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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 March 2017 (30.03.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/053220 A1

(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 11/20* (2006.01)
G06F 11/14 (2006.01)

(21) International Application Number:
PCT/US2016/052431

(22) International Filing Date:
19 September 2016 (19.09.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/865,442 25 September 2015 (25.09.2015) US

(71) Applicant: **NETAPP, INC.** [US/US]; 495 E. Java Drive, Sunnyvale, CA 94089 (US).

(72) Inventors: **NGAN, Ching-Yuk, Paul**; 495 E. Java Drive, Sunnyvale, CA 94089 (US). **SINGH, Kanwaldeep**; 495 E. Java Drive, Sunnyvale, CA 94089 (US). **MU, Yuedong**; 495 E. Java Drive, Sunnyvale, CA 94089 (US). **SUNDARARAJAN, Manoj, Kumar V**; 495 E. Java Drive, Sunnyvale, CA 94089 (US).

(74) Agent: **TATE, Daniel, A.**; Cooper Legal Group, LLC, 6505 Rockside Road, Suite 330, Independence, OH 44131 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,

[Continued on next page]

(54) Title: DATA SYNCHRONIZATION

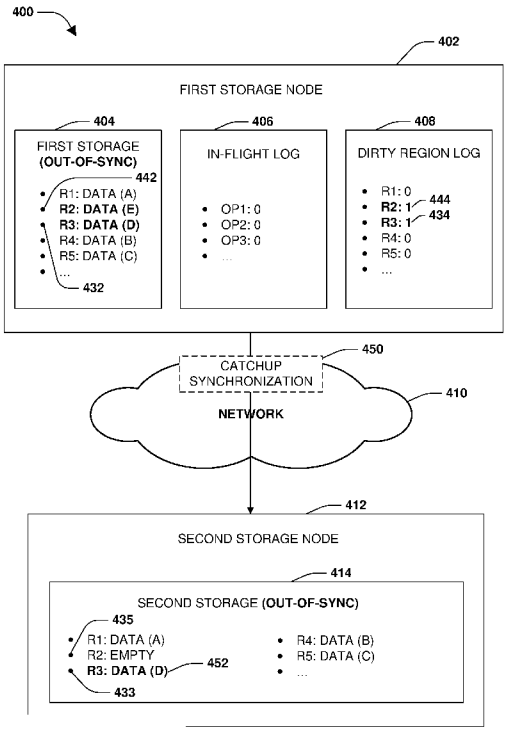


FIG. 4F

(57) Abstract: One or more techniques and/or computing devices are provided for data synchronization. For example, an in-flight log may be maintained to track storage operations that are received by a first storage node, but have not been committed to both first storage of the first storage node and second storage of a second storage node that has a replication relationship, such as a disaster recovery relationship, with the first storage node. A dirty region log may be maintained to track regions within the first storage that have been modified by storage operations that have not been replicated to the second storage. Accordingly, a catchup synchronization phase (e.g., asynchronous replication by a resync scanner) may be performed to replicate storage operations (e.g., replicate data within dirty regions of the first storage that were modified by such storage operations) to the second storage until the first storage and the second storage are synchronized.

WO 2017/053220 A1

SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

DATA SYNCHRONIZATION

DATA SYNCHRONIZATION

RELATED APPLICATION

[001] The present application claims priority to U.S. Non-Provisional Patent Application No.: 14/865,442, titled "DATA SYNCHRONIZATION", filed on September 25, 2015, which is incorporated herein by reference.

BACKGROUND

[002] Many storage networks may implement data replication for data loss protection. For example, a first storage cluster may comprise a first storage node configured to provide clients with primary access to data stored within a first storage device and/or other storage devices. A second storage cluster may comprise a second storage node configured to provide clients with access to data stored within a second storage device (e.g., failover access to replicated data within the second storage device) and/or other storage devices (e.g., primary access to data stored within a third storage device). The first storage node and the second storage node may be configured according to a disaster recovery relationship, such that the second storage node may provide failover access to replicated data that was replicated from the first storage device to the second storage device.

[003] When the first storage node and the second storage node are in a synchronous replication state, storage operations (e.g., data change fileops, offload fileops, metadata change fileops, abort operations, etc.) are synchronously replicated from the first storage node and first storage device to the second storage node and the second storage device. For example, a write storage operation may be received from a client by the first storage node. Before a response is provided back to the client that the write storage operation is complete, the write storage operation is to be both written to the first storage device and replicated to the second storage device. If a replication error or other issue occurs (e.g., the second storage node has a failure or reboots, a network connection between the first storage node and the second storage node is lost, etc.), then the first storage node and the second storage node may transition into an out-of-sync state without strict data consistency between the first storage device and the second storage device.

DESCRIPTION OF THE DRAWINGS

[004] Fig. 1 is a component block diagram illustrating an example clustered network in accordance with one or more of the provisions set forth herein.

[005] Fig. 2 is a component block diagram illustrating an example data storage system in accordance with one or more of the provisions set forth herein.

[006] Fig. 3 is a flow chart illustrating an exemplary method of data synchronization.

[007] Fig. 4A is a component block diagram illustrating an exemplary computing device for data synchronization, where a first storage node and a second storage node are implementing synchronous replication.

[008] Fig. 4B is a component block diagram illustrating an exemplary computing device for data synchronization, where network communication between a first storage node and a second storage node is lost.

[009] Fig. 4C is a component block diagram illustrating an exemplary computing device for data synchronization, where a first storage node processes a fourth storage operation while network communication is lost between the first storage node and a second storage node.

[010] Fig. 4D is a component block diagram illustrating an exemplary computing device for data synchronization, where a first storage node processes a fifth storage operation while network communication is lost between the first storage node and a second storage node.

[011] Fig. 4E is a component block diagram illustrating an exemplary computing device for data synchronization, where network communication between a first storage node and a second storage node is restored.

[012] Fig. 4F is a component block diagram illustrating an exemplary computing device for data synchronization, where catchup synchronization is performed between a first storage node and a second storage node.

[013] Fig. 4G is a component block diagram illustrating an exemplary computing device for data synchronization, where a sixth storage operation is

synchronously committed to first storage and replicated to second storage while a first storage node and a second storage node are out-of-sync.

[014] Fig. 4H is a component block diagram illustrating an exemplary computing device for data synchronization, where a sixth storage operation is synchronously committed to first storage and replicated to second storage while a first storage node and a second storage node are out-of-sync.

[015] Fig. 4I is a component block diagram illustrating an exemplary computing device for data synchronization, where catchup synchronization is performed between a first storage node and a second storage node.

[016] Fig. 5 is a component block diagram illustrating an exemplary computing device for maintaining data consistency between snapshots.

[017] Fig. 6 is an example of a computer readable medium in accordance with one or more of the provisions set forth herein.

DETAILED DESCRIPTION

[018] Some examples of the claimed subject matter are now described with reference to the drawings, where like reference numerals are generally used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide an understanding of the claimed subject matter. It may be evident, however, that the claimed subject matter may be practiced without these specific details. Nothing in this detailed description is admitted as prior art.

[019] One or more techniques and/or computing devices for data synchronization are provided. For example, an in-flight log may be maintained to track storage operations that are received by a first storage node, but have not been committed to both first storage of the first storage node and second storage of a second storage node having a replication relationship, such as a disaster recovery relationship, with the first storage node. In an example, a storage operation may be implemented to write to the first storage by the first storage node simultaneously with the creation of an entry, within the in-flight log, for the storage operation (e.g., the first storage and the in-flight log may be change simultaneously through shared write access provided by a file system).

[020] A dirty region log, such as a bitmap, may be used to track what regions of the first storage have been changed by storage operations that have not yet been replicated to the second storage. In this way, a catchup synchronization phase may be performed using the in-flight log and/or the dirty region log to replicate (e.g., asynchronous replication by a resync scanner) storage operations (e.g., dirty regions modified by storage operations) to the second storage and second storage node. A sync log, corresponding to the in-flight log and/or the dirty region log, may be efficiently maintained and used to reduce latency for responding to clients (e.g., a storage operation may be written to the first storage and a notification of the storage operation being complete may be sent to the client notwithstanding the storage operation not yet being replicated to the second storage because data of the storage operation will later be asynchronously replicated based upon the sync log), maintain low overhead (e.g., the sync log may comprise bitmaps that may be efficiently zeroed out), maintain consistency with snapshots (e.g., the sync log may be used to identify a snapshot difference between a snapshot of the first storage and of the second storage), provide persistent storage (e.g., the sync log may be stored on disk), etc.

[021] To provide context for data synchronization, Fig. 1 illustrates an embodiment of a clustered network environment 100 or a network storage environment. It may be appreciated, however, that the techniques, etc. described herein may be implemented within the clustered network environment 100, a non-cluster network environment, and/or a variety of other computing environments, such as a desktop computing environment. That is, the instant disclosure, including the scope of the appended claims, is not meant to be limited to the examples provided herein. It will be appreciated that where the same or similar components, elements, features, items, modules, etc. are illustrated in later figures but were previously discussed with regard to prior figures, that a similar (e.g., redundant) discussion of the same may be omitted when describing the subsequent figures (e.g., for purposes of simplicity and ease of understanding).

[022] Fig. 1 is a block diagram illustrating an example clustered network environment 100 that may implement at least some embodiments of the techniques and/or systems described herein. The example environment 100 comprises data storage systems or storage sites 102 and 104 that are coupled over a cluster fabric 106, such as a computing network embodied as a private Infiniband, Fibre Channel (FC), or Ethernet network facilitating communication between the storage systems 102 and

104 (and one or more modules, component, etc. therein, such as, nodes 116 and 118, for example). It will be appreciated that while two data storage systems 102 and 104 and two nodes 116 and 118 are illustrated in Fig. 1, that any suitable number of such components is contemplated. In an example, nodes 116, 118 comprise storage controllers (e.g., node 116 may comprise a primary or local storage controller and node 118 may comprise a secondary or remote storage controller) that provide client devices, such as host devices 108, 110, with access to data stored within data storage devices 128, 130. Similarly, unless specifically provided otherwise herein, the same is true for other modules, elements, features, items, etc. referenced herein and/or illustrated in the accompanying drawings. That is, a particular number of components, modules, elements, features, items, etc. disclosed herein is not meant to be interpreted in a limiting manner.

[023] It will be further appreciated that clustered networks are not limited to any particular geographic areas and can be clustered locally and/or remotely. Thus, in one embodiment a clustered network can be distributed over a plurality of storage systems and/or nodes located in a plurality of geographic locations; while in another embodiment a clustered network can include data storage systems (e.g., 102, 104) residing in a same geographic location (e.g., in a single onsite rack of data storage devices).

[024] In the illustrated example, one or more host devices 108, 110 which may comprise, for example, client devices, personal computers (PCs), computing devices used for storage (e.g., storage servers), and other computers or peripheral devices (e.g., printers), are coupled to the respective data storage systems 102, 104 by storage network connections 112, 114. Network connection may comprise a local area network (LAN) or wide area network (WAN), for example, that utilizes Network Attached Storage (NAS) protocols, such as a Common Internet File System (CIFS) protocol or a Network File System (NFS) protocol to exchange data packets. Illustratively, the host devices 108, 110 may be general-purpose computers running applications, and may interact with the data storage systems 102, 104 using a client/server model for exchange of information. That is, the host device may request data from the data storage system (e.g., data on a storage device managed by a network storage control configured to process I/O commands issued by the host device for the storage device), and the data storage system may return results of the request to the host device via one or more network connections 112, 114.

[025] The nodes 116, 118 on clustered data storage systems 102, 104 can comprise network or host nodes that are interconnected as a cluster to provide data storage and management services, such as to an enterprise having remote locations, cloud storage (e.g., a storage endpoint may be stored within a data cloud), etc., for example. Such a node in a data storage and management network cluster environment 100 can be a device attached to the network as a connection point, redistribution point or communication endpoint, for example. A node may be capable of sending, receiving, and/or forwarding information over a network communications channel, and could comprise any device that meets any or all of these criteria. One example of a node may be a data storage and management server attached to a network, where the server can comprise a general purpose computer or a computing device particularly configured to operate as a server in a data storage and management system.

[026] In an example, a first cluster of nodes such as the nodes 116, 118 (e.g., a first set of storage controllers configured to provide access to a first storage aggregate comprising a first logical grouping of one or more storage devices) may be located on a first storage site. A second cluster of nodes, not illustrated, may be located at a second storage site (e.g., a second set of storage controllers configured to provide access to a second storage aggregate comprising a second logical grouping of one or more storage devices). The first cluster of nodes and the second cluster of nodes may be configured according to a disaster recovery configuration where a surviving cluster of nodes provides switchover access to storage devices of a disaster cluster of nodes in the event a disaster occurs at a disaster storage site comprising the disaster cluster of nodes (e.g., the first cluster of nodes provides client devices with switchover data access to storage devices of the second storage aggregate in the event a disaster occurs at the second storage site).

[027] As illustrated in the exemplary environment 100, nodes 116, 118 can comprise various functional components that coordinate to provide distributed storage architecture for the cluster. For example, the nodes can comprise a network module 120, 122 and a data module 124, 126. Network modules 120, 122 can be configured to allow the nodes 116, 118 (e.g., network storage controllers) to connect with host devices 108, 110 over the network connections 112, 114, for example, allowing the host devices 108, 110 to access data stored in the distributed storage system. Further, the network modules 120, 122 can provide connections with one or more other

components through the cluster fabric 106. For example, in Fig. 1, a first network module 120 of first node 116 can access a second data storage device 130 by sending a request through a second data module 126 of a second node 118.

[028] Data modules 124, 126 can be configured to connect one or more data storage devices 128, 130, such as disks or arrays of disks, flash memory, or some other form of data storage, to the nodes 116, 118. The nodes 116, 118 can be interconnected by the cluster fabric 106, for example, allowing respective nodes in the cluster to access data on data storage devices 128, 130 connected to different nodes in the cluster. Often, data modules 124, 126 communicate with the data storage devices 128, 130 according to a network file system (NFS) protocol a storage area network (SAN) protocol, such as Small Computer System Interface (SCSI) or Fiber Channel Protocol (FCP), for example. Thus, as seen from an operating system on a node 116, 118, the data storage devices 128, 130 can appear as locally attached to the operating system. In this manner, different nodes 116, 118, etc. may access data blocks through the operating system, rather than expressly requesting abstract files.

[029] It should be appreciated that, while the example embodiment 100 illustrates an equal number of network and data modules, other embodiments may comprise a differing number of these modules. For example, there may be a plurality of network and data modules interconnected in a cluster that does not have a one-to-one correspondence between the network and data modules. That is, different nodes can have a different number of network and data modules, and the same node can have a different number of network modules than data modules.

[030] Further, a host device 108, 110 can be networked with the nodes 116, 118 in the cluster, over the networking connections 112, 114. As an example, respective host devices 108, 110 that are networked to a cluster may request services (e.g., exchanging of information in the form of data packets) of a node 116, 118 in the cluster, and the node 116, 118 can return results of the requested services to the host devices 108, 110. In one embodiment, the host devices 108, 110 can exchange information with the network modules 120, 122 residing in the nodes (e.g., network hosts) 116, 118 in the data storage systems 102, 104.

[031] In one embodiment, the data storage devices 128, 130 comprise volumes 132, which is an implementation of storage of information onto disk drives or disk arrays or other storage (e.g., flash) as a file-system for data, for example. Volumes

can span a portion of a disk, a collection of disks, or portions of disks, for example, and typically define an overall logical arrangement of file storage on disk space in the storage system. In one embodiment a volume can comprise stored data as one or more files that reside in a hierarchical directory structure within the volume.

[032] Volumes are typically configured in formats that may be associated with particular storage systems, and respective volume formats typically comprise features that provide functionality to the volumes, such as providing an ability for volumes to form clusters. For example, where a first storage system may utilize a first format for their volumes, a second storage system may utilize a second format for their volumes.

[033] In the example environment 100, the host devices 108, 110 can utilize the data storage systems 102, 104 to store and retrieve data from the volumes 132. In this embodiment, for example, the host device 108 can send data packets to the network module 120 in the node 116 within data storage system 102. The node 116 can forward the data to the data storage device 128 using the data module 124, where the data storage device 128 comprises volume 132A. In this way, in this example, the host device can access the storage volume 132A, to store and/or retrieve data, using the data storage system 102 connected by the network connection 112. Further, in this embodiment, the host device 110 can exchange data with the network module 122 in the host 118 within the data storage system 104 (e.g., which may be remote from the data storage system 102). The host 118 can forward the data to the data storage device 130 using the data module 126, thereby accessing volume 132B associated with the data storage device 130.

[034] It may be appreciated that data synchronization may be implemented within the clustered network environment 100. In an example, storage operations, associated with data within the data storage device 128 of the node 116, may be replicated to the data storage device 130 of the node 118. For example, a catchup synchronization phase may use an in-flight log and/or a dirty region log (e.g., used to track storage operations received by the node 116 and/or regions within the data storage device 128 modified by storage operations but not yet replicated to the data storage device 130) to asynchronously replicate data to the data storage device 130. It may be appreciated that data replication may be implemented for and/or between any type of computing environment, and may be transferrable between physical devices (e.g., node 116, node 118, etc.) and/or a cloud computing environment (e.g., remote to the clustered network environment 100).

[035] Fig. 2 is an illustrative example of a data storage system 200 (e.g., 102, 104 in Fig. 1), providing further detail of an embodiment of components that may implement one or more of the techniques and/or systems described herein. The example data storage system 200 comprises a node 202 (e.g., host nodes 116, 118 in Fig. 1), and a data storage device 234 (e.g., data storage devices 128, 130 in Fig. 1). The node 202 may be a general purpose computer, for example, or some other computing device particularly configured to operate as a storage server. A host device 205 (e.g., 108, 110 in Fig. 1) can be connected to the node 202 over a network 216, for example, to provides access to files and/or other data stored on the data storage device 234. In an example, the node 202 comprises a storage controller that provides client devices, such as the host device 205, with access to data stored within data storage device 234.

[036] The data storage device 234 can comprise mass storage devices, such as disks 224, 226, 228 of a disk array 218, 220, 222. It will be appreciated that the techniques and systems, described herein, are not limited by the example embodiment. For example, disks 224, 226, 228 may comprise any type of mass storage devices, including but not limited to magnetic disk drives, flash memory, and any other similar media adapted to store information, including, for example, data (D) and/or parity (P) information.

[037] The node 202 comprises one or more processors 204, a memory 206, a network adapter 210, a cluster access adapter 212, and a storage adapter 214 interconnected by a system bus 242. The storage system 200 also includes an operating system 208 installed in the memory 206 of the node 202 that can, for example, implement a Redundant Array of Independent (or Inexpensive) Disks (RAID) optimization technique to optimize a reconstruction process of data of a failed disk in an array.

[038] The operating system 208 can also manage communications for the data storage system, and communications between other data storage systems that may be in a clustered network, such as attached to a cluster fabric 215 (e.g., 106 in Fig. 1). Thus, the node 202, such as a network storage controller, can respond to host device requests to manage data on the data storage device 234 (e.g., or additional clustered devices) in accordance with these host device requests. The operating system 208 can often establish one or more file systems on the data storage system 200, where a file

system can include software code and data structures that implement a persistent hierarchical namespace of files and directories, for example. As an example, when a new data storage device (not shown) is added to a clustered network system, the operating system 208 is informed where, in an existing directory tree, new files associated with the new data storage device are to be stored. This is often referred to as "mounting" a file system.

[039] In the example data storage system 200, memory 206 can include storage locations that are addressable by the processors 204 and adapters 210, 212, 214 for storing related software application code and data structures. The processors 204 and adapters 210, 212, 214 may, for example, include processing elements and/or logic circuitry configured to execute the software code and manipulate the data structures. The operating system 208, portions of which are typically resident in the memory 206 and executed by the processing elements, functionally organizes the storage system by, among other things, invoking storage operations in support of a file service implemented by the storage system. It will be apparent to those skilled in the art that other processing and memory mechanisms, including various computer readable media, may be used for storing and/or executing application instructions pertaining to the techniques described herein. For example, the operating system can also utilize one or more control files (not shown) to aid in the provisioning of virtual machines.

[040] The network adapter 210 includes the mechanical, electrical and signaling circuitry needed to connect the data storage system 200 to a host device 205 over a computer network 216, which may comprise, among other things, a point-to-point connection or a shared medium, such as a local area network. The host device 205 (e.g., 108, 110 of Fig. 1) may be a general-purpose computer configured to execute applications. As described above, the host device 205 may interact with the data storage system 200 in accordance with a client/host model of information delivery.

[041] The storage adapter 214 cooperates with the operating system 208 executing on the node 202 to access information requested by the host device 205 (e.g., access data on a storage device managed by a network storage controller). The information may be stored on any type of attached array of writeable media such as magnetic disk drives, flash memory, and/or any other similar media adapted to store information. In the example data storage system 200, the information can be stored in data blocks on the disks 224, 226, 228. The storage adapter 214 can include

input/output (I/O) interface circuitry that couples to the disks over an I/O interconnect arrangement, such as a storage area network (SAN) protocol (e.g., Small Computer System Interface (SCSI), iSCSI, hyperSCSI, Fiber Channel Protocol (FCP), a network file system (NFS) protocol, etc.). The information is retrieved by the storage adapter 214 and, if necessary, processed by the one or more processors 204 (or the storage adapter 214 itself) prior to being forwarded over the system bus 242 to the network adapter 210 (and/or the cluster access adapter 212 if sending to another node in the cluster) where the information is formatted into a data packet and returned to the host device 205 over the network connection 216 (and/or returned to another node attached to the cluster over the cluster fabric 215).

[042] In one embodiment, storage of information on arrays 218, 220, 222 can be implemented as one or more storage "volumes" 230, 232 that are comprised of a cluster of disks 224, 226, 228 defining an overall logical arrangement of disk space. The disks 224, 226, 228 that comprise one or more volumes are typically organized as one or more groups of RAIDs. As an example, volume 230 comprises an aggregate of disk arrays 218 and 220, which comprise the cluster of disks 224 and 226.

[043] In one embodiment, to facilitate access to disks 224, 226, 228, the operating system 208 may implement a file system (e.g., write anywhere file system) that logically organizes the information as a hierarchical structure of directories and files on the disks. In this embodiment, respective files may be implemented as a set of disk blocks configured to store information, whereas directories may be implemented as specially formatted files in which information about other files and directories are stored.

[044] Whatever the underlying physical configuration within this data storage system 200, data can be stored as files within physical and/or virtual volumes, which can be associated with respective volume identifiers, such as file system identifiers (FSIDs), which can be 32-bits in length in one example.

[045] A physical volume corresponds to at least a portion of physical storage devices whose address, addressable space, location, etc. doesn't change, such as at least some of one or more data storage devices 234 (e.g., a Redundant Array of Independent (or Inexpensive) Disks (RAID system)). Typically the location of the physical volume doesn't change in that the (range of) address(es) used to access it generally remains constant.

[046] A virtual volume, in contrast, is stored over an aggregate of disparate portions of different physical storage devices. The virtual volume may be a collection of different available portions of different physical storage device locations, such as some available space from each of the disks 224, 226, and/or 228. It will be appreciated that since a virtual volume is not "tied" to any one particular storage device, a virtual volume can be said to include a layer of abstraction or virtualization, which allows it to be resized and/or flexible in some regards.

[047] Further, a virtual volume can include one or more logical unit numbers (LUNs) 238, directories 236, Qtrees 235, and files 240. Among other things, these features, but more particularly LUNS, allow the disparate memory locations within which data is stored to be identified, for example, and grouped as data storage unit. As such, the LUNs 238 may be characterized as constituting a virtual disk or drive upon which data within the virtual volume is stored within the aggregate. For example, LUNs are often referred to as virtual drives, such that they emulate a hard drive from a general purpose computer, while they actually comprise data blocks stored in various parts of a volume.

[048] In one embodiment, one or more data storage devices 234 can have one or more physical ports, wherein each physical port can be assigned a target address (e.g., SCSI target address). To represent respective volumes stored on a data storage device, a target address on the data storage device can be used to identify one or more LUNs 238. Thus, for example, when the node 202 connects to a volume 230, 232 through the storage adapter 214, a connection between the node 202 and the one or more LUNs 238 underlying the volume is created.

[049] In one embodiment, respective target addresses can identify multiple LUNs, such that a target address can represent multiple volumes. The I/O interface, which can be implemented as circuitry and/or software in the storage adapter 214 or as executable code residing in memory 206 and executed by the processors 204, for example, can connect to volume 230 by using one or more addresses that identify the LUNs 238.

[050] It may be appreciated that data synchronization may be implemented for the data storage system 200. In an example, storage operations, associated with data within the data storage devices 234 of the node 202, may be replicated to a second data storage device of a second node. For example, a catchup synchronization phase may use an in-flight log and/or a dirty region log (e.g., used to track storage

operations received by the node 202 and/or regions within the data storage devices 234 modified by storage operations not yet replicated to the second data storage device) to asynchronously replicate data to the second data storage device. It may be appreciated that data synchronization may be implemented for and/or between any type of computing environment, and may be transferrable between physical devices (e.g., node 202, host 205, etc.) and/or a cloud computing environment (e.g., remote to the node 202 and/or the host 205).

[051] One embodiment of data synchronization is illustrated by an exemplary method 300 of Fig. 3. A first storage node may be configured to provide clients with access to data stored within first storage. A second storage node may be configured to provide clients with access to data stored within second storage or other storage. The first storage node and the second storage node may be configured according to a disaster recovery relationship. For example, data from the first storage may be replicated to the second storage so that the second storage node may provide clients with failover access to replicated data within the second storage in the event the first storage node fails. While in a synchronous state, a storage operation, received by the first storage node from a client, may be synchronously stored within the first storage and replicated to the second storage before a storage operation complete response is provided back to the client (e.g., a storage operation may be split into two storage operations that may be committed to the first storage and the second storage in any order or in parallel). In an example, an in-flight log, used to track storage operations received by the first storage node but not yet committed to both the first storage and the second storage, may be maintained (e.g., a storage operation may be implemented to simultaneously write to the first storage and create an entry within the in-flight log). In an example, the in-flight log may be maintained as an in-core version (e.g., stored within memory), and may be persisted to an in-flight metadata file within a file system (e.g., persistent onto disk for surviving a failure).

[052] However, synchronous replication may be impractical under certain situations, such as where the second storage node fails or reboots, communication from the first storage node to the second storage node is lost, a relatively high network latency, etc. At 302, the first storage of the first storage node and the second storage of the second storage node may be determined as having transitioned into an out-of-sync state (e.g., an inability to synchronously commit storage operations to the first

storage and replicate to the second storage before responding back to clients within an acceptable time frame).

[053] While in the out-of-sync state, a dirty region log may be used to track storage operations to the first storage that are committed by the first storage node but are not committed by the second storage node for replication (e.g., storage operations, received while communication to the second storage node is unavailable, that are committed by the first storage node and indicated as complete to clients notwithstanding not being replicated to the second storage), at 304. In an example, while in the out-of-sync state, if a size of a file grows beyond a threshold size, then a size of the file at a time of transitioning into the out-of-sync state may be captured so that subsequent changes beyond that size are not tracked and thus anything beyond that size is implicitly dirty. In an example, the dirty region log may be persisted to disk, such as to a private inode space of a volume or a metadata file of a file system of the first storage node. In an example, the dirty region log may be maintained as a bitmap comprising bitmap entries corresponding to regions within the first storage. When a storage operation is committed by the first storage node to modify a region, a bit of a corresponding bitmap entry may be modified (e.g., changed from a "1" to a "0" or from a "0" to a "1") to indicate that the region is now a dirty region that has been modified and thus is not consistent with a corresponding region within the second storage because the storage operation (e.g., the dirty region) has not yet been replicated to the second storage. In this way, the dirty region log may be evaluated to identify dirty regions, modified by storage operations, not yet replicated from the first storage to the second storage. In an example, various versions of the dirty region may be maintained as a set of dirty region logs. A current version of the dirty region log may be either rolled forward to a new version or rolled back to an older version.

[054] At 306, a catchup synchronization phase (e.g., asynchronous replication of storage operations not yet committed to the second storage may be performed after communication to the second storage node becomes available; dirty regions within the first storage may be replicated to the second storage; etc.) may be performed to synchronize storage operations, designated as being associated with dirty regions of the first storage by the dirty region log, from the first storage to the second storage (e.g., replication of the dirty regions to corresponding regions within the second storage). In an example, the dirty region log and/or the in-flight log may be evaluated to determine whether to replicate a region of the first storage to the second storage.

For example, catchup synchronization functionality (e.g., asynchronous replication) may be used to replicate a region based upon at least one of the dirty region log indicating that the region is dirty (e.g., has been modified by a storage operation that has not been replicated to the second storage) or the in-flight log indicating that a storage operation, associated with the region, has not been committed to both the first storage and the second storage. In an example, a resync scanner may be invoked to evaluate the dirty region log and/or the in-flight log to identify one or more regions within the first storage to replicate to the second storage using catchup synchronization functionality. Responsive to synchronizing a region, specified as dirty by a dirty region entry within the dirty region log, as committed to the second storage, the dirty region entry may be cleared.

[055] During the catchup synchronization phase, new storage operations may be received and processed by the first storage node. If a new storage operation corresponds to a dirty region identified by the dirty region log, then the first storage operation may be committed to the dirty region within the first storage (e.g., and an operation complete response may be provided back to a client that issued the new storage operation) for subsequent synchronization by catchup synchronization functionality (e.g., asynchronous replication by the resync scanner). If the new storage operation corresponds to a non-dirty region within the first storage, then the new storage operation may be synchronously, such as in real-time, committed to the first storage and replicated to the second storage (e.g., and an operation complete response may be provided back to a client that issued the new storage operation and the new storage operation is committed to both the first storage and the second storage).

[056] In an example, the dirty region log and/or the in-flight log may be used to reconcile snapshots. For example, a first snapshot of the first storage may be generated. A second snapshot of the second storage may be generated, such as while the storage nodes are in the out-of-sync state, and thus the second snapshot may be different than the first snapshot because one or more storage operations may have been committed to the first storage but not the second storage. In an example, storage operation processing may be facilitated during the generation of the first snapshot and/or the second snapshot (e.g., snapshots may be generated without draining storage operations, thus reducing latency increases otherwise caused by draining storage operations). The dirty region log and/or the in-flight log may be used to identify a

snapshot difference between the first snapshot and the second snapshot. The second snapshot may be modified based upon the snapshot difference to create a reconciled snapshot having a data consistency with the first snapshot. In this way, the dirty region log and/or the in-flight log may be used to track data consistency between the first storage and the second storage for replication and/or snapshots.

[057] Figs. 4A-4I illustrate examples of a computing device comprising a system 400 for data synchronization. Fig. 4A illustrates a first storage node 402 configured to provide clients with access to data stored within first storage 404 (e.g., local storage locally attached to the first storage node 404, network storage communicatively coupled to the first storage node 404 over a network, etc.) and/or other storage not illustrated. A second storage node 412 is configured to provide clients to access to data stored within second storage 414 (e.g., provide clients with failover access to replicated data within the second storage 414) and/or other storage not illustrated (e.g., provide clients with primary access to data within third storage). In an example, the second storage 414 may be used to store replicated data that is replicated, such as synchronously replicated 416 over a network 410, from the first storage 404 to the second storage 414. For example, a first region comprising data (A), a second region that is empty, a third region that is empty, a fourth region comprising data (B), a fifth region comprising data (C), and/or other regions of the first storage 404 may be replicated to the second storage 414 as replicated regions. In this way, the first storage 404 and the second storage 414 may be in a synchronized state.

[058] An in-flight log 406 may be maintained for the first storage node 402. The in-flight log 406 may track storage operations that are received by the first storage node 402, but are not yet committed by both the first storage node 402 to the first storage 404 and the second storage node 412 to the second storage 414. For example, a first storage operation and a second storage operation may have been received by the first storage node 402, and thus a first entry for the first storage operation and a second entry for the second storage operation may have been created in the in-flight log 406 to indicate that the first storage operation and the second storage operation have not yet been committed to both the first storage 404 and the second storage 414. Accordingly, once the first storage node 402 commits the first storage operation and the second storage operation to the first storage 404 and the synchronous replication 416 replicates the first storage operation and the second storage operation to the second storage node 412 that then commits the first storage operation and the second

storage operation to the second storage 414, the first entry and the second entry may be modified (e.g., removed, zeroed out, etc.) to indicate that the storage operations have been synchronously committed to the first storage 404 and the second storage 414. Thus, the first storage node 402 may provide storage operation complete notifications to clients that sent the storage operations. In an example, a third storage operation may be received by the first storage node 402, and a third entry may be created (e.g., set to "1") within the in-flight log 406 to indicate that the third storage operation has not been committed to both the first storage 404 and the second storage 414.

[059] Fig. 4B illustrates the first storage node 402 and the second storage node 412 committing the third storage operation. Accordingly, the third entry within the in-flight log 406 may be modified (e.g., removed, zeroed out, etc.), thus indicating that the third storage operation has been synchronized between the first storage 404 and the second storage 414. Subsequently, a communication loss 422, over the network 410, between the first storage node 402 and the second storage node 412 may occur. Thus, the synchronous replication 416, over the network 410, may be lost and the first storage 404 and the second storage 414 may become out-of-sync due to the first storage node 402 processing storage operations that are unable to be adequately replicated to the second storage node 412 and second storage 414.

[060] Fig. 4C illustrates the first storage node 402 and the second storage node 412 transitioning into an out-of-sync state, where the first storage 404 is out-of-sync with the second storage 414 (e.g., a loss of data consistency because storage operations may be processed and responded back to clients as complete by the first storage node 402 without synchronous replication to the second storage node 412). For example, a client may send a forth storage operation 430 to the first storage node 402 for writing data (D) into the third region 432 of the first storage 404. A fourth entry, not illustrated, may be created within the in-flight log 406 to indicate that the fourth storage operation 430 was received by the first storage node 402, but has not been committed to both the first storage 404 and the second storage 414. A bit, of a third dirty region entry 434, within the dirty region log 408, associated with the third region 432, may be set to indicate that the third region 432 of the first storage 404 has been modified (e.g., set to 1 to indicate that the third region 432 has been made dirty with the data (D) of the fourth storage operation 430). In this way, third dirty region entry 434 of the dirty region log 408 indicates that the third region 432 of the first

storage 404 is out-of-sync and/or inconsistent with a corresponding third region 433 within the second storage 414.

[061] Fig. 4D illustrates the first storage node 402 and the second storage node 412 in the out-of-sync state. In an example, a second client may send a fifth storage operation 440 to the first storage node 402 for writing data (E) into the second region 442 of the first storage 404. A fifth entry, not illustrated, may be created within the in-flight log 406 to indicate that the fifth storage operation 440 was received by the first storage node 402, but has not been committed to both the first storage 404 and the second storage 414. A bit, of a second dirty region entry 444, of the dirty region log 408, associated with the second region 442, may be set to indicate that the second region 442 of the first storage 404 has been modified (e.g., set to 1 to indicate that the second region 442 has been made dirty with the data (E) of the fifth storage operation 440). In this way, second dirty region entry 444 of the dirty region log 408 indicates that the second region 442 of the first storage 404 is out-of-sync and/or inconsistent with a corresponding third region 435 within the second storage 414.

[062] Fig. 4E illustrates communication being reestablished over the network 410 between the first storage node 402 and the second storage node 412. Fig. 4F illustrates the first storage node 402 and the first storage 404 being out-of-sync with the second storage node 412 and the second storage 414 after the communication has been reestablished, and thus a catchup synchronization phase may be performed, such as asynchronous replication by a resync scanner, to synchronize storage operations, such as the fourth storage operation 430 (e.g., replication of data (D) written to the third region 432 by the fourth storage operation 430) and the fifth storage operation 440 (e.g., replication of data (E) written to the second region 443 by the fifth storage operation 440), based upon the in-flight log 406 and/or the dirty region log 408. For example, catchup synchronization 450 of the data (D) within the third region 432 of the first storage 404 may be performed to asynchronously replicate the data (D), written by the fourth storage operation 430 to the third region 432, into the corresponding third region 433 within the second storage 414 of the second storage node 412 as replicated data (D) 452 based upon at least one of the in-flight log 406 indicating that the fourth storage operation 430 was received by the first storage node 402 and has not been committed to both the first storage 404 and the second storage 412 or the third dirty region entry 434 of the dirty region log 408 indicating that the third region 432 of the first storage 404 is out-of-sync and/or inconsistent with the

corresponding third region 433 within the second storage 414. Once the third region 432 of the first storage 404 has been replicated to the second storage 414, the bit of the third dirty region entry 434 may be modified (e.g., removed, zeroed out, etc.) to indicate that the third region 432 is no longer dirty and thus is data consistent with the corresponding third region 433 of the second storage 414.

[063] Fig. 4G illustrates the first storage node 402 receiving a sixth storage operation 460 from a third client. The sixth storage operation 460 may write data (F) into a sixth region 462 of the first storage 404. Because the dirty region log indicates that the sixth region 462 is a non-dirty region, the sixth storage operation 460 may be synchronously committed by the first storage node 402 to the first storage 404 by writing the data (F) into the sixth region 462 and replicating (e.g., real-time synchronization 466) to the second storage 414 such as into a corresponding sixth region 467 as replicated data (F) 468, as illustrated by Fig. 4H. In an example where a storage operation corresponds to a dirty region, such as the second region 442, the storage operation may be committed into the first storage 404 and may be later synchronized to the second storage 414 by a resync scanner because the second region 442 is already indicated as being dirty by the dirty region log 408.

[064] Fig. 4I illustrates catchup synchronization 470 of the data (E) within the second region 442 of the first storage 404 being performed to asynchronously replicate the data (E), written by the fifth storage operation 440 to the second region 442, into the corresponding second region 435 within the second storage 414 of the second storage node 412 as replicated data (E) 472 based upon at least one of the in-flight log 406 indicating that the fifth storage operation 440 was received by the first storage node 402 and has not been committed to both the first storage 404 and the second storage 412 or the second dirty region entry 444 of the dirty region log 408 indicating that the second region 442 of the first storage 404 is out-of-sync and/or inconsistent with the corresponding second region 435 within the second storage 414. Once the second region 442 of the first storage 404 has been replicated to the second storage 414, the bit of the second dirty region entry 444 may be modified (e.g., removed, zeroed out, etc.) to indicate that the second region 442 is no longer dirty and thus is data consistent with the corresponding second region 435 of the second storage 414.

[065] Fig. 5 illustrates an example of a system 500 for snapshot reconciliation. A first storage node 502 may be associated with first storage 504 that is out-of-sync

with second storage 512 of a second storage node 510 having a disaster recovery relationship with the first storage node 502. An in-flight log 506 and/or a dirty region log 508 may be maintained for the first storage node 502 to indicate what storage operations have not been committed to both the first storage 504 and the second storage 512 and/or to indicate what regions within the first storage 504 are dirty because data within such regions have not been replicated to the second storage 512.

[066] While the first storage 504 and the second storage 512 are out-of-sync, a first snapshot 514 of the first storage 504 and a second snapshot 516 of the second storage 512 may be generated. The in-flight log 506 and/or the dirty region log 508 may be used to compare 518 the first snapshot 514 and the second snapshot 516 to identify a snapshot difference 520 between the first snapshot 514 and the second snapshot 516. The snapshot difference 502 may correspond to data within the first storage 504 that is not data consistent with the second storage 512, such as due to storage operation that were committed to the first storage 504 but not yet replicated to the second storage 512 while the first storage node 502 and the second storage node 510 are in an out-of-sync state. Accordingly, the second snapshot 516 may be modified based upon the snapshot difference 520 to create a reconciled snapshot 522 that has data consistency with the first snapshot 514.

[067] Still another embodiment involves a computer-readable medium comprising processor-executable instructions configured to implement one or more of the techniques presented herein. An example embodiment of a computer-readable medium or a computer-readable device that is devised in these ways is illustrated in Fig. 6, wherein the implementation 600 comprises a computer-readable medium 608, such as a CD-R, DVD-R, flash drive, a platter of a hard disk drive, etc., on which is encoded computer-readable data 606. This computer-readable data 606, such as binary data comprising at least one of a zero or a one, in turn comprises a set of computer instructions 604 configured to operate according to one or more of the principles set forth herein. In some embodiments, the processor-executable computer instructions 604 are configured to perform a method 602, such as at least some of the exemplary method 300 of Fig. 3, for example. In some embodiments, the processor-executable instructions 604 are configured to implement a system, such as at least some of the exemplary system 400 of Figs. 4A-4I and/or at least some of the exemplary system 500 of Fig. 5, for example. Many such computer-readable media are contemplated to operate in accordance with the techniques presented herein.

[068] It will be appreciated that processes, architectures and/or procedures described herein can be implemented in hardware, firmware and/or software. It will also be appreciated that the provisions set forth herein may apply to any type of special-purpose computer (e.g., file host, storage server and/or storage serving appliance) and/or general-purpose computer, including a standalone computer or portion thereof, embodied as or including a storage system. Moreover, the teachings herein can be configured to a variety of storage system architectures including, but not limited to, a network-attached storage environment and/or a storage area network and disk assembly directly attached to a client or host computer. 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 systems.

[069] In some embodiments, methods described and/or illustrated in this disclosure may be realized in whole or in part on computer-readable media. Computer readable media can include processor-executable instructions configured to implement one or more of the methods presented herein, and may include any mechanism for storing this data that can be thereafter read by a computer system. Examples of computer readable media include (hard) drives (e.g., accessible via network attached storage (NAS)), Storage Area Networks (SAN), volatile and non-volatile memory, such as read-only memory (ROM), random-access memory (RAM), EEPROM and/or flash memory, CD-ROMs, CD-Rs, CD-RWs, DVDs, cassettes, magnetic tape, magnetic disk storage, optical or non-optical data storage devices and/or any other medium which can be used to store data.

[070] Although the subject matter has been described in language specific to structural features or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing at least some of the claims.

[071] Various operations of embodiments are provided herein. The order in which some or all of the operations are described should not be construed to imply that these operations are necessarily order dependent. Alternative ordering will be appreciated given the benefit of this description. Further, it will be understood that not all operations are necessarily present in each embodiment provided herein. Also,

it will be understood that not all operations are necessary in some embodiments.

[072] Furthermore, the claimed subject matter is implemented as a method, apparatus, or article of manufacture using standard application or engineering techniques to produce software, firmware, hardware, or any combination thereof to control a computer to implement the disclosed subject matter. The term "article of manufacture" as used herein is intended to encompass a computer application accessible from any computer-readable device, carrier, or media. Of course, many modifications may be made to this configuration without departing from the scope or spirit of the claimed subject matter.

[073] As used in this application, the terms "component", "module," "system", "interface", and the like are generally intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component includes a process running on a processor, a processor, an object, an executable, a thread of execution, an application, or a computer. By way of illustration, both an application running on a controller and the controller can be a component. One or more components residing within a process or thread of execution and a component may be localized on one computer or distributed between two or more computers.

[074] Moreover, "exemplary" is used herein to mean serving as an example, instance, illustration, etc., and not necessarily as advantageous. As used in this application, "or" is intended to mean an inclusive "or" rather than an exclusive "or". In addition, "a" and "an" as used in this application are generally be construed to mean "one or more" unless specified otherwise or clear from context to be directed to a singular form. Also, at least one of A and B and/or the like generally means A or B and/or both A and B. Furthermore, to the extent that "includes", "having", "has", "with", or variants thereof are used, such terms are intended to be inclusive in a manner similar to the term "comprising".

[075] Many modifications may be made to the instant disclosure without departing from the scope or spirit of the claimed subject matter. Unless specified otherwise, "first," "second," or the like are not intended to imply a temporal aspect, a spatial aspect, an ordering, etc. Rather, such terms are merely used as identifiers, names, etc. for features, elements, items, etc. For example, a first set of information

and a second set of information generally correspond to set of information A and set of information B or two different or two identical sets of information or the same set of information.

[076] Also, although the disclosure has been shown and described with respect to one or more implementations, equivalent alterations and modifications will occur to others skilled in the art based upon a reading and understanding of this specification and the annexed drawings. The disclosure includes all such modifications and alterations and is limited only by the scope of the following claims. In particular regard to the various functions performed by the above described components (e.g., elements, resources, etc.), the terms used to describe such components are intended to correspond, unless otherwise indicated, to any component which performs the specified function of the described component (e.g., that is functionally equivalent), even though not structurally equivalent to the disclosed structure. In addition, while a particular feature of the disclosure may have been disclosed with respect to only one of several implementations, such feature may be combined with one or more other features of the other implementations as may be desired and advantageous for any given or particular application.

What is claimed is:

1. A method comprising:

determining, by a storage server, that first storage of a first storage node and second storage of a second storage node have transitioned into an out-of-sync state; and

while in the out-of-sync state:

tracking, within a dirty region log, storage operations to the first storage that are committed by the first storage node but are not committed by the second storage node for replication; and

performing a catchup synchronization phase to synchronize storage operations, designated as being associated with dirty regions of the first storage by the dirty region log, from the first storage to the second storage.

2. The method of claim 1, comprising:

maintaining an in-flight log to track storage operations received by the first storage node but not committed to both the first storage and the second storage.

3. The method of claim 2, the performing a catchup synchronization phase comprising:

evaluating the dirty region log and the in-flight log to determine whether to replicate a region of the first storage to the second storage.

4. The method of claim 2 or 3, the performing a catchup synchronization phase comprising:

replicating, using catchup synchronization functionality, a region based upon at least one of the dirty region log indicating that the region is dirty or the in-flight log indicating that a storage operation associated with the region has not been committed to both the first storage and the second storage.

5. The method of any of claims 2 to 4, comprising:

maintaining an in-core version of the in-flight log; and

persisting the in-core version of the in-flight log to an in-flight metadata file within a file system.

6. The method of any preceding claim, the performing a catchup synchronization phase comprising:

receiving a first storage operation;

responsive to the first storage operation corresponding to a dirty region identified by the dirty region log, committing the first storage operation to the dirty region within the first storage for subsequent synchronization by catchup synchronization functionality; and

responsive to the first storage operation corresponding to a non-dirty region within the first storage, synchronously committing the first storage operation to the first storage and replicating the first storage operation to the second storage.

7. The method of any preceding claim, comprising:

maintaining the dirty region log as a bitmap; and

modifying a bit of a bitmap entry, within the bitmap, corresponding to a region within the first storage based upon a storage operation modifying the region.

8. The method of any preceding claim, comprising:

generating a first snapshot of the first storage;

generating a second snapshot of the second storage;

utilizing the dirty region log to identify a snapshot difference between the first snapshot and the second snapshot; and

modifying the second snapshot based upon the snapshot difference to create a reconciled snapshot having data consistency with the first snapshot.

9. The method of claim 8, comprising:

utilizing an in-flight log, used to track storage operations received by the first storage node but not committed to both the first storage and the second storage, to identify the snapshot difference.

10. The method of claim 8 or 9, comprising:

facilitating storage operation processing during at least one of the generating of the first snapshot or the generating of the second snapshot.

11. The method of any preceding claim, comprising:

persisting the dirty region log to at least one of a private inode space of a volume or a metadata file of a file system of the first storage node.

12. The method of any preceding claim, comprising:

maintaining a set of dirty region logs corresponding to one or more versions of the dirty region log.

13. The method of claim 12, comprising:

rolling a current version of the dirty region log forwards to a newer version within the set of dirty region logs.

14. The method of claim 12, comprising:

rolling a current version of the dirty region log backwards to an older version within the set of dirty region logs.

15. The method of any preceding claim, comprising:

responsive to synchronizing a region, specified as dirty by a dirty region entry within the dirty region log, as committed to the second storage, clearing the dirty region entry.

16. The method of any preceding claim, the performing a catchup synchronization phase comprising:

invoking a resync scanner to evaluate the dirty region log to identify one or more regions within the first storage to replicate to the second storage using catchup synchronization functionality.

17. A non-transitory machine readable medium having stored thereon instructions for performing a method comprising machine executable code which when executed by at least one machine, causes the machine to perform the method of any preceding claim.

18. A non-transitory machine readable medium having stored thereon instructions for performing a method comprising machine executable code which when executed by at least one machine, causes the machine to:

maintain an in-flight log to track storage operations received by a first storage node but not committed to both first storage of the first storage node and second storage of a second storage node;

determine that the first storage of the first storage node and the second storage of the second storage node have transitioned into an out-of-sync state; and

while in the out-of-sync state:

track, within a dirty region log, storage operations to the first storage that are committed by the first storage node but are not committed by the second storage node for replication; and

perform a catchup synchronization phase to synchronize storage operations from the first storage to the second storage based upon the dirty region log and the in-flight log.

19. A computing device comprising:

a memory containing machine readable medium comprising machine executable code having stored thereon instructions for performing a method of data synchronization; and

a processor coupled to the memory, the processor configured to execute the machine executable code to cause the processor to:

determine, by a storage server, that first storage of a first storage node and second storage of a second storage node have transitioned into an out-of-sync state; and

while in the out-of-sync state:

track, within a dirty region log, storage operations to the first storage that are committed by the first storage node but are not committed to by second storage node for replication; and

perform a catchup synchronization phase to synchronize storage operations, designated as being associated with dirty regions of the first storage by the dirty region log, from the first storage to the second storage.

20. The computing device of claim 19, wherein the machine executable code causes the processor to:

maintain an in-flight log to track storage operations received by the first storage node but not committed to both the first storage and the second storage; and

replicate, using catchup synchronization functionality, a region based upon at least one of the dirty region log indicating that the region is dirty or the in-flight log indicating that a storage operation associated with the region has not been committed to both the first storage and the second storage.

21. The computing device of claim 19 or 20, wherein the machine executable code causes the processor to:

generate a first snapshot of the first storage;

generate a second snapshot of the second storage;

utilize the dirty region log and an in-flight log to identify a snapshot difference between the first snapshot and the second snapshot; and

modify the second snapshot based upon the snapshot difference to create a reconciled snapshot having data consistency with the first snapshot.

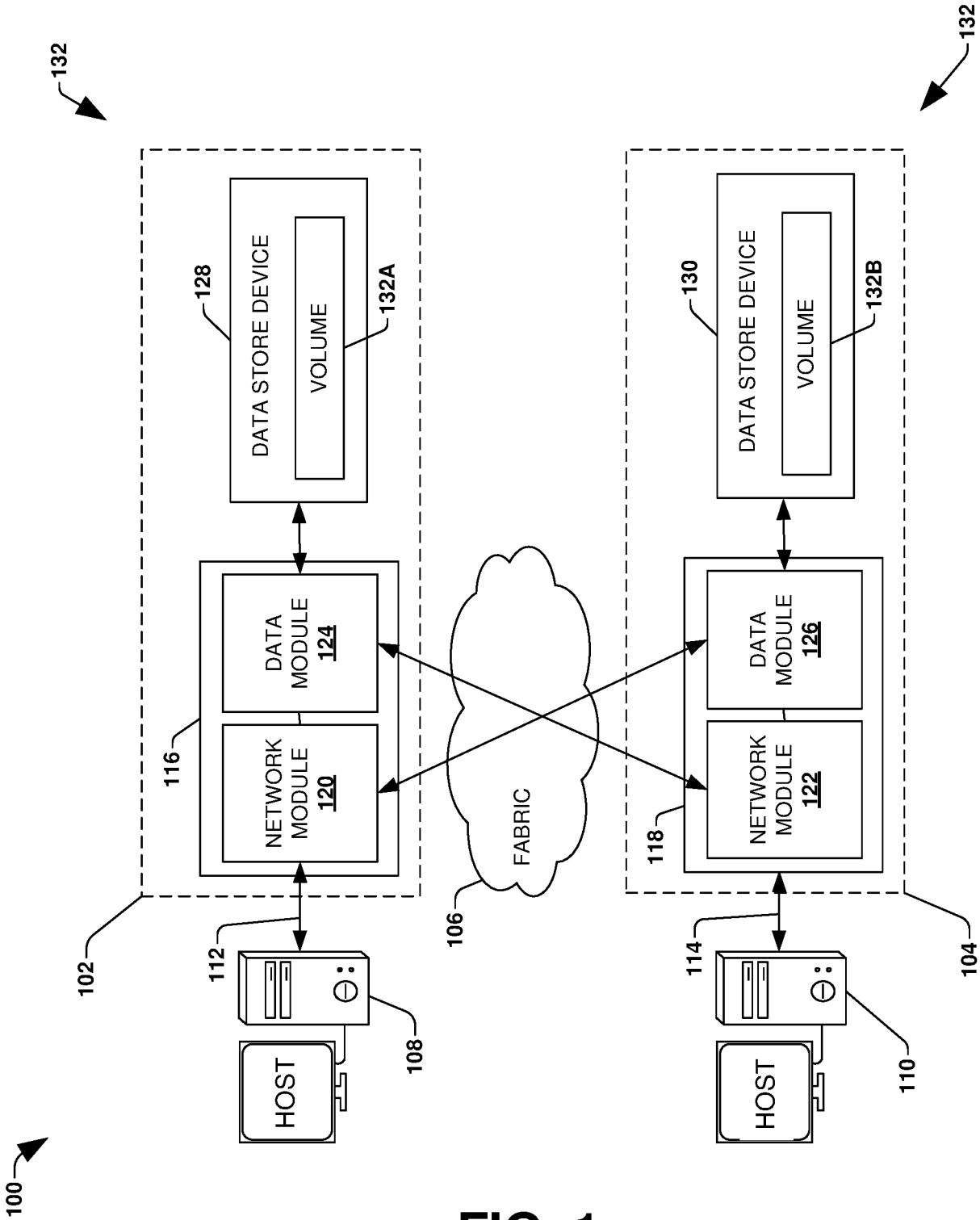

100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
130
132
132A
132B
132
132
DATA STORE DEVICE
VOLUME
DATA STORE DEVICE
VOLUME
DATA MODULE 124
NETWORK MODULE 120
DATA MODULE 126
NETWORK MODULE 122
FABRIC
HOST
HOST

FIG. 1

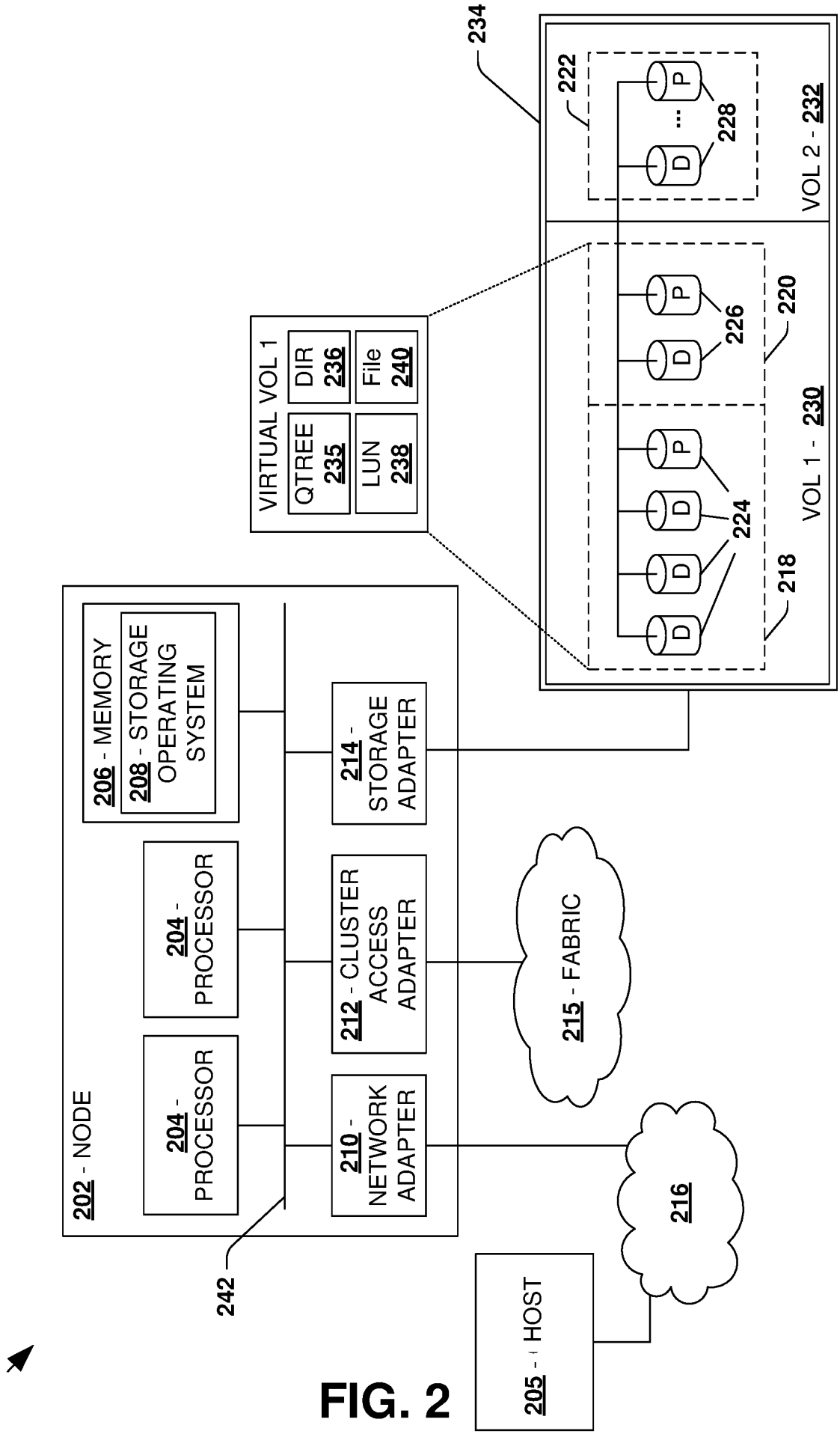

234
222
P
D
228
VOL 2 - 232
220
226
218
224
VOL 1 - 230
VIRTUAL VOL 1
QTREE 235
LUN 238
DIR 236
File 240
206 - MEMORY
208 - STORAGE OPERATING SYSTEM
204 - PROCESSOR
204 - PROCESSOR
212 - CLUSTER ACCESS ADAPTER
214 - STORAGE ADAPTER
210 - NETWORK ADAPTER
202 - NODE
215 - FABRIC
216
205 - HOST
242

FIG. 2

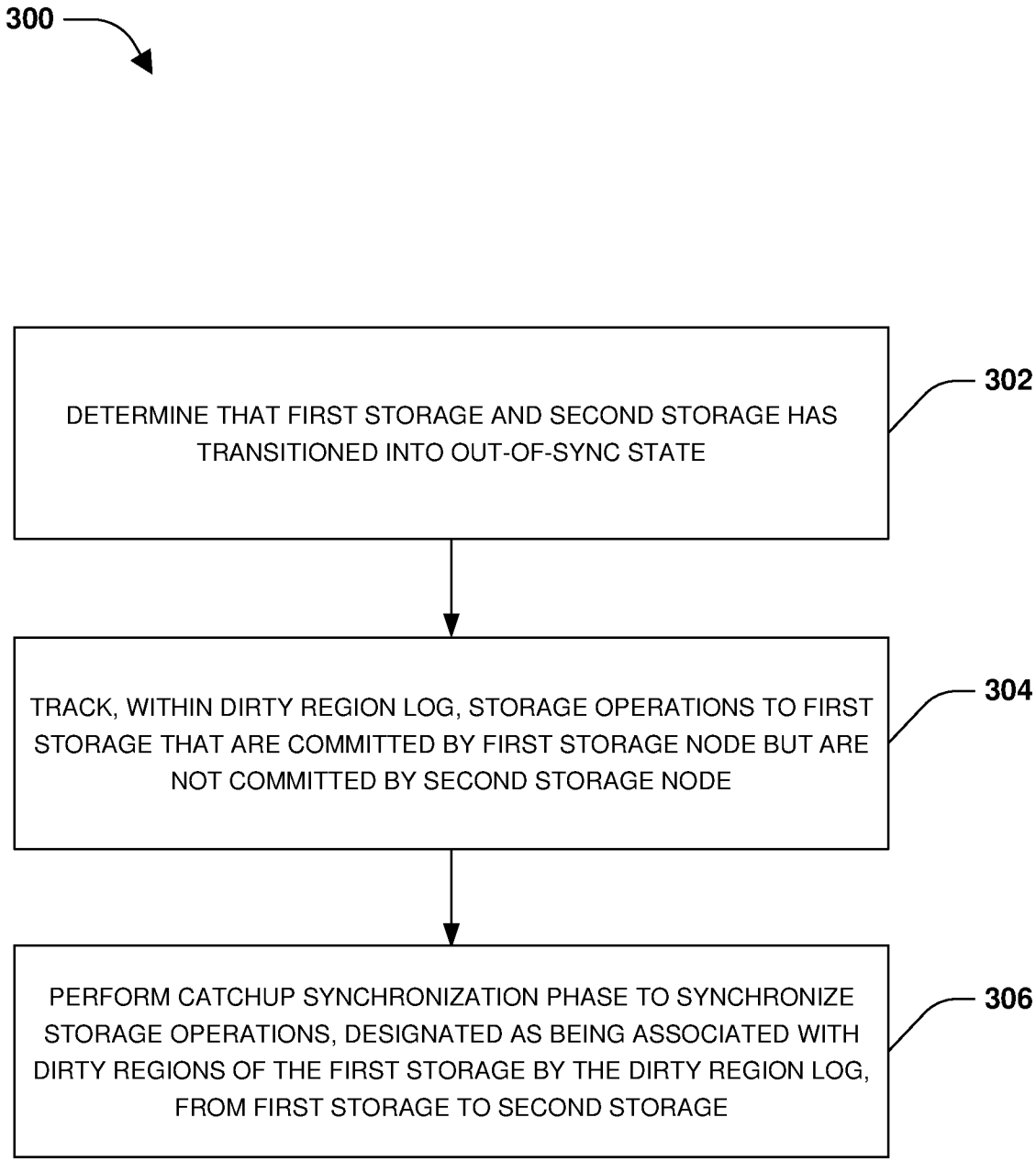

300
DETERMINE THAT FIRST STORAGE AND SECOND STORAGE HAS TRANSITIONED INTO OUT-OF-SYNC STATE
302
TRACK, WITHIN DIRTY REGION LOG, STORAGE OPERATIONS TO FIRST STORAGE THAT ARE COMMITTED BY FIRST STORAGE NODE BUT ARE NOT COMMITTED BY SECOND STORAGE NODE
304
PERFORM CATCHUP SYNCHRONIZATION PHASE TO SYNCHRONIZE STORAGE OPERATIONS, DESIGNATED AS BEING ASSOCIATED WITH DIRTY REGIONS OF THE FIRST STORAGE BY THE DIRTY REGION LOG, FROM FIRST STORAGE TO SECOND STORAGE
306

FIG. 3

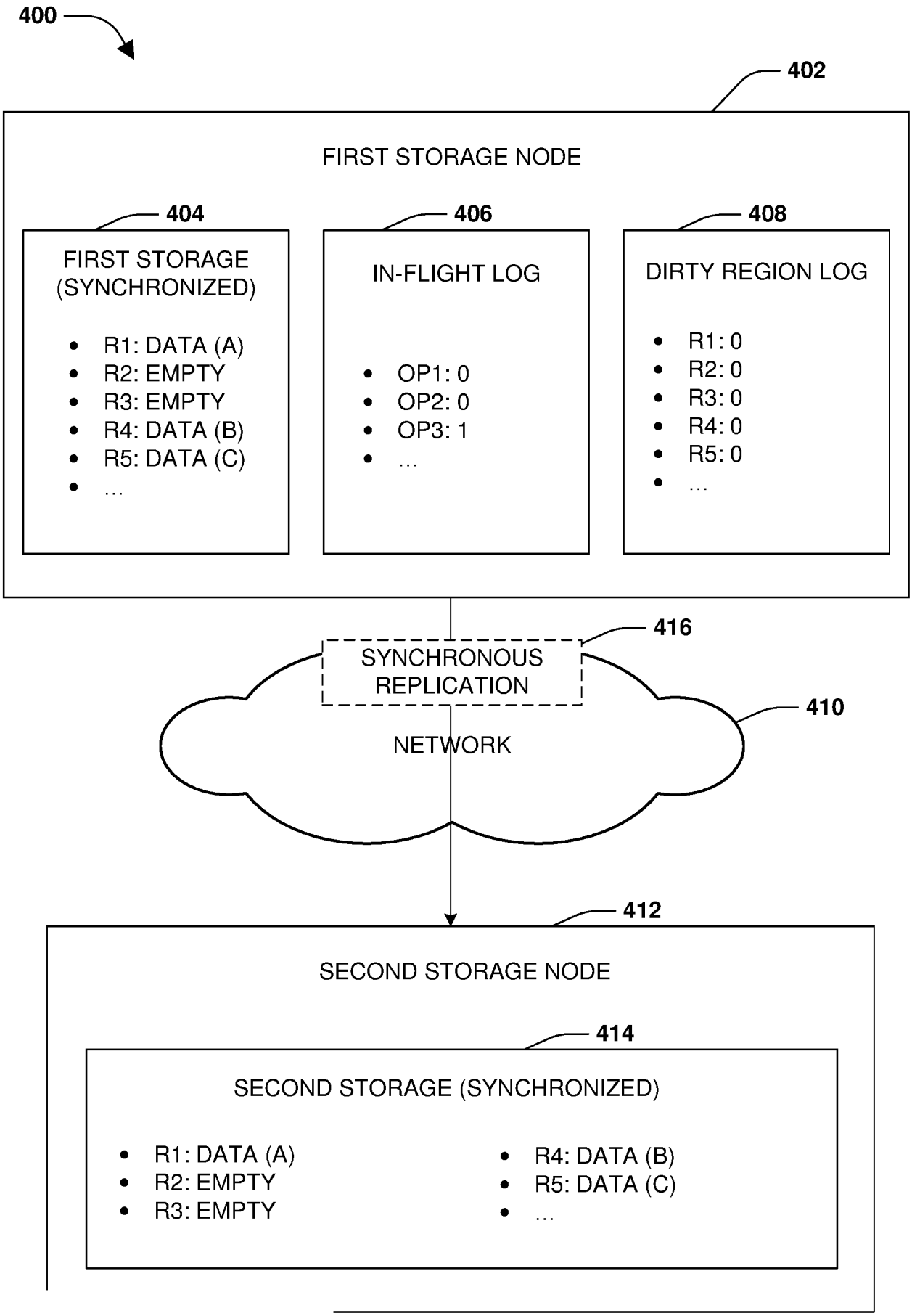

400
402
FIRST STORAGE NODE
404
FIRST STORAGE (SYNCHRONIZED)
R1: DATA (A)
R2: EMPTY
R3: EMPTY
R4: DATA (B)
R5: DATA (C)
…
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 1
…
408
DIRTY REGION LOG
R1: 0
R2: 0
R3: 0
R4: 0
R5: 0
…
416
SYNCHRONOUS REPLICATION
410
NETWORK
412
SECOND STORAGE NODE
414
SECOND STORAGE (SYNCHRONIZED)
R1: DATA (A)
R2: EMPTY
R3: EMPTY
R4: DATA (B)
R5: DATA (C)
…

FIG. 4A

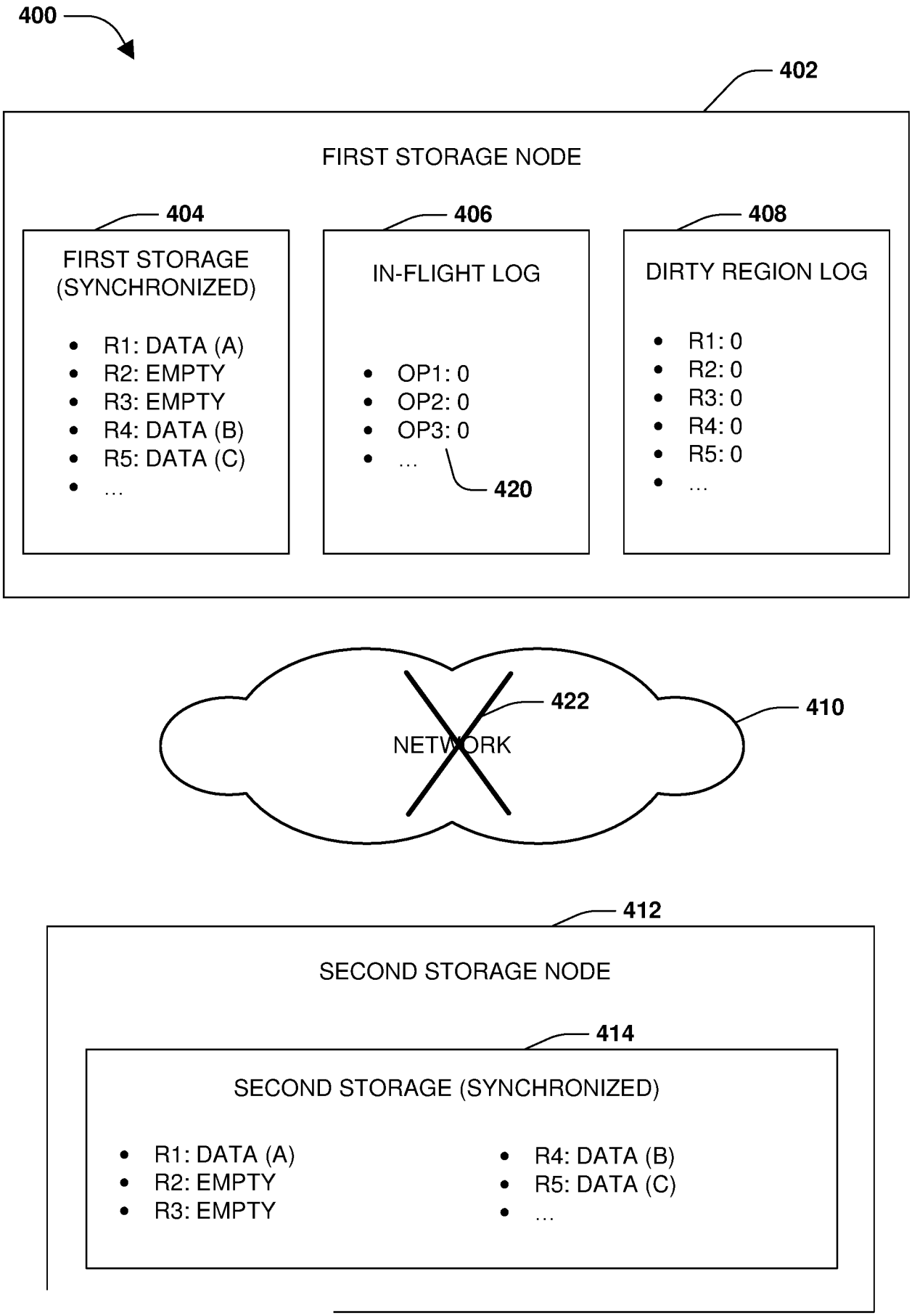

400
402
FIRST STORAGE NODE
404
FIRST STORAGE (SYNCHRONIZED)
R1: DATA (A)
R2: EMPTY
R3: EMPTY
R4: DATA (B)
R5: DATA (C)
…
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
…
420
408
DIRTY REGION LOG
R1: 0
R2: 0
R3: 0
R4: 0
R5: 0
…
422
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (SYNCHRONIZED)
R1: DATA (A)
R2: EMPTY
R3: EMPTY
R4: DATA (B)
R5: DATA (C)
…

FIG. 4B

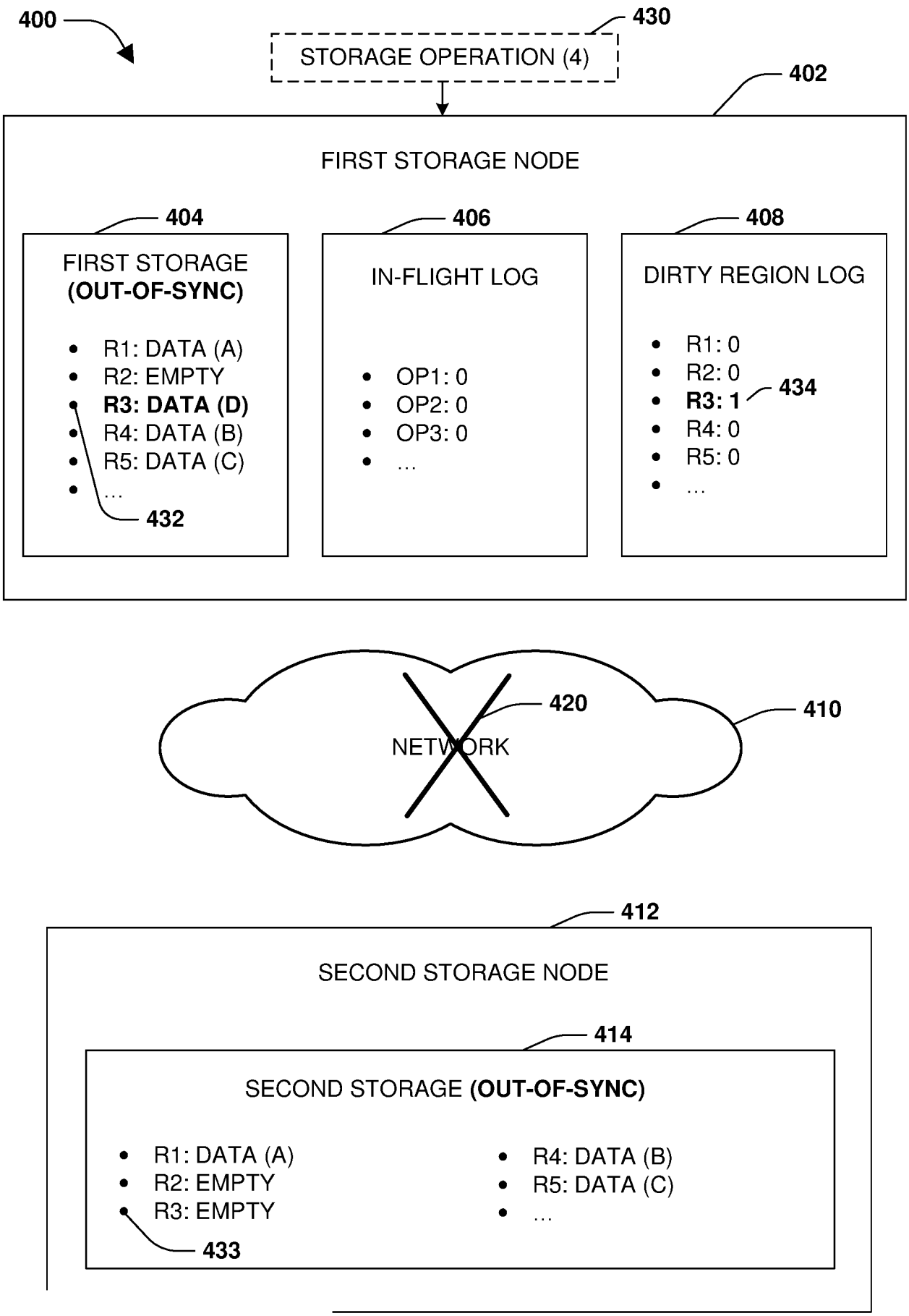

400
430
STORAGE OPERATION (4)
402
FIRST STORAGE NODE
404
FIRST STORAGE
(OUT-OF-SYNC)
R1: DATA (A)
R2: EMPTY
R3: DATA (D)
R4: DATA (B)
R5: DATA (C)
…
432
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
…
408
DIRTY REGION LOG
R1: 0
R2: 0
R3: 1
434
R4: 0
R5: 0
…
420
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (OUT-OF-SYNC)
R1: DATA (A)
R2: EMPTY
R3: EMPTY
433
R4: DATA (B)
R5: DATA (C)
…

FIG. 4C

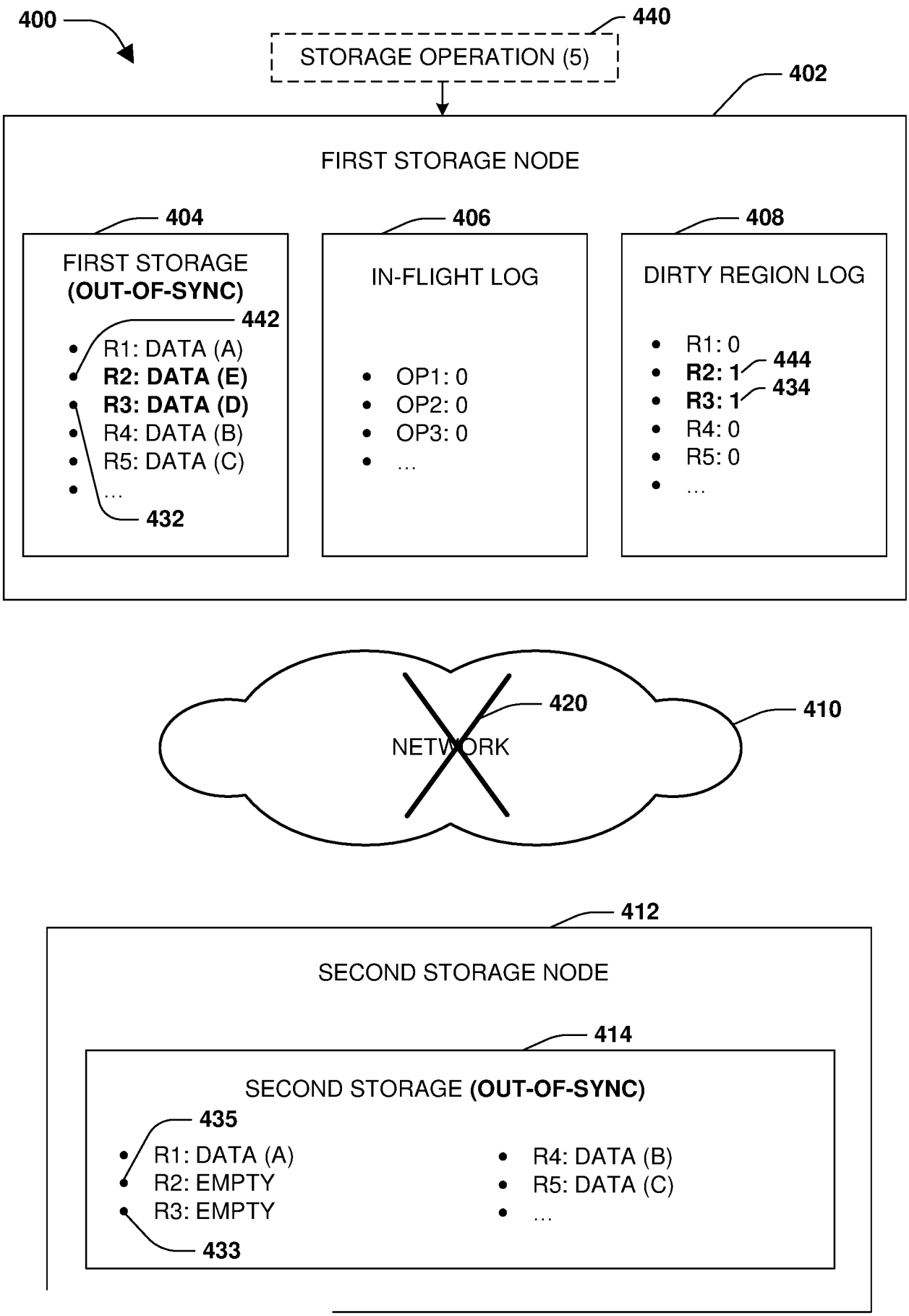

400
440
STORAGE OPERATION (5)
402
FIRST STORAGE NODE
404
FIRST STORAGE
(OUT-OF-SYNC)
442
R1: DATA (A)
R2: DATA (E)
R3: DATA (D)
R4: DATA (B)
R5: DATA (C)
…
432
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
…
408
DIRTY REGION LOG
R1: 0
R2: 1
R3: 1
444
434
R4: 0
R5: 0
…
420
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (OUT-OF-SYNC)
435
R1: DATA (A)
R2: EMPTY
R3: EMPTY
433
R4: DATA (B)
R5: DATA (C)
…

FIG. 4D

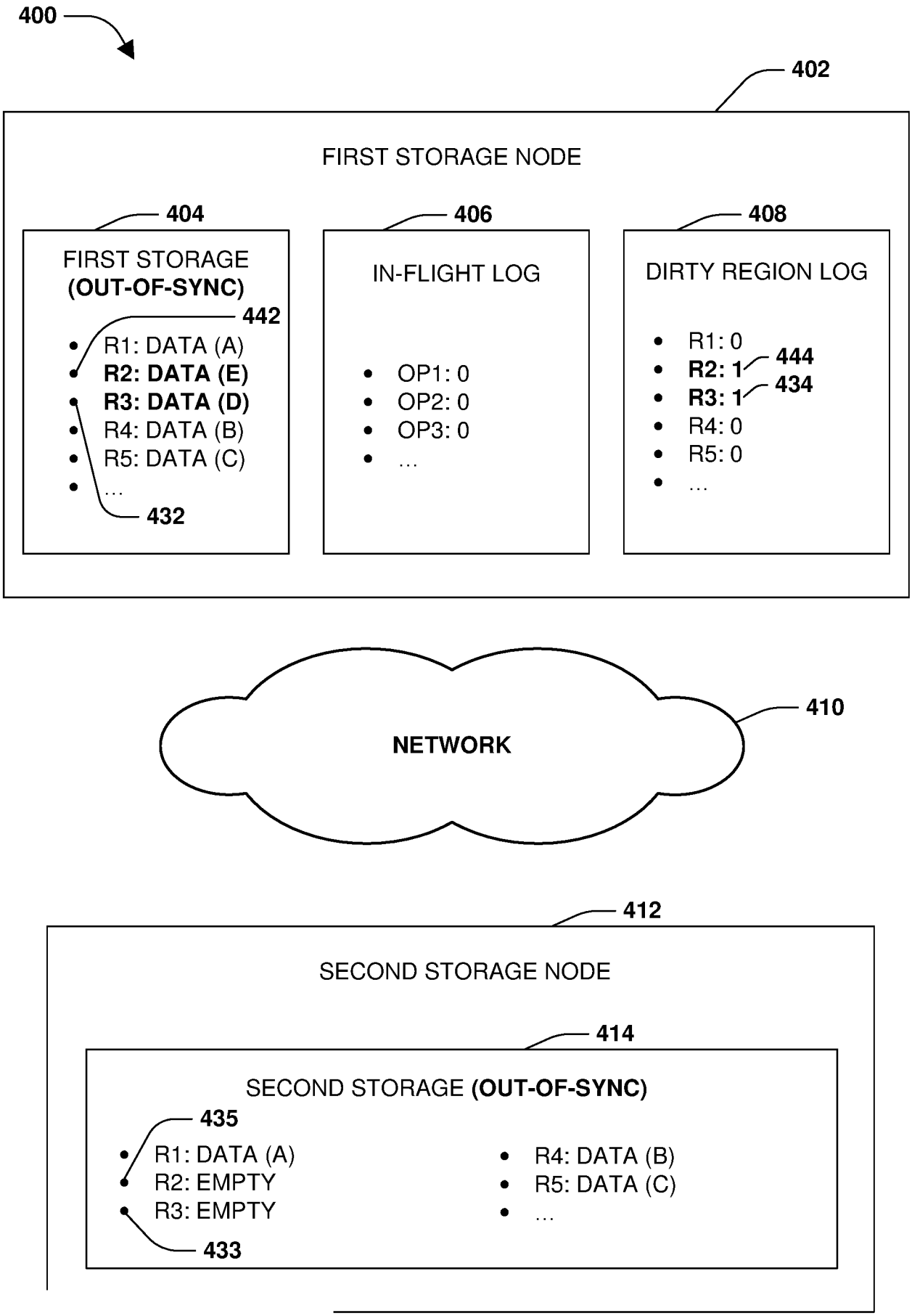

400
402
FIRST STORAGE NODE
404
FIRST STORAGE
(OUT-OF-SYNC)
442
R1: DATA (A)
R2: DATA (E)
R3: DATA (D)
R4: DATA (B)
R5: DATA (C)
…
432
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
…
408
DIRTY REGION LOG
R1: 0
R2: 1
R3: 1
444
434
R4: 0
R5: 0
…
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (OUT-OF-SYNC)
435
R1: DATA (A)
R2: EMPTY
R3: EMPTY
433
R4: DATA (B)
R5: DATA (C)
…

FIG. 4E

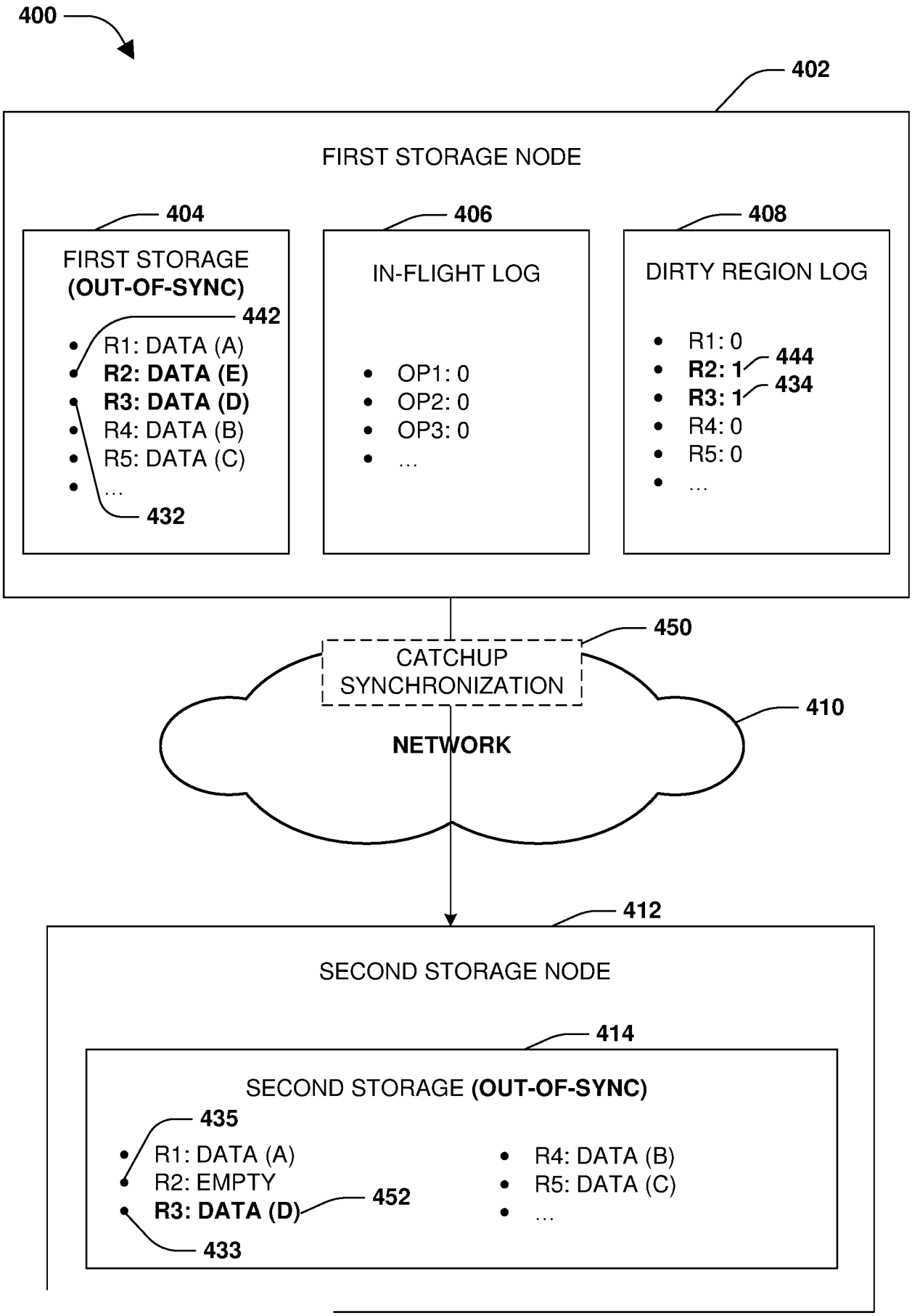

400
402
FIRST STORAGE NODE
404
FIRST STORAGE
(OUT-OF-SYNC)
442
R1: DATA (A)
R2: DATA (E)
R3: DATA (D)
R4: DATA (B)
R5: DATA (C)
…
432
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
…
408
DIRTY REGION LOG
R1: 0
R2: 1
R3: 1
R4: 0
R5: 0
…
444
434
450
CATCHUP
SYNCHRONIZATION
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (OUT-OF-SYNC)
435
R1: DATA (A)
R2: EMPTY
R3: DATA (D)
452
433
R4: DATA (B)
R5: DATA (C)
…

FIG. 4F

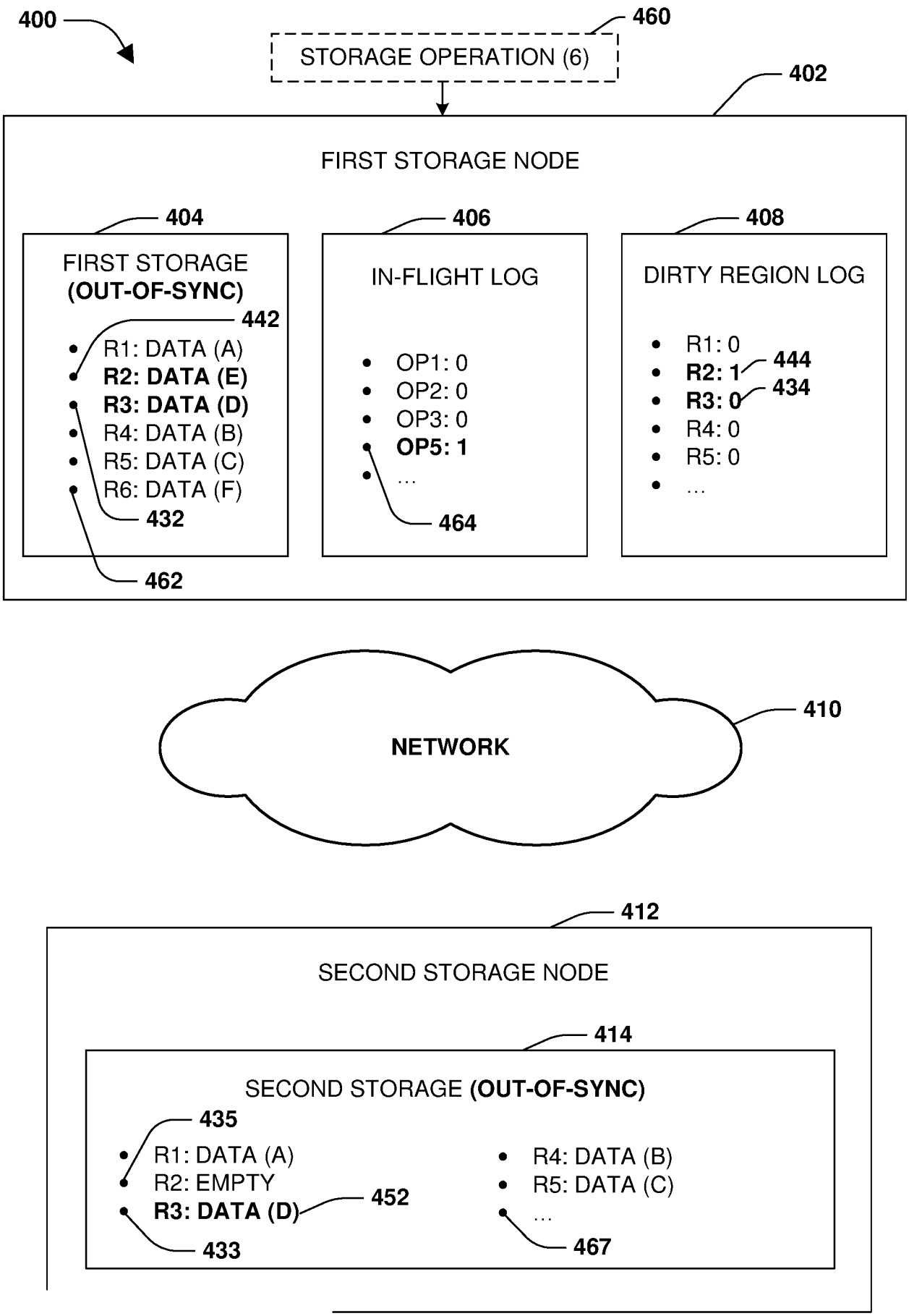

400
460
STORAGE OPERATION (6)
402
FIRST STORAGE NODE
404
FIRST STORAGE
(OUT-OF-SYNC)
442
R1: DATA (A)
R2: DATA (E)
R3: DATA (D)
R4: DATA (B)
R5: DATA (C)
R6: DATA (F)
432
462
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
OP5: 1
…
464
408
DIRTY REGION LOG
R1: 0
R2: 1
R3: 0
R4: 0
R5: 0
…
444
434
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (OUT-OF-SYNC)
435
R1: DATA (A)
R2: EMPTY
R3: DATA (D)
452
433
R4: DATA (B)
R5: DATA (C)
…
467

FIG. 4G

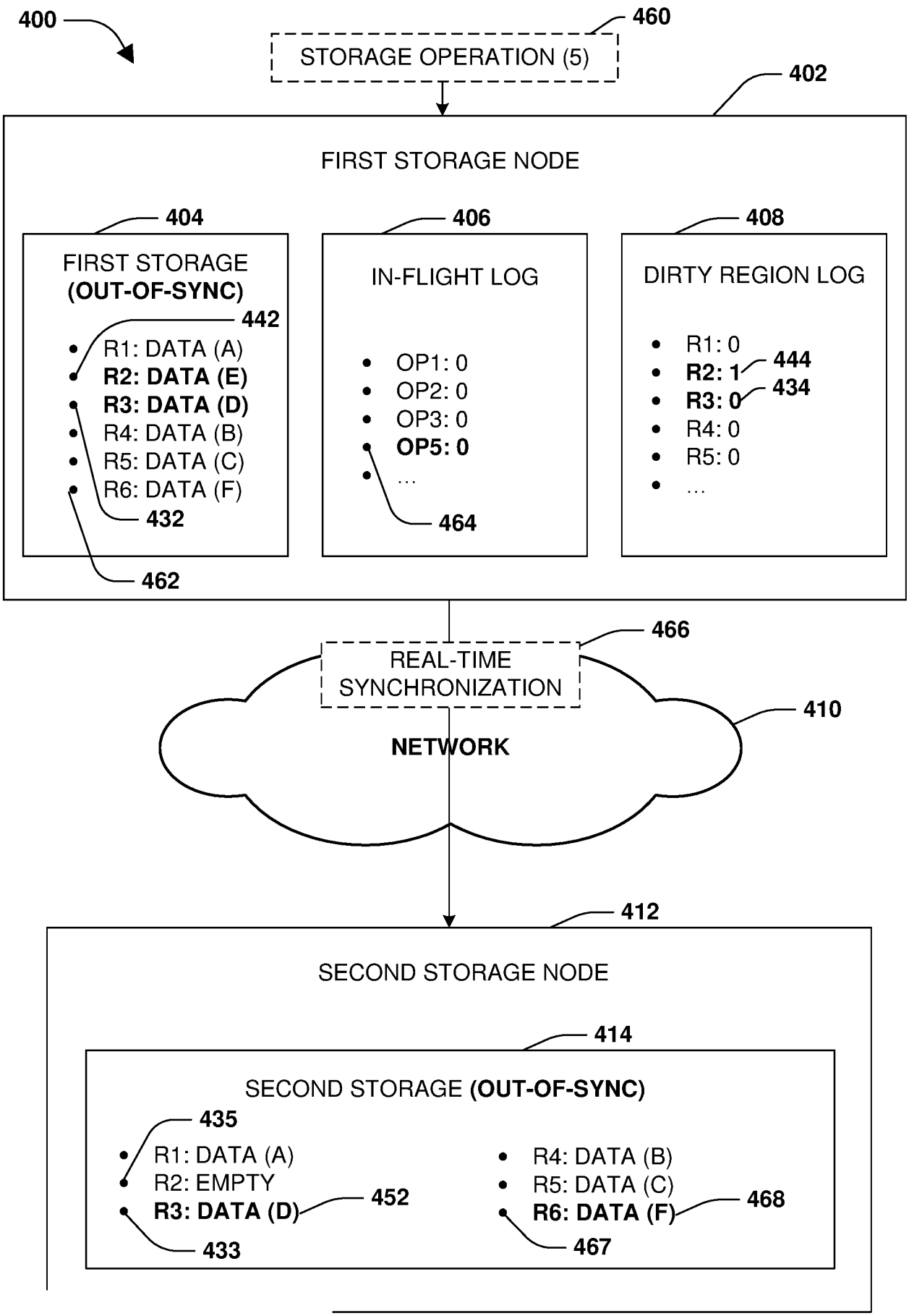

400
460
STORAGE OPERATION (5)
402
FIRST STORAGE NODE
404
FIRST STORAGE
(OUT-OF-SYNC)
442
R1: DATA (A)
R2: DATA (E)
R3: DATA (D)
R4: DATA (B)
R5: DATA (C)
R6: DATA (F)
432
462
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
OP5: 0
...
464
408
DIRTY REGION LOG
R1: 0
R2: 1
R3: 0
R4: 0
R5: 0
...
444
434
466
REAL-TIME
SYNCHRONIZATION
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (OUT-OF-SYNC)
435
R1: DATA (A)
R2: EMPTY
R3: DATA (D)
433
452
R4: DATA (B)
R5: DATA (C)
R6: DATA (F)
467
468

FIG. 4H

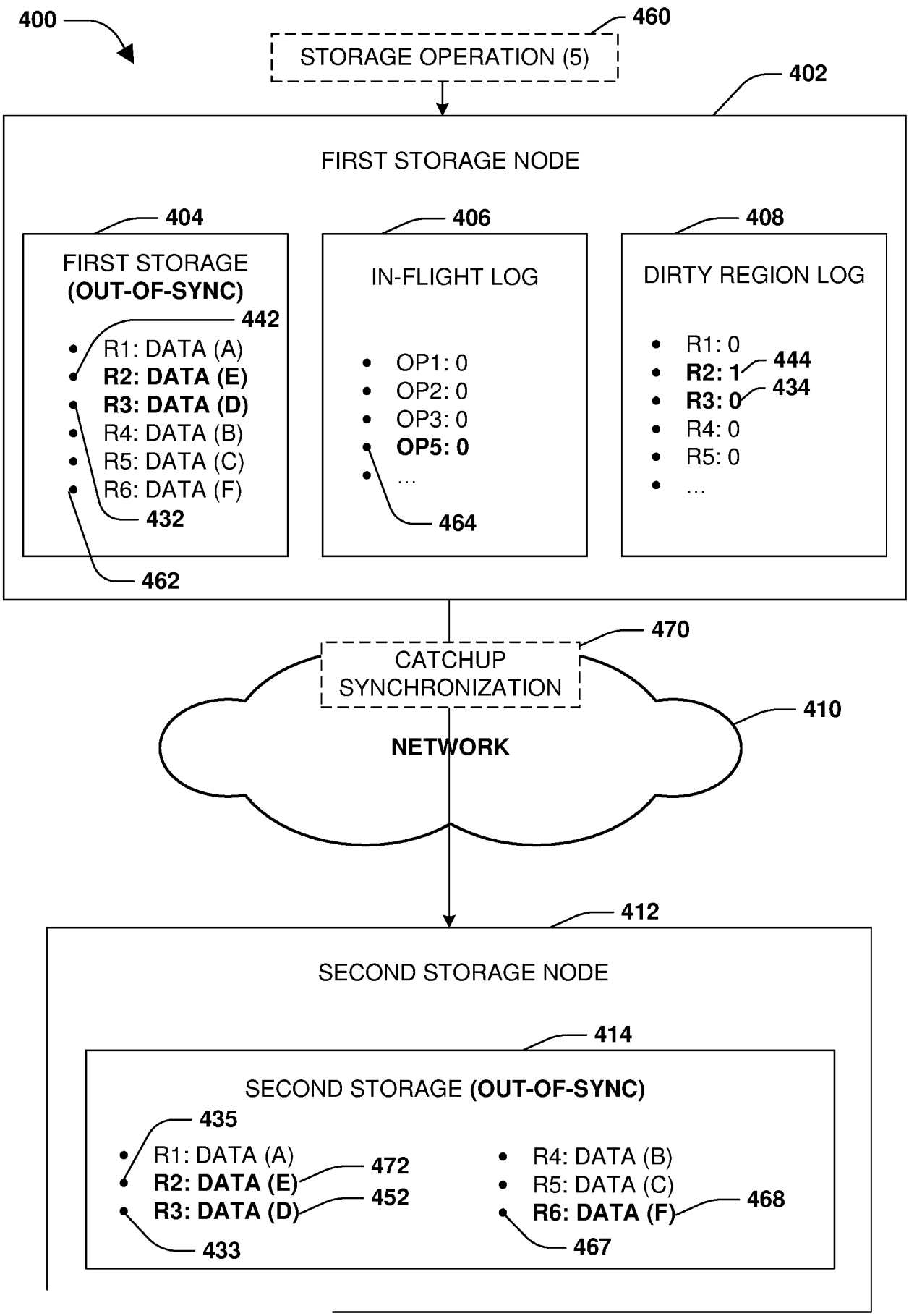

400
460
STORAGE OPERATION (5)
402
FIRST STORAGE NODE
404
FIRST STORAGE
(OUT-OF-SYNC)
442
R1: DATA (A)
R2: DATA (E)
R3: DATA (D)
R4: DATA (B)
R5: DATA (C)
R6: DATA (F)
432
462
406
IN-FLIGHT LOG
OP1: 0
OP2: 0
OP3: 0
OP5: 0
…
464
408
DIRTY REGION LOG
R1: 0
R2: 1
R3: 0
R4: 0
R5: 0
…
444
434
470
CATCHUP
SYNCHRONIZATION
NETWORK
410
412
SECOND STORAGE NODE
414
SECOND STORAGE (OUT-OF-SYNC)
435
R1: DATA (A)
R2: DATA (E)
R3: DATA (D)
433
472
452
R4: DATA (B)
R5: DATA (C)
R6: DATA (F)
467
468

FIG. 4I

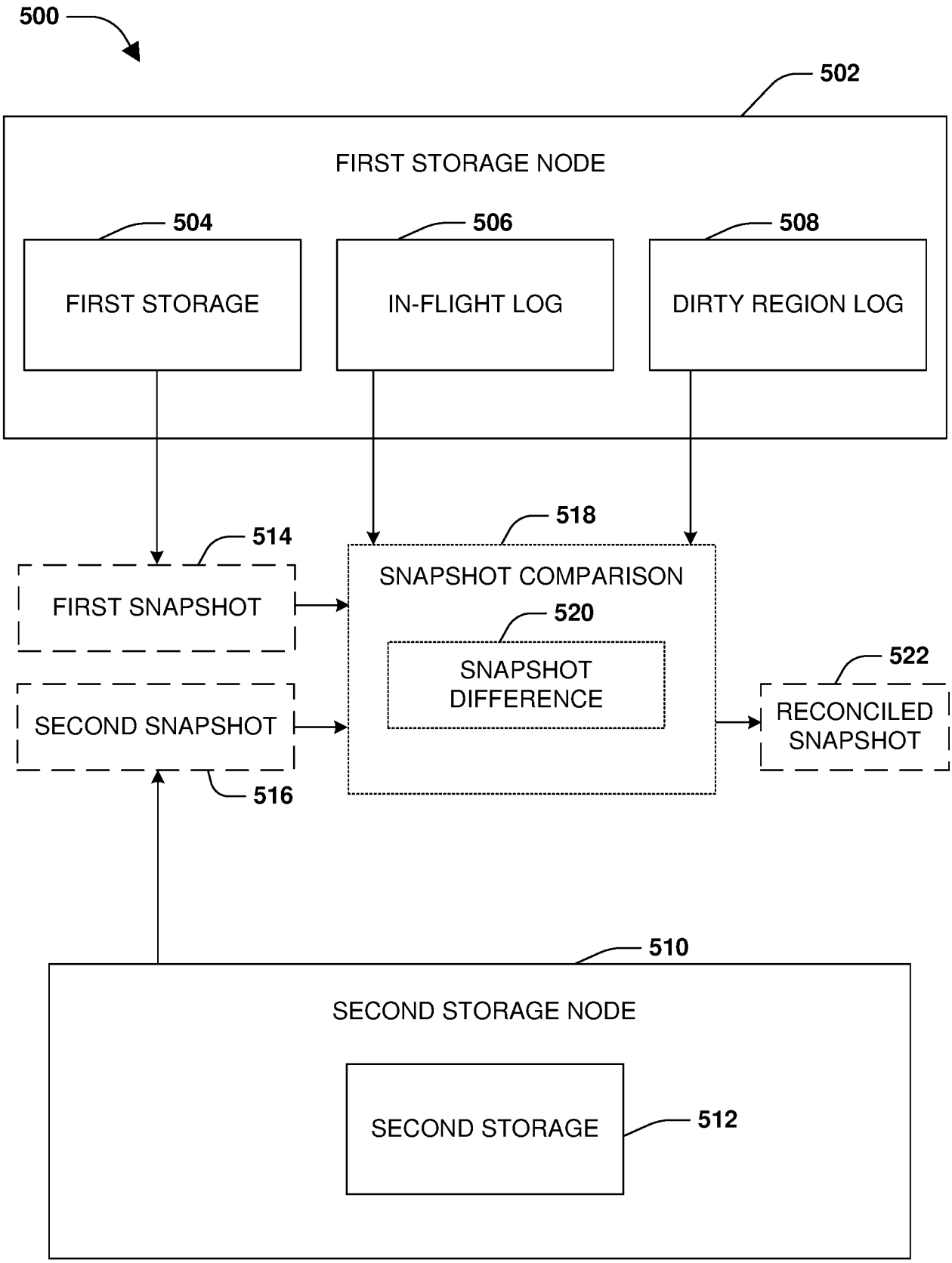

500
502
FIRST STORAGE NODE
504
FIRST STORAGE
506
IN-FLIGHT LOG
508
DIRTY REGION LOG
514
FIRST SNAPSHOT
518
SNAPSHOT COMPARISON
520
SNAPSHOT DIFFERENCE
522
RECONCILED SNAPSHOT
SECOND SNAPSHOT
516
510
SECOND STORAGE NODE
SECOND STORAGE
512

FIG. 5

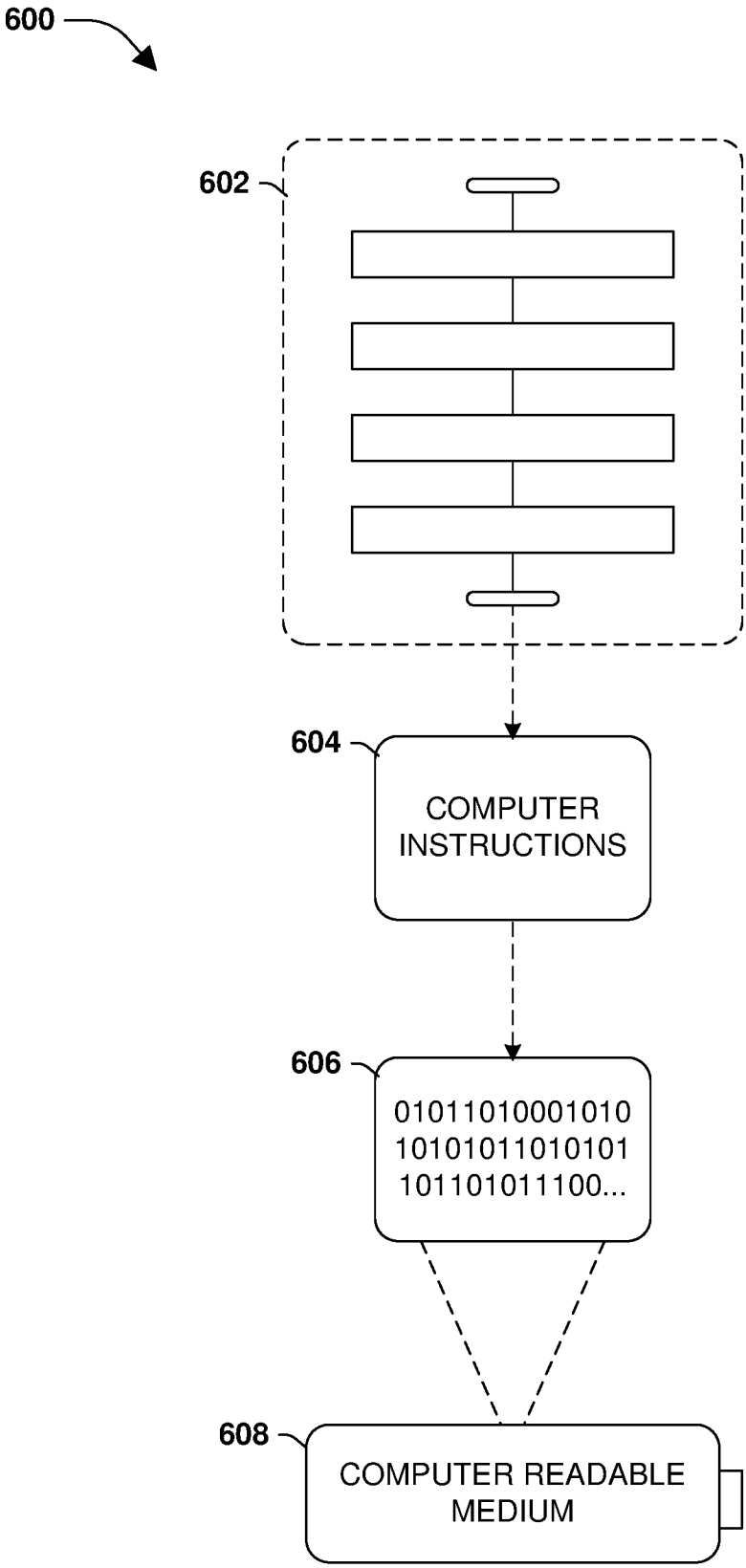

600
602
604
COMPUTER INSTRUCTIONS
606
01011010001010
10101011010101
101101011100...
608
COMPUTER READABLE MEDIUM

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/052431

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30 G06F11/14 G06F11/20
ADD.

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)
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 practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	AMZA C ET AL: "Data replication strategies for fault tolerance and availability on commodity clusters", DEPENDABLE SYSTEMS AND NETWORKS, 2000. DSN 2000. PROCEEDINGS INTERNATI ONAL CONFERENCE ON NEW YORK, NY, USA 25-28 JUNE 2000, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 25 June 2000 (2000-06-25), pages 459-467, XP010504357, DOI: 10.1109/ICDSN.2000.857576 ISBN: 978-0-7695-0707-1 abstract paragraphs [000I] - [00VI] ----- -/--	1-21

[X] Further documents are listed in the continuation of Box C.

[X] 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 application or patent 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
30 November 2016	09/12/2016

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, Fax: (+31-70) 340-3016	Breche, Philippe

1

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/052431

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 7 395 378 B1 (PENDHARKAR NIRANJAN S [IN] ET AL) 1 July 2008 (2008-07-01) abstract; figures 1-5 column 2, line 58 - column 8, line 35 -----	1-21
A	US 8 055 613 B1 (MU PAUL YUEDONG [US] ET AL) 8 November 2011 (2011-11-08) abstract; figures 1-7 column 3, line 45 - column 12, line 50 -----	1-21

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2016/052431

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7395378 B1	01-07-2008	NONE	
US 8055613 B1	08-11-2011	NONE	