shown and described according to this invention can be implemented as software, consisting of a computer readable medium including programmed instructions executing on a computer, as hardware or firmware using state machines and the like, or as a combination of hardware, software and firmware. Accordingly, this description is meant to be taken only by way of example and not to otherwise limit the scope of the invention.

What is claimed is:

CLAIMS

1. A method for performing fencing in a clustered storage environment, the method comprising the steps of:

providing a plurality of nodes configured in a cluster for sharing data, each node being a cluster member;

providing a storage system that supports a plurality of data containers, said storage system supporting a protocol that configures export lists that assign each cluster member certain access permission rights including read-write access permission or read only access permission as to each respective data container associated with the storage system;

coupling the cluster to the storage system by a network; and

providing a fencing program in each cluster member such that when a change in cluster membership is detected, a surviving member transmits an application program interface message to said storage system commanding said storage system to modify one or more of said export lists such that the access permission rights of one or more identified cluster members are modified.

2. The method as defined in claim 1 wherein said protocol used by said storage system is the Network File System protocol.

3. The method as defined in claim 1 wherein said cluster is coupled to the storage system over a network using Transmission Control Protocol/Internet Protocol.

4. The method as defined in claim 1 wherein said cluster member transmits said application program interface messages encapsulated in HyperText Transfer Protocol messages.

5. The method as defined in claim 1 further comprising the step of initially configuring the storage system such that each cluster member has a set of access permission rights set forth in a configuration file.

6. The method as defined in claim 5 wherein said application program interface message provides said modifications to said access permission rights of said initial configuration.

7. The method as defined in claim 1 including the further step of validating the storage system and the exports configuration.

8. The method as defined in claim 1 further comprising checking the initial configuration for the presence of any wildcard entries in said export lists, and if said wildcard entries are found, issuing an error message

9. The method as defined in claim 1 wherein a change in cluster membership is determined by the associated cluster infrastructure.

10. A system adapted to perform fencing operations in a clustered storage environment, the system comprising:

a cluster of interconnected cluster members that provide storage services for one or more clients;

a networked storage system coupled to said cluster by way of a network, said storage system having a plurality of storage devices, said storage system including export lists that contain rules regarding access permission rights for specified cluster members as to specified associated storage devices, and/or files and/or directories stored therein; and

a fencing program executing within each cluster member that includes program instructions for sending an application program interface message from one of said cluster members to said storage system and that contains instructions for modifying one or more of said export lists to change the access permission rights of one or more of said cluster members.

11. The system as defined in claim 10 wherein said networked storage system is coupled to said cluster by way of a network that uses Transmission Control Protocol/Internet Protocol.

12. The system as defined in claim 10 wherein said networked storage system wherein said storage system includes a Network File System server that uses Network File system export lists that contain rules regarding access permission rights to data of the storage system that is shared by said cluster members.

13. The system as defined in claim 10 where said storage system is initially configured with a set of export lists that sets forth read-only and/or read-write access permissions for each cluster member of the cluster that is interfaced with the storage system with respect to one of an entire storage system, a file of a storage system, and a directory of a storage system.

14. The system as defined in claim 10 wherein said fencing program is initiated upon a change in cluster membership as determined by the associated cluster infrastructure.

15. The system as defined in claim 10 wherein said cluster members are interfaced with a client requesting data from the storage system via one or more networks.

16. The system as defined in claim 10 further comprising a quorum device that is directly coupled to each cluster member and which quorum device is configured in such a manner that the cluster member that asserts a claim to the quorum device first is thereby granted access to the storage system.

17. The system as defined in claim 16 wherein the cluster member that asserts a claim to the quorum device first transmits the application program interface message to the storage system containing instructions for modifying one or more of said export lists to change an access permission right of one or more of said cluster members.

18. A computer readable medium for performing failure fencing in a clustered environment with networked storage, including program instructions for performing the steps of:

providing a plurality of nodes configured in a cluster for sharing data, each node being a cluster member;

providing a storage system that has access to multiple storage devices including disks, said storage system supporting a protocol that configures export lists that assign each cluster member certain access permission rights including read-write access permission or read only access permission as to each respective storage device, and files and directories of said storage device, associated with the storage system;

coupling the cluster to the storage system by a network; and

providing a fencing program in each cluster member such that when a change in cluster membership is detected, a surviving member transmits an application program interface message via said protocol over said network to said storage system commanding said storage system to modify one or more of said export lists such that the access permission rights of one or more identified cluster members are modified.

19. The computer readable medium for performing failure fencing as defined in claim 18 further comprising program instructions for sending an application program interface message that provides said modifications to said access permission rights of an initial configuration by way of a config file or a command line.

20. The computer readable medium for performing failure fencing as defined in claim 18 wherein changes in cluster membership are determined by the associated cluster infrastructure.

21. The computer readable medium for performing failure fencing as defined in claim 18 further comprising a simple and complete user interface that can be plugged into a host cluster framework which can accommodate different types of shared data containers.

22. The computer readable medium as defined in claim 18 including means for supporting NFS as a shared data source in a high-availability environment that includes one or more storage system clusters and one or more host clusters having end-to-end availability in mission-critical deployments having substantially continuous availability.

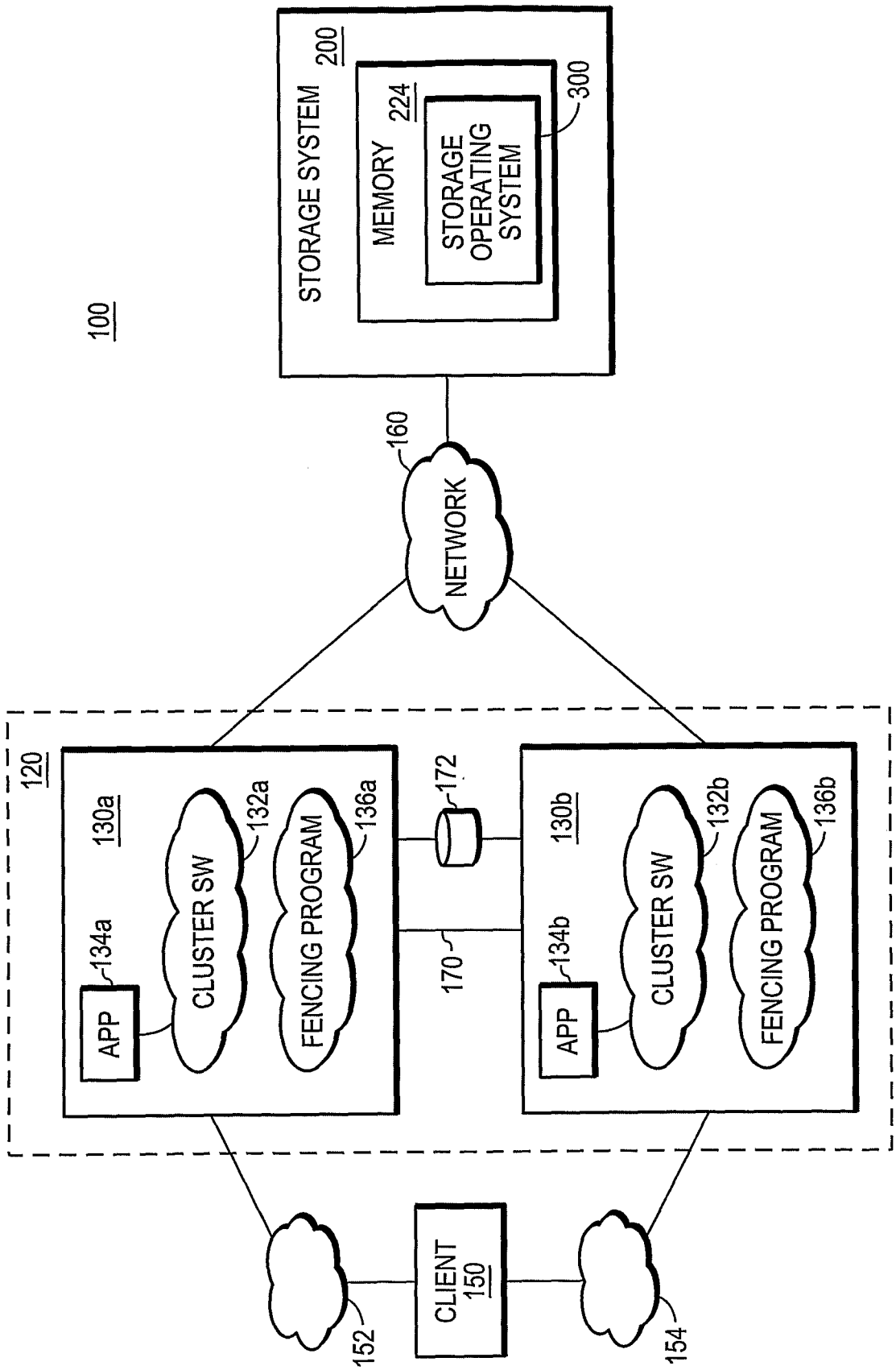

100
STORAGE SYSTEM
200
MEMORY
224
STORAGE OPERATING SYSTEM
300
160
NETWORK
120
130a
APP
134a
CLUSTER SW
132a
FENCING PROGRAM
136a
172
170
130b
APP
134b
CLUSTER SW
132b
FENCING PROGRAM
136b
CLIENT
150
152
154

FIG. 1

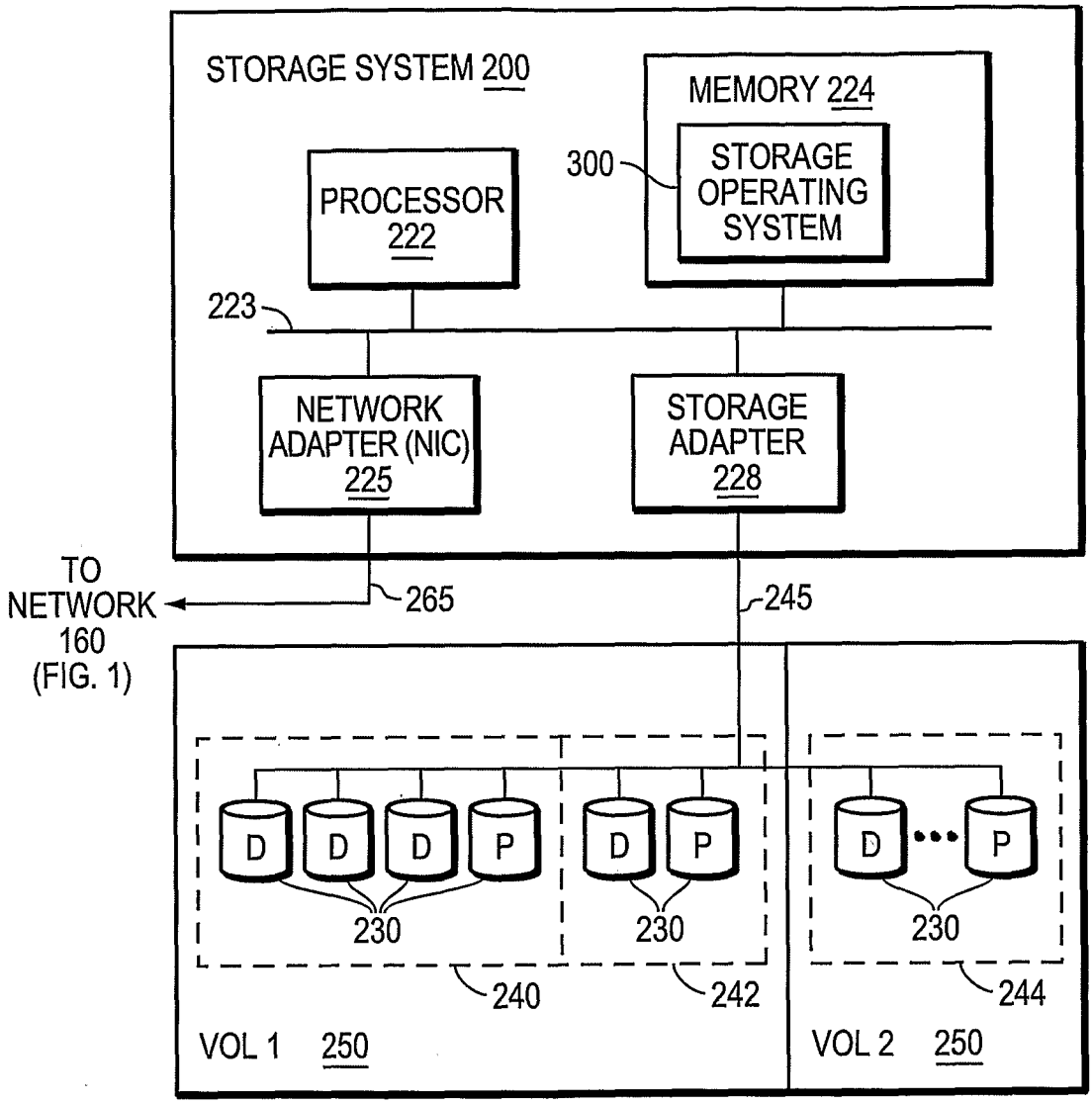
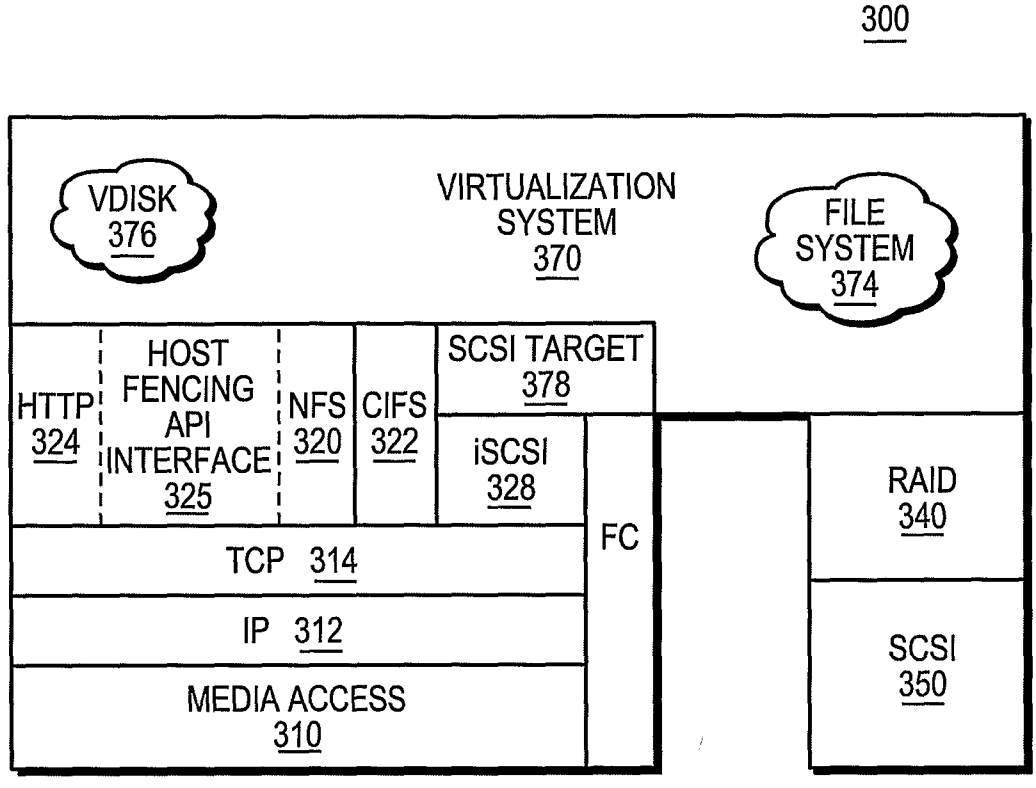

STORAGE SYSTEM 200
MEMORY 224
300
STORAGE OPERATING SYSTEM
PROCESSOR 222
223
NETWORK ADAPTER (NIC) 225
STORAGE ADAPTER 228
TO NETWORK 160 (FIG. 1)
265
245
D
D
D
P
D
P
D
P
230
230
230
240
242
244
VOL 1 250
VOL 2 250

FIG. 2


300
VDISK
376
VIRTUALIZATION
SYSTEM
370
FILE
SYSTEM
374
HTTP
324
HOST
FENCING
API
INTERFACE
325
NFS
320
CIFS
322
SCSI TARGET
378
iSCSI
328
FC
RAID
340
TCP 314
IP 312
MEDIA ACCESS
310
SCSI
350

. 3

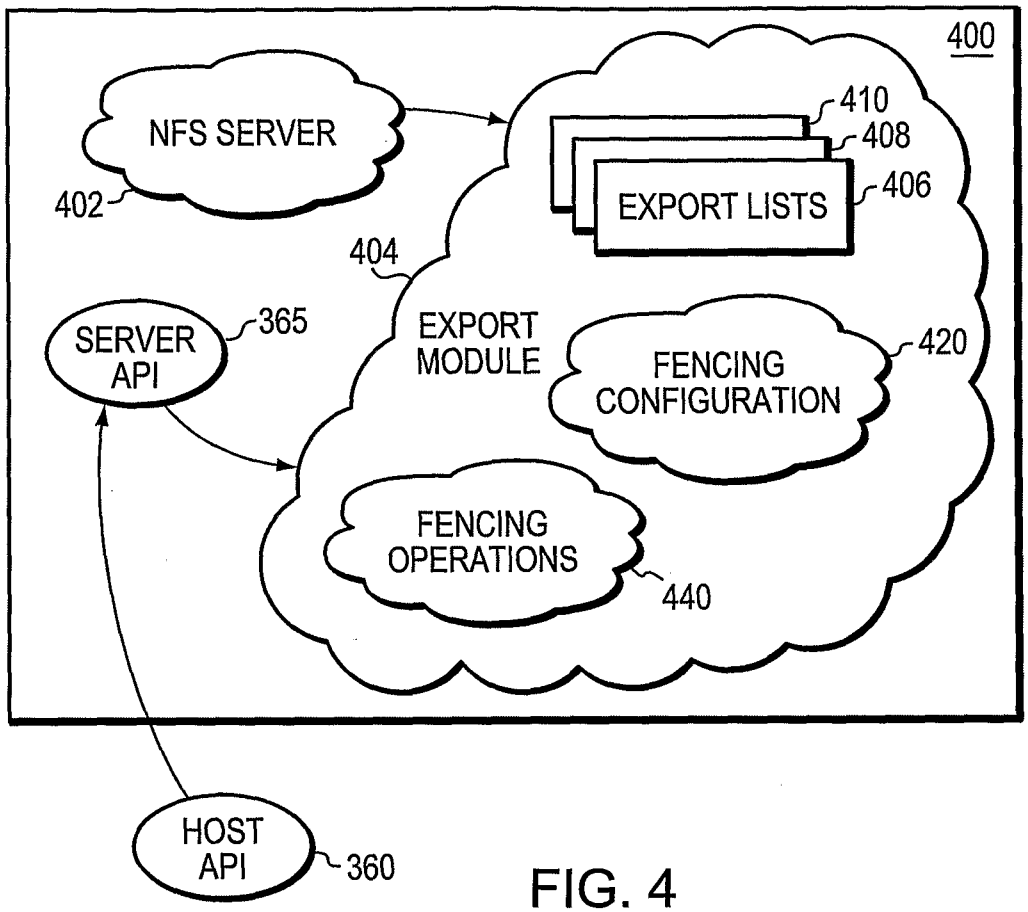

400
NFS SERVER
402
410
408
EXPORT LISTS
406
404
SERVER API
365
EXPORT MODULE
FENCING CONFIGURATION
420
FENCING OPERATIONS
440
HOST API
360

FIG. 4

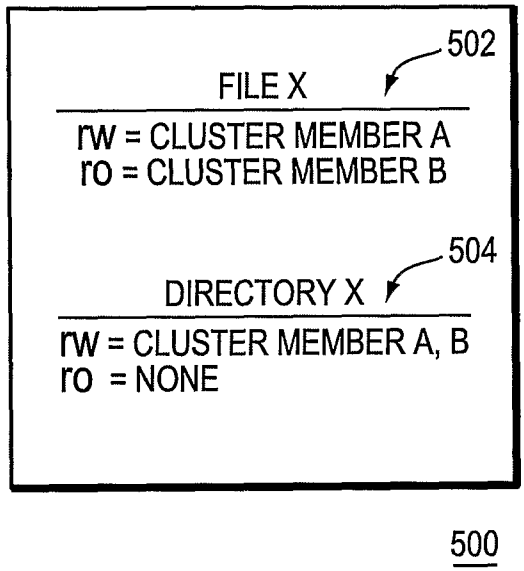

502
FILE X
rw = CLUSTER MEMBER A
ro = CLUSTER MEMBER B
504
DIRECTORY X
rw = CLUSTER MEMBER A, B
ro = NONE
500

FIG. 5

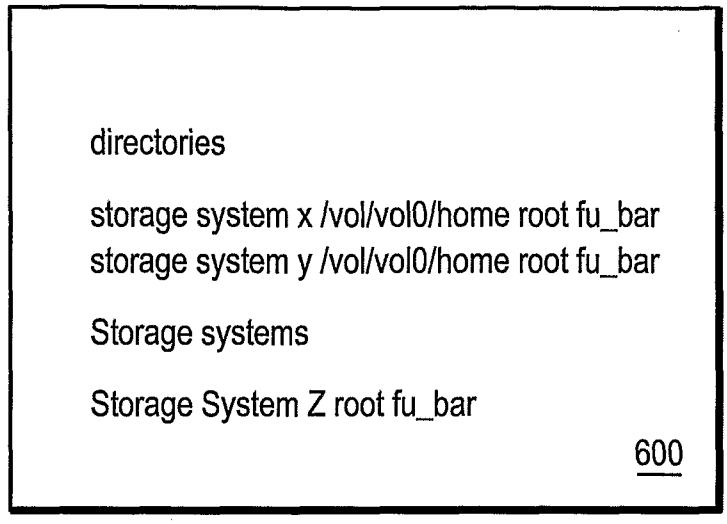

directories
storage system x /vol/vol0/home root fu_bar
storage system y /vol/vol0/home root fu_bar
Storage systems
Storage System Z root fu_bar
600

FIG. 6A

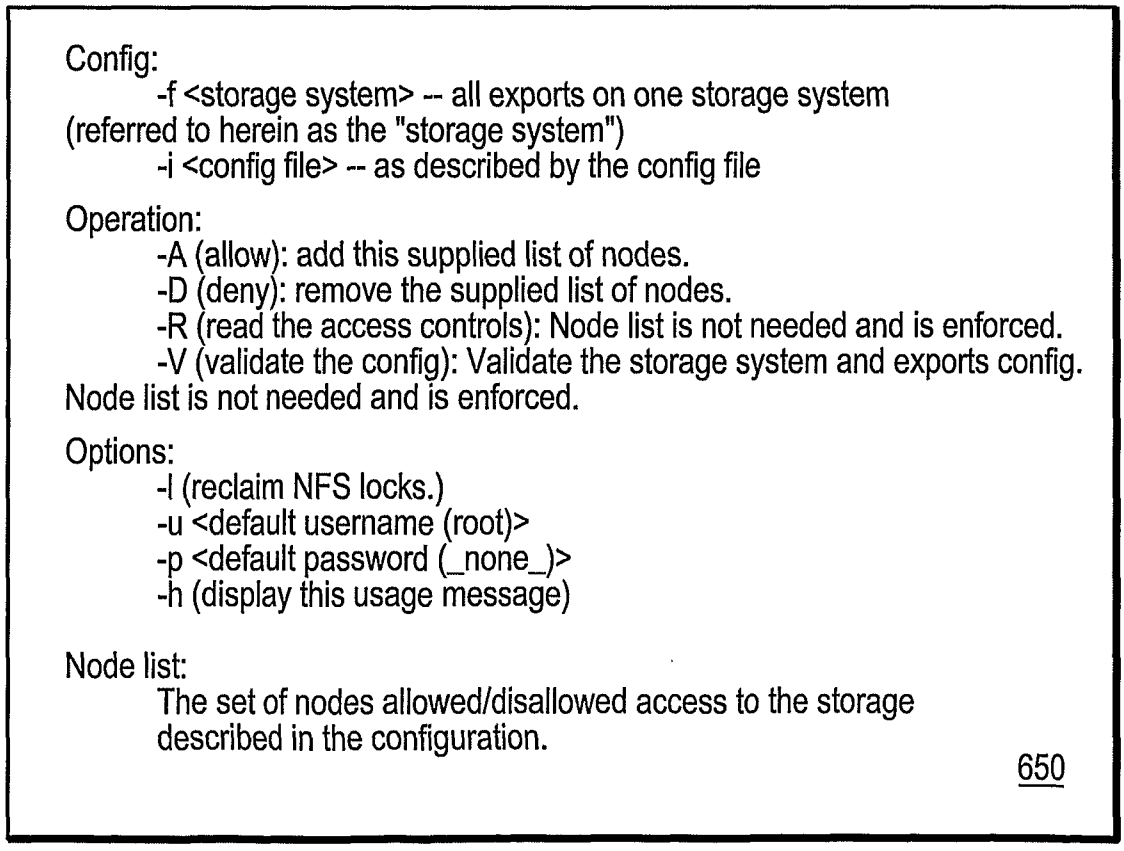

Config:
-f <storage system> -- all exports on one storage system
(referred to herein as the "storage system")
-i <config file> -- as described by the config file
Operation:
-A (allow): add this supplied list of nodes.
-D (deny): remove the supplied list of nodes.
-R (read the access controls): Node list is not needed and is enforced.
-V (validate the config): Validate the storage system and exports config.
Node list is not needed and is enforced.
Options:
-l (reclaim NFS locks.)
-u <default username (root)>
-p <default password (_none_)>
-h (display this usage message)
Node list:
The set of nodes allowed/disallowed access to the storage
described in the configuration.
650

FIG. 6B

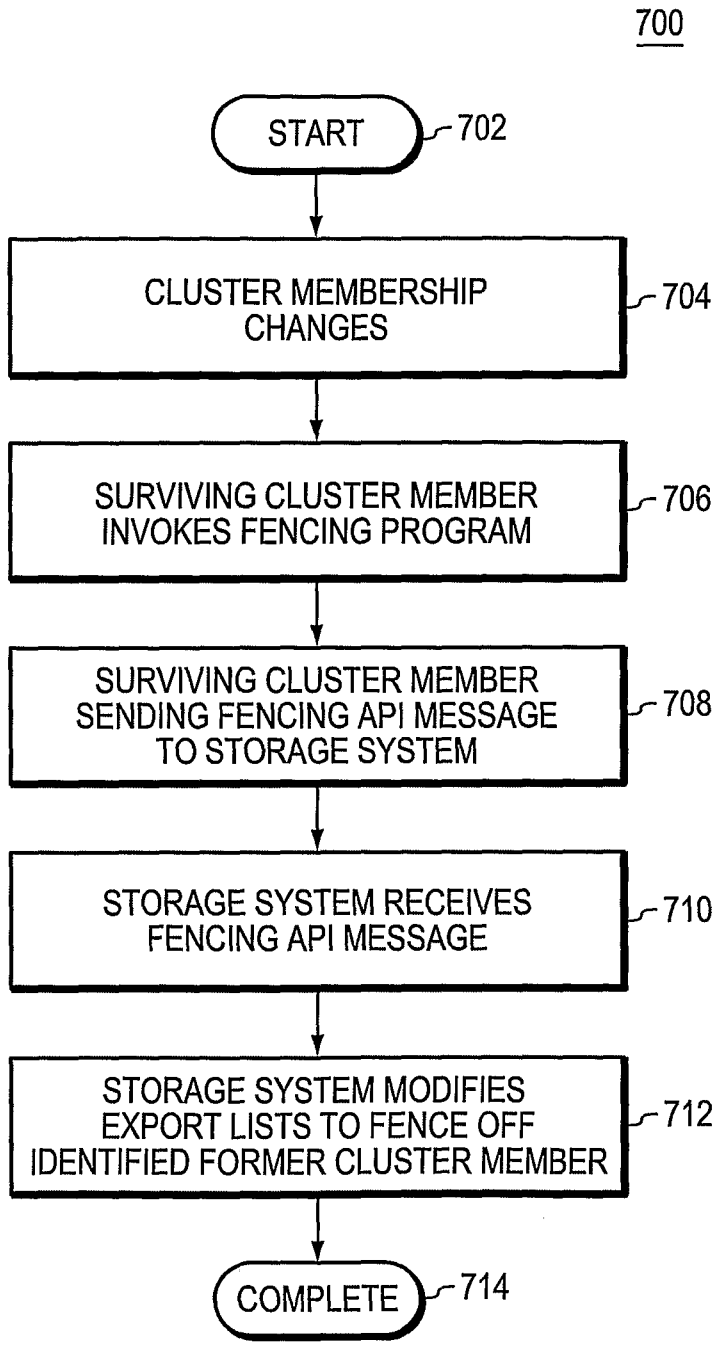

700
START
702
CLUSTER MEMBERSHIP CHANGES
704
SURVIVING CLUSTER MEMBER INVOKES FENCING PROGRAM
706
SURVIVING CLUSTER MEMBER SENDING FENCING API MESSAGE TO STORAGE SYSTEM
708
STORAGE SYSTEM RECEIVES FENCING API MESSAGE
710
STORAGE SYSTEM MODIFIES EXPORT LISTS TO FENCE OFF IDENTIFIED FORMER CLUSTER MEMBER
712
COMPLETE
714

FIG. 7

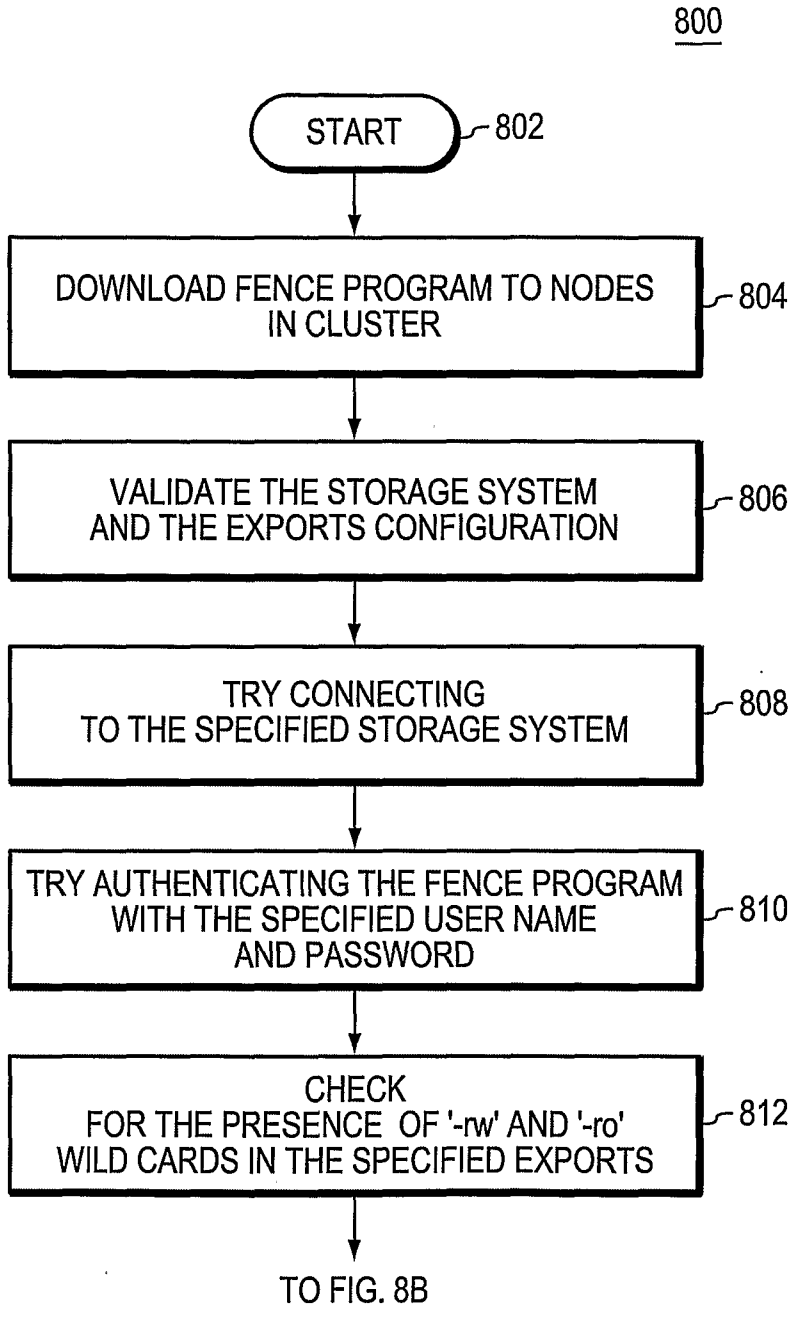

800
START
802
DOWNLOAD FENCE PROGRAM TO NODES IN CLUSTER
804
VALIDATE THE STORAGE SYSTEM AND THE EXPORTS CONFIGURATION
806
TRY CONNECTING TO THE SPECIFIED STORAGE SYSTEM
808
TRY AUTHENTICATING THE FENCE PROGRAM WITH THE SPECIFIED USER NAME AND PASSWORD
810
CHECK FOR THE PRESENCE OF '-rw' AND '-ro' WILD CARDS IN THE SPECIFIED EXPORTS
812
TO FIG. 8B

FIG. 8A

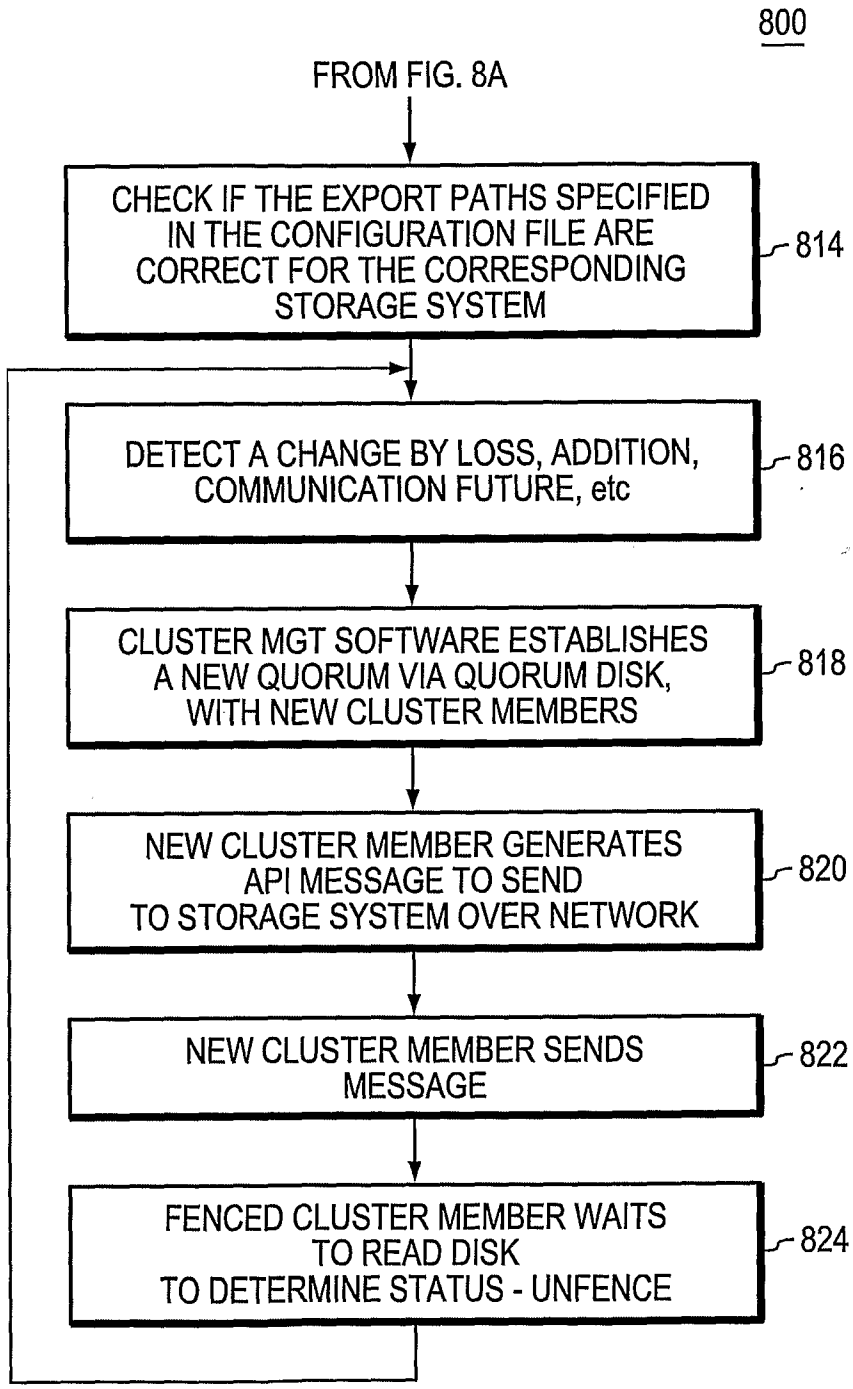

800
FROM FIG. 8A
CHECK IF THE EXPORT PATHS SPECIFIED IN THE CONFIGURATION FILE ARE CORRECT FOR THE CORRESPONDING STORAGE SYSTEM
814
DETECT A CHANGE BY LOSS, ADDITION, COMMUNICATION FUTURE, etc
816
CLUSTER MGT SOFTWARE ESTABLISHES A NEW QUORUM VIA QUORUM DISK, WITH NEW CLUSTER MEMBERS
818
NEW CLUSTER MEMBER GENERATES API MESSAGE TO SEND TO STORAGE SYSTEM OVER NETWORK
820
NEW CLUSTER MEMBER SENDS MESSAGE
822
FENCED CLUSTER MEMBER WAITS TO READ DISK TO DETERMINE STATUS - UNFENCE
824

FIG. 8B

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/028494

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/08 H04L29/14 G06F3/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2004/210656 A1 (BECK KENNETH S [US] ET AL) 21 October 2004 (2004-10-21) figure 2 paragraphs [0040], [0091] - [0095]	1-22
A	US 2004/123053 A1 (KARR RONALD S [US] ET AL) 24 June 2004 (2004-06-24) figure 5 paragraphs [0008] - [0011], [0032]	1-22
A	EP 1 107 119 A2 (SUN MICROSYSTEMS INC [US]) 13 June 2001 (2001-06-13) paragraph [0052] figures 1,2,5	1-22

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
1 December 2006	08/12/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Tyszka, Krzysztof

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/028494

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004210656 A1	21-10-2004	NONE	
US 2004123053 A1	24-06-2004	AU 2003303433 A1 WO 2004059498 A2	22-07-2004 15-07-2004
EP 1107119 A2	13-06-2001	NONE	