US 20050044310A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0044310 A1

Cameron (43) Pub. Date: Feb. 24, 2005

(54) METHOD OF MANAGING VIRTUAL VOLUMES IN A UTILITY STORAGE SERVER SYSTEM

(76) Inventor: **Douglas J. Cameron**, Woodinville, WA (US)

Correspondence Address:
PATENT LAW GROUP LLP
2635 NORTH FIRST STREET
SUITE 223
SAN JOSE, CA 95134 (US)

(21) Appl. No.: **10/959,569**

(22) Filed: **Oct. 5, 2004**

Related U.S. Application Data

(63) Continuation of application No. 10/840,956, filed on May 7, 2004, now Pat. No. 6,823,442.

(60) Provisional application No. 60/470,018, filed on May 12, 2003.

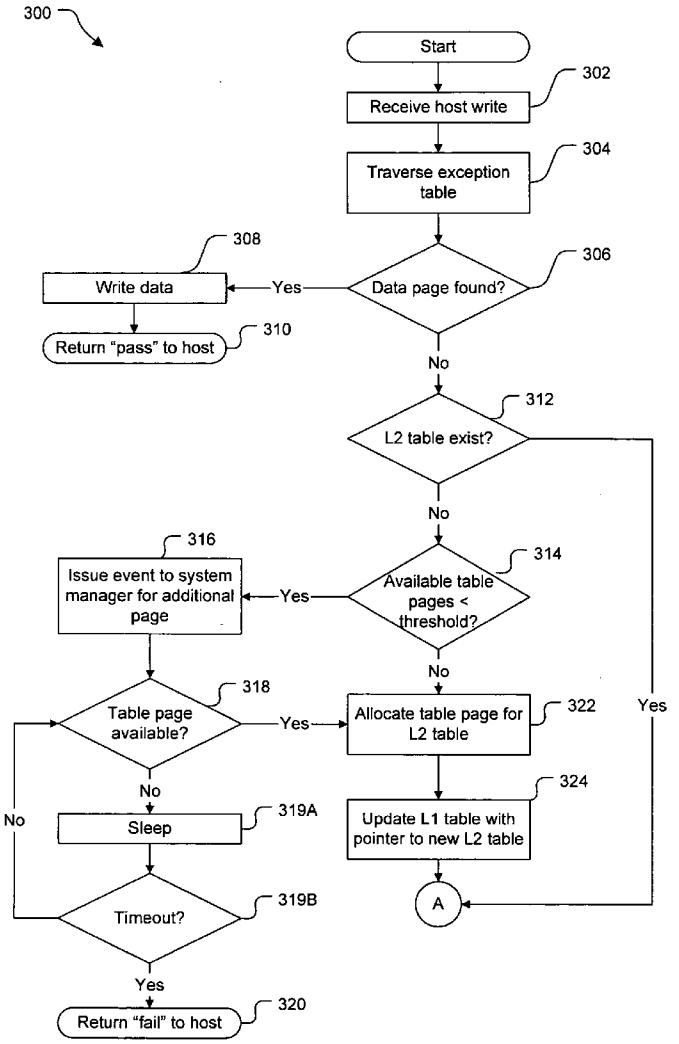
Publication Classification

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

(52) **U.S. Cl.** **711/112**; 711/203; 711/170

(57) **ABSTRACT**

A method is provided to allow a system administrator of a utility storage server to provision virtual volumes several times larger than the amount of physical storage within the storage server. A virtual volume is a virtual representation of multiple disks as a single large volume to a host or an application. In one embodiment, a virtual volume comprises an exception list containing the set of differences from dummy base volume consisting of all zeros. This exception list can be made up of address tables that map virtual volume pages to logical disk pages. As storage demand grows, additional storage is allocated for the address tables and the data pages from separate pools of storage. If any of the pools runs low, more logical disk regions are allocated to that pool.



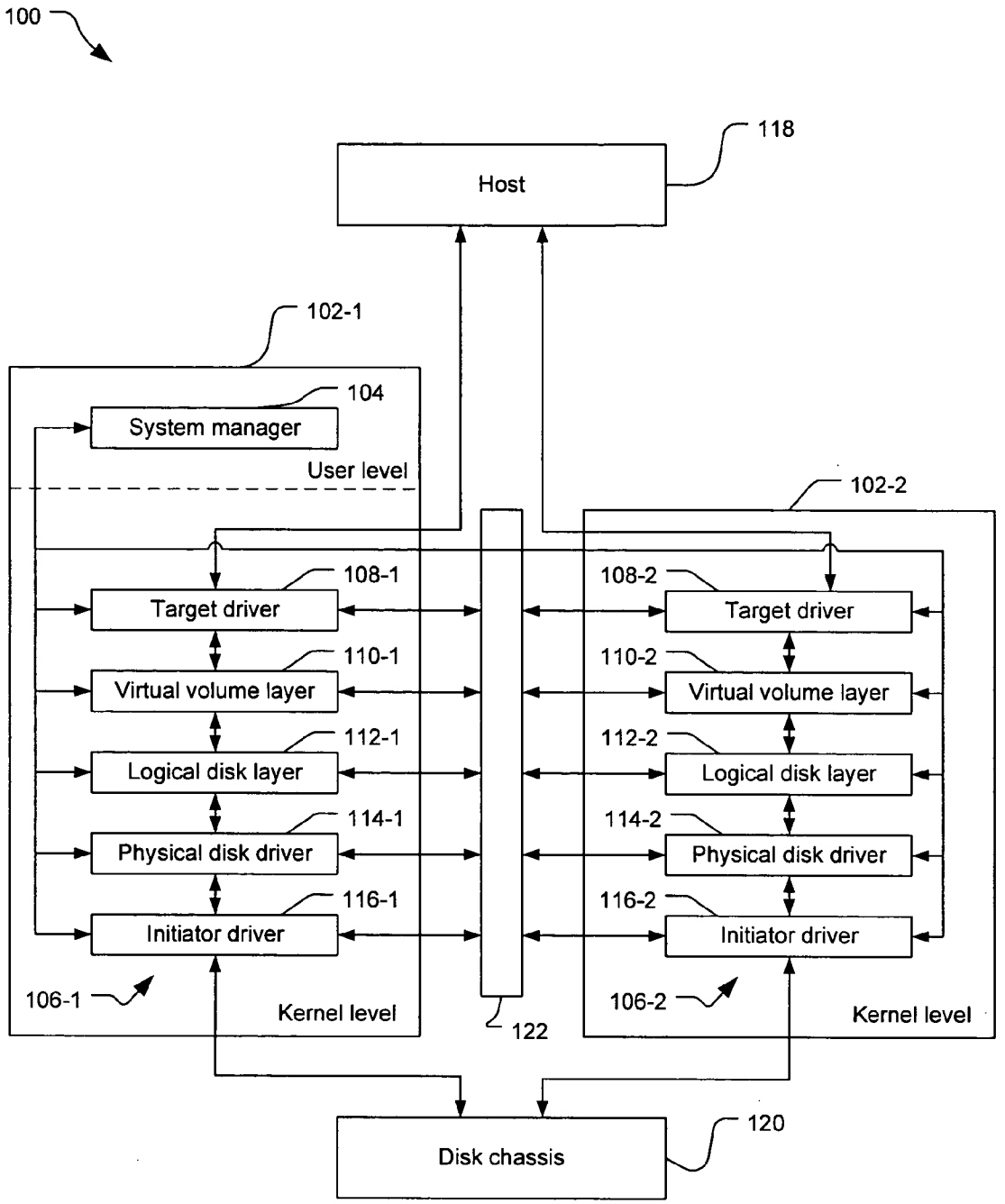

100
Host
118
102-1
104
System manager
User level
102-2
108-1
Target driver
110-1
Virtual volume layer
112-1
Logical disk layer
114-1
Physical disk driver
116-1
Initiator driver
106-1
Kernel level
108-2
Target driver
110-2
Virtual volume layer
112-2
Logical disk layer
114-2
Physical disk driver
116-2
Initiator driver
106-2
Kernel level
122
Disk chassis
120

Fig. 1

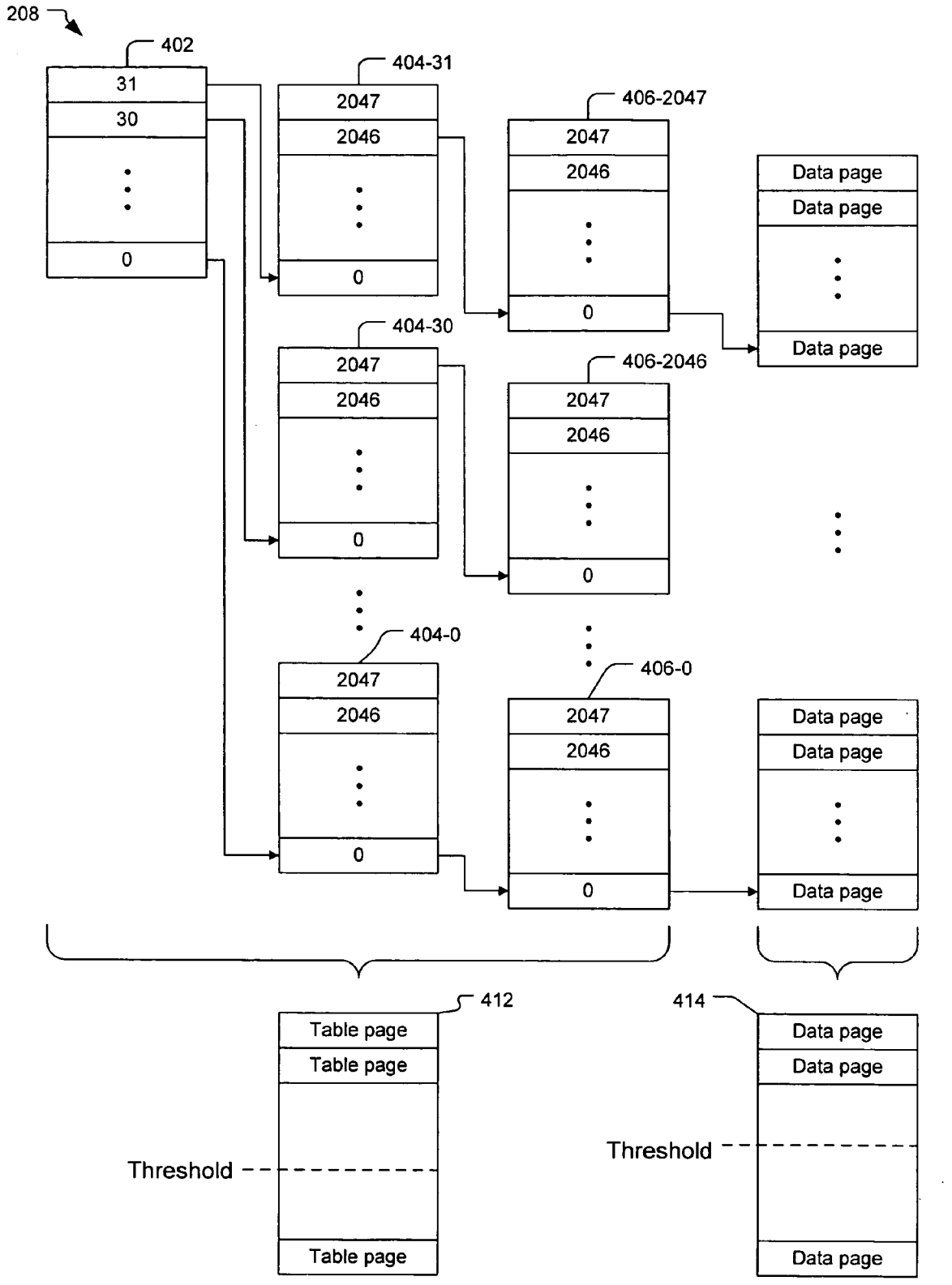

208
402
31
30
0
404-31
2047
2046
0
406-2047
2047
2046
0
Data page
Data page
Data page
404-30
2047
2046
0
406-2046
2047
2046
0
404-0
2047
2046
0
406-0
2047
2046
0
Data page
Data page
Data page
412
Table page
Table page
Threshold
Table page
414
Data page
Data page
Threshold
Data page

Fig. 2

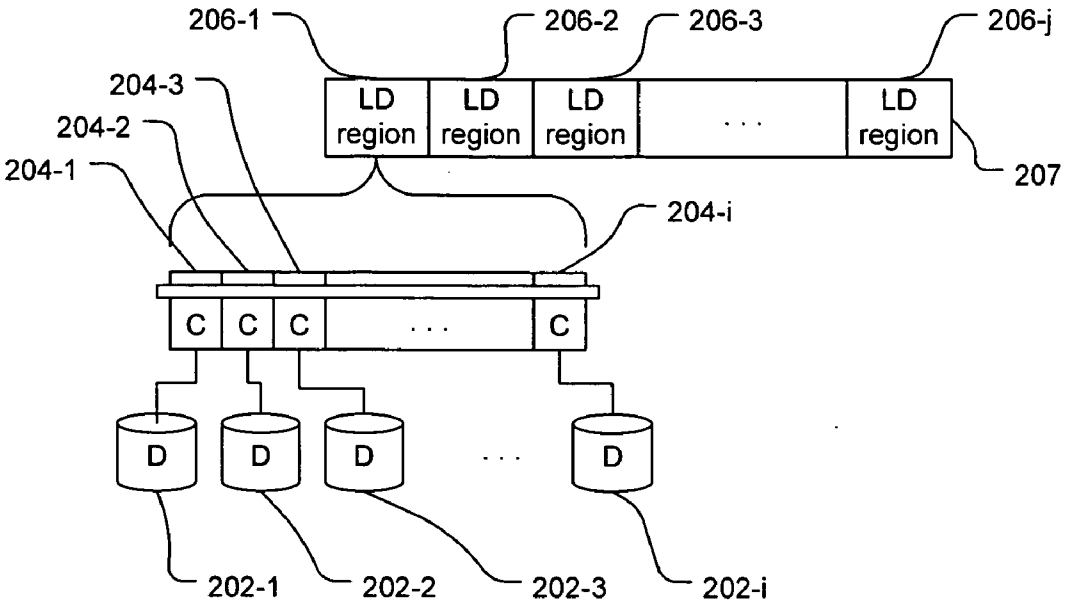

206-1
206-2
206-3
206-j
204-3
204-2
204-1
LD region
LD region
LD region
LD region
207
204-i
C
C
C
C
D
D
D
D
202-1
202-2
202-3
202-i

Fig. 3

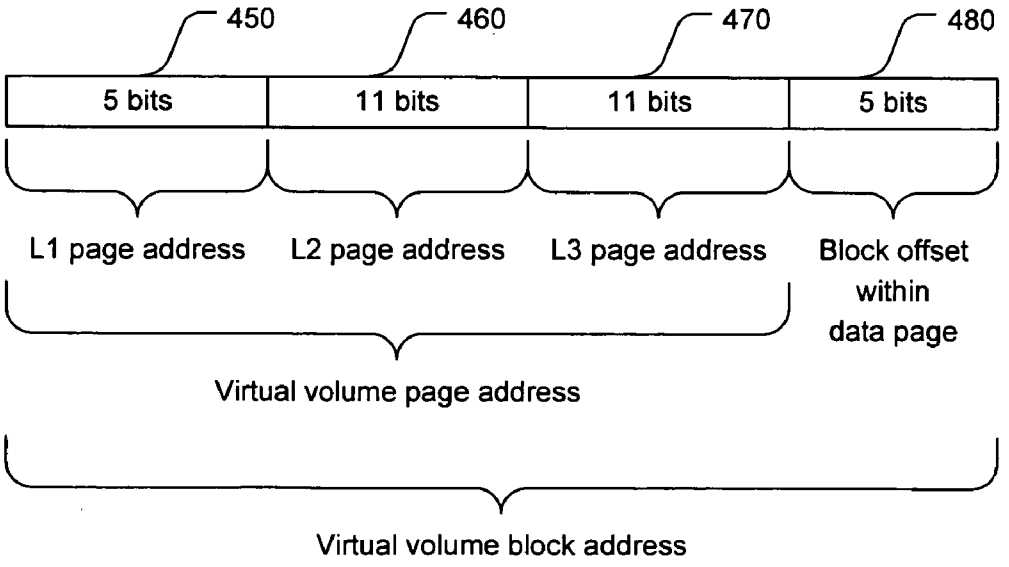

450
460
470
480
5 bits
11 bits
11 bits
5 bits
L1 page address
L2 page address
L3 page address
Block offset within data page
Virtual volume page address
Virtual volume block address

Fig. 3A

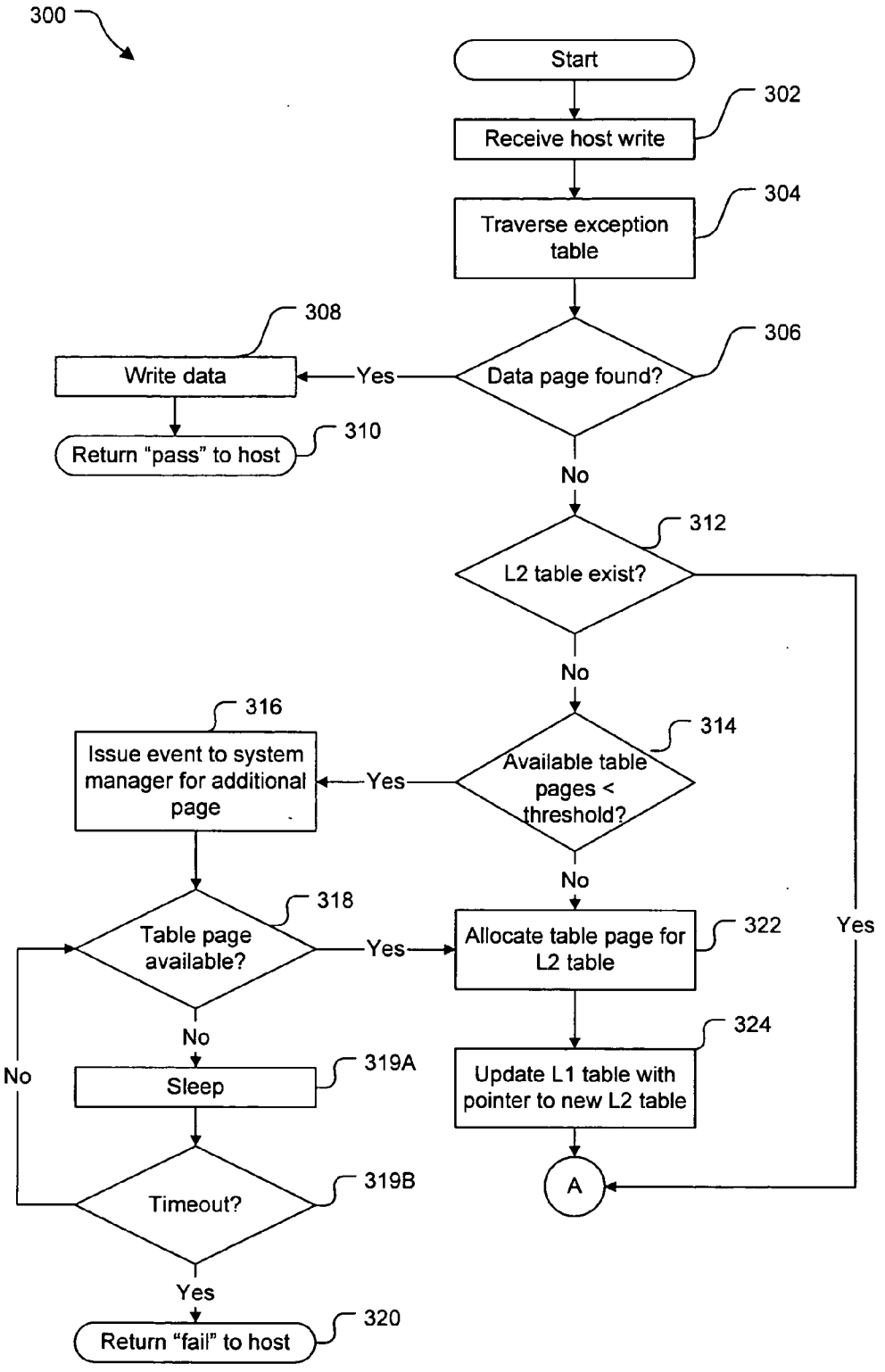

300
Start
302
Receive host write
304
Traverse exception table
306
Data page found?
Yes
308
Write data
310
Return "pass" to host
No
312
L2 table exist?
Yes
No
314
Available table pages < threshold?
Yes
316
Issue event to system manager for additional page
No
318
Table page available?
Yes
322
Allocate table page for L2 table
No
319A
Sleep
No
319B
Timeout?
Yes
320
Return "fail" to host
324
Update L1 table with pointer to new L2 table
A

Fig. 4A

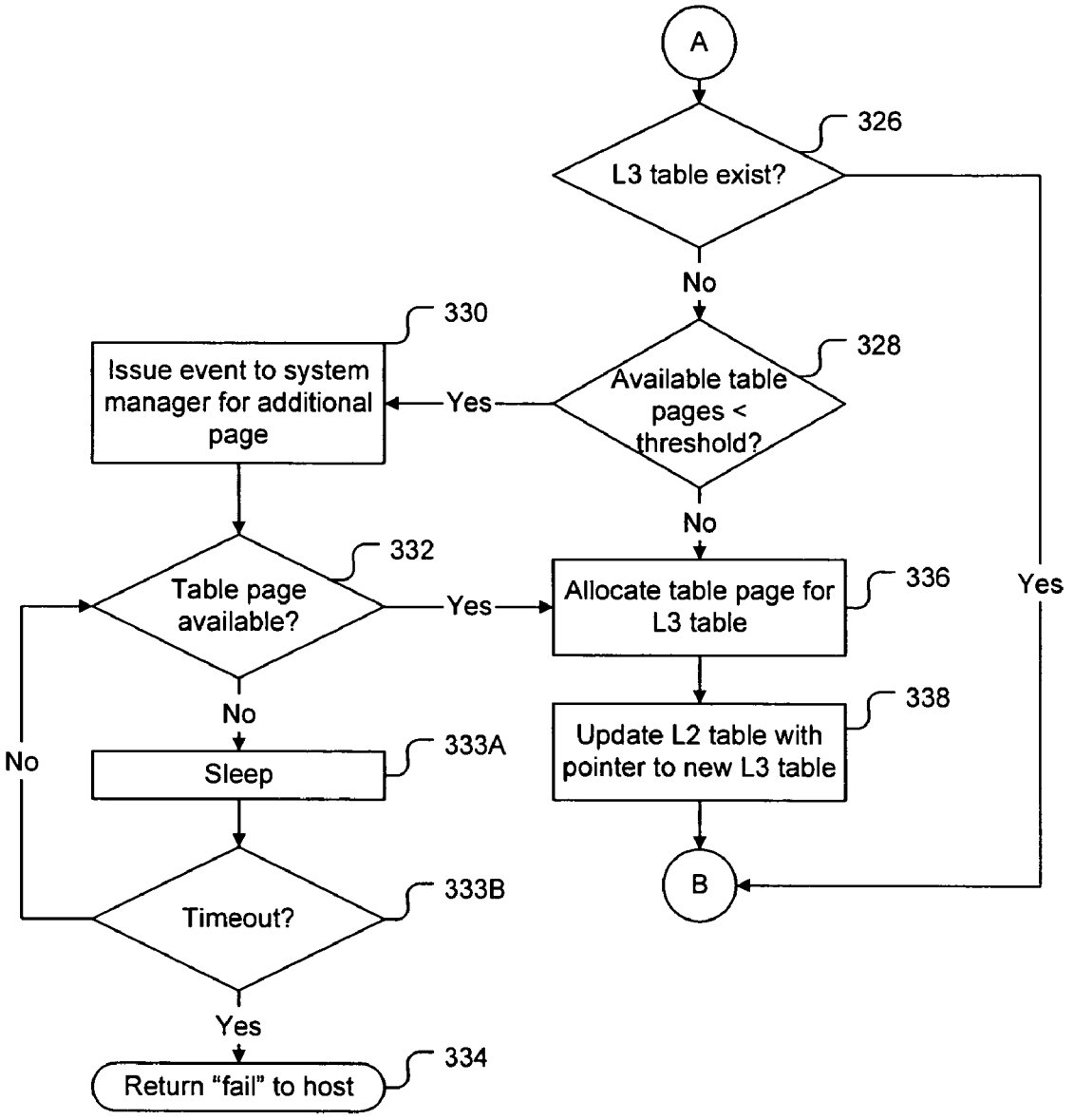

A
326
L3 table exist?
Yes
No
328
Available table pages < threshold?
Yes
330
Issue event to system manager for additional page
No
332
Table page available?
Yes
336
Allocate table page for L3 table
338
Update L2 table with pointer to new L3 table
B
No
333A
Sleep
333B
Timeout?
No
Yes
334
Return "fail" to host

Fig. 4B

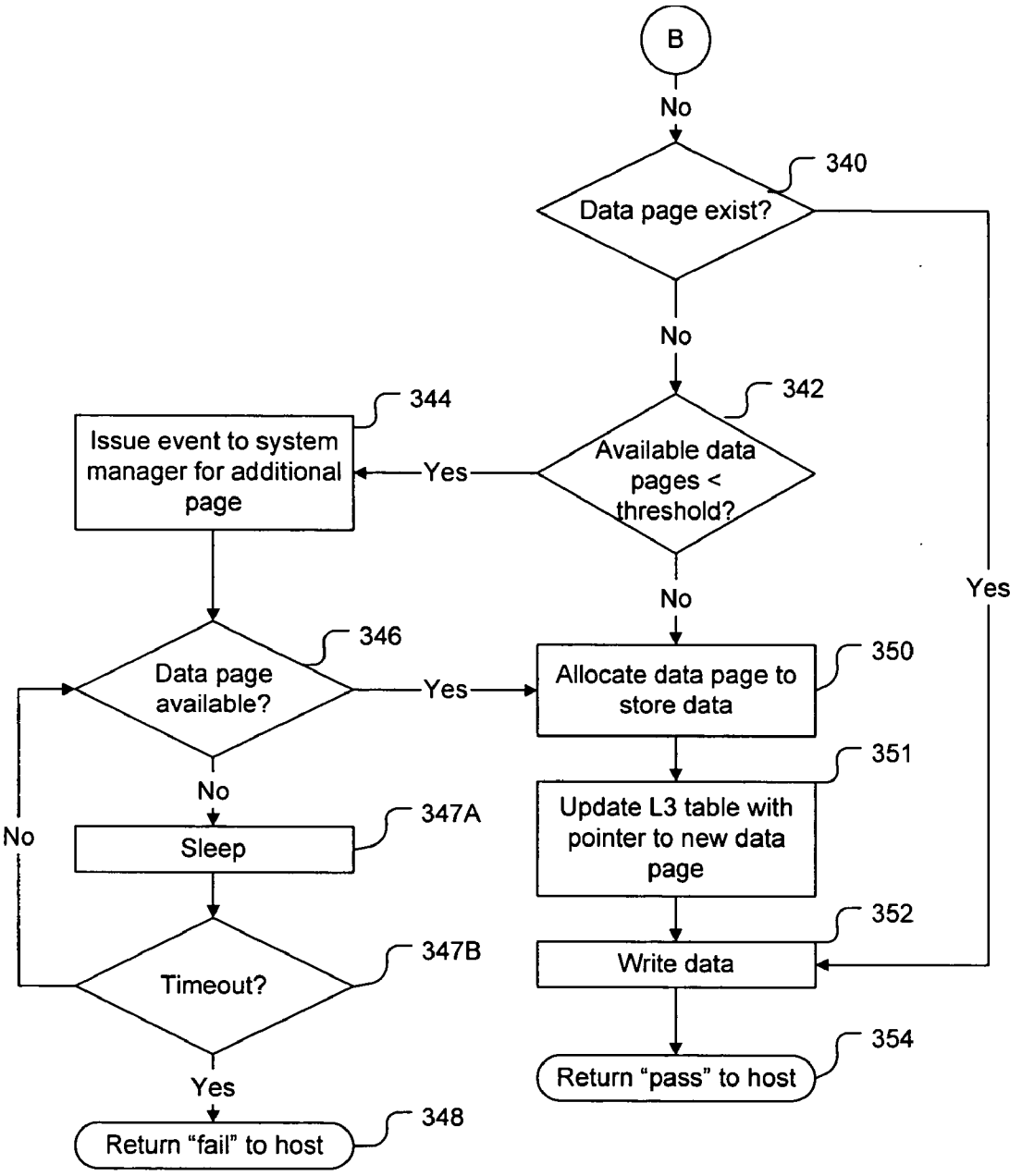

B
No
340
Data page exist?
No
342
Available data pages < threshold?
Yes
344
Issue event to system manager for additional page
No
Yes
346
Data page available?
Yes
350
Allocate data page to store data
351
Update L3 table with pointer to new data page
352
Write data
354
Return "pass" to host
No
347A
Sleep
347B
Timeout?
Yes
348
Return "fail" to host

Fig. 4C

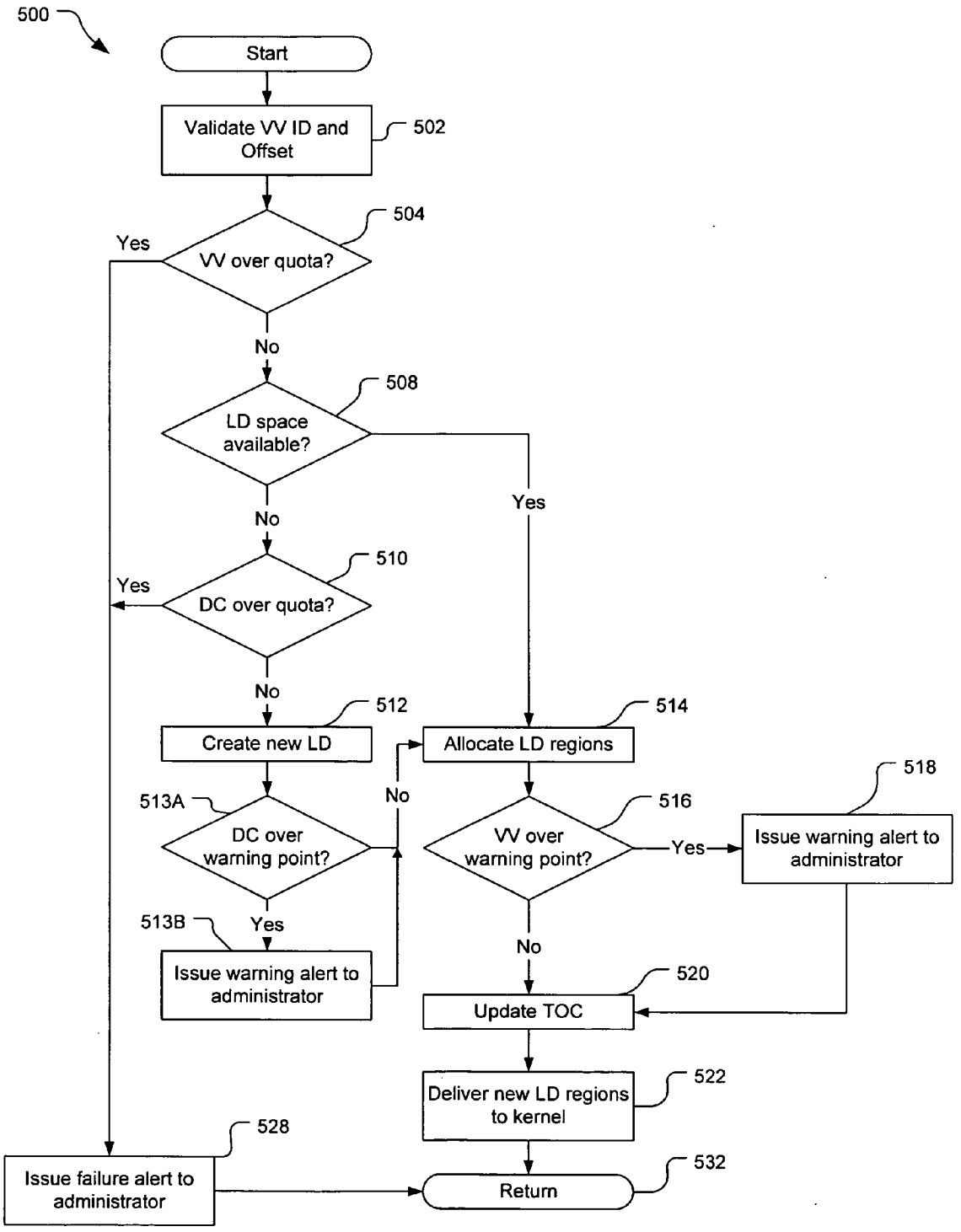

500
Start
Validate VV ID and Offset
502
VV over quota?
504
Yes
No
LD space available?
508
Yes
No
DC over quota?
510
Yes
No
Create new LD
512
Allocate LD regions
514
513A
DC over warning point?
No
513B
Yes
Issue warning alert to administrator
VV over warning point?
516
Yes
Issue warning alert to administrator
518
No
Update TOC
520
Deliver new LD regions to kernel
522
Issue failure alert to administrator
528
Return
532

Fig. 5

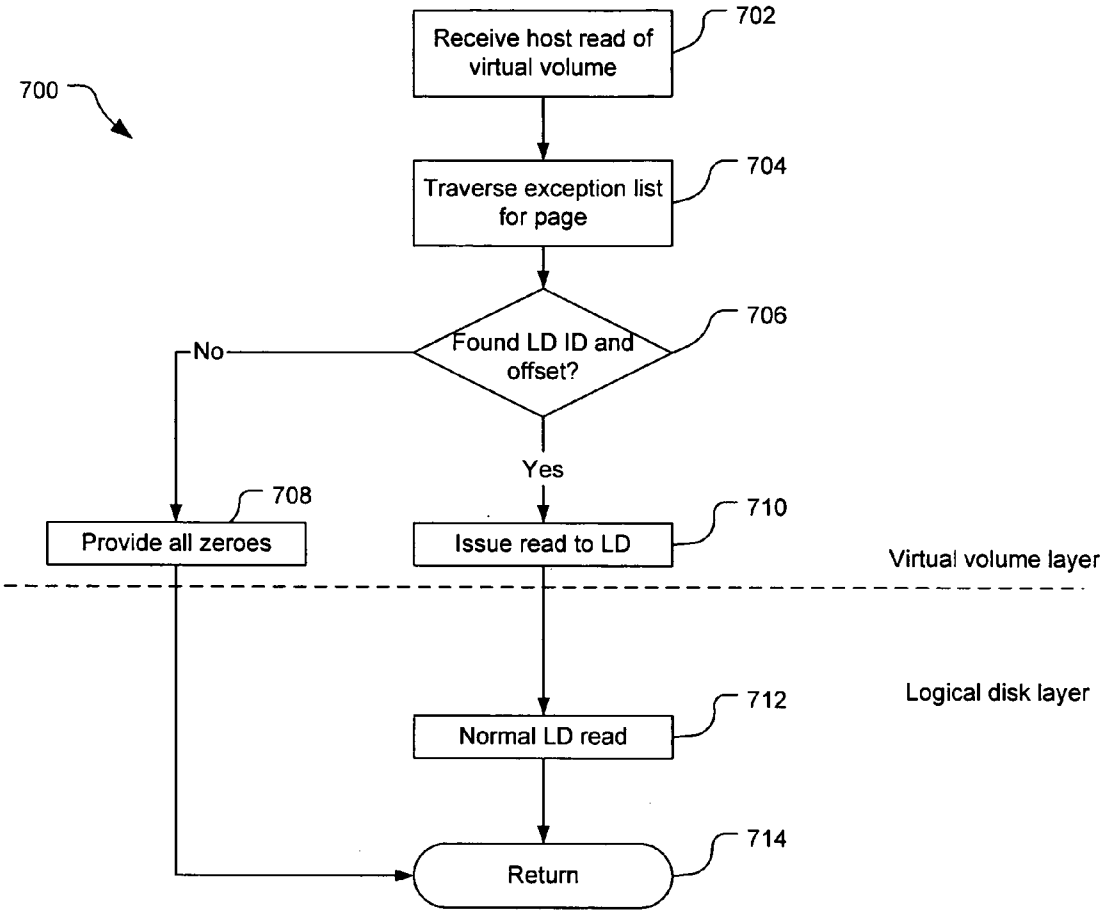

700
Receive host read of virtual volume
702
Traverse exception list for page
704
Found LD ID and offset?
706
No
Yes
Provide all zeroes
708
Issue read to LD
710
Virtual volume layer
Logical disk layer
Normal LD read
712
Return
714

Fig. 6

METHOD OF MANAGING VIRTUAL VOLUMES IN A UTILITY STORAGE SERVER SYSTEM

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This is a Continuation of U.S. patent application Ser. No. 10/840,956 filed on May 7, 2004, which claims the benefit of U.S. Provisional Application No. 60/470,018 filed on May 12, 2003, which are hereby incorporated by reference.

FIELD OF INVENTION

[0002] This invention relates to a utility storage server and more particularly to virtual volume management of the utility storage server.

DESCRIPTION OF RELATED ART

[0003] A utility storage server may be defined as any carrier-class storage system that provisions physical storage to multiple users and/or applications. To meet the demands of multiple users and applications, a system administrator has to purchase enough physical storage for the users and the applications. Often the purchased physical storage is underutilized as the users and the applications generally fill their provisioned storage over time. Thus, what is needed is a method that allows the system administrator to increase asset utilization and defer expenses spent on the physical storage.

SUMMARY

[0004] In one embodiment of the invention, a method is provided to allow a system administrator of a utility storage server to provision virtual volumes several times larger than the amount of physical storage within the storage server. A virtual volume is a virtual representation of multiple disks as a single large volume to a host or an application. In one embodiment, a virtual volume comprises an exception list containing the set of differences from a dummy base volume consisting of all zeros. This exception list can be made up of address tables that map virtual volume pages to logical disk pages. As storage demand grows, additional storage is allocated for the address tables and the data pages from separate pools of storage. If any of the pools runs low, more logical disk regions are allocated to that pool.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] **FIG. 1** illustrates a software architecture of a utility storage server in one embodiment of the invention.

[0006] **FIG. 2** illustrates a representation of the mapping of pages in a virtual volume pages to pages in a logical disk of a node in one embodiment.

[0007] **FIG. 3** illustrates the mapping of logical disk regions to physical disks in one embodiment of the invention.

[0008] **FIG. 3A** illustrates a virtual volume block address and its component segments in one embodiment of the invention.

[0009] **FIGS. 4A, 4B**, and 4C illustrate a method for a virtual volume layer to respond to a write request to the virtual volume in one embodiment of the invention.

[0010] **FIG. 5** illustrates a method for a system manager to respond to an event requesting an additional logical disk region from the virtual volume layer in one embodiment of the invention.

[0011] **FIG. 6** illustrates a method for the virtual volume layer and the logical disk layer to respond to a read request from the host in one embodiment.

DETAILED DESCRIPTION

[0012] For a description of a utility storage server, please see U.S. Pat. No. 6,658,478, entitled "Data Storage System," attorney docket no. M-8494 US, and U.S. patent application Ser. No. 09/883,681, entitled "Node Controller for a Data Storage System," attorney docket no. M-8496 US, which are incorporated by reference in their entirety.

[0013] **FIG. 1** illustrates a software architecture of a utility storage server **100** in one embodiment. For simplicity, utility storage server **100** is shown to include a cluster of nodes **102-1** and **102-2** although the cluster may include additional nodes.

[0014] Node **102-1** includes a system manager **104** residing in the user level above the operating system and a data stack **106-1** residing in the kernel level of the operating system. Data stack **106-1** includes a target driver **108-1**, a virtual volume layer **110-1**, a logical disk layer **112-1**, a physical disk layer **114-1**, and an initiator driver **116-1**.

[0015] Initiator driver **116-2** performs the actual reads and writes to the physical disk drive using, e.g., the SCSI protocol. Physical disk layer **114-1** routes the read and write requests to the appropriate node with access to the physical disk drives on disk chassis **120**.

[0016] Logical disk layer **112-1** organizes "chunklets" of physical disks **202-1** to **202-*i*** (**FIG. 3**) into logical disks (LDs) of specific RAID levels. In one embodiment, a chunklet is 256 megabytes of contiguous disk space. System manager **104** allocates logical disk storage space, in units of logical disk regions (shown as a stripe through chunklets **204-1** to **204-*i*** from physical disks **202-1** to **202-*i*** in **FIG. 3**), to virtual volume layer **110-1**. In one embodiment, an LD region is 256 megabytes of logical disk storage space. Virtual volume layer **110-1** divides up each LD region into pages for storing data. In one embodiment, a page has a size of 16 kilobytes and holds thirty-two 512 byte data blocks. Virtual volume layer **110-1** maps pages in a virtual volume to pages in the logical disk and sends the read/write requests to the proper logical disk addresses within logical disk layer **112-1**.

[0017] Target driver **108-1** communicates the read and write requests to virtual volume layer **110-1**. A host **118** sends read and write requests to the virtual volume via target driver **108-1** using, e.g., the SCSI protocol.

[0018] Similarly, node **102-2** includes a data stack **106-2** residing in the kernel level. Data stack **106-2** also includes a target driver **108-2**, a virtual volume layer **110-2**, a logical disk layer **112-2**, a physical disk layer **114-2**, and an initiator driver **116-2**. Components of data stacks **106-1** and **106-2** communicate by a node-to-node link **122**.

[0019] System manager **104** resides only on one of the nodes of utility storage server **100**. If system manager **104** fails at one of the nodes, it can be restarted at another node.

System manager **104** keeps a single system image of utility storage server **100**. System manager **104** also services events from the data stacks, delivers configuration commands to the data stacks, and records system configuration information in a table of contents (TOC) on a physical disk drive.

[**0020**] **FIG. 2** illustrates the mapping of a virtual volume **208** to a logical disk (e.g., logical disk **207** in **FIG. 3**) in one embodiment of the invention. Virtual volume **208** is an exception list made up of address tables that map pages of a virtual volume **208** to pages of logical disk **207** storage. Virtual volume **208** also includes pools **412** and **414** of logical disk storage from which these exception list tables and logical disk data pages are allocated. In one embodiment, virtual volume **208** is implemented using a snapshot mechanism in which the exception list tracks changes to a dummy base volume consisting of only zeroes.

[**0021**] The address tables are divided into three levels. This is because virtual volume **208** is written or read in blocks each identified by a virtual volume block address. The virtual volume block address includes a virtual volume page address consisting of parts **450**, **460**, and **470** (**FIG. 3A**). Parts **450**, **460**, and **470** lead to a page in virtual volume **208**, which is mapped to a corresponding page in logical disk **207**. The virtual volume block address further includes a block offset **480** (**FIG. 3A**) that leads to a data block in the corresponding page in logical disk **207**.

[**0022**] A level **1** table (e.g., table **402**) consists of entries that can be indexed by the first part **450** of the page address. Specifically, part **450** provides an offset from the start of the level **1** table. Each entry in the level **1** table stores a pointer to the start of a level **2** table that shares the first part of the page address.

[**0023**] Each of the level **2** tables (e.g., table **404-0** to **404-31**) consists of entries that can be indexed by the second part **460** of the page addresses. Specifically, part **460** provides an offset from the start of a level **2** table. Each entry in the level **2** table stores a pointer to the start of a level **3** table that shares the first and the second part of the page address.

[**0024**] Each of the level **3** tables (e.g., tables **406-0** to **406-2047** in one embodiment) consists of entries that can be indexed by the third part **470** of the page addresses. Specifically, part **470** provides an offset from the start of a level **3** table. Each entry in the level **3** table stores a pointer to a page in a logical disk. Accordingly, a page in the virtual volume (e.g., VV data page) is mapped to a page in a logical disk (e.g., LD data page). Part **480** of the page address identifies an offset of a data block (i.e., block offset) from the start of the LD data page.

[**0025**] Virtual volume layer **110-1** initially creates virtual volume **208** with only a blank level **1** table. As data is written to virtual volume **208** (described later), virtual volume layer **110-1** allocates LD data pages and adds the level **2** and level **3** tables that are necessary to manage these LD data pages.

[**0026**] **FIGS. 4A, 4B**, and 4C illustrate a method **300** for virtual volume layer **110-1** to respond to a write request from host **118** (or an application) to virtual volume **208** in one embodiment.

[**0027**] In step **302**, virtual volume layer **110-1** receives a write request to a data block in virtual volume **208**. The write request identifies the data block by the ID of virtual volume **208** and its virtual volume block address. Step **302** is followed by step **304**.

[**0028**] In step **304**, virtual volume layer **110-1** traverses tables **402**, **404**, and **406** to find the LD data page that corresponds to the virtual volume block address. Step **304** is followed by step **306**.

[**0029**] In step **306**, virtual volume layer **110-1** determines if the corresponding LD data page has been found. If so, step **306** is followed by step **308**. Otherwise step **306** is followed by step **312**. The corresponding LD data page cannot be found if the VV data page to-be-written in virtual volume **208** has not been mapped to an LD data page by pointers and address tables as described above.

[**0030**] In step **308**, virtual volume layer **110-1** instructs logical disk layer **112-1** to issue a write to the corresponding LD data page. Specifically, virtual volume **110-1** identifies the block by a logical disk ID and an offset from the start of the logical disk. The offset from the start of the logical disk is determined from the sum of (1) the offset of the LD region from the start of the logical disk, (2) the offset of the LD data page from the start of the LD region, and (3) the block offset of the data block from the start of the LD data page. In one embodiment, the write is replicated to the other nodes (e.g., node **102-2**) for failover protection. Step **308** is followed by step **310**.

[**0031**] In step **310**, virtual volume layer **110-1** returns a "pass" status to the host and ends method **300**.

[**0032**] In step **312**, virtual volume layer **110-1** determines if a level **2** (L2) address table exists for the requested VV data page. If so, then step **312** is followed by step **326**. Otherwise step **312** is followed by step **314**.

[**0033**] In step **314**, virtual volume layer **110-1** determines if the number of available LD table pages in a pool **412** (**FIG. 2**) for storing address tables is less than a threshold. If so, then step **314** is followed by step **316**. Otherwise step **314** is followed by step **322**.

[**0034**] In step **316**, virtual volume layer **110-1** issues an event to system manager **104**. In response to the event, system manager **104** may allocate an available LD region to virtual volume layer **110-1**. System manager **104** may allocate the LD region in a method **500** described later in reference to **FIG. 5**. Virtual volume layer **110-1** divides the allocated LD region into LD table pages and increments pool **412** with these new pages. Step **316** is followed by step **318**.

[**0035**] In step **318**, virtual volume layer **110-1** determines if an LD table page is available to be used as an L2 table. If so, then step **318** is followed by step **322**. Otherwise step **318** is followed by step **319**A.

[**0036**] In step **319**A, virtual volume layer **110-1** sleeps for a predetermined amount of time. Step **319**A is followed by step **319**B.

[**0037**] In step **319**B, virtual volume **110-1** determines if a timeout has been reached. If so, then step **319**B is followed by step **320**. Otherwise step **319**B is followed by step **318**.

[**0038**] In step **320**, virtual volume **110-1** returns a "fail" status to the host and ends method **300**.

[0039] In step **322**, virtual volume **110-1** creates an L2 table from an available LD table page in pool **412**. Step **322** is followed by step **324**.

[0040] In step **324**, virtual volume **110-1** updates the level one (L1) table with a pointer to the newly created L2 table. Specifically, virtual volume **110-1** writes the pointer in the L1 table entry having an offset identified by the first part of the virtual volume page address. Step **324** is followed by step **326**.

[0041] In step **326** (**FIG. 4B**), virtual volume layer **110-1** determines if a level **3** (L3) table exists for the requested VV data page. If so, then step **326** is followed by step **340**. Otherwise step **326** is followed by step **328**.

[0042] In step **328**, virtual volume layer **110-1** determines if the number of available LD table pages in pool **412** (**FIG. 2**) for storing address tables is less than a threshold. If so, then step **328** is followed by step **330**. Otherwise step **328** is followed by step **336**.

[0043] In step **330**, virtual volume layer **110-1** issues an event to system manager **104**. In response to the event, system manager **104** may allocate an available LD region to virtual volume layer **110-1**. System manager **104** may allocate the LD region in a method **500** described later in reference to **FIG. 5**. Virtual volume layer **110-1** divides the LD region into LD table pages and increments pool **412** with these new pages. Step **330** is followed by step **332**.

[0044] In step **332**, virtual volume layer **110-1** determines if an LD table page is available to be used as an L3 table. If so, then step **332** is followed by step **336**. Otherwise step **332** is followed by step **333**A.

[0045] In step **333**A, virtual volume layer **110-1** sleeps for a predetermined amount of time. Step **333**A is followed by step **333**B.

[0046] In step **333**B, virtual volume **110-1** determines if a timeout has been reached. If so, then step **333**B is followed by step **334**. Otherwise step **333**B is followed by step **332**.

[0047] In step **334**, virtual volume **110-1** returns a "fail" status to the host and ends method **300**.

[0048] In step **336**, virtual volume **110-1** creates an L3 table from an available page in pool **412**. Step **336** is followed by step **338**.

[0049] In step **338**, virtual volume **110-1** updates the corresponding L2 table with a pointer to the newly created L3 table. Specifically, virtual volume **110-1** writes the pointer in the L2 table entry having an offset identified by the second part of the virtual volume page address. Step **338** is followed by step **340**.

[0050] In step **340** (**FIG. 4C**), virtual volume layer **110-1** determines if an LD data page in the logical disk exists for the virtual volume block address received. If so, then step **340** is followed by step **352**. Otherwise step **340** is followed by step **342**.

[0051] In step **342**, virtual volume layer **110-1** determines if the number of available LD data pages in a pool **414** (**FIG. 2**) for storing data is less than a threshold. If so, then step **342** is followed by step **344**. Otherwise step **342** is followed by step **350**.

[0052] In step **344**, virtual volume layer **110-1** issues an event to system manager **104**. In response to the event, system manager **104** may allocate an available LD region to pool **414**. System manager **104** may allocate the LD region in method **500** described later in reference to **FIG. 5**. Virtual volume layer **110-1** divides the LD region into LD data pages and increments pool **414** with these new pages. Step **344** is followed by step **346**.

[0053] In step **346**, virtual volume layer **110-1** determines if an LD data page is available to be used. If so, then step **346** is followed by step **350**. Otherwise step **346** is followed by step **347**A.

[0054] In step **347**A, virtual volume layer **110-1** sleeps for a predetermined amount of time. Step **347**A is followed by step **347**B.

[0055] In step **347**B, virtual volume layer **110-1** determines if a timeout has been reached. If so, then step **347**B is followed by step **348**. Otherwise step **347**B is followed by step **346**.

[0056] In step **348**, virtual volume layer **110-1** returns a "fail" status to the host and ends method **300**.

[0057] In step **350**, virtual volume layer **110-1** allocates an LD data page to virtual volume **208** from pool **414** to store data. Step **350** is followed by step **351**.

[0058] In step **351**, virtual volume layer **110-1** updates the corresponding L3 table with a pointer to the new LD data page in virtual volume **208**. Specifically, virtual volume **110-1** writes the pointer in the L3 table entry having an offset identified by the third part of the virtual volume page address. Step **351** is followed by step **352**.

[0059] In step **352**, virtual volume layer **110-1** writes the data into the new LD data page at an offset identified by the block offset of the virtual volume block address. Specifically, virtual volume layer **110-1** identifies the block by a logical disk ID and an offset from the start of the logical disk. The offset from the start of the logical disk is determined from the sum of (1) the offset of the LD region from the starting address of the logical disk, (2) the offset of the LD data page from the starting address of the LD region, and (3) the block offset of the data block from the starting address of the LD data page. Step **352** is followed by step **354**.

[0060] In step **354**, virtual volume **110-1** returns a "pass" status to the host and ends method **300**.

[0061] **FIG. 5** illustrates a method **500** for system manager **104** to respond to the event from a virtual volume layer (e.g., virtual volume layer **110-1**) in one embodiment.

[0062] In step **502**, system manager **104** validates the virtual volume ID and retrieves a data allocation control structure (DC) for the virtual volume identified by the virtual volume ID. DC, also known common provisioning group (CPG), is a part of system manager **104** that sets the maximum physical allocation for each virtual volume, the maximum physical allocation of all the virtual volumes owned (i.e., controlled) by the DC, and warning points for the physical allocation of each virtual volume and the DC itself. DC also sets the RAID characteristics of the logical disks and the set of nodes in a cluster from which the physical disk drive chunklets are allocated when creating additional logical disks.

[0063] In step **504**, system manager **104** determines if the physical allocation of the identified virtual volume (e.g., virtual volume **208**) is over the maximum physical allocation specified by the DC. The maximum physical allocation can have a default value or be set by the user. If the size of the virtual volume is over the maximum physical allocation, step **504** is followed by step **528**. If not, step **504** is followed by step **508**.

[0064] In step **508**, system manager **104** determines if one or more LD regions are available in existing logical disks (e.g., logical disk **207**). If one or more LD regions are available, step **508** is followed by step **514**. If not, step **508** is followed by step **510**.

[0065] In step **510**, system manager **104** determines if the total physical allocation of the virtual volumes owned by the DC is over the maximum physical allocation specified by the DC. If the size of the DC is over the maximum physical allocation, step **510** is followed by step **528**. If not, step **510** is followed by step **512**.

[0066] In step **512**, system manager **104** instructs logical disk layer **112-1** to create one or more new logical disks from chunklets in the physical disks. Step **512** is followed by step **513**A.

[0067] In step **513**A, system manager **104** determines if the size of the DC is over a warning point specified by the DC. The warning point provides an early warning to the user that the physical limit is approaching. The warning point can have a default value or be set by the user. If the size of the DC is over the warning point, step **513**A is followed by step **513**B. If not, step **513**A is followed by step **514**.

[0068] In step **513**B, system manager **104** issues an allocation warning alert to the administrator of system **100**. Step **513**B is followed by step **514**.

[0069] In step **514**, system manager **104** allocates (e.g., assigns) one or more available LD regions in the logical disk (whether existing or newly created) to be mapped to virtual volume pages. Each LD region is identified by a logical disk ID and an offset from the start of the logical disk. Step **514** is followed by step **516**.

[0070] In step **516**, system manager **104** determines if the physical allocation of the virtual volume is over a warning point specified by the DC. The warning point provides an early warning to the user that the physical limit is approaching. The warning point can have a default value or be set by the user. If the size of the virtual volume is over the warning point, step **516** is followed by step **518**. If not, step **516** is followed by step **520**.

[0071] In step **518**, system manager **104** issues an allocation warning alert to the administrator of system **100**. Step **518** is followed by step **520**.

[0072] In step **520**, system manager **104** updates the table of contents (TOC). TOC stores the organization of the virtual volumes, the logical disks, and the chunklets of server **100** on one or more physical disk drives. Step **520** is followed by step **522**.

[0073] In step **522**, system manager **104** delivers the one or more LD regions to virtual volume layer **110-1**. As described above, virtual volume layer **110-1** divides the one or more LD regions into LD table or data pages and increments pool **412** or **414** with these new pages. Step **522** is followed by step **532**.

[0074] In step **528**, system manager **104** issues an allocation failure alert to the administrator of system **100**. Step **528** is followed by step **532**.

[0075] In step **532**, system manager **104** ends method **500**.

[0076] **FIG. 6** illustrates a method **700** for a virtual volume layer (e.g., virtual volume layer **110-1**) and a LD layer (e.g., logical disk layer **112-1**) to respond to a read request from host **118** in one embodiment.

[0077] In step **702**, virtual volume layer **110-1** receives from host **118** a read request of a data block in a virtual volume (e.g., virtual volume **208**). The read request identifies the VV data block by a virtual volume ID and a virtual volume block address.

[0078] In step **704**, volume layer **110-1** traverses tables **402**, **404**, and **406** to find a LD data page corresponding to the virtual volume block address. As described above, if such an LD data page exists, the virtual volume block address can be mapped by a pointer identifying a logical disk ID and an offset from the start of the logical disk. Step **704** is followed by step **706**.

[0079] In step **706**, volume layer **110-1** determines if it has found the logical disk ID and the offset from the start of the logical disk. If so, step **706** is followed by step **710**. Otherwise step **706** is followed by step **708**.

[0080] In step **708**, virtual volume layer **110-1** returns all zeros to host **118** in response to the read request. In one embodiment when the virtual volume (also referred to as a "Thin Provisioned Virtual Volume" or "TPVV") is implemented using snapshot technology as a list of differences (exceptions) from a dummy base volume having all zeros. When a data block for a read request cannot be found in the virtual volume exception list, virtual volume layer **110-1** will look to the dummy base volume for the requested data block having the specified virtual volume block address. When searching for the requested data block in the dummy base volume, virtual volume layer **110-1** will find only zeros and thus return only zero data to host **118**. For a description of a snapshot implemented as an exception list, please see U.S. patent application Ser. No. 10/655,951, entitled "Time-And-Space Efficient Mechanism To Create Virtual Storage Volume Copies," attorney docket no. 3PD-M-8497 US; U.S. patent application Ser. No. 10/655,963, entitled "Efficient And Reliable Virtual Volume Mapping," attorney docket no. 3PD-M-8498 US; and U.S. patent application Ser. No. 10/655,961, entitled "Read/Write Snapshot," attorney docket no. 3PD-P100, which are incorporated by reference in their entirety. Step **708** is followed by step **714**.

[0081] In step **710**, virtual volume layer **110-1** issues a read command to the data block identified by the logical disk ID and the identified offset to logical disk layer **112-1**. Step **708** is followed by step **712**.

[0082] In step **712**, logical disk layer **112-1** performs a normal read to the data block identified by the logical disk ID and the identified offset. Step **712** is followed by step **714**.

[0083] In step **714**, logical disk layer **112-1** ends method **600**.

[0084] Various other adaptations and combinations of features of the embodiments disclosed are within the scope of the invention. Numerous embodiments are encompassed by the following claims.

What is claimed is:

1. A method for writing to a virtual volume that is mapped to at least one logical disk, the virtual volume comprising an exception list, the method comprising:

receiving a request to write a data to the virtual volume at an address;

traversing the exception list with a path based on a portion of the address to find a pointer;

if the pointer does not exist,

determining if a number of available data pages in a data page pool is less than a threshold;

if the number of available data pages in the data page pool is not less than a threshold,

allocating one data page from the data page pool to form the data page;

writing the pointer in the exception list with the path based on the portion of the address;

writing the data in the data page at a location based on another portion of the address.

2. The method of claim 1, further comprising:

if the pointer exists,

following the pointer to the data page;

writing the data in the data page at the location based on another portion of the address.

3. The method of claim 1, further comprising:

if the number of available data pages in the data page pool is less than the threshold,

requesting additional data pages to be added to the data page pool;

determining if the additional data pages have been added before timing out;

if the additional data pages have been added before timing out,

allocating one data page from the data page pool to form the data page;

writing the pointer in the exception list with the path based on the portion of the address;

writing the data in the data page at the location based on another portion of the address.

4. The method of claim 3, further comprising:

if the additional data pages have not been added before timing out, returning a write failure message.

5. The method of claim 1, in response to said requesting additional data pages to be added to the data page pool, further comprising:

determining if said one logical disk has one storage unit ("logical disk region") available;

if said one logic disk has one logical disk region available,

dividing said one logical disk region into data pages;

allocating the data pages to the data page pool.

6. The method of claim 4, further comprising:

if said one logic disk does not have one logical disk region available,

creating a new logical disk;

dividing a logical disk region from the new logical disk into data pages;

allocating the data pages to the data page pool.

* * * * *