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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 August 2015 (06.08.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/114643 A1

(51) International Patent Classification:
G06F 11/00 (2006.01)

(21) International Application Number:
PCT/IN2014/000074

(22) International Filing Date:
30 January 2014 (30.01.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445, Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventors; and
(71) Applicants : **MOHANTA, Taranisen** [IN/IN]; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore 560 048 (IN). **BASIREDDY, Ranjith Reddy** [IN/IN]; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore 560 048 (IN). **NAYAKA B, Govindaraja** [IN/IN]; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore 560 048 (IN). **MAJEED, Zameer** [IN/IN]; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore 560 048 (IN).

(74) Agents: **THIAGARAJAN, Srinivasan** et al.; Lakshmikumaran & Sridharan, B6/10, Safdarjung Enclave, New Delhi 110 029 (IN).

(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, LT, 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, 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).

[Continued on next page]

(54) Title: DATA STORAGE SYSTEM REBUILD

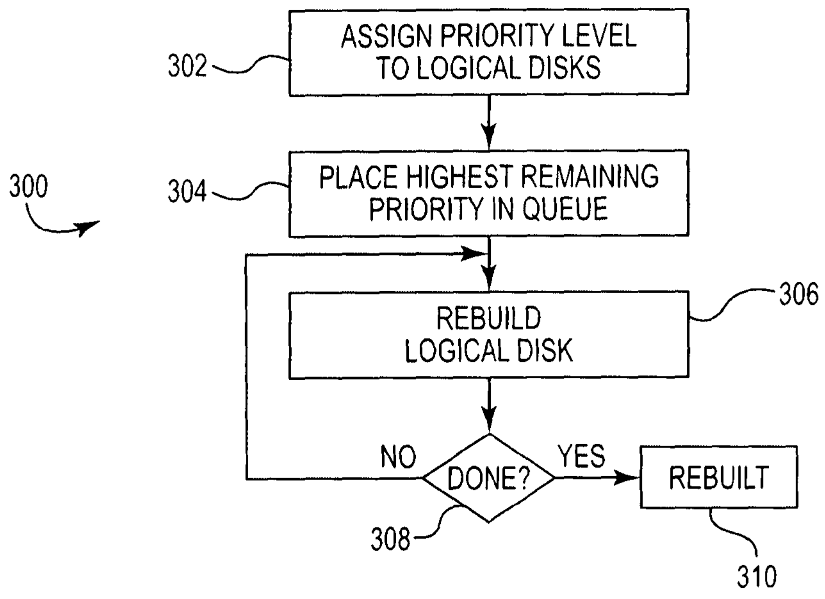

ASSIGN PRIORITY LEVEL TO LOGICAL DISKS
302
PLACE HIGHEST REMAINING PRIORITY IN QUEUE
304
300
REBUILD LOGICAL DISK
306
NO
DONE?
YES
REBUILT
308
310

Fig. 6

(57) **Abstract**: A method of rebuilding data stored on a failed physical disk in a storage array is disclosed. Each logical disk in a set of logical disks having data stored on the failed physical disk is assigned a distinguishable priority level from highest priority to lowest priority. The set of logical disks are rebuilt in an order of highest priority to lowest priority. Other priorities are dynamically assigned during rebuilding the failed disk from the logical disk assigned with the highest priority.

WO 2015/114643 A1

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

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

DATA STORAGE SYSTEM REBUILD

BACKGROUND

[0001] Disk array data storage systems have multiple storage disk drive devices that are arranged and coordinated to form a single mass storage system that provides high data availability. Data availability relates to the ability of a mass storage system to access data stored in the storage system while ensuring continued operation in the event of a disk or component failure. Data availability is often provided through the use of redundancy where data, or relationships among data, are stored in multiple locations in the storage system. In the event of disk failure, redundant data is retrieved from the operable portion of the system and used to regenerate the original data that is lost due to the component failure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Figure 1 is a block diagram illustrating a system having a host computer system and a data storage system.

[0003] Figure 2 is a block diagram illustrating the host computer system and the data storage system of Figure 1 in more detail.

[0004] Figure 3 is a block diagram illustrating an example of a portion of the data storage system of Figure 1 as it corresponds to logical disks.

[0005] Figure 4 is a block diagram illustrating a portion of a spare disk in the data storage system of Figure 3 during a first stage of rebuild.

[0006] Figure 5 is a block diagram illustrating a portion of the data storage system of Figure 3 in a second stage of rebuild.

[0007] Figure 6 is a flow diagram illustrating a process of rebuilding a failed disk in the storage system of Figure 1.

[0008] Figure 7 is a block diagram illustrating a portion of the data storage system of Figure 2 in more detail.

DETAILED DESCRIPTION

[0009] In the following detailed description, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific examples in which the disclosure may be practiced. It is to be understood that other examples may be utilized and structural or logical changes may be made without departing from the scope of the present disclosure. The following detailed description, therefore, is not to be taken in a limiting sense, and the scope of the present disclosure is defined by the appended claims. It is to be understood that features of the various examples described herein may be combined, in part or whole, with each other, unless specifically noted otherwise.

[0010] Two common mechanisms for storing redundant data on disk drives include mirrored redundancy and parity redundancy. In mirrored redundancy, the data being stored is duplicated and stored in two separate areas of the storage system that are the same size and include an original data storage area and a redundant storage area. In parity redundancy, the original data is stored in an original data storage area and the redundant data is parity data is stored in a storage area size less than the size of the original data storage area. Additionally, a data storage system may permit data to be stored in multiple redundancy groups co-existing within the system. For example, RAID (Redundant Array of Independent Disks) storage systems are disk array systems in which part of the physical storage capacity is used to store redundant data. RAID systems are typically characterized as one of several architectures or levels, enumerated under the acronym RAID. RAID architectures can include storage disks configured according to mirrored redundancy, parity-type redundant storage, and RAID systems can include both mirrored redundancy architectures and parity redundancy architectures.

[0011] In the event a disk in a RAID system fails, the data in the array is "rebuilt," which is a process that typically involves issuing multiple read and/or write requests to the disk array. Typically, the RAID system is also available for read and write requests from a host computer during this rebuilding process. Unfortunately, host requests and rebuild requests often require access to the same resources in the RAID systems. The contention for resources during rebuilds often affects either host requests, rebuild requests, or both requests. To compound matters, many storage systems employ larger and larger disks capable of storing more and more amounts of data. Accordingly, rebuilding data in the storage array can take a very long time. Further, failures of other disks during rebuild can result in permanent data loss.

[0012] Figure 1 illustrates an example computing system 20 having a host 22 connected to a data storage system 24 via an input/output (I/O) interface bus 26. Host 22 can be a general-purpose computing device that can be configured, for example, as a server or workstation. Host 22 can include a visual display device 28, one or more processing units 30, such as central processing units (CPU) or general purpose-computing on graphics processing units (GPGPU), input devices such as a keyboard 32, and a mouse 34. Other data entry and output peripherals may also be included, such as a printer, storage, optical drive, network interfaces, and so forth. In the example, host 22 is coupled to a network 36, such as a wide area network (WAN) or Internet, to serve data from the data storage system 24 to one or more clients 38.

[0013] Data storage system 24 represents a storage array configured to hold user data and other information in a computer readable form. In one implementation, data storage system 24 is a hierarchical RAID storage system that is capable of storing data according to different and/or multiple redundancy schemes. Host 22 provides an interface for an administrator or other user to manage the RAID storage system, such as to run diagnostics, evaluate performance, or set operating parameters. For ease of explanation, data storage system 24 is described herein as a RAID system, although other types of storage arrays could alternatively be used.

[0014] Figure 2 illustrates host 22 and data storage system 24 in more detail. The host 22 includes one or more processors 30, a volatile memory 42, a keyboard 32, a mouse 34, a non-volatile memory 44, and a display 28. An administrator module 46 is stored in memory 44 and executes on processor 40. Administrator module 46 can set various parameters of data storage system 24 and also provide management functions such as diagnostics, performance review, or capacity analysis. Administrator module 46 supports a storage manager user interface (UI) 48 that presents a visual interface on the display 28. An administrator, via UI 48, can alter various parameters in data storage system 24 to control the priority given to host I/O requests versus rebuild I/O requests, as discussed in more detail below.

[0015] Data storage system 24 includes a disk array 50 with multiple storage disks 52, one or more disk array controllers 54, and a system manager 56, such as a RAID management system. As used in this disclosure, a "disk" refers to any mass storage device, typically a non-volatile, randomly accessible, and rewriteable mass storage device. Examples of disks include magnetic disks, optical disks, and solid state storage devices (SSD), which can include one or more non-volatile electronic storage elements such as PROMs, EPROMs, or EEPROMs.

[0016] Disk array controller 54 is coupled to the disk array 50 via one or more interface buses 58, such as a small computer system interface (SCSI). system manager 56 is coupled to the disk array controllers 54 via an interface protocol 60. The disk array controller can include memory 62. The system manager 56 can be embodied in software, firmware, hardware, or a combination thereof, and can be embodied as a separate component (as shown), within the disk array controller 54, within the host 22, or other implementation. In one implementation, system manager 56 is a software module that runs on a processing unit of data storage system 24, or on processor 30 of computer 22. Alternatively, system manager 56 may be executed by another processing unit, be embodied in firmware, or be embodied in hardware such as an application specific integrated circuit (ASIC).

[0017] Disk array controller 54 coordinates data transfers to and from the disk array 50. In some examples, multiple disk array controllers are used. One or more redundancy groups can be implemented on disk array 50 and one or more RAID levels can be implemented in each redundancy group. Multiple disk array controllers enhance reliability by providing continuous backup and redundancy in the event that one controller becomes inoperable. Parallel disk array controllers can have respective mirrored memories 62 coupled together through a link. In one implementation, the memory 62 is a battery-backed, non-volatile RAM (NVRAM), although other types of memories could alternatively be used. The memory 62 can store several types of information, such as a memory map of the storage space in disk array 50, a read cache for data being read from disk array 50, a write cache for data before it is written to disk array 50. Different configurations of disk array controller 54 and memory 62 can be used.

[0018] Failure of a disk 52 refers to all or a portion of a disk 52 becoming inaccessible due to a problem with the disk itself or a component used in accessing the disk such as a bus. RAID systems are equipped with mechanisms to detect disk failures. Disk failure can result in a degraded RAID level if the disk stores data for the RAID level.

[0019] Due to the different RAID levels that may possibly be stored on disk array 50, failures of different disks can result in different RAID levels being rebuilt. The manner in which the data in the storage array is rebuilt can vary depending on numerous factors, such as the nature of the failure, the RAID levels employed in the array, the number of disks in the redundancy group. Examples of rebuilding data in an array include migrating data to other disks, RAID levels, or both so that a failed disk is not used, copying data to (or determining what data to write and writing that data to) a newly installed disk, writing data to a designated backup or spare disk. Rebuilding of a RAID level typically involves the reading of data from and/or the writing of data to disk array 50. Additionally, the rebuilding process may be performed automatically in response to detecting the failure, or alternatively may be performed in response to a user or administrator request.

[0020] Storage system 24 may use virtualization concepts as a tool to enable additional functionality and more advanced features. A logical disk is a device that provides an area of usable storage capacity on one or more physical disk drive components in a computer system. Other terms for logical disk include partition, logical volume, and a virtual disk (vdisk). In virtualized storage area networks, logical disks are presented to the operating system on the host 22. RAID architectures can provide logical disks from the arrays. A logical unit number, or LUN, is a number used to identify a logical unit, such as a logic disk.

[0021] A virtualization tool presents a logical space for data storage and manages the process of mapping it to the actual physical location. Virtualization tools include a meta-data mapping table that translates the incoming (virtual) disk identifier, and LBA (logical block addressing) to a physical disk identifier and LBA. Virtualization granularity depends on the implementation. Some virtualized systems simply provide minimum granularity with disk aggregation and the granularity is a physical disk itself. Other virtualization systems break down the physical disks into smaller chunks, or extents, that spread a single virtual disk across many physical disks. A larger chunk includes more data. A smaller chunk has less data and has higher granularity than a larger chunk.

[0022] Figure 3 illustrates a schematic diagram of a set of logical disks 100, in particular LUN 1 101, LUN 2 102, LUN 3 103, and LUN 4 104, having data allocated in chunks 106 across a set of physical disks 108 of a selected redundancy group 109 in a storage array 110. In the example of Figure 3, storage array 108 can correspond with storage array 50. Redundancy group 109 in storage array 110 includes disk 1 111, disk 2 112, disk 3 113, disk 4 114, disk 5 115 and can be configured to include one or more spare disks such as spare disk 116. In the example, the redundancy group 109 is configured in a RAID level. Chunks 106 include data in a size of a selected granularity. Figure 3 shows chunks 106 similarly allocated on each disk 1-5, 111, 112, 113, 114, 115, on physical disks 108, for ease of illustration, and other allocation configurations are possible. A table 120 is used to map addresses of the data in the logical disks 100 to addresses in the physical disks 108. The table 120 can also includes metadata 122. In the example, metadata 122 includes information 124 as to which chunks include usable data, such data written to the disk vial I/O operations, or have chunks that have data that has been erased or zeroed, or not written. In the example, disk 5 115 has failed, and will be rebuilt on spare disk 116.

[0023] As cost-per-megabyte of storage continues to decrease and data appears to have an unlimited potential for growth, storage systems are incentivized to increase data capacity. The time used to complete a data rebuild for a failed disk increases proportionately with the data capacity of the failed disk using legacy techniques. The number of disk failures before permanent data loss occurs varies by architecture, and some architectures can permanently lose data after two disk failures. If the disk is not rebuilt before the failure of a critical number of disks in the RAID level, data on the storage system can be permanently lost, i.e., the data is lost and the lost data is unrecoverable. While the time to complete a rebuild is proportional to the data capacity of the disk, the time to complete a rebuild before permanent data loss occurs corresponds with other factors and typically is not proportional to data capacity. Accordingly some storage systems may encounter permanent data loss if the time to complete rebuild is not made faster.

[0024] Legacy systems rebuild data from a disk that has failed onto a spare disk in one of several manners. In a first manner, a legacy system will rebuild the logical disks one at a time such as in the order the logical disks were created. When one logical disk is rebuilt, the legacy system will rebuild another logical disk until the rebuild of the failed disk is complete. In a second manner, a legacy system will rebuild a physical disk in some convenient method, such as by physical address, cylinder or stripe, sector, line number, or other grouping of data on a physical disk. RAID systems typically perform an online rebuild, i.e., where data is read from remaining disks and written to the spare disk while the host also performs I/O operations on the remaining disks. This contention for resources causes a degradation in performance in both serving host I/O requests and rebuild requests.

[0025] Legacy systems have attempted to improve the performance of the disk array during rebuild in a number of ways. One example legacy method includes prioritizing the host workload with the rebuild workload such that one or the other is preferred. In order to reduce the affect of a rebuild on host workload, request for host I/O may take priority over the rebuild. This method can extend rebuild time or reduce host throughput. Another example legacy system a storage controller selects a number of redundant storage devices over which to distribute a write-data based on loads of the processor of the storage controller and the internal network.

[0026] Figures 4 and 5 illustrate a process of rebuilding the failed disk 5 115 onto spare disk 6 116 while host 22 continues to serve I/O requests and in accordance with the present disclosure. In many storage systems, logical disks exist to serve various purposes. Some logical disks 100 can have critical data where the cost of permanent data loss is relatively very high. Other logical disks 100 can have less critical data where the cost of permanent data loss in the storage array 108 is not as high or relatively low. For example, some logical disks 100 are replicated in other storage arrays or in other storage systems. For these logical disks, the cost of permanent data loss in the storage array 108 is relatively low because other backups exist to rebuild the logical disk. Some logical disks may not include remote backups because of cost or infrastructure limitations.

[0027] In order to address this concern, logical disks 100 are assigned a relative priority, such as in the system manager 56, and the order the logical disks are rebuilt is based on the priority. For example, each of the logical disks 100 is assigned a separate relative priority. The order of the rebuild is in order of highest priority to lowest until the all of the logical disks 100 are rebuilt on the spare disk 116. In another example, one or more logical disks are given a relatively higher priority than other logical disks. The order of rebuild is based first on that priority. Once the prioritized logical disks are rebuilt, the remaining logical disks can be rebuilt using one or more of the several manners of legacy systems or other systems.

[0028] In the example, logical disk 3 103 is assigned the highest relative priority of the logical disks 100 having data stored in the storage 108. Logical disk 4

104 is assigned the next highest priority. Figure 4 illustrates data chunks 206 of logical disk 3 103 being rebuilt first onto spare disk 116. Once logical disk 3 103 has been rebuilt onto spare disk 116, the system manager 56 can begin rebuilding the logical disk with the next highest priority, i.e., logical disk 4 104. Figure 5 illustrates data chunks 216 of logical disk 4 104 being rebuilt next onto spare disk 116. Data chunks from logical disk 1 101 and logical disk 2 102 can be rebuilt onto spare disk similarly, in order of priority, or in another manner.

[0029] Figure 6 illustrates an example method 300 to rebuild a failed physical disk 110 in a storage array 108 in order of priority assigned to the logical disks 100. One or more selected logical disks are assigned a priority level with respect to one or more of the other logical disks 100 at 302. In one example, a system administrator or other user can assign the priority. In another example, a tool can analyze metadata regarding the logical disks 100 to assign the priority. The tool can take into consideration such information as the type of data stored in each of the logical disks, whether a logical disks is backed up in another redundancy group in array 110 or in a remote storage area network, and other relevant consideration. The tool can weigh these considerations and assign one or more, such as each, of the logical disks a priority. The highest priority logical disk is provided to a rebuild queue in the management system 56 at 304. The logical drive in the rebuild queue is rebuilt on the spare disk according to RAID techniques at 306. For example, rebuilding a physical disk in a RAID 5 architecture will include rebuilding data and constructing associated parity information. Once the highest priority logical disk is rebuilt, the next highest priority logical disk is rebuilt from the queue at 308 until all the logical disks 100 have been rebuilt at 310.

[0030] Often, logical disks 100 can includes irrelevant data such as at unused portions of the logical disks. Such irrelevant data includes data that has been nullified or erased or data at addresses that have not been written to. Read operations to irrelevant data waste resources particularly if such reads contend for resources with host I/O requests. Chunks 106 having irrelevant data can be marked as such in metadata 122.

[0031] As part of rebuild 306, the metadata 122 can be read from table 120 to determine which chunks 106 contain irrelevant data. Data chunks with irrelevant data are not read from the physical disks 108 during the rebuild. Read operations to the table 120 stored in memory of the storage system 24 is performed much more quickly than reads to the physical disks 108. Further, reads to table 120 to determine irrelevant data instead of to physical disks 108 eliminates a contention with host I/O requests to the physical disks 108. Chunks 106 containing irrelevant data can be written to the spare disk 116 during rebuild without interference with host I/O requests thus improving performance of the storage system 24 during rebuild.

[0032] Rebuild 306 can also be based on other policy considerations. In one example, either the rebuild or the host I/O requests can be given priority over the other. For example, a system administrator or other tool can assign or preselect a priority to either host requests or rebuild requests. If a priority is assigned, that priority is taken into consideration during the rebuild operation 306. In one example, rebuild requests are assigned a high, neutral, or low priority. In the case of a high priority, host requests are throttled in favor of rebuild requests. In the case of a low priority, rebuild requests are throttled in favor of host requests. In the case of a neutral priority, neither host requests nor rebuild request are throttled, and rebuild is performed in a default mode. In another example, additional priority levels, such as very high, high, neutral, low, very low, can be assigned with corresponding relative throttles. In one example of throttling, the number of threads in a process dedicated to host requests or rebuild request can be increased up to a maximum amount, decreased to a minimum amount, or adjusted to some other amount in between maximum and minimum.

[0033] In addition to a preselected rebuild priorities, or instead of them, the system manager 56 can dynamically set rebuild priority during rebuild. A dynamically set priority takes into consideration system performance and other information to change priorities from either default or otherwise preselected priorities to further improve performance of the storage system 24.

[0034] In one example of dynamically setting priorities during the rebuild 306, one or more of the physical disks 108 in redundancy group 109 can include features to detect and report on various indicators of reliability that are applied to anticipate failures in the physical disks. One such set of features can be found in the Self-Monitoring. Analysis and Reporting Technologies, or SMART, having technical documentation in the AT Attachment, or ATA, Standard, such as Section 4.21 in the revision of September 6, 2008. Rebuild priority can be increased automatically if a failure of another physical disk 108 in the redundancy group 109 is anticipated. Alternatively, rebuild priority can be decreased or set to neutral if physical disks 108 are indicated to be in better than expected working order.

[0035] In another example of dynamically setting priorities during the rebuild 306, the system manager 56 can anticipate periods of lower host request. The system manager can be equipped with a tool to track patterns of host requests. Based on the tracked patterns, the system manager 56 can provide a prediction as to periods of high and low host activity with respect to the storage system 24. For example, the patterns can be used to predict periods where the relative amount host requests is high or the type of host requests are intensive in contrast with periods where the relative host usage of storage system 23 is low. The system manager applies these predictions to reduce or set to neutral the rebuild priority during periods of anticipated high host usage to improve host I/O performance. The system manager can also apply these predictions to accelerate rebuilds during periods of anticipated low host usage to improve rebuild time.

[0036] Additionally, typical RAID systems serve requests unevenly on the physical disks 108. For example, some physical disks 108 may be busier than other disk in the redundancy group 109 serving host requests or rebuild requests. System manager 56 can track which disks are busier than others, and then allocate the rebuild 306 to the less busy drives. In the case where one or more drives are busy performing the rebuild 306, the less busy disks can be accessed to perform rebuild in parallel.

[0037] The RAID level of the failed disk can also be a factor on rebuilding priority or on rebuilding acceleration. In one example, a rebuild may be accelerated if the failed disk is part of a RAID level that will permanently lose data if another disk in the redundancy group fails, or if two disks fail and one is indicated as anticipated failing from SMART technology. Further, if two different redundancy groups have failed disks, priority can be given to certain RAID levels over other RAID levels. In one example, a failed disk in a RAID 5 redundancy group can received priority over a failed disk in a RAID 6 redundancy group.

[0038] Figure 7 illustrates system manger 56 in more detail including features described above. System manager 56 includes a request queue 66, a queue controller 68, a priority identifier 70, a request dispatcher 72, a rebuild controller 74, and a request processor 76. Rebuild controller 74 manages the rebuilding of data in any of the one or more RAID levels maintained by disk array 50 in the event that one or more of disks 52 fails. System manager 56 can be implemented as hardware, firmware, software, or any combination thereof. For example, all of part of the system manager 56 can be implemented as a computer storage medium storing computer executable instructions to perform a method. Computer storage media in this disclosure does not include a transitory propagating signal.

[0039] Request queue 66 can include a logical queue structure that is an abstraction of the actual queue mechanism implementation. Request queue 66 can be implemented in any of a wide variety of manners. Examples of such implementations include: a single queue may be used into which all I/O requests -- regardless of whether the request is from the host or rebuild -- are placed; multiple queues may be used and I/O requests placed in one queue while rebuild requests are placed in another queue, or alternatively requests of different priorities may be placed into different queues; a simple list may be maintained of the requests and time of receipt of each request and optionally priority of each request.

[0040] Request queue 66 stores I/O requests targeting disk array 50 and to be processed by controller 54. These I/O requests can originate with host 22 or alternatively rebuild controller 74, and can be input and/or output requests such as read requests, write requests, or other requests. In one example, requests are input to request queue 66 via queue controller 68, although alternatively requests could be input to request queue 66 directly via host 22 or rebuild controller 74. Requests can be input to request queue 66 in different manners, such as in the order they are received or according to some other ordering as discussed.

[0041] Requests are retrieved from request queue 66 by request dispatcher 72 and forwarded to request processor 76 for processing. Processing of a request varies based on the type of request, but generally refers to performing any of the actions necessary to carry out the request. These actions may include, for example, calculating parity values, writing data to one or more disks, or reading data from one or more disks. Request dispatcher 72 can retrieve requests from request queue 66 in the order they were placed in the queue, or alternatively according to some other ordering.

[0042] Priority identifier 70 determines whether host I/O requests or rebuild I/O requests should have priority over the other or how much priority one should have over the other. The behavior of queue controller 68 and/or request dispatcher 72 may change based on whether host I/O requests or rebuild I/O requests should have priority over the other.

[0043] The system manager 56 also includes a rebuild priority controller 80 for improving the performance of a rebuild. The rebuild priority controller 80 includes a prioritizer 82, a priority queue 84, a tracker 86, and a predictor 88. The prioritizer 82 assigns relative priorities to the logical disks and determines an order to the logical disks 100 to be rebuilt. The order is presented to rebuild queue 84, which can be constructed similarly to the request queue 66. The prioritizer 82 also assigns RAID level priority, and other rebuild priorities applied during rebuild. Dynamic rebuild priorities can be determined from the tracker 86 and predictor 88. In one example, the tracker 86 receives information regarding the reliability of the working physical disks and provides it to the predictor 88, which anticipates failures. Further, the tracker 86 receives data from the priority identifier 70 regarding host requests and rebuild request, and predicts periods of low host requests. Further the tracker 86 receives data from the request processor 76 to determine which disks 108 are being read for rebuild operations and which disks 108 are processing host requests. This information can be provided to the predictor 88 to make dynamic determinations.

[0044] An example process is set forth below:

```
Disk Rebuild Performance Optimizer (inparam: redundancyGroup)
{

         //Luns are rebuilt in order of priority
         //List all the policies from the engine


         lun_priority        //in this example, values are 1 - n

         rebuild_priority    // Values can be high/neutral/low)( high -> rebuild has
                             // priority, neutral -> default  policy, low -> host
requests
                             // have priority )

         rebuild_adaptive_operation    // Values can be  yes/no . If yes-> firmware will
                                       // track the host request patterns and predict the
                                       // future host workload and adapt the rebuild
                                       // operation priority. No-> default  behavior)

//  Get all the affected luns from the redundancy group and arrange them based on the
//  given lun priority and raid level policy

         For each lun in the redundancy group
         {
           //  Get rebuild priority for the lun

           If(lun_priority OR raid_level_priority)
           {
              //  Lun will be inserted into the queue based on priority

              InsertLunToPriorityQueue(lun , priorityQueue , priority);

           }
         }

// Rebuild priority is set higher than the host workload option
         //  Option to make the rebuild process high priority than the host workload

         rebuild_priority   //For high = 1 and low =0 (default)

                IF(rebuild_priority)
           {
                //  Get the resources by reducing the host workload and accelerating
                //  rebuild, i.e., throttle the host workload. The rebuild level = 0 to
                //  100 where 0 is least priority

                 ChangeLunsQueueDepthAndThrottlingLimit ( level);
          }

         For each lun in priority Queue
         {

           // Select the next highest priority lun from priority queue

           PerformRebuild(lun);
         }

}

Sub perform_rebuild(lun)
{
         While(Chunks remaining to rebuild for the lun)
         {
         //  3. Option to make the rebuild having higher priority with maximum resource
         //  when the host workload is less  rebuild_adaptive_operation   //For high = 1
         //  and low=0(default)

             If(rebuild_adaptive_operation)
             {

              //  Monitor the host work load and predict the dip time to accelerate the
              //  rebuild operation.

                   If(ExpectedHostWorkload(Chunks_rebuild_time) == LESS)
                   {
                     SetNumRebuildThreads(MAX);
                   }
                   Else
                   {
                     SetNumRebuildThreads(DEFAULT);
                   }

             }

If (another disk drive failure predicted in the same redundancy group)
             {

              //  Accelerate the rebuild and throttle the host workload

                     SetNumRebuildThreads(MAX);
                     ChangeLunsQueueDepthAndThrottlingLimit ( level);

              }


         //  If rebuild has the highest priority then send the rebuild request as SCSI HOQ
         // (Head Of Queue) to the busy drives. Perform Rebuild for data chuncks by
issuing
         // the reads doing data regeneration.  For example : For RAID 5 read the data
from
         // the remaining drives and XOR

                For each disk in set of remaining drives
                  {
                       If(disk extent is irrelevant data)
                       {
                           //  Skip the read operation
                       }
                  Else
                   {
                           //  Issue the read operation
                       }
              //  Perform XOR

                  }

         }
  }
```

[0045] Although specific examples have been illustrated and described herein, a variety of alternate and/or equivalent implementations may be substituted for the specific examples shown and described without departing from the scope of the present disclosure. This application is intended to cover any adaptations or variations of the specific examples discussed herein. Therefore, it is intended that this disclosure be limited only by the claims and the equivalents thereof.

CLAIMS

1. A method of rebuilding data stored on a failed physical disk in a storage array, comprising:

assigning a highest priority to a logical disk of a plurality of logical disks having data stored in the storage array;

selecting the logical disk assigned the highest priority to be rebuilt prior to other logical disks in the plurality of logical disks; and

dynamically assigning other priorities during rebuilding the failed disk from the logical disk assigned with the highest priority.

2. The method of claim 1 wherein selecting the other logical disk to be rebuilt based on an assigned relative priority.

3. The method of claim 1 wherein selecting includes placing the logical disk assigned the highest priority in a rebuild queue of a storage system manger.

4. The method of claim 1 wherein dynamically assigning other priorities includes detecting indicators related to anticipating failures of a working physical disk in the storage array, and accelerating the rebuilding if the working physical disk if indications that an anticipated failure are detected.

5. The method of claim 1 wherein dynamically assigning other priorities includes predicting periods of low host requests and accelerating rebuilding during periods of predicted low host requests.

6. The method of claim 5 wherein predicting periods of low host requests includes predicting periods of high host requests and decelerating rebuilding during periods of predicted high host requests.

7. The method of claim 1 wherein dynamically assigning other priorities includes increasing an amount of threads for rebuilding the failed disk to accelerate rebuilding.

8. The method of claim 7 including decreasing an amount of threads for host requests to accelerate rebuilding.

9. A computer readable storage medium storing computer executable instructions for controlling a computing device to perform a method of rebuilding data stored on a failed physical disk in a storage array, the method comprising:

assigning a highest priority to a logical disk of a plurality of logical disks having data stored in the storage array;

tracking data chunks of the logical disks having irrelevant data;

selecting the logical disk assigned the highest priority to be rebuilt prior to other logical disks in the plurality of logical disks wherein rebuilding includes not reading data from data chunks having irrelevant data; and

dynamically assigning other priorities during rebuilding the failed disk from the logical disk assigned with the highest priority.

10. The computer readable storage medium of claim 9 including writing irrelevant data to a spare disk to data chunks having irrelevant data.

11. A system, comprising: to rebuilding data stored on a failed physical disk in a storage array, comprising:

a host having a memory and a processor; and

a storage array including a plurality of physical disks configured to store data from a plurality of logical disks, wherein:

each logical disk in the plurality of logical disks having data stored on the failed physical disk is assigned a distinguishable priority level from highest priority to lowest priority; and

the plurality of logical disks are rebuilt in an order of highest priority to lowest priority; and

dynamic priorities are assigned to the logical disks during rebuilding of the failed disk from the logical disk assigned with the highest priority.

12. The system of claim 11 wherein the plurality of logical disks includes all logical disks stored in a redundancy group of the failed physical disk.

13. The system of claim 11 wherein the priority is assigned based on whether a logical disk is backed-up in another storage array.

14. The system of claim 11 wherein the priority is assigned based on types of data stored in the plurality of logical disks.

15. The system of claim 11 wherein the highest priority logical disk is completely rebuilt prior to rebuilding a next highest priority logical disk is begun.

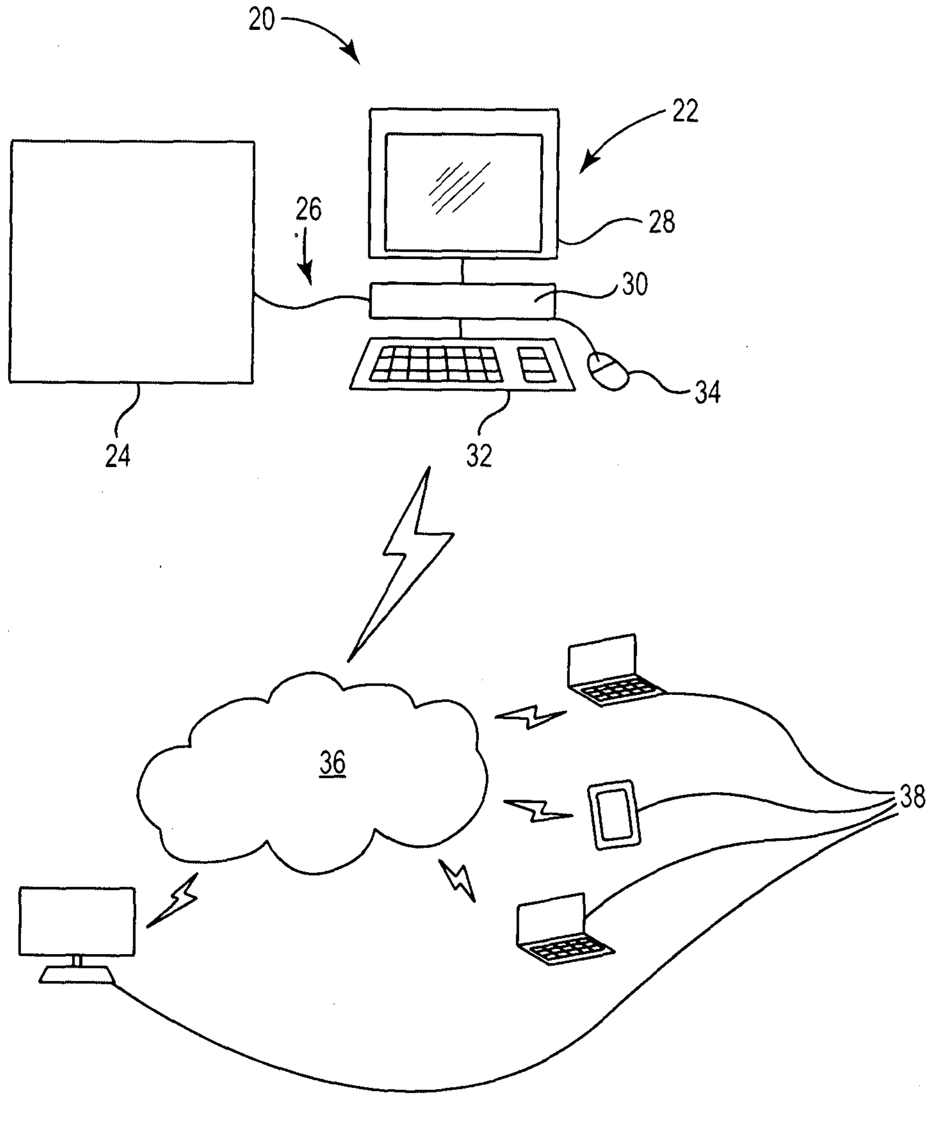


Fig. 1

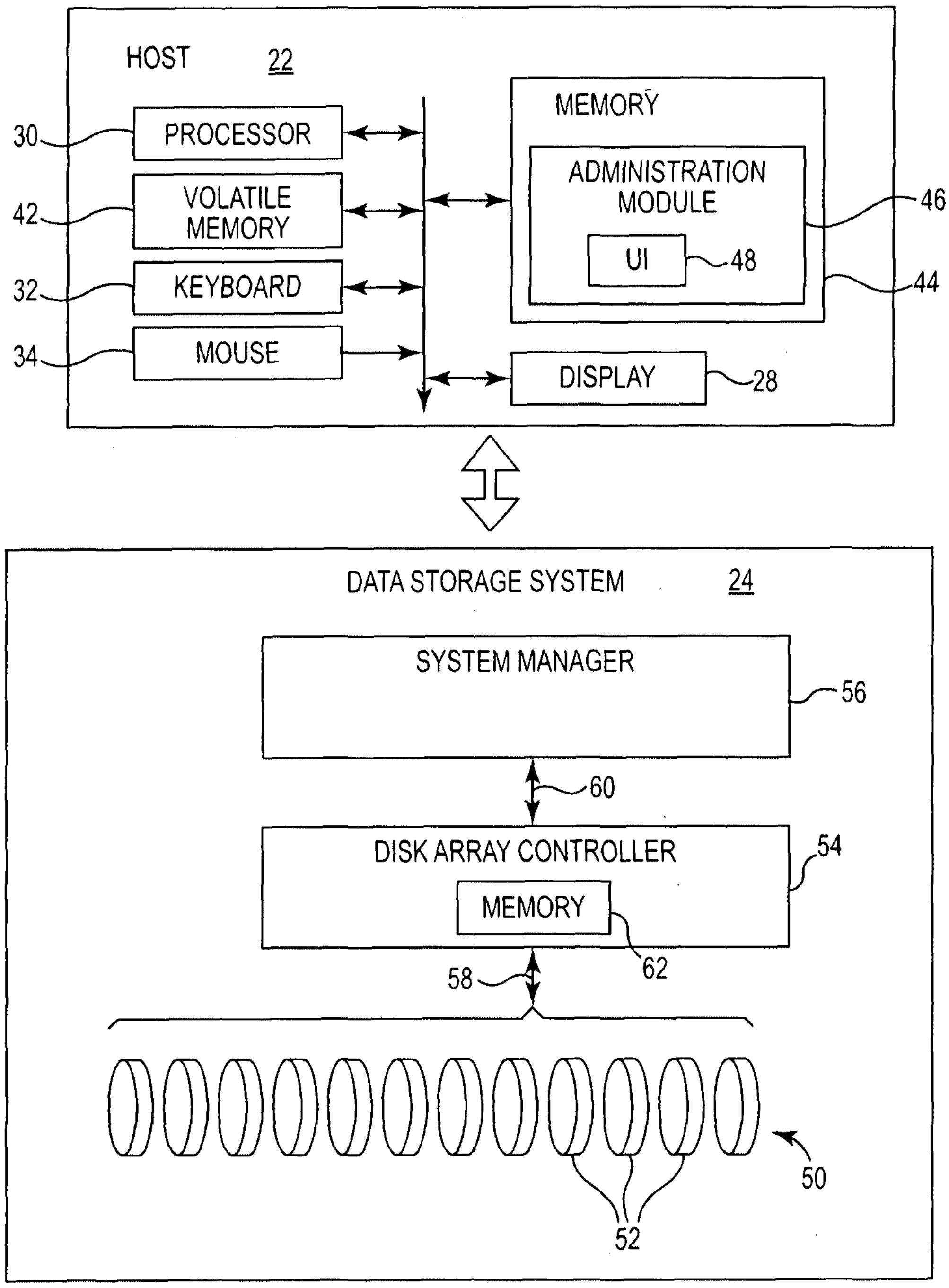

HOST 22
PROCESSOR
VOLATILE MEMORY
KEYBOARD
MOUSE
MEMORY
ADMINISTRATION MODULE
UI
DISPLAY
30
42
32
34
46
48
44
28
DATA STORAGE SYSTEM 24
SYSTEM MANAGER
56
60
DISK ARRAY CONTROLLER
MEMORY
54
62
58
50
52

Fig. 2

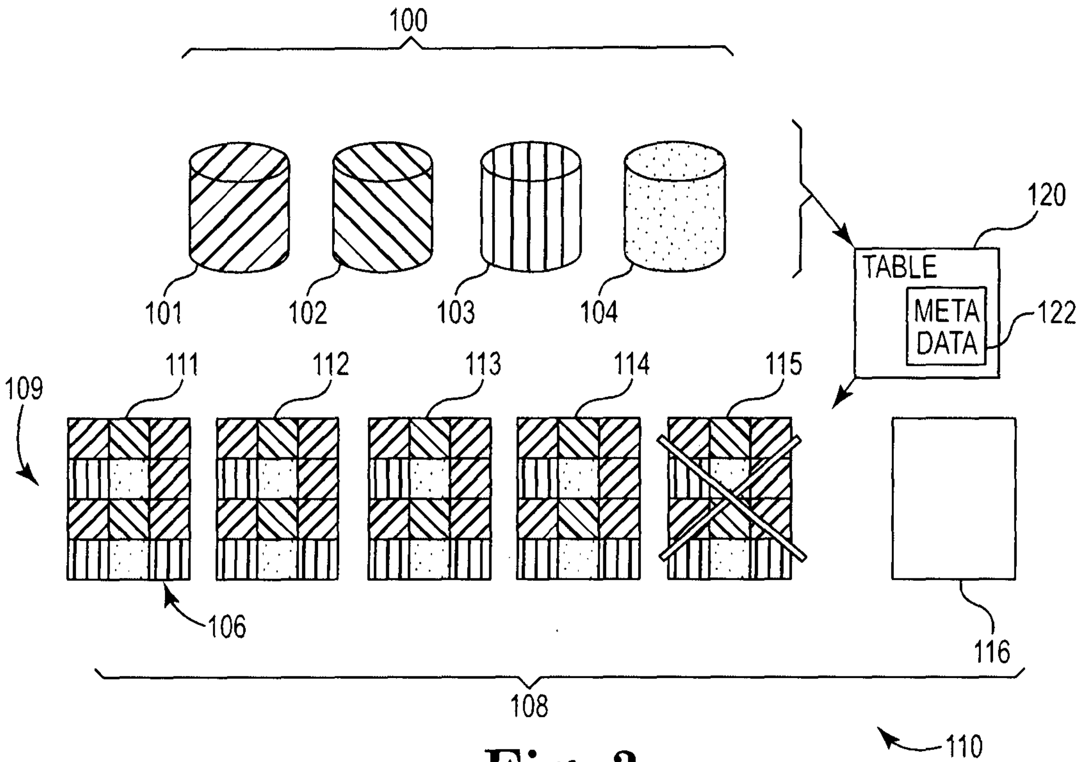


Fig. 3

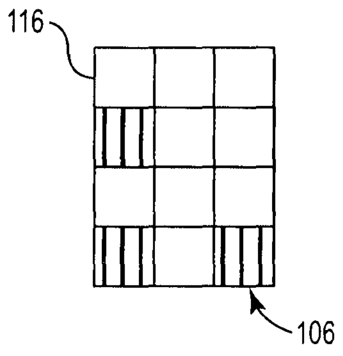


Fig. 4

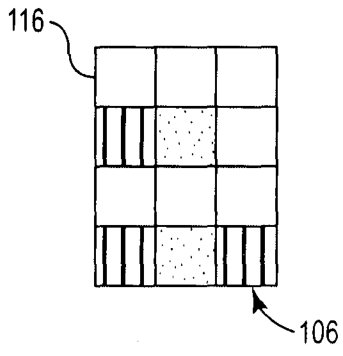


Fig. 5

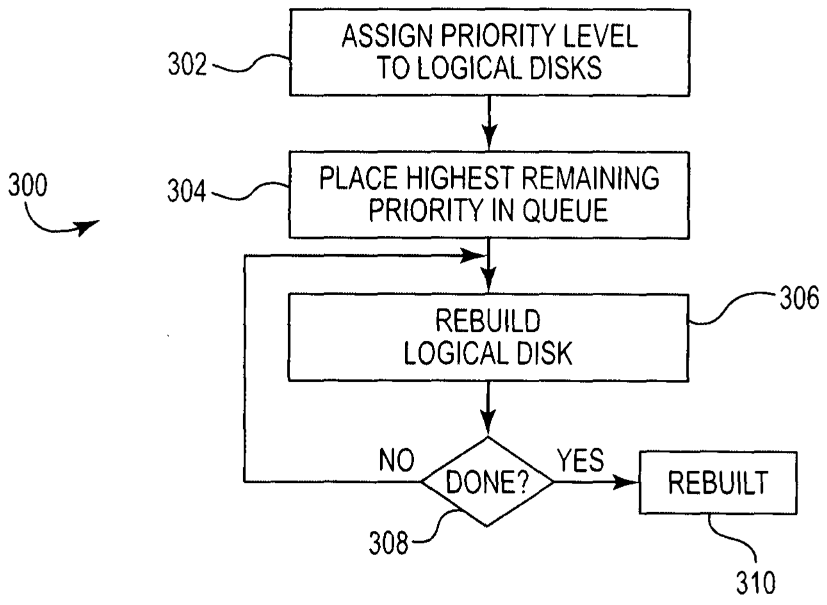
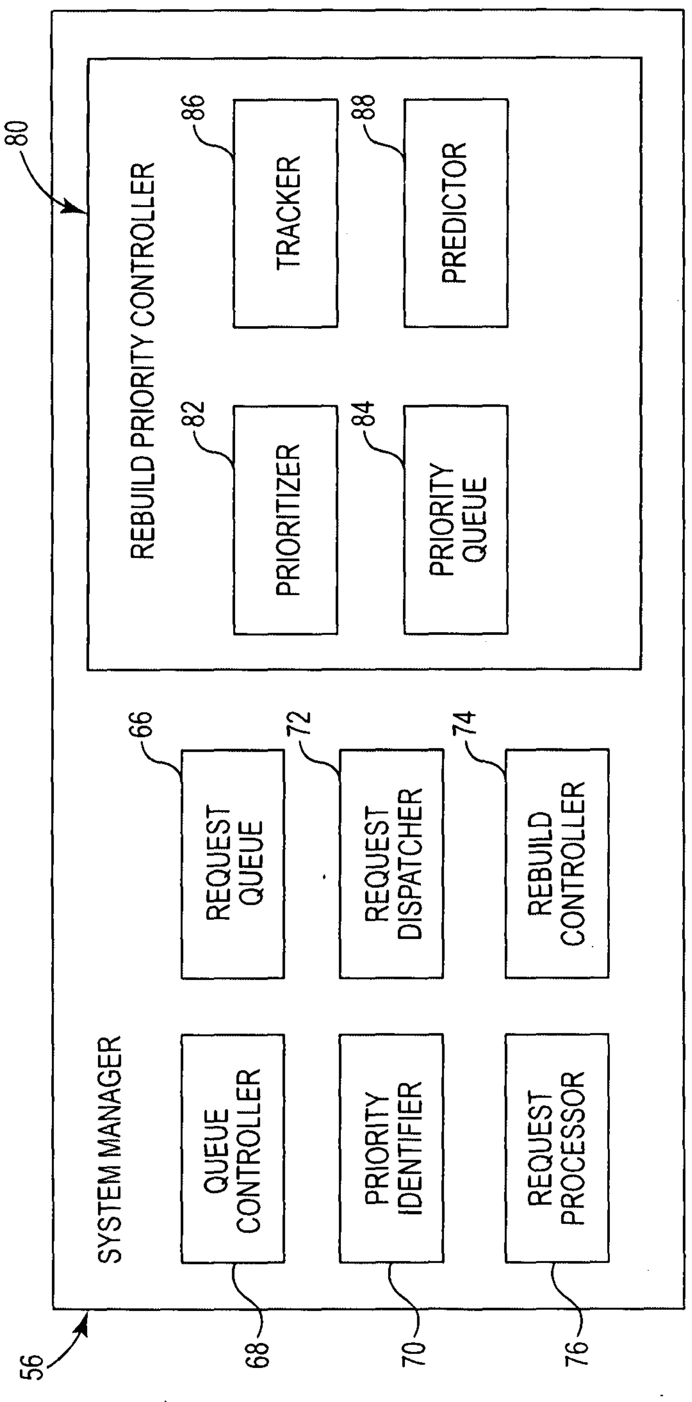

ASSIGN PRIORITY LEVEL TO LOGICAL DISKS
302
PLACE HIGHEST REMAINING PRIORITY IN QUEUE
304
300
REBUILD LOGICAL DISK
306
NO
DONE?
YES
308
REBUILT
310

Fig. 6



56
SYSTEM MANAGER
QUEUE CONTROLLER
68
PRIORITY IDENTIFIER
70
REQUEST PROCESSOR
76
REQUEST QUEUE
66
REQUEST DISPATCHER
72
REBUILD CONTROLLER
74
80
REBUILD PRIORITY CONTROLLER
PRIORITIZER
82
PRIORITY QUEUE
84
TRACKER
86
PREDICTOR
88

Fig. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IN2014/000074

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/00(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; H04L

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)

WPI, EPODOC, GOOGLE, CNPAT, CNKI, IEEE: rebuild, reconstruct, radi, array, priority, order, dynamically, logical, virtual, disk, stripe, volume, irrelevant

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6516425 B1 (HEWLETT-PACKARD CO.) 04 February 2003 (2003-02-04) description, column 1, line 65 to column 2, line 35, column 5, line 65 to column 9, line 3 and column 12, line 42 to column 14, line 19, figures 1-3, abstract and claims 18-20	1-4, 7-8, 11-15
Y	US 6516425 B1 (HEWLETT-PACKARD CO.) 04 February 2003 (2003-02-04) description, column 1, line 65 to column 2, line 35, column 5, line 65 to column 9, line 3 and column 12, line 42 to column 14, line 19, figures 1-3, abstract and claims 18-20	5-6, 9-10
Y	US 6609145 B1 (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 19 August 2003 (2003-08-19) description, column 3, lines 1-13 and abstract	5-6
Y	US 2009271659 A1 (TROPPENS, ULF ET AL.) 29 October 2009 (2009-10-29) description, paragraph [0016] and abstract	9-10
A	US 2004059958 A1 (UMBERGER, DAVID K. ET AL.) 25 March 2004 (2004-03-25) the whole document	1-15
A	US 2012084600 A1 (LSI CORPORATION) 05 April 2012 (2012-04-05) the whole document	1-15

☐ 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	"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
"E"	earlier application or patent but published on or after the international filing date	"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
"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)	"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
"O"	document referring to an oral disclosure, use, exhibition or other means	"&"	document member of the same patent family
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search **10 October 2014**	Date of mailing of the international search report **29 October 2014**
Name and mailing address of the ISA/CN **STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA(ISA/CN) 6,Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088 China** Facsimile No. **(86-10)62019451**	Authorized officer **WU,Shaohong** Telephone No. **(86-10)82245502**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IN2014/000074

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 6516425 B1	04 February 2003	JP 2001147785 A	29 May 2001
US 6609145 B1	19 August 2003	US 5822584 A	13 October 1998
US 2009271659 A1	29 October 2009	None	
US 2004059958 A1	25 March 2004	JP 2001290746 A	19 October 2001
		US 6647514 B1	11 November 2003
US 2012084600 A1	05 April 2012	None	