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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 October 2016 (06.10.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/160634 A1

(51) International Patent Classification:
*G06F 3/06* (2006.01)

(21) International Application Number:
PCT/US2016/024391

(22) International Filing Date:
27 March 2016 (27.03.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/671,519 27 March 2015 (27.03.2015) US

(71) Applicant: **PURE STORAGE, INC.** [US/US]; Suite 400, 650 Castro St., Mountain View, CA 94041 (US).

(72) Inventors: **HAYES, John**; Suite 400, 650 Castro St., Mountain View, CA 94041 (US). **BOTES, Par**; Suite 400, 650 Castro St., Mountain View, CA 94041 (US).

(74) Agent: **GENCARELLA, Michael, L.**; Womble Carlyle Sandridge & Rice LLP, P.O. Box 7037, Atlanta, GA 30357-0037 (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, 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, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Declarations under Rule 4.17**:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

**Published**:

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

(54) Title: DATA STRIPING ACROSS STORAGE NODES THAT ARE ASSIGNED TO MULTIPLE LOGICAL ARRAYS

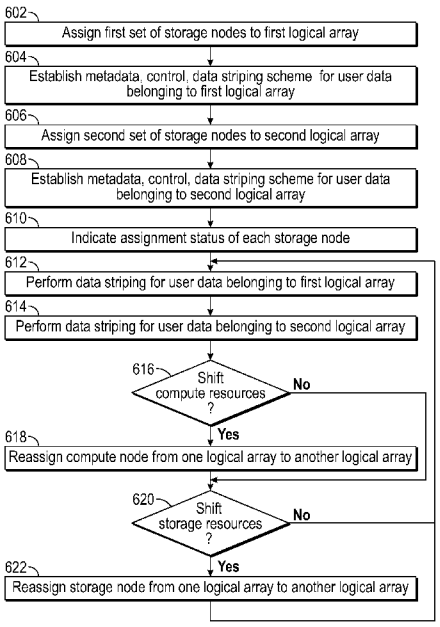


FIG. 6

(57) Abstract: A storage cluster is provided. The storage cluster includes a plurality of storage nodes coupled together as the storage cluster. The plurality of storage nodes is configured to assign data to two or more logical arrays and the plurality of storage nodes is configured to establish data striping across the plurality of storage nodes for user data of each of the two or more logical arrays.

WO 2016/160634 A1

# DATA STRIPING ACROSS STORAGE NODES THAT ARE ASSIGNED TO MULTIPLE LOGICAL ARRAYS

## BACKGROUND

**[0001]** Solid-state memory, such as flash, is currently in use in solid-state drives (SSD) to augment or replace conventional hard disk drives (HDD), writable CD (compact disk) or writable DVD (digital versatile disk) drives, collectively known as spinning media, and tape drives, for storage of large amounts of data. Flash and other solid-state memories have characteristics that differ from spinning media. Yet, many solid-state drives are designed to conform to hard disk drive standards for compatibility reasons, which makes it difficult to provide enhanced features or take advantage of unique aspects of flash and other solid-state memory, such as the ability to partition arrays.

**[0002]** It is within this context that the embodiments arise.

## SUMMARY

**[0003]** In some embodiments, a storage cluster is provided. The storage cluster includes a plurality of storage nodes coupled together as the storage cluster. The plurality of storage nodes is configured to assign data to two or more logical arrays and the plurality of storage nodes is configured to establish data striping across the plurality of storage nodes for user data of each of the two or more logical arrays.

**[0004]** In some embodiments, a storage cluster is provided. The storage cluster includes a plurality of storage nodes coupled together as the storage cluster. The plurality of storage nodes is configured to assign data to two or more logical arrays and the plurality of storage nodes is configured to establish data striping across the plurality of storage nodes for user data of each of the two or more logical arrays.

**[0005]** In some embodiments, a method for data striping across storage nodes in a storage cluster is provided. The method includes creating at least a first logical array and a second logical array in a plurality of storage nodes of a storage cluster and performing data striping for first user data, of which the first logical array has ownership, across the plurality of storage nodes. The method includes performing data striping for second user data, of which the second logical array has ownership, across the plurality of storage nodes.

**[0006]** Other aspects and advantages of the embodiments will become apparent from the following detailed description taken in conjunction with the accompanying drawings which illustrate, by way of example, the principles of the described embodiments.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0007]** The described embodiments and the advantages thereof may best be understood by reference to the following description taken in conjunction with the accompanying drawings. These drawings in no way limit any changes in form and detail that may be made to the described embodiments by one skilled in the art without departing from the spirit and scope of the described embodiments.

**[0008]** Fig. 1 is a perspective view of a storage cluster with multiple storage nodes and internal storage coupled to each storage node to provide network attached storage, in accordance with some embodiments.

**[0009]** Fig. 2 is a block diagram showing an interconnect switch coupling multiple storage nodes in accordance with some embodiments.

**[0010]** Fig. 3 is a multiple level block diagram, showing contents of a storage node and contents of one of the non-volatile solid state storage units in accordance with some embodiments.

**[0011]** Fig. 4 is a front view of a storage cluster, in a single chassis, partitioned into multiple logical arrays or clusters.

**[0012]** Fig. 5 is a front view of a storage cluster spanning two chassis, partitioned into multiple logical arrays or clusters.

**[0013]** Fig. 6 is a flow diagram of a method for partitioning a storage cluster into multiple logical arrays and performing data striping of user data in accordance with the multiple logical arrays, which can be practiced on embodiments of the storage cluster, storage nodes and storage units of Figs. 1-5.

**[0014]** Fig. 7 is an illustration showing an exemplary computing device which may implement the embodiments described herein.

## DETAILED DESCRIPTION

**[0015]** A storage cluster, with storage nodes and storage units that have storage memory, can be partitioned into two or more logical arrays. In some embodiments, this occurs in a single chassis, and in some embodiments this can occur across multiple chassis. Each logical

array acts as a storage cluster and can have software independent of another logical array. Various degrees of sharing or isolation from one logical array to another are possible in various embodiments. Aspects of the storage cluster, storage nodes and storage units are described herein with reference to Figs. 1-3. Aspects of partitioning into multiple logical arrays are described with reference to Figs. 4-6.

**[0016]** The embodiments below describe a storage cluster that stores user data, such as user data originating from one or more user or client systems or other sources external to the storage cluster. The storage cluster distributes user data across storage nodes housed within a chassis, using erasure coding and redundant copies of metadata. Erasure coding refers to a method of data protection or reconstruction in which data is stored across a set of different locations, such as disks, storage nodes or geographic locations. Flash memory is one type of solid-state memory that may be integrated with the embodiments, although the embodiments may be extended to other types of solid-state memory or other storage medium, including non-solid state memory. Control of storage locations and workloads are distributed across the storage locations in a clustered peer-to-peer system. Tasks such as mediating communications between the various storage nodes, detecting when a storage node has become unavailable, and balancing I/Os (inputs and outputs) across the various storage nodes, are all handled on a distributed basis. Data is laid out or distributed across multiple storage nodes in data fragments or stripes that support data recovery in some embodiments. Ownership of data can be reassigned within a cluster, independent of input and output patterns. This architecture described in more detail below allows a storage node in the cluster to fail, with the system remaining operational, since the data can be reconstructed from other storage nodes and thus remain available for input and output operations. In various embodiments, a storage node may be referred to as a cluster node, a blade, or a server.

**[0017]** The storage cluster is contained within a chassis, i.e., an enclosure housing one or more storage nodes. A mechanism to provide power to each storage node, such as a power distribution bus, and a communication mechanism, such as a communication bus that enables communication between the storage nodes are included within the chassis. The storage cluster can run as an independent system in one location according to some embodiments. In one embodiment, a chassis contains at least two instances of both the power distribution and the communication bus which may be enabled or disabled independently. The internal communication bus may be an Ethernet bus, however, other technologies such as Peripheral Component Interconnect (PCI) Express, InfiniBand, and others, are equally suitable. The chassis provides a port for an external communication bus for enabling communication

between multiple chassis, directly or through a switch, and with client systems. The external communication may use a technology such as Ethernet, InfiniBand, Fibre Channel, etc. In some embodiments, the external communication bus uses different communication bus technologies for inter-chassis and client communication. If a switch is deployed within or between chassis, the switch may act as a translation between multiple protocols or technologies. When multiple chassis are connected to define a storage cluster, the storage cluster may be accessed by a client using either proprietary interfaces or standard interfaces such as network file system (NFS), common internet file system (CIFS), small computer system interface (SCSI) or hypertext transfer protocol (HTTP). Translation from the client protocol may occur at the switch, chassis external communication bus or within each storage node.

**[0018]** Each storage node may be one or more storage servers and each storage server is connected to one or more non-volatile solid state memory units, which may be referred to as storage units. One embodiment includes a single storage server in each storage node and between one to eight non-volatile solid state memory units, however this one example is not meant to be limiting. The storage server may include a processor, dynamic random access memory (DRAM) and interfaces for the internal communication bus and power distribution for each of the power buses. Inside the storage node, the interfaces and storage unit share a communication bus, e.g., PCI Express, in some embodiments. The non-volatile solid state memory units may directly access the internal communication bus interface through a storage node communication bus, or request the storage node to access the bus interface. The non-volatile solid state memory unit contains an embedded central processing unit (CPU), solid state storage controller, and a quantity of solid state mass storage, e.g., between 2-32 terabytes (TB) in some embodiments. An embedded volatile storage medium, such as DRAM, and an energy reserve apparatus are included in the non-volatile solid state memory unit. In some embodiments, the energy reserve apparatus is a capacitor, super-capacitor, or battery that enables transferring a subset of DRAM contents to a stable storage medium in the case of power loss. In some embodiments, the non-volatile solid state memory unit is constructed with a storage class memory, such as phase change or magnetoresistive random access memory (MRAM) that substitutes for DRAM and enables a reduced power hold-up apparatus.

**[0019]** One of many features of the storage nodes and non-volatile solid state storage is the ability to proactively rebuild data in a storage cluster. The storage nodes and non-volatile solid state storage can determine when a storage node or non-volatile solid state storage in the

storage cluster is unreachable, independent of whether there is an attempt to read data involving that storage node or non-volatile solid state storage. The storage nodes and non-volatile solid state storage then cooperate to recover and rebuild the data in at least partially new locations. This constitutes a proactive rebuild, in that the system rebuilds data without waiting until the data is needed for a read access initiated from a client system employing the storage cluster. These and further details of the storage memory and operation thereof are discussed below.

**[0020]** Fig. 1 is a perspective view of a storage cluster 160, with multiple storage nodes 150 and internal solid-state memory coupled to each storage node to provide network attached storage or storage area network, in accordance with some embodiments. A network attached storage, storage area network, or a storage cluster, or other storage memory, could include one or more storage clusters 160, each having one or more storage nodes 150, in a flexible and reconfigurable arrangement of both the physical components and the amount of storage memory provided thereby. The storage cluster 160 is designed to fit in a rack, and one or more racks can be set up and populated as desired for the storage memory. The storage cluster 160 has a chassis 138 having multiple slots 142. It should be appreciated that chassis 138 may be referred to as a housing, enclosure, or rack unit. In one embodiment, the chassis 138 has fourteen slots 142, although other numbers of slots are readily devised. For example, some embodiments have four slots, eight slots, sixteen slots, thirty-two slots, or other suitable number of slots. Each slot 142 can accommodate one storage node 150 in some embodiments. Chassis 138 includes flaps 148 that can be utilized to mount the chassis 138 on a rack. Fans 144 provide air circulation for cooling of the storage nodes 150 and components thereof, although other cooling components could be used, or an embodiment could be devised without cooling components. A switch fabric 146 couples storage nodes 150 within chassis 138 together and to a network for communication to the memory. In an embodiment depicted in Fig. 1, the slots 142 to the left of the switch fabric 146 and fans 144 are shown occupied by storage nodes 150, while the slots 142 to the right of the switch fabric 146 and fans 144 are empty and available for insertion of storage node 150 for illustrative purposes. This configuration is one example, and one or more storage nodes 150 could occupy the slots 142 in various further arrangements. The storage node arrangements need not be sequential or adjacent in some embodiments. Storage nodes 150 are hot pluggable, meaning that a storage node 150 can be inserted into a slot 142 in the chassis 138, or removed from a slot 142, without stopping or powering down the system. Upon insertion or removal of storage node 150 from slot 142, the system automatically reconfigures in order to

recognize and adapt to the change. Reconfiguration, in some embodiments, includes restoring redundancy and/or rebalancing data or load.

**[0021]** Each storage node 150 can have multiple components. In the embodiment shown here, the storage node 150 includes a printed circuit board 158 populated by a CPU 156, i.e., processor, a memory 154 coupled to the CPU 156, and a non-volatile solid state storage 152 coupled to the CPU 156, although other mountings and/or components could be used in further embodiments. The memory 154 has instructions which are executed by the CPU 156 and/or data operated on by the CPU 156. As further explained below, the non-volatile solid state storage 152 includes flash or, in further embodiments, other types of solid-state memory.

**[0022]** Referring to Fig. 1, storage cluster 160 is scalable, meaning that storage capacity with non-uniform storage sizes is readily added, as described above. One or more storage nodes 150 can be plugged into or removed from each chassis and the storage cluster self-configures in some embodiments. Plug-in storage nodes 150, whether installed in a chassis as delivered or later added, can have different sizes. For example, in one embodiment a storage node 150 can have any multiple of 4 TB, e.g., 8 TB, 12 TB, 16 TB, 32 TB, etc. In further embodiments, a storage node 150 could have any multiple of other storage amounts or capacities. Storage capacity of each storage node 150 is broadcast, and influences decisions of how to stripe the data. For maximum storage efficiency, an embodiment can self-configure as wide as possible in the stripe, subject to a predetermined requirement of continued operation with loss of up to one, or up to two, non-volatile solid state storage units 152 or storage nodes 150 within the chassis.

**[0023]** Fig. 2 is a block diagram showing a communications interconnect 170 and power distribution bus 172 coupling multiple storage nodes 150. Referring back to Fig. 1, the communications interconnect 170 can be included in or implemented with the switch fabric 146 in some embodiments. Where multiple storage clusters 160 occupy a rack, the communications interconnect 170 can be included in or implemented with a top of rack switch, in some embodiments. As illustrated in Fig. 2, storage cluster 160 is enclosed within a single chassis 138. External port 176 is coupled to storage nodes 150 through communications interconnect 170, while external port 174 is coupled directly to a storage node. External power port 178 is coupled to power distribution bus 172. Storage nodes 150 may include varying amounts and differing capacities of non-volatile solid state storage 152 as described with reference to Fig. 1. In addition, one or more storage nodes 150 may be a compute only storage node as illustrated in Fig. 2. Authorities 168 are implemented on the

non-volatile solid state storages 152, for example as lists or other data structures stored in memory. In some embodiments the authorities are stored within the non-volatile solid state storage 152 and supported by software executing on a controller or other processor of the non-volatile solid state storage 152. In a further embodiment, authorities 168 are implemented on the storage nodes 150, for example as lists or other data structures stored in the memory 154 and supported by software executing on the CPU 156 of the storage node 150. Authorities 168 control how and where data is stored in the non-volatile solid state storages 152 in some embodiments. This control assists in determining which type of erasure coding scheme is applied to the data, and which storage nodes 150 have which portions of the data. Each authority 168 may be assigned to a non-volatile solid state storage 152. Each authority may control a range of inode numbers, segment numbers, or other data identifiers which are assigned to data by a file system, by the storage nodes 150, or by the non-volatile solid state storage 152, in various embodiments.

**[0024]** Every piece of data, and every piece of metadata, has redundancy in the system in some embodiments. In addition, every piece of data and every piece of metadata has an owner, which may be referred to as an authority. If that authority is unreachable, for example through failure of a storage node, there is a plan of succession for how to find that data or that metadata. In various embodiments, there are redundant copies of authorities 168. Authorities 168 have a relationship to storage nodes 150 and non-volatile solid state storage 152 in some embodiments. Each authority 168, covering a range of data segment numbers or other identifiers of the data, may be assigned to a specific non-volatile solid state storage 152. In some embodiments the authorities 168 for all of such ranges are distributed over the non-volatile solid state storages 152 of a storage cluster. Each storage node 150 has a network port that provides access to the non-volatile solid state storage(s) 152 of that storage node 150. Data can be stored in a segment, which is associated with a segment number and that segment number is an indirection for a configuration of a RAID (redundant array of independent disks) stripe in some embodiments. The assignment and use of the authorities 168 thus establishes an indirection to data. Indirection may be referred to as the ability to reference data indirectly, in this case via an authority 168, in accordance with some embodiments. A segment identifies a set of non-volatile solid state storage 152 and a local identifier into the set of non-volatile solid state storage 152 that may contain data. In some embodiments, the local identifier is an offset into the device and may be reused sequentially by multiple segments. In other embodiments the local identifier is unique for a specific segment and never reused. The offsets in the non-volatile solid state storage 152 are applied

to locating data for writing to or reading from the non-volatile solid state storage 152 (in the form of a RAID stripe). Data is striped across multiple units of non-volatile solid state storage 152, which may include or be different from the non-volatile solid state storage 152 having the authority 168 for a particular data segment.

**[0025]** If there is a change in where a particular segment of data is located, e.g., during a data move or a data reconstruction, the authority 168 for that data segment should be consulted, at that non-volatile solid state storage 152 or storage node 150 having that authority 168. In order to locate a particular piece of data, embodiments calculate a hash value for a data segment or apply an inode number or a data segment number. The output of this operation points to a non-volatile solid state storage 152 having the authority 168 for that particular piece of data. In some embodiments there are two stages to this operation. The first stage maps an entity identifier (ID), e.g., a segment number, inode number, or directory number to an authority identifier. This mapping may include a calculation such as a hash or a bit mask. The second stage is mapping the authority identifier to a particular non-volatile solid state storage 152, which may be done through an explicit mapping. The operation is repeatable, so that when the calculation is performed, the result of the calculation repeatably and reliably points to a particular non-volatile solid state storage 152 having that authority 168. The operation may include the set of reachable storage nodes as input. If the set of reachable non-volatile solid state storage units changes the optimal set changes. In some embodiments, the persisted value is the current assignment (which is always true) and the calculated value is the target assignment the cluster will attempt to reconfigure towards. This calculation may be used to determine the optimal non-volatile solid state storage 152 for an authority in the presence of a set of non-volatile solid state storage 152 that are reachable and constitute the same cluster. The calculation also determines an ordered set of peer non-volatile solid state storage 152 that will also record the authority to non-volatile solid state storage mapping so that the authority may be determined even if the assigned non-volatile solid state storage is unreachable. A duplicate or substitute authority 168 may be consulted if a specific authority 168 is unavailable in some embodiments.

**[0026]** With reference to Figs. 1 and 2, two of the many tasks of the CPU 156 on a storage node 150 are to break up write data, and reassemble read data. When the system has determined that data is to be written, the authority 168 for that data is located as above. When the segment ID for data is already determined the request to write is forwarded to the non-volatile solid state storage 152 currently determined to be the host of the authority 168 determined from the segment. The host CPU 156 of the storage node 150, on which the non-

volatile solid state storage 152 and corresponding authority 168 reside, then breaks up or shards the data and transmits the data out to various non-volatile solid state storage 152. The transmitted data is written as a data stripe in accordance with an erasure coding scheme. In some embodiments, data is requested to be pulled, and in other embodiments, data is pushed. In reverse, when data is read, the authority 168 for the segment ID containing the data is located as described above. The host CPU 156 of the storage node 150 on which the non-volatile solid state storage 152 and corresponding authority 168 reside requests the data from the non-volatile solid state storage and corresponding storage nodes pointed to by the authority. In some embodiments the data is read from flash storage as a data stripe. The host CPU 156 of storage node 150 then reassembles the read data, correcting any errors (if present) according to the appropriate erasure coding scheme, and forwards the reassembled data to the network. In further embodiments, some or all of these tasks can be handled in the non-volatile solid state storage 152. In some embodiments, the segment host requests the data be sent to storage node 150 by requesting pages from storage and then sending the data to the storage node making the original request.

**[0027]** In some systems, for example in UNIX-style file systems, data is handled with an index node or inode, which specifies a data structure that represents an object in a file system. The object could be a file or a directory, for example. Metadata may accompany the object, as attributes such as permission data and a creation timestamp, among other attributes. A segment number could be assigned to all or a portion of such an object in a file system. In other systems, data segments are handled with a segment number assigned elsewhere. For purposes of discussion, the unit of distribution is an entity, and an entity can be a file, a directory or a segment. That is, entities are units of data or metadata stored by a storage system. Entities are grouped into sets called authorities. Each authority has an authority owner, which is a storage node that has the exclusive right to update the entities in the authority. In other words, a storage node contains the authority, and that the authority, in turn, contains entities.

**[0028]** A segment is a logical container of data in accordance with some embodiments. A segment is an address space between medium address space and physical flash locations, i.e., the data segment number, are in this address space. Segments may also contain meta-data, which enable data redundancy to be restored (rewritten to different flash locations or devices) without the involvement of higher level software. In one embodiment, an internal format of a segment contains client data and medium mappings to determine the position of that data. Each data segment is protected, e.g., from memory and other failures, by breaking

the segment into a number of data and parity shards, where applicable. The data and parity shards are distributed, i.e., striped, across non-volatile solid state storage 152 coupled to the host CPUs 156 (See Fig. 5) in accordance with an erasure coding scheme. Usage of the term segments refers to the container and its place in the address space of segments in some embodiments. Usage of the term stripe refers to the same set of shards as a segment and includes how the shards are distributed along with redundancy or parity information in accordance with some embodiments.

**[0029]** A series of address-space transformations takes place across an entire storage system. At the top are the directory entries (file names) which link to an inode. Inodes point into medium address space, where data is logically stored. Medium addresses may be mapped through a series of indirect mediums to spread the load of large files, or implement data services like deduplication or snapshots. Medium addresses may be mapped through a series of indirect mediums to spread the load of large files, or implement data services like deduplication or snapshots. Segment addresses are then translated into physical flash locations. Physical flash locations have an address range bounded by the amount of flash in the system in accordance with some embodiments. Medium addresses and segment addresses are logical containers, and in some embodiments use a 128 bit or larger identifier so as to be practically infinite, with a likelihood of reuse calculated as longer than the expected life of the system. Addresses from logical containers are allocated in a hierarchical fashion in some embodiments. Initially, each non-volatile solid state storage 152 may be assigned a range of address space. Within this assigned range, the non-volatile solid state storage 152 is able to allocate addresses without synchronization with other non-volatile solid state storage 152.

**[0030]** Data and metadata is stored by a set of underlying storage layouts that are optimized for varying workload patterns and storage devices. These layouts incorporate multiple redundancy schemes, compression formats and index algorithms. Some of these layouts store information about authorities and authority masters, while others store file metadata and file data. The redundancy schemes include error correction codes that tolerate corrupted bits within a single storage device (such as a NAND flash chip), erasure codes that tolerate the failure of multiple storage nodes, and replication schemes that tolerate data center or regional failures. In some embodiments, low density parity check (LDPC) code is used within a single storage unit. Reed-Solomon encoding is used within a storage cluster, and mirroring is used within a storage grid in some embodiments. Metadata may be stored using

an ordered log structured index (such as a Log Structured Merge Tree), and large data may not be stored in a log structured layout.

**[0031]** In order to maintain consistency across multiple copies of an entity, the storage nodes agree implicitly on two things through calculations: (1) the authority that contains the entity, and (2) the storage node that contains the authority. The assignment of entities to authorities can be done by pseudorandomly assigning entities to authorities, by splitting entities into ranges based upon an externally produced key, or by placing a single entity into each authority. Examples of pseudorandom schemes are linear hashing and the Replication Under Scalable Hashing (RUSH) family of hashes, including Controlled Replication Under Scalable Hashing (CRUSH). In some embodiments, pseudo-random assignment is utilized only for assigning authorities to nodes because the set of nodes can change. The set of authorities cannot change so any subjective function may be applied in these embodiments. Some placement schemes automatically place authorities on storage nodes, while other placement schemes rely on an explicit mapping of authorities to storage nodes. In some embodiments, a pseudorandom scheme is utilized to map from each authority to a set of candidate authority owners. A pseudorandom data distribution function related to CRUSH may assign authorities to storage nodes and create a list of where the authorities are assigned. Each storage node has a copy of the pseudorandom data distribution function, and can arrive at the same calculation for distributing, and later finding or locating an authority. Each of the pseudorandom schemes requires the reachable set of storage nodes as input in some embodiments in order to conclude the same target nodes. Once an entity has been placed in an authority, the entity may be stored on physical devices so that no expected failure will lead to unexpected data loss. In some embodiments, rebalancing algorithms attempt to store the copies of all entities within an authority in the same layout and on the same set of machines.

**[0032]** Examples of expected failures include device failures, stolen machines, datacenter fires, and regional disasters, such as nuclear or geological events. Different failures lead to different levels of acceptable data loss. In some embodiments, a stolen storage node impacts neither the security nor the reliability of the system, while depending on system configuration, a regional event could lead to no loss of data, a few seconds or minutes of lost updates, or even complete data loss.

**[0033]** In the embodiments, the placement of data for storage redundancy is independent of the placement of authorities for data consistency. In some embodiments, storage nodes that contain authorities do not contain any persistent storage. Instead, the storage nodes are connected to non-volatile solid state storage units that do not contain authorities. The

communications interconnect between storage nodes and non-volatile solid state storage units consists of multiple communication technologies and has non-uniform performance and fault tolerance characteristics. In some embodiments, as mentioned above, non-volatile solid state storage units are connected to storage nodes via PCI express, storage nodes are connected together within a single chassis using Ethernet backplane, and chassis are connected together to form a storage cluster. Storage clusters are connected to clients using Ethernet or fiber channel in some embodiments. If multiple storage clusters are configured into a storage grid, the multiple storage clusters are connected using the Internet or other long-distance networking links, such as a "metro scale" link or private link that does not traverse the internet.

**[0034]** Authority owners have the exclusive right to modify entities, to migrate entities from one non-volatile solid state storage unit to another non-volatile solid state storage unit, and to add and remove copies of entities. This allows for maintaining the redundancy of the underlying data. When an authority owner fails, is going to be decommissioned, or is overloaded, the authority is transferred to a new storage node. Transient failures make it non-trivial to ensure that all non-faulty machines agree upon the new authority location. The ambiguity that arises due to transient failures can be achieved automatically by a consensus protocol such as Paxos, hot-warm failover schemes, via manual intervention by a remote system administrator, or by a local hardware administrator (such as by physically removing the failed machine from the cluster, or pressing a button on the failed machine). In some embodiments, a consensus protocol is used, and failover is automatic. If too many failures or replication events occur in too short a time period, the system goes into a self-preservation mode and halts replication and data movement activities until an administrator intervenes in accordance with some embodiments.

**[0035]** As authorities are transferred between storage nodes and authority owners update entities in their authorities, the system transfers messages between the storage nodes and non-volatile solid state storage units. With regard to persistent messages, messages that have different purposes are of different types. Depending on the type of the message, the system maintains different ordering and durability guarantees. As the persistent messages are being processed, the messages are temporarily stored in multiple durable and non-durable storage hardware technologies. In some embodiments, messages are stored in RAM, NVRAM and on NAND flash devices, and a variety of protocols are used in order to make efficient use of each storage medium. Latency-sensitive client requests may be persisted in replicated

NVRAM, and then later NAND, while background rebalancing operations are persisted directly to NAND.

**[0036]** Persistent messages are persistently stored prior to being transmitted. This allows the system to continue to serve client requests despite failures and component replacement. Although many hardware components contain unique identifiers that are visible to system administrators, manufacturer, hardware supply chain and ongoing monitoring quality control infrastructure, applications running on top of the infrastructure address virtualize addresses. These virtualized addresses do not change over the lifetime of the storage system, regardless of component failures and replacements. This allows each component of the storage system to be replaced over time without reconfiguration or disruptions of client request processing.

**[0037]** In some embodiments, the virtualized addresses are stored with sufficient redundancy. A continuous monitoring system correlates hardware and software status and the hardware identifiers. This allows detection and prediction of failures due to faulty components and manufacturing details. The monitoring system also enables the proactive transfer of authorities and entities away from impacted devices before failure occurs by removing the component from the critical path in some embodiments.

**[0038]** Communication within the storage cluster can take place through a suitable protocol for a distributed storage system. The protocol may designate a master authority in some embodiments or the protocol could be token-based in further embodiments. In some embodiments, in order to communicate and represent the ownership of an authority and the right to record persistent changes on behalf of the authority, the authority may provide some evidence (token) of authority ownership that can be independently verifiable. Independent verification of the authority token requires some shared state, and means for evolving the shared state, across each of the servers in the cluster in some embodiments. One possibility for representing the authority token would be to globally propagate each renewed token out to each server/device in the storage cluster. Alternatively, authority ownership token may be represented as a time-bounded lease.

**[0039]** In order to acquire or renew the time-bounded lease, the server secures agreement from a majority of a witness set for a proposed lease in some embodiments. Before the time bound lease expires, the authority owner attempts to renew the lease by re-polling the witness set. A failure to renew the authority ownership should generally result in quiescing the authority (or tearing it down), but the timeliness of that quiescence is not necessary for

correctness as all commands are validated by the receiver with the lease token held by the authority at the time of command submission.

**[0040]** At any point, a server that succeeds in getting votes from a majority of witnesses may take authority. Since witnesses will not vote for a new owner while an existing lease is still active, a change of ownership occurs after a lease has expired in some embodiments. Accordingly, once the lease has expired, a new lessee can be given ownership without concern for creating conflicts. It should be appreciated that these example communication protocols are provided for illustrative purposes and not meant to be limiting as any suitable communication protocol for a distributed system may be integrated with the embodiments.

**[0041]** Fig. 3 is a multiple level block diagram, showing contents of a storage node 150 and contents of a non-volatile solid state storage 152 of the storage node 150. Data is communicated to and from the storage node 150 by a network interface controller (NIC) 202 in some embodiments. Each storage node 150 has a CPU 156, and one or more non-volatile solid state storage 152, as discussed above. Moving down one level in Fig. 3, each non-volatile solid state storage 152 has a relatively fast non-volatile solid state memory, such as nonvolatile random access memory (NVRAM) 204, and flash memory 206. In some embodiments, NVRAM 204 may be a component that does not require program/erase cycles (DRAM, MRAM, PCM), and can be a memory that can support being written vastly more often than the memory is read from. Moving down another level in Fig. 3, the NVRAM 204 is implemented in one embodiment as high speed volatile memory, such as dynamic random access memory (DRAM) 216, backed up by energy reserve 218. Energy reserve 218 provides sufficient electrical power to keep the DRAM 216 powered long enough for contents to be transferred to the flash memory 206 in the event of power failure. In some embodiments, energy reserve 218 is a capacitor, super-capacitor, battery, or other device, that supplies a suitable supply of energy sufficient to enable the transfer of the contents of DRAM 216 to a stable storage medium in the case of power loss. The flash memory 206 is implemented as multiple flash dies 222, which may be referred to as packages of flash dies 222 or an array of flash dies 222. It should be appreciated that the flash dies 222 could be packaged in any number of ways, with a single die per package, multiple dies per package (i.e. multichip packages), in hybrid packages, as bare dies on a printed circuit board or other substrate, as encapsulated dies, etc. In the embodiment shown, the non-volatile solid state storage 152 has a controller 212 or other processor, and an input output (I/O) port 210 coupled to the controller 212. I/O port 210 is coupled to the CPU 156 and/or the network interface controller 202 of the flash storage node 150. Flash input output (I/O) port 220 is

coupled to the flash dies 222, and a direct memory access unit (DMA) 214 is coupled to the controller 212, the DRAM 216 and the flash dies 222. In the embodiment shown, the I/O port 210, controller 212, DMA unit 214 and flash I/O port 220 are implemented on a programmable logic device (PLD) 208, e.g., a field programmable gate array (FPGA). In this embodiment, each flash die 222 has pages, organized as sixteen kB (kilobyte) pages 224, and a register 226 through which data can be written to or read from the flash die 222. In further embodiments, other types of solid-state memory are used in place of, or in addition to flash memory illustrated within flash die 222.

**[0042]** Storage cluster 160, in various embodiments as disclosed herein, can be contrasted with storage arrays in general. The storage nodes 150 are part of a collection that creates the storage cluster 160. Each storage node 150 owns a slice of data and computing required to provide the data. Multiple storage nodes 150 cooperate to store and retrieve the data. Storage memory or storage devices, as used in storage arrays in general, are less involved with processing and manipulating the data. Storage memory or storage devices in a storage array receive commands to read, write, or erase data. The storage memory or storage devices in a storage array are not aware of a larger system in which they are embedded, or what the data means. Storage memory or storage devices in storage arrays can include various types of storage memory, such as RAM, solid state drives, hard disk drives, etc. The storage units 152 described herein have multiple interfaces active simultaneously and serving multiple purposes. In some embodiments, some of the functionality of a storage node 150 is shifted into a storage unit 152, transforming the storage unit 152 into a combination of storage unit 152 and storage node 150. Placing computing (relative to storage data) into the storage unit 152 places this computing closer to the data itself. The various system embodiments have a hierarchy of storage node layers with different capabilities. By contrast, in a storage array, a controller owns and knows everything about all of the data that the controller manages in a shelf or storage devices. In a storage cluster 160, as described herein, multiple controllers in multiple storage units 152 and/or storage nodes 150 cooperate in various ways (e.g., for erasure coding, data sharding, metadata communication and redundancy, storage capacity expansion or contraction, data recovery, and so on).

**[0043]** Fig. 4 is a front view of a storage cluster 160, in a single chassis 138, partitioned into multiple logical arrays 404, 406 or clusters. Each logical array 404 functions as a storage cluster, with a unique cluster identifier in some embodiments. The chassis 138 houses the storage nodes 150 of the storage cluster 160, and also houses a switch fabric 146 or other bus or network that couples the storage nodes 150 as the storage cluster. A power supply 402

occupies part of the chassis 138, or could be external to the chassis 138 in various embodiments. As shown in Fig. 4, the storage cluster 160, which may be considered a physical array, is partitioned into two logical arrays 404, 406. In various embodiments, the storage cluster 160 could be partitioned into further logical arrays. A visual indicator 412 on each storage node 150, in some embodiments, shows a number, letter, symbol, color or other visual indication corresponding to the assignment of the storage node 150 relative to one or more logical arrays 404, 406. That is, the visual indicator 412 shows the membership of a particular storage node 150 to a particular logical array 404, or other membership or assignment status of the storage node 150. For example, the visual indicator 412 could show the letter "A", the number "1", or other name or label for a storage node 150 assigned to a particular logical array 404. In some embodiments, the visual indicator 412 can indicate when a storage node 150 is unassigned, for example when the storage node 150 is an unassigned spare. In further embodiments, the visual indicator 412 can indicate when a storage node 150 is shared by multiple logical arrays 404, 406. Since each storage unit 152 and each storage node 150 has a processor and local memory, the storage units 152 and storage nodes 150 of the storage cluster 160 can be programmed for various features as will be further described below. It should be appreciated that various embodiments can have one or a small number of these features, or a larger number of features, in various combinations.

**[0044]** Fig. 5 is a front view of a storage cluster 160 spanning two chassis 138, partitioned into multiple logical arrays 404, 406, 408, 410 or clusters. As in Fig. 4, storage nodes 150 in one chassis 138 can be assigned to a particular logical array 404, 406. In the embodiment shown, a logical array 408 can span more than one chassis 138. That is, storage nodes 150 in two or more chassis 138 can be assigned to a particular logical array 408. To facilitate communication among such storage nodes 150 in a logical array 408 spanning two or more chassis 138, these chassis 138 are coupled together, for example by a bus or a network that couples the storage nodes 150 of the two or more chassis 138 together.

**[0045]** With reference to Figs. 4 and 5, various features and combinations of these features are possible with the embodiments described herein. In some embodiments, the assignment of a storage node 150 to a logical array 404 depends on the slot 142 of the chassis 138 (see Fig. 1). In other embodiments, the assignment of a storage node 150 to a logical array 404, 406, 408, 410 is slot independent. It is not necessary to have all slots 142 in a chassis 138 occupied by storage nodes 150, since a storage node 150 can be hot plugged into a slot 142, and assigned to a logical array 404, 406, 408, 410. Alternatively, a storage node 150 could occupy a slot but be unassigned, and then assigned to one of the logical arrays 404,

406, 408, 410. Various degrees and types of shared and separate facilities are also possible. For example, the storage cluster 160 could have a shared physical network, and also have network separation using virtual local area networks, one for each logical array 404, 406, 408, 410. Each logical array 404, 406 could have a unique Internet protocol address, a unique virtual local area network name, unique software image and version, etc. Flow-based control could be applied to reconfigure one or more of these virtual local area networks or route communication over different paths in the virtual local area networks. Software separation allows for independent software, independent operating systems, independent upgrades of software, independent upgrades of operating systems and so on. For example, each logical array 404, 406 could have its own operating system, software, software upgrades, operating system upgrades, hardware upgrades, etc.

**[0046]** Management isolation allows for separate ports and separate management of each logical array 404, 406. For example, each logical array 404 could be accessed and managed via a port belonging to one of the storage nodes 150 assigned to that logical array 404, or a port belonging to a virtual local area network assigned to that logical array 404. The number of storage nodes 150 assigned to one logical array 404 is independent of the number of storage nodes 150 assigned to another logical array 406. Total storage capacity, or utilized or spare storage capacity, of one logical array 404 is independent of that of another logical array 406. Redundancy schemes and/or encryption schemes can differ from one logical array 404 to another logical array 406. Logical arrays 404, 406, 408, 410 can have administrative domain isolation.

**[0047]** In some embodiments, there could be shared data and/or shared data striping across storage nodes 150 of multiple logical arrays 404, 406, and separate metadata and control for each logical array 404, 406. In other embodiments, there could be shared data and/or shared data striping, and shared metadata and control across storage nodes 150 of multiple logical arrays 404, 406. There could be shared data and/or shared data striping, and shared metadata across storage nodes 150 of multiple logical arrays 404, 406, and separate control and processing for each logical array 404, 406. In embodiments with shared data striping, there could be wider stripes (e.g., across most or all of the storage nodes 150 of a storage cluster 160) than would be possible for embodiments with the same total number of storage nodes 150 but data striping limited to the storage nodes 150 assigned to a particular logical array 404, 406. Data striping could even extend across storage nodes 150 of two or more chassis 138.

**[0048]** Some embodiments have a high degree of isolation, with data, data striping, metadata, control, processing, and communication of one logical array 404 isolated and independent of that of another logical array 406. With this degree of isolation, there is no communication from members of one storage cluster (e.g., storage nodes 150 assigned to one logical array 404) to members of another storage cluster (e.g., storage nodes 150 assigned to another logical array 406), and vice versa. No data from one logical array 404, and no metadata from one logical array 404, is found in storage nodes 150 assigned to another logical array 406, and vice versa. This high degree of isolation allows for corruption isolation, for example arising from faults in processing. That is, a corruption in one logical array 404 or cluster does not affect another logical array 406 or cluster.

**[0049]** Some embodiments can dynamically shift compute resources from one logical array 404 to another logical array 406. In some versions, the storage cluster 160 has one or more compute nodes, such as shown in Fig. 2 as a "storage nodes compute only" storage node 150. Logical arrays 404, 406 could be provisioned with one or more compute nodes each in some embodiments. The compute node or nodes communicate with other storage nodes 150 of the logical array 404. With one or more compute nodes, a logical array 404, 406, 408, 410 can function as both a storage array and a computing facility, and execute applications as well as store user data, for example. In versions supporting multiple chassis 138, one or more compute nodes from one chassis 138 could be assigned to logical arrays 404, 406, 408, 410 with storage nodes 150 of either or any of the multiple chassis 138. This is regardless of the slot 142 (see Fig. 1) any storage node occupies.

**[0050]** Some embodiments have ability to remove a storage node 150 from a logical array 404. This can happen in various ways. If a storage node 150 fails, the system can recover user data, using erasure coding, reconfigure, and redistribute user data among remaining storage nodes 150, using the same or a differing version of erasure coding. Physically removing a storage node 150 from a chassis 138 could trigger such actions, as if the storage node 150 had failed. In this case, the removed storage node 150 still contains portions of user data and metadata, which can be recovered upon reinsertion of the storage node 150 into the same chassis 138, or a differing chassis 138 in some embodiments. However, in some embodiments, a command to evacuate a storage node 150 of data and metadata causes the corresponding logical array 404 (of which that storage node 150 is a member) to do so and to redistribute the data and the metadata throughout the remaining storage nodes 150 of that logical array 404. The evacuated storage node 150 can then be physically removed from the chassis 138 as empty, left in the chassis 138 as unassigned (e.g., a spare, or reserve capacity),

or assigned to another logical array 406. In some versions, one or more of the storage nodes 150 has the capability of being evacuated and removed or reassigned, and one or more of the storage nodes 150 lacks this capability and should not be removed from a logical array 404 once assigned to that logical array 404. The visual indicator 412 could show a symbol, color or message, etc., to indicate that this storage node 150 is prevented from being removed. Similarly, the visual indicator 412 could show a storage node 150 is removable, if such is the case. A request to remove a storage node 150 could be followed by a determination of whether that storage node 150 is of a type that is removable, or a type that is not removable, which could then be communicated by message or report.

**[0051]** Some embodiments have automatic provisioning of a logical array 404. Other embodiments have manual provisioning of each logical array 404. A storage cluster 160 could be setup for manual provisioning of one or more logical arrays 404, and automatic provision of one or more logical arrays 406. A licensing model can be implemented in some embodiments of the storage cluster 160. A manufacturer would supply one or more chassis 138 populated by storage nodes 150, but not all of the storage nodes 150 would be assigned to a particular logical array 404 (or multiple logical arrays 404, 406). The storage cluster 160 would self-monitor, and communicate back to the manufacturer regarding utilized storage capacity. When the storage cluster 160 assigns an unassigned storage node 150 to a logical array 404, whether automatically or by client instruction, the storage cluster 160 communicates this event and situation (e.g., in a report or message) back to the manufacturer. The manufacturer can then bill or debit the user for the additional storage capacity at the time this additional capacity is brought online. This may be referred to as a "purchase on first use model". In some versions, the storage cluster 160 communicates when a storage node 150 is removed from a logical array 404 and associated storage cluster. The manufacturer could then refund or credit a user, or discontinue billing for any removed storage node 150, in a flexible billing plan based on storage usage. In some embodiments, a storage node 150 can be assigned to be removable or non-removable. Non-removable storage nodes 150 would remain as assigned to a logical array 404, and could not be removed, nor would a refund or credit be issued if storage capacity of such a non-removable storage node 150 is not used, in the flexible billing plan.

**[0052]** Fig. 6 is a flow diagram of a method for partitioning a storage cluster into multiple logical arrays and performing data striping of user data in accordance with the multiple logical arrays, which can be practiced on embodiments of the storage cluster, storage nodes and storage units of Figs. 1-5. The method can be practiced by processors of storage nodes

and/or storage units. In an action 602, a first set of storage nodes is assigned to a first logical array. Metadata, control, and data striping scheme for user data belonging to the first logical array are established, in an action 604. In various embodiments, there are various capabilities for each of these to be isolated to storage nodes of the first logical array, or shared across storage nodes of the storage cluster (i.e., shared across storage nodes assigned to multiple logical arrays) in various combinations. For example, authorities and wards (i.e., owners of user data) could be established in storage nodes assigned to the first logical array, with metadata, control and data striping confined to nodes assigned to the first logical array, or metadata, control and/or data striping shared across nodes of the first logical array and another logical array, etc.

**[0053]** In an action 606, a second set of storage nodes is assigned to a second logical array. Metadata, control, and data striping scheme for user data belonging to the second logical array are established, in an action 608. The second logical array, and storage nodes assigned to the second logical array has similar capabilities and possibilities in various embodiments as the first logical array and corresponding storage nodes. Assignment status of each storage node is indicated, in an action 610. For example, the visual indicator described above with reference to Fig. 4 shows a visual indication of whether a storage node is assigned to a particular logical array, or is shared or unassigned. Data striping for user data belonging to the first logical array is performed in an action 612. This is according to the data striping scheme established in the action 604, which could be for data striping across storage nodes assigned to the first logical array, or for data striping across storage nodes assigned to the first logical array and the second logical array, etc. Data striping for user data belonging to the second logical array is performed in an action 614. This is according to the data striping scheme established in the action 608, which could be for data striping across storage nodes assigned to the second logical array, or for data striping across storage nodes assigned to the first logical array and the second logical array, etc.

**[0054]** In a decision action 616, it is determined whether to shift compute resources. This could be based on a request, e.g., from a client, or automatic load balancing of compute resources, etc. If the answer is no, flow proceeds to the decision action 620. If the answer is yes, compute resources should be shifted, flow proceeds to the action 618, where a compute node is reassigned from one logical array to another logical array. Flow then proceeds to the decision action 620 where it is determined whether to shift storage resources. This determination could be based on a request, e.g., from a client, or automatic load balancing of storage resources, etc. If the answer is no, flow branches back to the action 612, to continue

performing data striping. Alternatively, flow could branch elsewhere to perform further actions or determinations. If the answer is yes, flow proceeds to the action 622, where a storage node is reassigned from one logical array to another logical array. The reassignment could include evacuating data and metadata from the storage node, removing the storage node from one logical array, declaring that the storage node is unassigned, and then assigning the storage node to the other logical array, as described above with reference to Figs. 4 and 5. After reassigning the storage node, the flow proceeds back to the action 612, to continue data striping. Alternatively, flow could proceed elsewhere to perform further actions or determinations.

**[0055]** It should be appreciated that the methods described herein may be performed with a digital processing system, such as a conventional, general-purpose computer system. Special purpose computers, which are designed or programmed to perform only one function may be used in the alternative. Fig. 7 is an illustration showing an exemplary computing device which may implement the embodiments described herein. The computing device of Fig. 7 may be used to perform embodiments of the functionality for a storage node or a non-volatile solid state storage in accordance with some embodiments. The computing device includes a central processing unit (CPU) 701, which is coupled through a bus 705 to a memory 703, and mass storage device 707. Mass storage device 707 represents a persistent data storage device such as a disc drive, which may be local or remote in some embodiments. The mass storage device 707 could implement a backup storage, in some embodiments. Memory 703 may include read only memory, random access memory, etc. Applications resident on the computing device may be stored on or accessed via a computer readable medium such as memory 703 or mass storage device 707 in some embodiments. Applications may also be in the form of modulated electronic signals modulated accessed via a network modem or other network interface of the computing device. It should be appreciated that CPU 701 may be embodied in a general-purpose processor, a special purpose processor, or a specially programmed logic device in some embodiments.

**[0056]** Display 711 is in communication with CPU 701, memory 703, and mass storage device 707, through bus 705. Display 711 is configured to display any visualization tools or reports associated with the system described herein. Input/output device 709 is coupled to bus 705 in order to communicate information in command selections to CPU 701. It should be appreciated that data to and from external devices may be communicated through the input/output device 709. CPU 701 can be defined to execute the functionality described herein to enable the functionality described with reference to Figs. 1-6. The code embodying

this functionality may be stored within memory 703 or mass storage device 707 for execution by a processor such as CPU 701 in some embodiments. The operating system on the computing device may be MS-WINDOWS™, UNIX™, LINUX™, iOS™, CentOS™, Android™, Redhat Linux™, z/OS™, or other known operating systems. It should be appreciated that the embodiments described herein may be integrated with virtualized computing system also.

**[0057]** Detailed illustrative embodiments are disclosed herein. However, specific functional details disclosed herein are merely representative for purposes of describing embodiments. Embodiments may, however, be embodied in many alternate forms and should not be construed as limited to only the embodiments set forth herein.

**[0058]** It should be understood that although the terms first, second, etc. may be used herein to describe various steps or calculations, these steps or calculations should not be limited by these terms. These terms are only used to distinguish one step or calculation from another. For example, a first calculation could be termed a second calculation, and, similarly, a second step could be termed a first step, without departing from the scope of this disclosure. As used herein, the term "and/or" and the "/" symbol includes any and all combinations of one or more of the associated listed items.

**[0059]** As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "comprises", "comprising", "includes", and/or "including", when used herein, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof. Therefore, the terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting.

**[0060]** It should also be noted that in some alternative implementations, the functions/acts noted may occur out of the order noted in the figures. For example, two figures shown in succession may in fact be executed substantially concurrently or may sometimes be executed in the reverse order, depending upon the functionality/acts involved.

**[0061]** With the above embodiments in mind, it should be understood that the embodiments might employ various computer-implemented operations involving data stored in computer systems. These operations are those requiring physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or

magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. Further, the manipulations performed are often referred to in terms, such as producing, identifying, determining, or comparing. Any of the operations described herein that form part of the embodiments are useful machine operations. The embodiments also relate to a device or an apparatus for performing these operations. The apparatus can be specially constructed for the required purpose, or the apparatus can be a general-purpose computer selectively activated or configured by a computer program stored in the computer. In particular, various general-purpose machines can be used with computer programs written in accordance with the teachings herein, or it may be more convenient to construct a more specialized apparatus to perform the required operations.

**[0062]** A module, an application, a layer, an agent or other method-operable entity could be implemented as hardware, firmware, or a processor executing software, or combinations thereof. It should be appreciated that, where a software-based embodiment is disclosed herein, the software can be embodied in a physical machine such as a controller. For example, a controller could include a first module and a second module. A controller could be configured to perform various actions, e.g., of a method, an application, a layer or an agent.

**[0063]** The embodiments can also be embodied as computer readable code on a non-transitory computer readable medium. The computer readable medium is any data storage device that can store data, which can be thereafter read by a computer system. Examples of the computer readable medium include hard drives, network attached storage (NAS), read-only memory, random-access memory, CD-ROMs, CD-Rs, CD-RWs, magnetic tapes, and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network coupled computer system so that the computer readable code is stored and executed in a distributed fashion. Embodiments described herein may be practiced with various computer system configurations including hand-held devices, tablets, microprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers and the like. The embodiments can also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a wire-based or wireless network.

**[0064]** Although the method operations were described in a specific order, it should be understood that other operations may be performed in between described operations, described operations may be adjusted so that they occur at slightly different times or the

described operations may be distributed in a system which allows the occurrence of the processing operations at various intervals associated with the processing.

**[0065]** In various embodiments, one or more portions of the methods and mechanisms described herein may form part of a cloud-computing environment. In such embodiments, resources may be provided over the Internet as services according to one or more various models. Such models may include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In IaaS, computer infrastructure is delivered as a service. In such a case, the computing equipment is generally owned and operated by the service provider. In the PaaS model, software tools and underlying equipment used by developers to develop software solutions may be provided as a service and hosted by the service provider. SaaS typically includes a service provider licensing software as a service on demand. The service provider may host the software, or may deploy the software to a customer for a given period of time. Numerous combinations of the above models are possible and are contemplated.

**[0066]** Various units, circuits, or other components may be described or claimed as "configured to" perform a task or tasks. In such contexts, the phrase "configured to" is used to connote structure by indicating that the units/circuits/components include structure (e.g., circuitry) that performs the task or tasks during operation. As such, the unit/circuit/component can be said to be configured to perform the task even when the specified unit/circuit/component is not currently operational (e.g., is not on). The units/circuits/components used with the "configured to" language include hardware--for example, circuits, memory storing program instructions executable to implement the operation, etc. Reciting that a unit/circuit/component is "configured to" perform one or more tasks is expressly intended not to invoke 35 U.S.C. 112, sixth paragraph, for that unit/circuit/component. Additionally, "configured to" can include generic structure (e.g., generic circuitry) that is manipulated by software and/or firmware (e.g., an FPGA or a general-purpose processor executing software) to operate in manner that is capable of performing the task(s) at issue. "Configured to" may also include adapting a manufacturing process (e.g., a semiconductor fabrication facility) to fabricate devices (e.g., integrated circuits) that are adapted to implement or perform one or more tasks.

**[0067]** The foregoing description, for the purpose of explanation, has been described with reference to specific embodiments. However, the illustrative discussions above are not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations are possible in view of the above teachings. The embodiments

were chosen and described in order to best explain the principles of the embodiments and its practical applications, to thereby enable others skilled in the art to best utilize the embodiments and various modifications as may be suited to the particular use contemplated. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

# CLAIMS

What is claimed is:

1. A storage cluster comprising:

a plurality of storage nodes coupled together as the storage cluster;

the plurality of storage nodes configured to assign data to two or more logical arrays; and

the plurality of storage nodes configured to establish data striping across the plurality of storage nodes for user data of each of the two or more logical arrays.

2. The storage cluster of claim 1, further comprising:

the plurality of storage nodes configured to establish metadata and control for user data of a first logical array in a first subset of the plurality of storage nodes; and

the plurality of storage nodes configured to establish metadata and control for user data of a second logical array in a second subset of the plurality of storage nodes, wherein the first subset and the second subset do not overlap, and wherein the data striping for the user data of each of the first logical array and the second logical array is across the first subset and the second subset.

3. The storage cluster of claim 1, further comprising:

the plurality of storage nodes configured to establish metadata and control, for both user data of a first logical array and user data of a second logical array, across the plurality of storage nodes.

4. The storage cluster of claim 1, further comprising:

the plurality of storage nodes configured to assign a first subset of the plurality of storage nodes to a first logical array and a second subset of the plurality of storage nodes to a second logical array, wherein the data striping across the plurality of storage nodes is wider as compared to data striping across only the first subset or data striping across only the second subset.

5. The storage cluster of claim 1, further comprising:

the plurality of storage nodes configured to establish first metadata and first control for first user data of a first logical array in a first subset of the plurality of storage nodes; and

the plurality of storage nodes configured to establish second metadata and second control for second user data of a second logical array in a second subset of the plurality of storage nodes, wherein the first metadata and the first control are isolated from the second metadata and the second control.

6. The storage cluster of claim 1, further comprising:

a single chassis housing the plurality of storage nodes.

7. The storage cluster of claim 1, further comprising:

two or more chassis having the plurality of storage nodes therein, wherein the data striping across the plurality of storage nodes spans at least two of the two or more chassis.

8. A plurality of storage nodes, comprising:

one or more chassis;

a plurality of storage nodes in the one or more chassis, wherein the one or more chassis couples the plurality of storage nodes as a storage cluster;

the plurality of storage nodes configured to establish a first logical array and a second logical array; and

the plurality of storage nodes configured to perform data striping for first user data of the first logical array across the plurality of storage nodes and data striping for second user data of the second logical array across the plurality of storage nodes.

9. The plurality of storage nodes of claim 8, wherein:

metadata and control for the first user data are assigned to a first subset of the plurality of storage nodes;

metadata and control for the second user data are assigned to a second subset of the plurality of storage nodes; and

a fault in processing the first user data is confined to the first subset of the plurality of storage nodes and isolated from the second subset of the plurality of storage nodes.

10. The plurality of storage nodes of claim 8, further comprising:

the plurality of storage nodes configured to share metadata and control for each of the first user data and the second user data across the plurality of storage nodes.

11. The plurality of storage nodes of claim 8, wherein:

the one or more chassis includes two chassis, with the plurality of storage nodes occupying the two chassis;

the data striping for the first user data spans the two chassis; and

the data striping for the second user data spans the two chassis.

12. The plurality of storage nodes of claim 8, further comprising:

the plurality of storage nodes configured to perform automatic provisioning, including determining arrangement of metadata, control and user data relative to the first logical array and the second logical array.

13. The plurality of storage nodes of claim 8, further comprising:

the plurality of storage nodes configured to shift compute resources of a one of the plurality of storage nodes from the first logical array to the second logical array; and

the plurality of storage nodes configured to assign compute resources of a further storage node to either the first logical array or the second logical array regardless of which of the one or more chassis the further storage node occupies or which slot in the one or more chassis the further storage node occupies.

14. A method for data striping across storage nodes in a storage cluster, the method comprising:

creating at least a first logical array and a second logical array in a plurality of storage nodes of a storage cluster;

performing data striping for first user data, of which the first logical array has ownership, across the plurality of storage nodes; and

performing data striping for second user data, of which the second logical array has ownership, across the plurality of storage nodes.

15. The method of claim 14, further comprising:

sharing metadata and control for the first user data and the second user data across the plurality of storage nodes.

16. The method of claim 14, further comprising:

establishing first metadata and first control for the first user data in a first subset of the plurality of storage nodes;

establishing second metadata and second control for the second user data in a second subset of the plurality of storage nodes, wherein the first subset and the second subset are non-overlapping.

17. The method of claim 14, wherein assigning the plurality of storage nodes further comprises:

assigning a first subset of the plurality of storage nodes to the first logical array to control the first user data; and

assigning a second subset of the plurality of storage nodes to the second logical array to control the second user data, wherein assigning the first subset and assigning the second subset includes an automatic provisioning.

18. The method of claim 14, wherein assigning the plurality of storage nodes further comprises:

assigning a first subset of the plurality of storage nodes to the first logical array to control the first user data; and

assigning a second subset of the plurality of storage nodes to the second logical array to control the second user data, wherein assigning the first subset and assigning the second subset includes a manual provisioning.

19. The method of claim 14, further comprising:

reassigning one of the plurality of storage nodes, from being assigned to the first logical array and controlling a portion of the first user data, to being assigned to the second logical array and controlling a portion of the second user data.

20. The method of claim 14, further comprising:

indicating, at a visual indicator of each storage node of the plurality of storage nodes, responsive to the assigning, whether the storage node is unassigned, whether the storage node is assigned to the first logical array, whether the storage node is assigned to the second logical array, or whether the storage node is shared by two or more logical arrays.

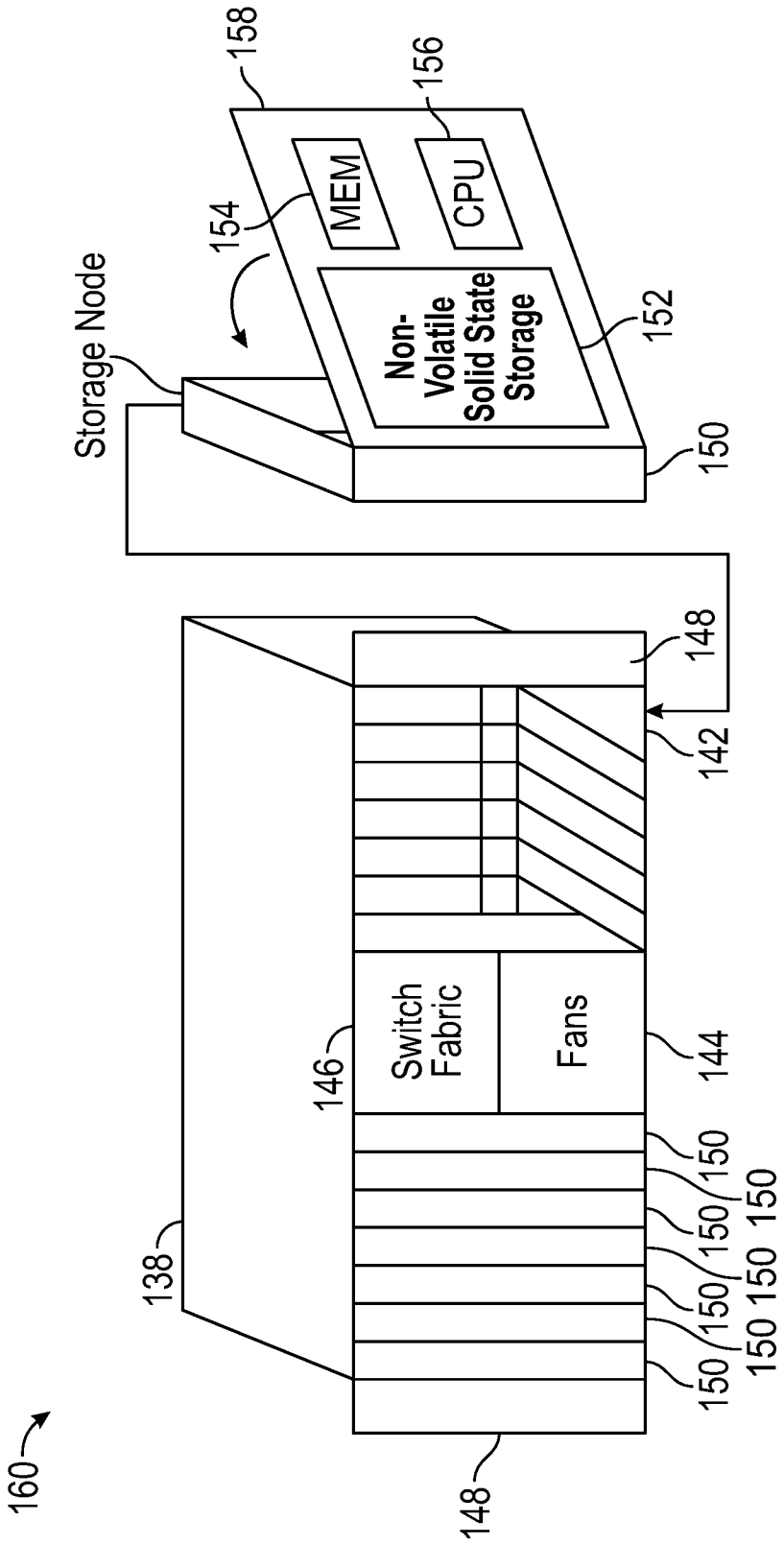
158
156
MEM
CPU
154
Storage Node
Non-Volatile Solid State Storage
152
150
148
142
Switch Fabric
Fans
146
144
150
150
150
150
150
150
150
150
138
148
160

FIG. 1

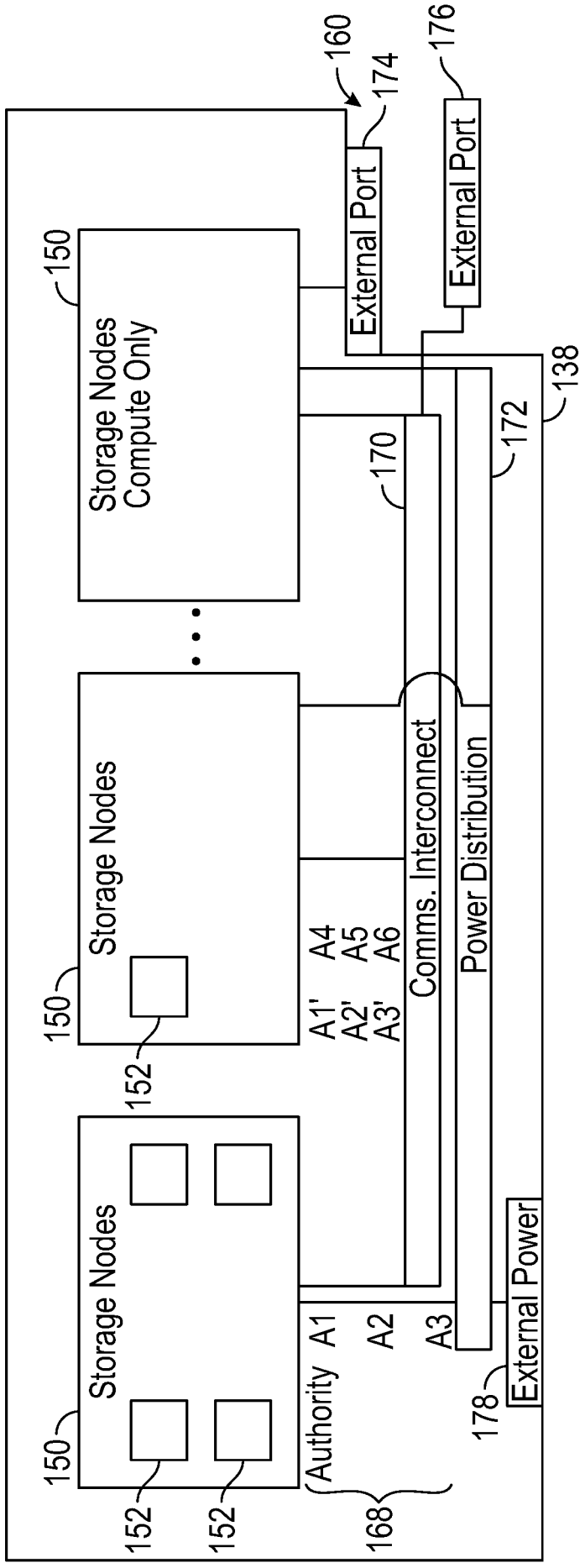
160
174
176
External Port
External Port
150
Storage Nodes
Compute Only
138
170
172
150
Storage Nodes
152
A1' A4
A2' A5
A3' A6
Comms. Interconnect
Power Distribution
150
Storage Nodes
152
152
Authority A1
A2
A3
168
178
External Power

FIG. 2

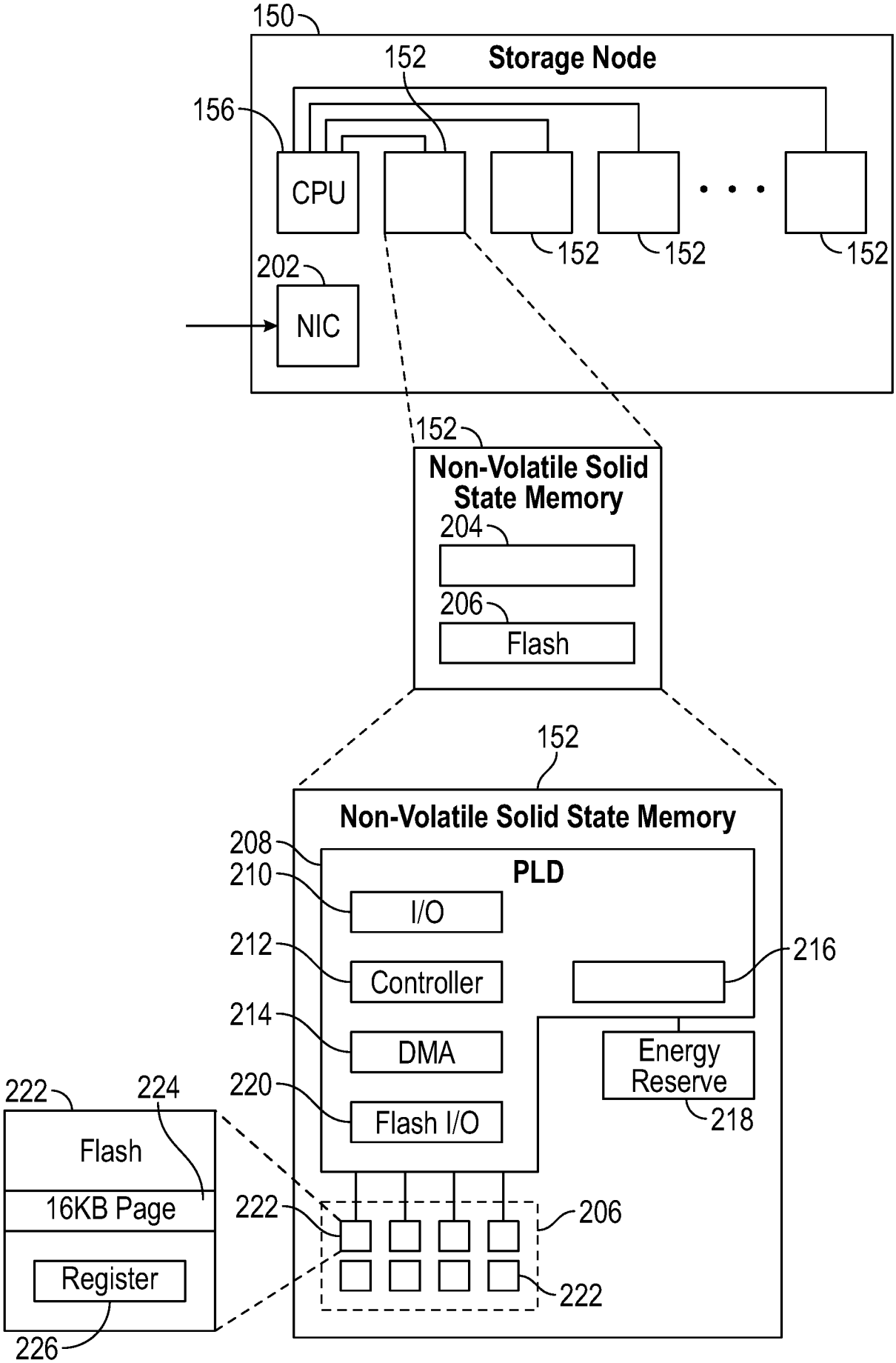
150
Storage Node
152
156
CPU
152
152
152
202
NIC
152
Non-Volatile Solid State Memory
204
206
Flash
152
Non-Volatile Solid State Memory
208
PLD
210
I/O
212
Controller
216
214
DMA
Energy Reserve
220
Flash I/O
218
222
224
Flash
16KB Page
Register
226
222
206
222

FIG. 3

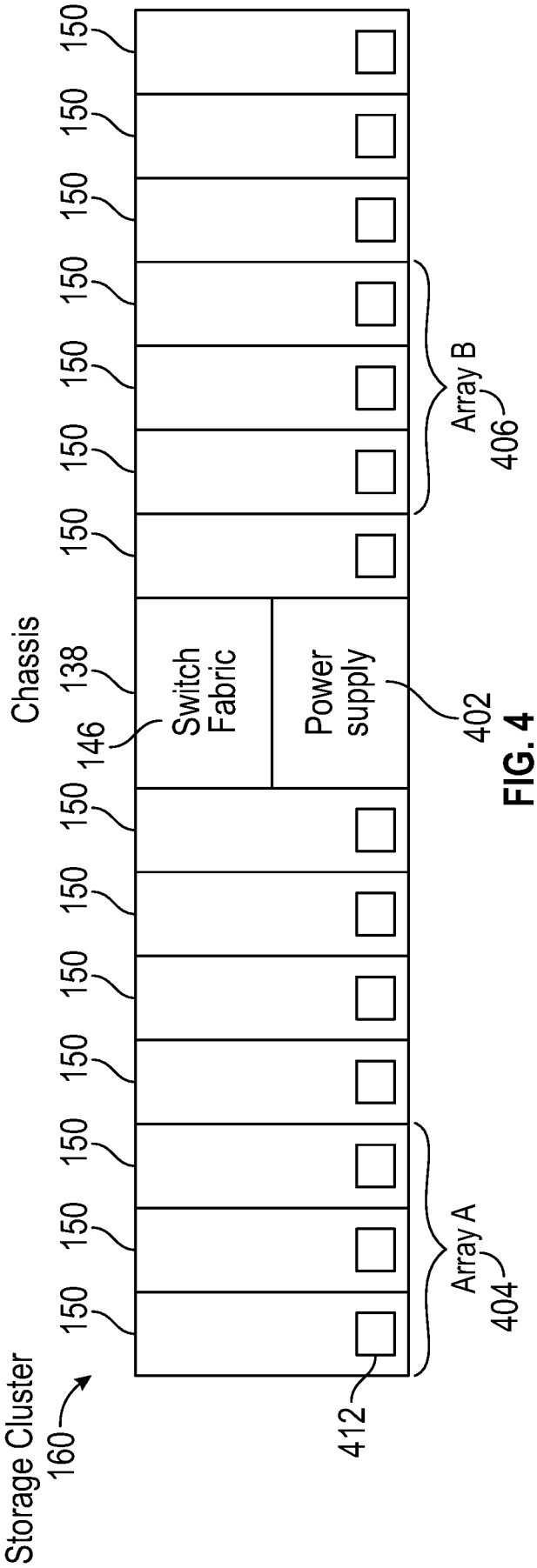
Storage Cluster
160
Chassis
150
138
146
Switch Fabric
Power supply
402
Array B
406
Array A
404
412

FIG. 4

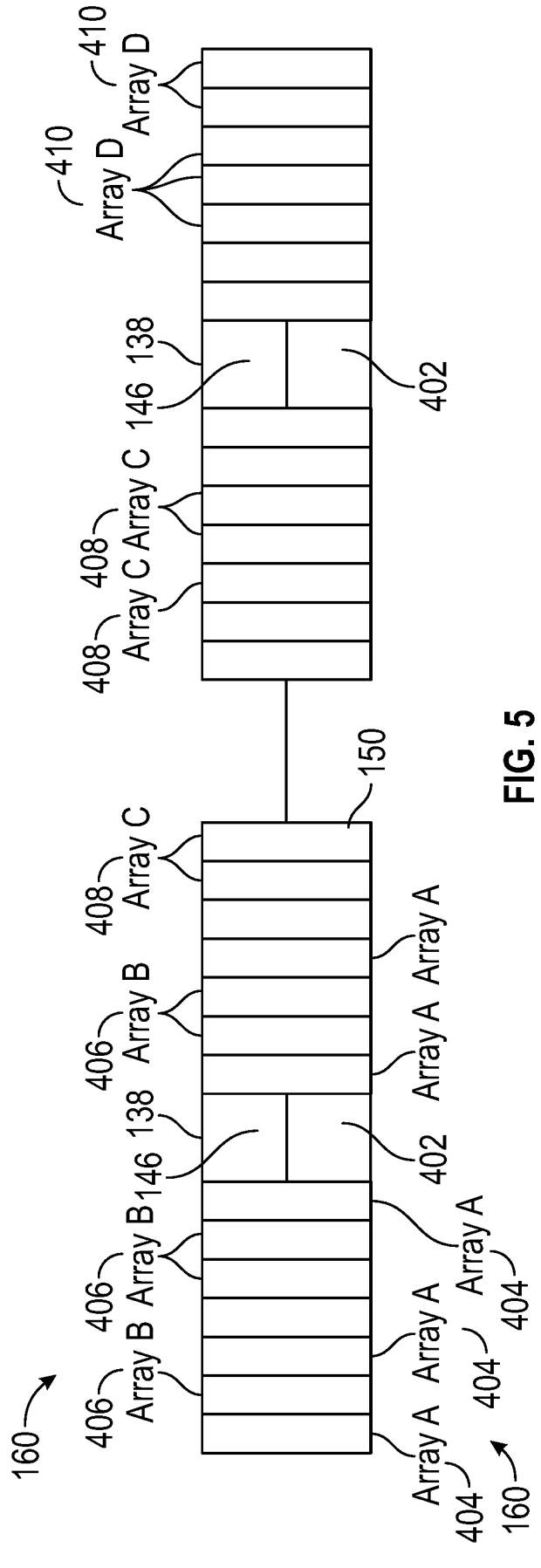
160
410
Array D
408
Array C
406
Array B
404
Array A
138
146
402
150

FIG. 5

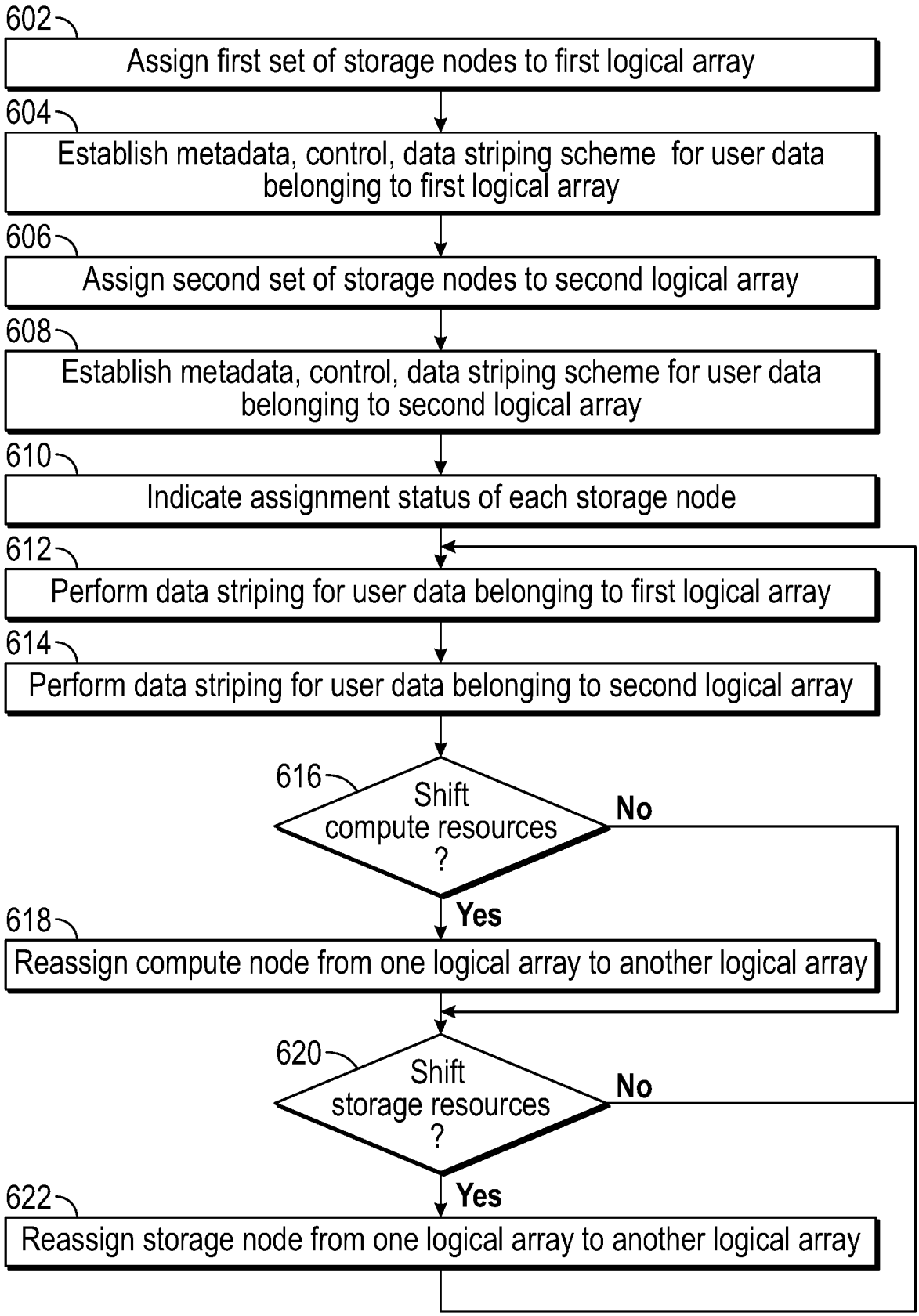
602
Assign first set of storage nodes to first logical array
604
Establish metadata, control, data striping scheme for user data belonging to first logical array
606
Assign second set of storage nodes to second logical array
608
Establish metadata, control, data striping scheme for user data belonging to second logical array
610
Indicate assignment status of each storage node
612
Perform data striping for user data belonging to first logical array
614
Perform data striping for user data belonging to second logical array
616
Shift compute resources ?
No
Yes
618
Reassign compute node from one logical array to another logical array
620
Shift storage resources ?
No
Yes
622
Reassign storage node from one logical array to another logical array

FIG. 6

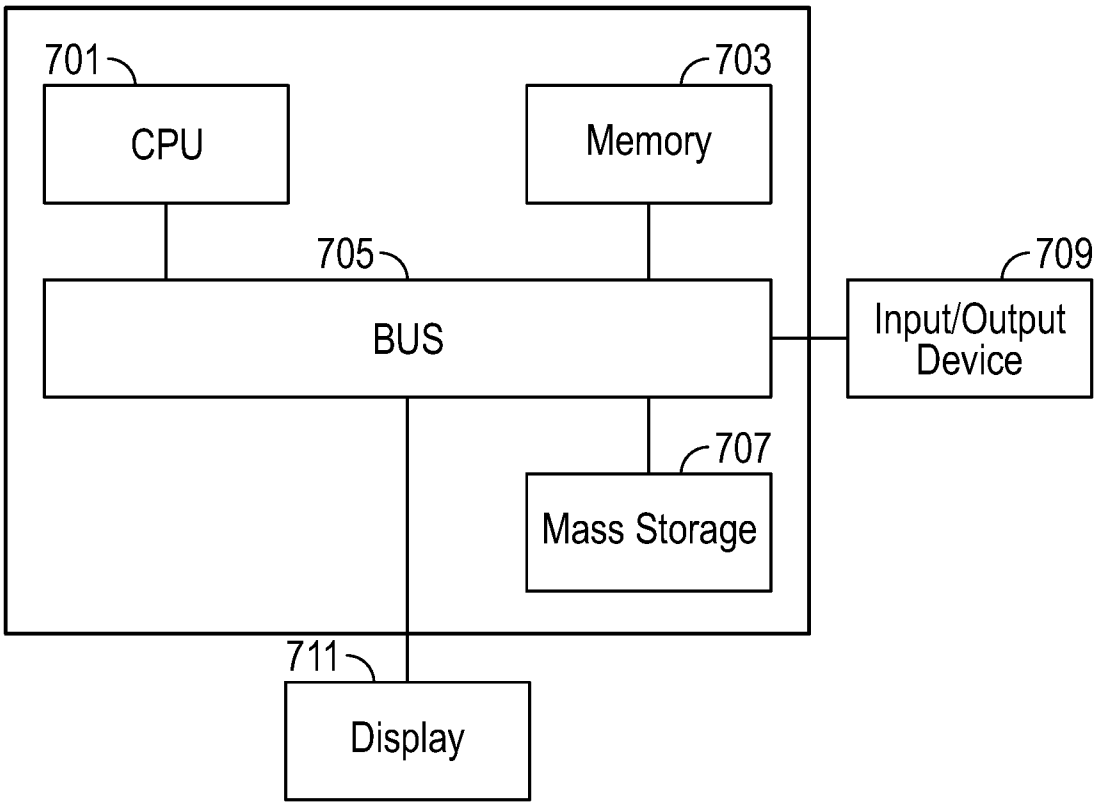
701
CPU
703
Memory
705
BUS
709
Input/Output Device
707
Mass Storage
711
Display

FIG. 7

**INTERNATIONAL SEARCH REPORT**

International application No.
**PCT/US2016/024391**

**A. CLASSIFICATION OF SUBJECT MATTER**

**G06F 3/06(2006.01)i**

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 3/06; G06F 12/02; G06F 12/16; G06F 11/10; G06F 12/00; G06F 11/00; G06F 13/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords:storage node, storage cluster, assign, logical array, data striping, user data

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | US 8850108 B1 (PURE STORAGE, INC.) 30 September 2014<br>See column 1, lines 20-27; column 2, lines 1-56; column 12, lines 3-5; and figures 1, 6. | 1-20 |
| Y | US 2009-0198940 A1 (KEVIN JOHN ASH et al.) 06 August 2009<br>See paragraphs [0009]-[0022]; and figures 2-3. | 1-20 |
| Y | US 7032125 B2 (KEITH W. HOLT et al.) 18 April 2006<br>See column 3, line 47 - column 5, line 4; claim 1; and figures 3-5. | 2,5,9,16 |
| A | WO 2012-044488 A1 (PURE STORAGE, INC.) 05 April 2012<br>See paragraphs [0007]-[0008]; and figure 2. | 1-20 |
| A | US 2004-0024963 A1 (NISHA TALAGALA et al.) 05 February 2004<br>See paragraphs [0008]-[0010]; and figures 2-4. | 1-20 |

☐ 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 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 |
|---|---|
| 12 July 2016 (12.07.2016) | **12 July 2016 (12.07.2016)** |


Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
BYUN, Sung Cheal
Telephone No. +82-42-481-8262


Form PCT/ISA/210 (second sheet) (January 2015)

**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2016/024391**

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| US 8850108 B1 | 30/09/2014 | AU 2015-268889 A1 | 25/02/2016 |
| | | US 2015-0355848 A1 | 10/12/2015 |
| | | US 2015-0355969 A1 | 10/12/2015 |
| | | US 2016-085628 A1 | 24/03/2016 |
| | | US 9201600 B1 | 01/12/2015 |
| | | WO 2015-187218 A1 | 10/12/2015 |
| US 2009-0198940 A1 | 06/08/2009 | US 8423739 B2 | 16/04/2013 |
| US 7032125 B2 | 18/04/2006 | US 2003-204670 A1 | 30/10/2003 |
| WO 2012-044488 A1 | 05/04/2012 | CN 103348326 A | 09/10/2013 |
| | | CN 103348326 B | 30/03/2016 |
| | | EP 2622475 A1 | 07/08/2013 |
| | | JP 2013-539132 A | 17/10/2013 |
| | | JP 5894167 B2 | 23/03/2016 |
| | | KR 10-2013-0118876 A | 30/10/2013 |
| | | US 2012-0079318 A1 | 29/03/2012 |
| | | US 2014-317447 A1 | 23/10/2014 |
| | | US 8775868 B2 | 08/07/2014 |
| US 2004-0024963 A1 | 05/02/2004 | US 7051155 B2 | 23/05/2006 |