SYSTEMS AND METHODS FOR INTELLIGENT DISK REBUILD AND LOGICAL GROUPING OF SAN STORAGE ZONES

Inventors: Dean Kalman, Cary, NC (US); Jeffrey MacFarland, Wake Forest, NC (US)

Assignee: Adaptec, Inc.

Filed: Jul. 3, 2008

Publication Classification

Int. Cl.
G06F 11/14 (2006.01)
G06F 12/00 (2006.01)

U.S. Cl. 714/20; 711/114; 714/E11.113; 711/E12.001

ABSTRACT

A method of rebuilding a replacement drive used in a RAID group of drives is disclosed. The rebuilding method includes tracking data modification operations continuously during use of the drives. The method also includes saving the tracked data modifications to a log in a persistent storage, where the tracked data modifications are associated with stripe data present on the drives. Then, rebuilding a failed one of the drives with a replacement drive. The rebuilding is facilitated by referencing the log from the persistent storage, and the log facilitating reading only portions of stripe data from surviving drives and omitting reading of portions from the drives where no data was written. Thus, the rebuilding only rebuilds the stripe data to the replacement drive. Also provided is a zoning method, which enables logical zone creation from storage area networks.

<table>
<thead>
<tr>
<th>Stripe</th>
<th>Written</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>True</td>
</tr>
<tr>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
</tr>
<tr>
<td>6</td>
<td>False</td>
</tr>
<tr>
<td>Stripe</td>
<td>Drive 0</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>0</td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>D1</td>
</tr>
<tr>
<td>2</td>
<td>XX</td>
</tr>
<tr>
<td>3</td>
<td>D2</td>
</tr>
<tr>
<td>4</td>
<td>XX</td>
</tr>
<tr>
<td>5</td>
<td>XX</td>
</tr>
<tr>
<td>6</td>
<td>XX</td>
</tr>
</tbody>
</table>

**FIG. 1**

<table>
<thead>
<tr>
<th>Stripe</th>
<th>Written</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>True</td>
</tr>
<tr>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
</tr>
<tr>
<td>6</td>
<td>False</td>
</tr>
</tbody>
</table>

**FIG. 2**
SYSTEMS AND METHODS FOR INTELLIGENT DISK REBUILD AND LOGICAL GROUPING OF SAN STORAGE ZONES

CLAIM OF PRIORITY


FIELD OF THE INVENTION

[0002] Embodiments of this invention generally relates to replacing a failed disk drive that is part of a RAID drive group and rebuilding the replaced disk drive and creating logical grouping of the SAN Storages.

BACKGROUND OF THE INVENTION

[0003] When a drive fails that is part of a RAID 1, RAID 5, or RAID 6 drive group, the failing drive should be replaced. Once the failing drive is replaced, RAID controllers go through a process called rebuild. For RAID 1, this would involve a copy operation from the surviving drive to the replaced drive. For RAID 5 and RAID 6, this would involve a reconstruction of the data or parity from the surviving drives to the replaced drive.

[0004] Currently storage is allocated from individual storage enclosures. When provisioning the storage in a SAN environment, the user must understand the location, capabilities, reliability and access control associated with each storage enclosure. Therefore, the user needs to keep track of each storage enclosure for its location, reliability, capabilities, and access control characteristics.

[0005] In view of these issues, embodiments of the invention arise.

SUMMARY

[0006] Broadly speaking, embodiments of the invention provide methods and systems for intelligent rebuilding of the replaced disk drive after disk failure, and creating SAN storage zones to logically group a plurality of storage devices.

[0007] In one embodiment, with the increase in disk drive sizes, rebuild times are becoming exorbitantly long, taking many hours or days. Long rebuild times are a detriment since they impact the overall RAID controller performance and in addition leaving user data exposed without protection. If for example a second drive fails while a RAID 5 drive group is rebuilding, the drive group will go offline and the data on that drive group will be lost. Speeding up rebuild times is therefore an essential requirement going forward. In this embodiment, an embodiment to speed up rebuild times is to use a host write tracking persistent log. The log is configured to keep track of what areas on the disk group have been written by the host since the drive group was constructed. As result, there is no need to reconstruct an unwritten area since there is no data to reconstruct.

[0008] In another embodiment, a method of rebuilding a replacement drive used in a RAID group of drives is disclosed. The method includes tracking data modification operations continuously during use of the drives. The method also includes saving the tracked data modifications to a log in a persistent storage, where the tracked data modifications are associated with stripe data present on the drives. Then, rebuilding a failed one of the drives with a replacement drive. The rebuilding is facilitated by referencing the log from the persistent storage, and the log facilitating reading only portions of stripe data from surviving drives and omitting reading of portions from the drives where no data was written. Thus, the rebuilding only rebuilds the stripe data to the replacement drive.

[0009] In another embodiment, storage zones are defined. The logical grouping of SAN storage based on location or other characteristics is established, instead of based upon individual storage enclosures within a SAN. For example, the storage zone can consist of all the storage located within one computer rack, the storage contained within a building, or storage with particular characteristics, such as performance, cost, and reliability. Along these lines, initiator permissions are defined for each created storage zone. One benefit of zoning is it allows for simplified storage administration, simplified storage allocation and/or use. Initiator permissions and policy are then associated with storage zones. Thus, SAN storage can be allocated via "logical grouping" and not individual storage enclosures.

[0010] In yet another embodiment, a method of creating storage area network zones is disclosed. The method includes identifying a plurality of storage devices. Then, assigning each of the plurality of storage devices to a logical group, where the logical group being identified by characteristics. Then, presenting the plurality of storage devices as part of the logical group without regard to enclosure identifications. Access and control properties are then assigned to the logical group, which provide access to the plurality of storage devices. Administration is also now carried out for the logical group, instead of the physical characteristics or individual SANs. Thus, easy SAN grouping can be carried out, where administration is simplified.

[0011] Other aspects of the invention will become more apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIGS. 1 and 2 show stripe data tables, illustrating data associated with rebuilding of a replacement disk drive after disk failure, in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0013] Embodiments of the invention provide methods and systems for intelligent rebuilding of the replaced disk drive after disk failure and creating SAN storage zones to logically group a plurality of storage devices.
In iSCSI (Internet Small Computer Systems Interface) compliant Storage Area Networks, the SCSI commands are sent in IP packets. Use of IP packets to send SCSI commands to the disk arrays enables implementation of a SAN over an existing Ethernet. Leveraging the IP network for implementing SAN also permits use of IP and Ethernet features, such as sorting out packet routes and alternate paths for sending the packets.

In the iSCSI therefore, there are main functional entities, initiators and targets. Initiators are machines that need to access data and targets are machines that provide the data. A target could be a RAID array or another computer system. Targets handle iSCSI requests from initiators. Target machines may include hot standby machines with "mirrored" storage. If the active machine fails, the standby machine will take over to provide the iSCSI service, and when the failed machine returns, the failed machine will re-synchronize with the standby machine and then take back the iSCSI service.

With the increase in disk drive sizes, rebuild times are becoming exorbitantly long taking many hours or days. Long rebuild times are a detriment since they impact the overall RAID controller performance and in addition leave the customers data exposed and possibly not protected. If for example a second drive fails while a RAID 5 drive group is rebuilding, the drive group will go offline and the data on that drive group will be lost. Speeding up rebuild times is therefore an essential requirement going forward. The embodiments of the present invention typically provide a faster rebuild of the replaced drive.

The main performance-limiting issues with disk storage relate to the slow mechanical components that are used for positioning and transferring data. Since a RAID drive group has many drives in it, an opportunity presents itself to improve performance by using the hardware in all these drives in parallel. For example, if we need to read a large file, instead of pulling it all from a single hard disk, it is much faster to chop it up into pieces, store some of the pieces on each of the drives in the group, and then use all the disks to read back the file when needed. This technique of chopping up pieces of files is called striping.

Striping can be done at the byte level, or in blocks. Byte-level striping means that the file is broken into "byte-sized pieces". The first byte of the file is sent to the first drive, then the second to the second drive, and so on. Sometimes byte-level striping is done as a sector of 512 bytes. Block-level striping means that each file is split into blocks of a certain size and those are distributed to the various drives. The size of the blocks used is also called the stripe size (or block size, or several other names), and can be selected from a variety of choices when the drive group is set up.

The advantages of the present invention are numerous. Most notably, the system and methods described herein provides a faster way of rebuilding the replaced drive in a RAID group by tracking data modification operations (or stripping information) (e.g., write, delete, update) continuously and rebuilding the replaced drive by reading only the portions of stripe from one or more surviving disk drives in the RAID array.

In one embodiment, the disk rebuild time is enhanced by the use of a persistent write operations tracking module. The persistent write operations tracking module keeps track of what areas on the disk group have been written by the host since the drive group was constructed. The tracking information is stored in a persistent tracking log. With the information contained in the persistent tracking log, a replaced disk drive can be rebuilt quickly by selectively reading only parts (e.g., stripping information) of one or more surviving disk drives. There is no need to reconstruct an unwritten area since there is no data to reconstruct. A simplified example using a RAID 1 drive group is shown in FIG. 1.

The persistent tracking log is used to track the stripes that have been written. FIG. 2 illustrates an example of the persistent tracking log.

When the rebuild algorithm starts, it looks at the persistent log and determines which stripes need to be rebuilt. In this example illustrated by FIG. 2, stripes 0, 1, and 3 need to be rebuilt and stripes 2, 4, 5, and 6 do not need to be rebuilt because the "written" flag is "false", which means that after no data was written in stripes 2, 4, 5, and 6 after the disk drive group was constructed or put to work in the RAID. This simple example would result in a >50% increased rebuild time. Thus, a percentage savings can be identified as a function of used and unused space on disk drives being rebuilt.

In one embodiment, the persistent tracking log is maintained by the RAID controller. In other embodiment, the persistent tracking log may be maintained by any component of the computing system to which the RAID array is in communication with so long as the persistent tracking log can be retrieved at a later time to rebuild the replacement drive. The persistent tracking log, in one embodiment, is stored in a relational database. In other embodiment, the persistent tracking log is stored in a non-volatile memory, including a disk drive, ROM, Flash Memory, or any similar storage media.

In accordance with another embodiment, methods and systems for creating SAN storage zones to logically group a plurality of storage devices is provided. The advantages provided by this embodiment are numerous. Most notably, the system and methods described herein eliminate a need for the user to keep track of the storage characteristics, and location of each individual storage enclosure.

Instead, a logical group consisting of a plurality of storage enclosures that may be located at different locations and having different storage characteristics is created. The logical group of storage enclosures is then made available as a single storage enclosure to the user. The administrator of the logical group may modify the characteristics of the logical group by adding or removing one or more storage enclosures, changing locations of the one or more storage enclosures in a logical group.

In one embodiment, the storage enclosures in a logical group are hidden from the user. Hence, any change (e.g., adding or removing enclosures, changing location, etc.) in the structure of logical groups does not affect overall system configuration and usage. Therefore, the logical grouping of the storage enclosures simplifies the management of the Storage Area Network (SAN) and permits efficient storage, configuration and privilege management.
With the creation of the storage zone, i.e., the logical grouping of the storage enclosures, SAN storage is no longer viewed at the enclosure level. The storage enclosures are logically grouped together to meet customers’ unique requirements for administrating, provisioning, and usage of the storage enclosures.

The storage administrator defines the storage zone by creating a logical group and adding the selected storage enclosures to the local group. The access control properties are then defined and permissions to individual storage initiators e.g., iSCSI (Internet Small Computer Systems Interface), Fibre Channel (FC), SAS, etc. Initiator permissions can be unique for each initiator within a storage zone. In one embodiment, logical groups of initiators can also be defined and added to a particular storage zone.

In one embodiment, the SAN administrator(s) defines grouping properties for each of the physical and logical storage coupled to the SAN appliances. The SAN appliance as described herein a box including slots for a plurality of server blades, RAID disk arrays, and SAN control and management software to control and manage the server blades, RAID, data buses, and other necessary components of the SAN. The properties may include location of the storage, names of special characteristics, capabilities, and type of the storage. In one embodiment, each property in the properties is structured in a tree structure format. For example, under a “Location” named node in the property tree structure, a nod named “Building 23” is created. Under the “Building 23” node, a child node named “Server Room A” may be created. More sibling and child nodes may be created to properly identify a location. The properties may be stored anywhere in the SAN so long as the appliance in which the zone grouping is being created may read the properties.

One or more zone grouping rules are then created and stored in the SAN. The zone grouping rule may define a set of properties that if matched would trigger creation of a zone group. A zone grouping rule may be set to be active or inactive. The appliance discovers all the storages that are coupled to the appliance and retrieves the properties associated with each of the storage. Further, based on one or more active zone grouping rules, the appliance attempt to match the properties of the storages. If a matching rule is satisfied, the appliance creates a zone group of the storages that provides matching properties as defined by one or more zone grouping rule. The zone groups are then permanently stored in the appliance. The SAN administrator may edit the zone groups if a change in the group is necessary.

A set of default group properties is provided. One or more default group properties are attached to a newly created zone group. The zone group rule would include which default group properties are to be used for a newly created group. The group properties may include permissions and privilege grants to one or more storage initiators.

In one embodiment, storage zones may be created by grouping the storage enclosures based on a location. In another embodiment, storage zones may be created by grouping the storage enclosures based on reliability characteristics of the storage enclosures. In yet another embodiment, a zone group may be created based on any physical or logical characteristics so long as the physical or logical characteristics is defined in the property of the storage enclosures and one or more zone group rules are defined to use the physical or logical characteristics to create zone groups.

By providing a layer of abstraction over the storage initiators and storage enclosures, initiator storage allocation does not require involvement of the Storage Area Network (SAN) administrator. The storage initiators work with the storage zones and not with the physical storage enclosures. Furthermore, more storage enclosures can be seamlessly added to a storage zone without impacting availability of storage interface to the initiators of users and without a need to create access control properties for the newly added storage enclosure. Similarly, new storage initiators may be added to a storage zone without impacting the usage of the physical storage enclosures in the storage zone.

Since from usage view point, a storage zone is treated same as a physical storage enclosures, a unique set of permission may be associated with the storage zone, similar to associating access control properties to a physical storage enclosure. Therefore, the logical grouping of SAN greatly simplified the administration and use of the storage enclosures.

With the above embodiments in mind, it should be understood that the invention may employ various hardware and software 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 invention are useful machine operations. The invention also relates to a device or an apparatus for performing these operations. The apparatus may be specially constructed for the required purposes, such as the carrier network discussed above, or it may be a general purpose computer selectively activated or configured by a computer program stored in the computer. In particular, various general purpose machines may 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.

The programming modules, page modules, and subsystems described in this document can be implemented using a programming language such as Flash, JAVA, C++, C, C#, Visual Basic, JAVA Script, PHP, XML, HTML etc., or a combination of programming languages. Commonly available application programming interface (API) such as HTTP API, XML API and parsers etc. are used in the implementation of the programming modules. As would be known to those skilled in the art that the components and functionality described above and elsewhere in this document may be implemented on any desktop operating system which provides a support for a display screen, such as different versions of Microsoft Windows, Apple Mac, Unix/Linux, Linux etc. using any programming language suitable for desktop software development.

The programming modules and ancillary software components, including configuration file or files, along with setup files required for installing and related functionality as described in this document, are stored on a computer readable medium. Any computer medium such as a flash drive, a CD-ROM disk, an optical disk, a floppy disk, a hard drive, a shared drive, and an storage suitable for providing downloads from connected computers, could be used for storing the program-
ming modules and ancillary software components. It would be known to a person skilled in the art that any storage medium could be used for storing these software components so long as the storage medium can be read by a computer system. 

[0040] The invention may be practiced with other computer system configurations including hand-held devices, microprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers and the like. The invention may also be practiced in distributing computing environments where tasks are performed by remote processing devices that are linked through a network. 

[0041] As used herein, a storage area network (SAN) is an architecture to attach remote computer storage devices (such as disk arrays, tape libraries and optical jukeboxes) to servers in such a way that, to the operating system, the devices appear as locally attached. 

[0042] The invention can also be embodied as a computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data, which can thereafter be 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, DVDs, Flash, magnetic tapes, and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network coupled computer systems so that the computer readable code is stored and executed in a distributed fashion. 

[0043] While this invention has been described in terms of several preferable embodiments, it will be appreciated that those skilled in the art upon reading the specifications and studying the drawings will realize various alternation, additions, permutations and equivalents thereof. It is therefore intended that the present invention includes all such alterations, additions, permutations, and equivalents as fall within the true spirit and scope of the claims.

What is claimed is:

1. A method of rebuilding a replacement drive used in a RAID group of drives, comprising:
   tracking data modification operations continuously during use of the drives;
   saving the tracked data modifications to a log in a persistent storage, the tracked data modifications being associated with stripe data present on the drives; and
   rebuilding a failed one of the drives with a replacement drive, the rebuilding being facilitated by referencing the log from the persistent storage, and the log facilitating reading only portions of stripe data from surviving drives and omitting reading of portions from the drives where no data was written, so that the rebuilding only rebuilds the stripe data to the replacement drive.

2. The method of rebuilding a replacement drive as recited in claim 1, wherein RAID-level-5 writes data in stripes across multiple drives.

3. The method of rebuilding a replacement drive as recited in claim 1, wherein the replacement drive is rebuilt using the stripe data present on surviving drives that did not experience a failure, and the replacement drive completes the RAID group of drives.

4. The method of rebuilding a replacement drive as recited in claim 1, wherein modification operation include one or more of write operations, delete operations, or update operations.

5. The method of rebuilding a replacement drive as recited in claim 1, wherein the log identifies particular stripes to rebuild.

6. The method of rebuilding a replacement drive as recited in claim 6, wherein the log provides flags identifying written or no data.

7. The method of rebuilding a replacement drive as recited in claim 6, wherein rebuild time is reduced as a percentage of amount of stripes not requiring rebuild.

8. The method of rebuilding a replacement drive as recited in claim 1, wherein the log is stored in a relational database, a disk drive, a ROM, or a Flash Memory.

9. A method of creating storage area network zones, comprising:
   identifying a plurality of storage devices;
   assigning each of the plurality of storage devices to a logical group, the logical group being identified by characteristics;
   presenting the plurality of storage devices as part of the logical group without regard to enclosure identifications;
   assigning access control properties to the logical group, which provide access to the plurality of storage devices.

10. A method of creating storage area network zones as recited in claim 9, wherein one or more grouping rules are created and stored in a storage area network zone.

11. A method of creating storage area network zones as recited in claim 10, further comprising:
   discovering each storage device in the zone; and
   retrieving properties of each storage.

12. A method of creating storage area network zones as recited in claim 10, wherein the characteristics include one or more of location, name, purpose, physical attribute, or logical attribute.