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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 June 2005 (02.06.2005)

PCT

(10) International Publication Number
WO 2005/050386 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US2004/038324

(22) International Filing Date: 15 November 2004 (15.11.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/519,876 13 November 2003 (13.11.2003) US
60/519,576 13 November 2003 (13.11.2003) US

(71) Applicant *(for all designated States except US)*: **COMMVAULT SYSTEMS, INC.** [US/US]; Two Crescent Place, Oceanport, NJ 07757-0090 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **PRAHLAD, Anand** [US/US]; 3 Bucknell Drive, East Brunswick, NJ 00816 (US). **NGO, David** [US/US]; 152 Dorchester Way, Shrewsbury, NJ 07702 (US).

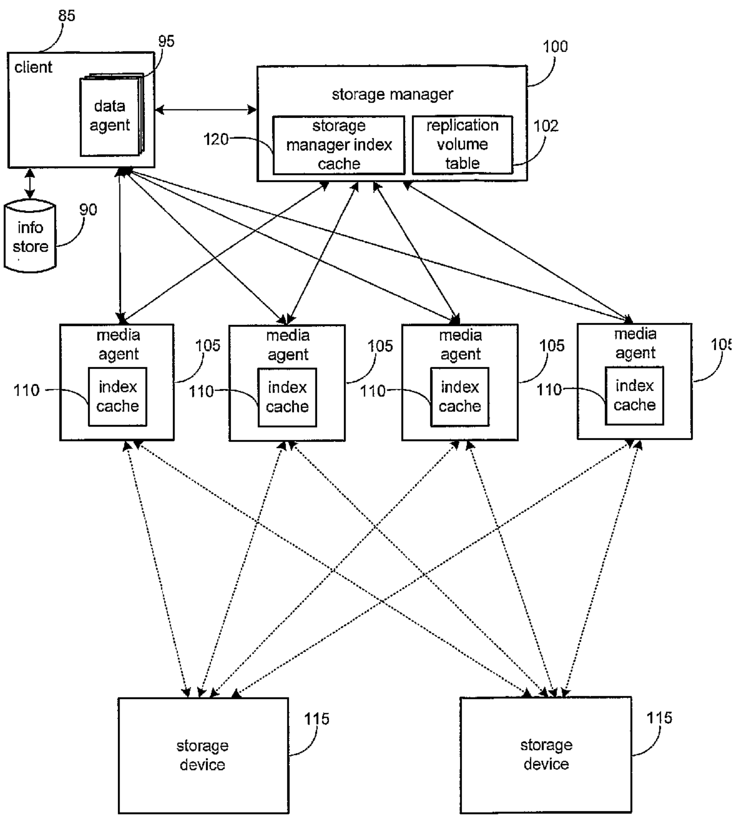
(74) Agent: **SHANAHAN, Michael**; Brown Raysman Millstein, Felder & Steiner LLP, 900 Third Avenue, New York, NY 10022 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR PERFORMING A SNAPSHOT AND FOR RESTORING DATA



(57) Abstract: The present invention relates to a method for tracking a plurality of snapshots of an information store. The present invention comprises performing a first snapshot of an information store that indexes the contents of the information store, copying the contents of the information store to a first storage device, using the first snapshot, and updating a replication volume table indicating the storage of the contents of the first information store using the first snapshot on the first storage device. A second snapshot is performed of the information store that indexes the contents of the information store, the contents of the information store are copied to a second storage device using the second snapshot and the replication volume table is updated to indicate the storage of the contents of the first information store using the second snapshot on the second storage device.

WO 2005/050386 A2

ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR PERFORMING A SNAPSHOT AND FOR RESTORING DATA

Applicant(s) hereby claim the benefit of provisional patent application serial nos. 60/519,876 and 60/519,576, entitled "SYSTEM AND METHOD FOR PERFORMING A SNAPSHOT AND FOR RESTORING DATA," and "SYSTEM AND METHOD FOR PERFORMING AN IMAGE LEVEL SNAPSHOT AND FOR RESTORING PARTIAL VOLUME DATA," respectively, each filed on November 13, 2003, attorney docket nos. 4982/41PROV and 4982/47PROV, respectively. These applications are incorporated by reference herein in their entirety.

COPYRIGHT NOTICE

RELATED APPLICATIONS

This application is related to the following patents and pending patent applications, each of which is hereby incorporated herein by reference in its entirety:

- U.S. Patent No. 6,418,478, entitled "PIPELINED HIGH SPEED DATA TRANSFER MECHANISM," issued July 9, 2002, attorney docket number 4982/6;
- Application Serial No. 09/610,738, entitled "MODULAR BACKUP AND RETRIEVAL SYSTEM USED IN CONJUNCTION WITH A STORAGE AREA NETWORK," filed July 6, 2000, attorney docket number 4982/8;
- Application Serial No. 09/744,268, entitled "LOGICAL VIEW AND ACCESS TO PHYSICAL STORAGE IN MODULAR DATA AND STORAGE MANAGEMENT SYSTEM," filed January 30, 2001, attorney docket number 4982/10;

- Application Serial No. 60/409,183, entitled "DYNAMIC STORAGE DEVICE POOLING IN A COMPUTER SYSTEM," filed September 9, 2002, attorney docket number 4982/18PROV;
- Application Serial No. 10/681,386 entitled "SYSTEM AND METHOD FOR MANAGING STORED DATA," filed October 7, 2003, attorney docket number 4982/29; and
- Application Serial No. 60/460,234, entitled "SYSTEM AND METHOD FOR PERFORMING STORAGE OPERATIONS IN A COMPUTER NETWORK," filed April 3, 2003, attorney docket number 4982/35PROV.

BACKGROUND OF THE INVENTION

The invention disclosed herein relates generally to a system and method for performing a snapshot and for restoring data. More particularly, the present invention relates to a system and method for performing snapshots of an information store, which are stored across multiple storage devices, and for restoring partial or full snapshots.

To obtain a more thorough understanding of the present invention, the following discussion provides additional understanding regarding the manner is which magnetic media is used to store information. Using traditional techniques, copies of an information store are performed using the operating system's file system. Copying is done by accessing the operating system's (OS) file system for the information store to be backed-up, such as the Windows NTFS file system. The file allocation system of the operating system typically uses a file allocation table to keep track of the physical or logical clusters across which each file in the information store is stored. Also called an allocation unit, a cluster is a given number of disk sectors that are treated as a unit, each disk sector storing a number of bytes of data. This unit, the cluster, is the smallest unit of storage the operating system can manage. For example, on a computer running Microsoft's Windows 95 operating system, the OS uses the Windows FAT32 32-bit file allocation table having a cluster size to 4K. The number of sectors is determined when the disk is formatted by a formatting program, generally, but not necessarily, when the OS is installed.

The operating system allocates disk space for a file only when needed. That is, the data space is not preallocated but allocated dynamically. The space is allocated one cluster at a time, where a cluster is a given number of consecutive disk sectors. The clusters for a file are chained together, and kept track of, by entries in a file allocation table (FAT).

The clusters are arranged on the disk to minimize the disk head movement. For example, all of the space on a track is allocated before moving on to the next track. This is accomplished by using the sequential sectors on the lowest-numbered cylinder of the lowest numbered platter, then all sectors in the cylinder on the next platter, and so on, until all sectors on all platters of the cylinder are used. This is performed sequentially across the entire disk, for example, the next sector to be used will be sector 1 on platter 0 of the next cylinder.

For a hard (fixed) disk, FAT, sector, cluster, etc. size is determined when a disk formatting program formats the disk, and are based on the size of the partition. To locate all of the data that is associated with a particular file stored on a hard disk, the starting cluster of the file is obtained from the directory entry, then the FAT is referenced to locate the next cluster associated with the file. Essentially, the FAT is a linked list of pointers to clusters on the disk, e.g., each 16-bit FAT entry for a file points to the next sequential cluster used for that file. The last entry for a file in the FAT has a number indicating that no more clusters follow. This number can be from FFF8 to FFFF (base 16) inclusive.

FIG. 1 shows an example directory entry 2 of a Windows-formatted hard disk and accompanying FAT 20. The exemplary directory entry 2 consists of 32 bytes of data. The name of the file and its extension are stored in the first eleven bytes 4 of the directory entry 2 and a file attribute byte 6 is provided. By definition, ten bytes 8 are reserved for future use and four bytes are provided to store time 10 and date 12 information (two bytes each). Two cluster bytes 14 point to the first cluster of sectors used to store the file information. The last four bytes 18 of the directory entry 2 are used to store the size of the file.

A sixteen-byte section of a FAT 20 is depicted. The first four bytes 21 store system information. A two-byte pair, bytes four and five (16), are the beginning bytes of the FAT 20 used to track file information. The first cluster for data space on all disks is cluster "02." Therefore, bytes four and five (16) are associated with the first cluster of disk sectors "02" used to store file information. Bytes six and seven (22) are associated with cluster "03" . . . and bytes fourteen and fifteen (24) are associated with cluster "07."

This example illustrates how sectors associated with a file referenced in a directory are located. The cluster information bytes 14 in the directory 2 point to cluster number "02." The sectors in cluster "02" (not shown), contain the initial sector of data for the referenced file. Next, the FAT is referenced to see if additional clusters are used to store the file information. FAT bytes four and five (16) were pointed to by the cluster information bytes 14, and the information stored in bytes four and five (16) in the FAT 20 point to the next cluster used for the file. Here, the next cluster is "05". Accordingly, cluster "05" contains the next sector of data for the referenced file. FAT bytes ten and eleven (26) contain an end-of-file flag, "FFFF," indicating there are no more clusters associated with the referenced file. All of the information comprising the referenced file, therefore, is contained in clusters "02" and "05" on the disk.

As with other applications running on the computer, a typical backup application provides a read request to the operating system, which handles interpretation of the information contained in the FAT and reading of each file for the backup application. A file system is provided on the storage device that is used by the backup application to write files that are copied to the device. Similarly, the recovery portion of the backup application, or a separate recovery application, may read files from the storage device for recovery of the information.

Inherent problems and disadvantages have been discovered with currently available systems and methods for archiving data contained in an information store. One technique is to perform a full copy of the data contained in the information store. Utilizing this technique results in two separate copies of the information store, and the length of time it takes to make this kind of copy is related to the amount of data copied and the speed of the disk subsystem. For example, assuming a transfer rate of 25 MB/sec, the approach will take one hour to copy 90GB of data. These techniques, however, in addition to other disadvantages, require the applications on the information store to be quiesced during the copy routine. This places a significant burden on system administrators to complete copying and get critical systems back into the production environment as, absent a high-speed data bus, the copying may consume a significant amount of time to complete.

Administrators typically keep multiple copies of a given information store. Unfortunately, this has the drawback of requiring *n* times the amount of space of the information store to maintain *n* copies, which can be quite expensive to store, in addition to requiring complex and time consuming techniques for restoration of the copied data.

One currently available alternative is to perform snapshots of an information store. With current snapshot systems and methods, administrators create an incremental copy that is an exact point-in-time replica of the source volume each time a snapshot is taken. A series of snapshot are stored locally on the information store from which it was taken and track incremental changes to the data in the information store. Furthermore, changed data is written to a new location in the information store as tracked by the snapshot. With knowledge regarding the change, as well as the changed data, the snapshot can be used to "roll back" changes to an information store to the point in time when the snapshot was taken. If there should be any logical corruption in the information store's data that went un-detected for a period of time, however, these incremental updates faithfully replicate that logical corruption to the data when copying. Additionally, other drawbacks are associated with currently know snapshot techniques, including the significant drawback of preventing restoration from the snapshot in the event that the information store fails, as both the snapshot and the information store become unavailable.

Systems and methods are needed, therefore, that overcome problems associated with currently known techniques for taking, maintaining and restoring snapshots.

SUMMARY OF THE INVENTION

The present invention addresses, among other things, the problems discussed above with copying up data using systems and methods known to those of skill in the art. The invention provides systems and methods for performing *n* snapshots of an information store, without requiring *n* times the space of the information store, and storing those snapshots in multiple destinations across a network.

One embodiment of the system of the present invention creates the snapshots by taking a snapshot that indexes only clusters for files that were created or changed since the last snapshot. A snapshots, t_n, is restored by restoring the clusters from the snapshot t_n. The clusters that were not restored from snapshot t_n are restored from snapshot t_{n-1}, etc., until the remaining clusters are restored from the first snapshot, snapshot t_0.

In accordance with some aspects of the present invention, multiple snapshots are kept on a storage device, without requiring *n* times the space of the total volume of the information store. The system creates snapshots at various points in time that index only clusters for files that were created or changed since the last snapshot, and creates a copy of the data that has been changed or created. This allows users to keep several snapshots without requiring *n* times the space of the total volume of the information store.

In some embodiments, the system stores a map, which may be part of a snapshot, to track specific files and folders with their corresponding copied clusters. The map created by reading data from the file allocation table of the information store and associates files and folders with the clusters stored in the snapshots. In this way, even though the snapshot was performed at the cluster level, individual or groups of files and/or folders may be restored without unnecessarily restoring the entire information store.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated in the figures of the accompanying drawings which are meant to be exemplary and not limiting, in which like references are intended to refer to like or corresponding parts, and in which:

FIG. 1 is an example directory entry for a file in a prior art FAT of a Windows-formatted hard disk;

Fig. 2 is a block diagram illustrating a network architecture for performing snapshot operations according to one embodiment of the present invention;

Fig. 3 is a flow diagram illustrating a method for creating a snapshot according to one embodiment of the present invention;

Fig. 4 is a block diagram illustrating the relationship between a map and a snapshot according to one embodiment of the present invention;

Fig. 5 is a flow diagram illustrating a method for restoring a snapshot according to one embodiment of the present invention; and

Fig. 6 is a flow diagram illustrating a method for restoring specific files or folders from a snapshot according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

With reference to Figs. 2 through 6, embodiments of the present invention are shown. Fig. 2 presents a block diagram illustrating the components of a system for performing storage and restoration operations on electronic data in a computer network according to one embodiment of the invention. It should be understood that the invention is not limited to networked environments, and may also be implemented on a stand-alone computer or electronic device.

As shown, the system of Fig. 2 includes a storage manager 100, including a volume replication table 102 and a storage manager index cache 120, and one or more of the following: a client 85, an information store 90, a data agent 95, a media agent 105, a media agent index cache 110, and a storage device 115. One exemplary embodiment of the present system is the CommVault QuNetix three-tier system available from CommVault Systems, Inc. of Oceanport, NJ, further described in U.S. Patent Application No. 09/610,738 and hereby incorporated by reference in its entirety.

A data agent 95 is a software module that is generally responsible for retrieving data from an information store 90 for copies, snapshots, archiving, migration, and recovery of data stored in an information store 90 or other memory location, e.g., hard disc drive. Each client computer 85 preferably has at least one data agent 95 and the system can support many client computers 85. The data agent 95 provides an interface to an information store 90 to execute copies, snapshots, archiving, migration, recovery and other storage operations on data in conjunction with one or more media agents 105. According to one embodiment, each client 85 runs a number of data agents 95, wherein each data agent is configured to interface with data generated by or from a specific application, e.g., a first data agent to interface with Microsoft Exchange data and a second data agent to interface with Oracle database data. As is explained in greater detail herein, a data agent 95 is in communication with one or more media agents 105 to effect the distributed storage of snapshots on one or more storage devices 115 that are remote from the information store that is the source of the snapshot 90.

The storage manager 100 is a software module or application that coordinates and controls other components comprising the system, e.g., data and media agents, 95 and 105, respectively. The storage manager 100 communicates with data 95 and media 105 agents to control and manage snapshot creation, migration, recovery and other storage operations. According to one embodiment, the storage manger 100 maintains data in a storage manager index cache 120 that instructs a given data agent 95 to work in conjunction with a specific media agent 105 to store snapshots on one or more storage devices 115.

The storage manager 100 maintains a storage manager index cache 120. Data in the storage manager index cache 120, which the storage manager 100 collects from data agents 95, media agents 105, user and other applications, is used to indicate, track and associate: logical relationships and associations between components of the system, user preferences, management tasks, and other data that is useful to the system. For example, the storage manager index cache 120 may contain data that tracks logical associations between media agents 105 and storage devices 115. The storage manager index cache 120 may also contain data that tracks the status of storage operations to be performed, storage patterns such as media use, storage space growth, network bandwidth, service level agreement ("SLA") compliance levels, data protection levels, storage policy information, storage criteria associated with user preferences, data retention criteria, storage operation preferences, and other storage-related information.

A media agent 105 is a software module that transfers data in conjunction with one or more data agents 95, as directed by the storage manager 100, between an information store 90 and one or more storage devices 115, such as a tape library, a magnetic media storage device, an optical media storage device, or other storage device. The media agent 105 communicates with and controls the one or more storage devices 115. According to one embodiment, the media agent 105 may communicate with the storage device 115 via a local bus, such as a SCSI adaptor. Alternatively, the storage device 115 may communicate with the data agent 105 via a Storage Area Network ("SAN"). Other types of communication techniques, protocols and media are contemplated as falling within the scope of the invention.

The media agent 105 receives snapshots, preferably with the changed data that is tracked by the snapshot, from one or more data agents 95 and determines one or more storage devices 115 to which it should write the snapshot. According to one embodiment, the media agent 105 applies load-balancing algorithms to select a storage device 115 to which it writes the snapshot. Alternatively, the storage manager 100 instructs the media agent 105 as to which storage device 115 the snapshot should be written. In this manner, snapshots from a given information store 90 may be written to one or more storage devices 115, ensuring data is available for restoration purposes in the event that the information store fails. Either the media agent or the storage manager 100 records the storage device on which the snapshot is written in a replication volume table 102, thereby allowing the snapshot to be located when required for restoring the information store 90.

A media agent 105 maintains a media agent index cache 110 that stores index data the system generates during snapshot, migration, and restore operations. For example, storage operations for Microsoft Exchange data generate application specific index data regarding the substantive Exchange data. Similarly, other applications may be capable of generating application specific data during a copy or snapshot. This data is generally described as metadata, and may be stored in the media agent index cache 110. The media agent index cache 110 may track data that includes, for example, information regarding the location of stored data on a given volume. The media agent index cache 110 may also track data that includes, but is not limited to, file names, sizes, creation dates, formats, application types, and other file-related information, information regarding one or more clients associated stored data, information regarding one or more storage policies, storage criteria, storage preferences, compression information, retention-related information, encryption related information, and stream related information. Index data provides the system with an efficient mechanism for locating user files during storage operations such as copying, performing snapshots and recovery.

This index data is preferably stored with the snapshot that is backed up to the storage device 115, although it is not required, and the media agent 105 that controls the storage operation may also write an additional copy of the index data to its media agent index cache 110. The data in the media agent index cache 110 is thus readily available to the system for use in storage operations and other activities without having to be first retrieved from the storage device 115.

In order to track the location of snapshots, the system uses a database table or similar data structure, referred to herein as a replication volume table 102. The replication volume table 102, among other advantages, facilitates the tracking of multiple snapshots across multiple storage devices 115. For example, the system might, as directed by a policy or a user, store a first snapshot $t_0$on first storage device A, such as a tape drive or library, and then store subsequent snapshots containing only the changed culster(s), t_n, on a second storage device B, such as an optical drive or library. Alternatively, instructions may be stored within system components, e.g., a storage manger 100 or media agent 105, directing the storage devices 115 used to store snapshots. Information regarding the storage device 115 to which the snapshot is written, as well as other information regarding the snapshot generally, is written to the replication volume table 102. An exemplary structure according to one embodiment is as follows:

```
(
    id                    serial,      //   PRIMARY KEY FOR THIS TABLE
    PointInTime           integer,     //
    CreationTime          integer,     //   Timestamp of RV creation
    ModifyTime            integer,     //   Timestamp of last RV update
    CurrentState          integer,     //   Current state of RV
    CurrentRole           integer,     //   Current role of RV
    PrimaryVolumeId       integer,     //   FOREIGN KEY FOR SNRVolume
                                            TABLE
    PhysicalVolumeId      integer,     //   FOREIGN KEY FOR SNRVolume
                                            TABLE
    ReplicationPolicyId   integer,     //   FOREIGN KEY FOR
                                            ReplicationPolicy TABLE
    RVScratchVolumeId     integer,     //   FOREIGN KEY FOR
                                            RVScratchVolume table
    Flags                 integer,
    JobId                 LONGLONG,
    SnapVolumeId          integer,     //   FOREIGN KEY FOR SNRVolume
                                            TABLE
}
```

In the exemplary replication volume table, id is a unique identification number assigned by the system to the snapshot; PointInTime represents the date and time that the snapshot was created; CreationTime represents the date and time that the snapshot was completed; ModifyTime is the recorded date and time of the snapshot taken prior to the current snapshot; CurrentState is an identifier used to indicate a current status of the snapshot (e.g. pending, completed, unfinished, etc.); PrimaryVolumeId is the identifier for the information store 90 from which the snapshot is being made; PhysicalVolumeId is a hardware identifier for the information store 90; RVScratchVolumeId is an identifier for a scratch volume, which in some embodiments may be used to buffer additional memory as known to those of skill in the art; Flags contains a 32 bit word for various settings such as whether a snapshot has been taken previously, etc.; JobId stores the identifier for the job as assigned by a storage management module; and the SnapVolumeId points to the physical destination storage device 115 to which the snapshot is written.

As each snapshot indexes an information store at a given point in time, a mechanism must be provided that allows the snapshots taken of an information store to be chronologically related so that they are properly used for restoring an information store 90. According to the replication volume table 102, the CurrentRole integer may store a value for the relative position of a given snapshot in hierarchy of snapshots taken from a given information store 90 (e.g. first (t0), second (t1), t2, t3, etc.)

In some embodiments, components of the system may reside on and be executed by a single computer. According to this embodiment, a data agent 95, media agent 105 and storage manager 100 are located at the client computer 85 to coordinate and direct local copying, archiving, migration, and retrieval application functions among one or more storage devices 115 that are remote or distinct from the information store 90. This embodiment is further described in U.S. Patent Application Number 09/610,738.

One embodiment of a method for using the system of the present invention to perform snapshots is illustrated in the flow diagram of Fig. 3. When the system is initialized, or at other times as directed by a user or rules, e.g., policies or other instructions, the storage manager directs the data agent to perform an initial full snapshot of the data stored in the information store, e.g., indexing the location of all data in the information store, in conjunction with one or more media agents. The system copies all of the data on the information store with the initial snapshot to a storage device, step 300.

Advantageously, the snapshot and data copied from the information store may be written to a storage device that is remote or different from the information store, step 302, e.g., local data from a given information store is written to a storage device attached to a network. The selection of a destination storage device for the snapshot may be accomplished using one or more techniques known to those of skill in the art. For example, a fixed mapping may be provided indicating a storage device for which all snapshots and copied or changed data should be written. Alternatively, an algorithm may be implemented to dynamically select a storage device from among a number of storage devices available on a network. For example, a storage manager may select a media agent to handle the transfer of the snapshot and copied data to a specific storage device based on criteria such as available bandwidth, other scheduled storage operations, media availability, storage policies, storage preferences, or other consider considerations. The snapshot, preferably along with the data from the information store, is written to the selected destination storage device, step 304. According to certain embodiments, the snapshot contains information regarding the files and folders that are tracked by the snapshot. Alternatively, the information regarding the files and folders that are indexed by the snapshot, e.g., file system information, are stored on the storage device.

One embodiment of a snapshot used to track clusters read from the information store to clusters in a snapshot, as well as to map file and folder names corresponding to the snapshot clusters, is illustrated in Fig. 4. It should be noted that clusters are but one level of granularity that may be indexed by a snapshot, e.g., blocks, extents, etc. During the scan, the data agent creates a snapshot 350 and writes data, e.g., new or changed data, to a storage device 115. According to the present embodiment, the snapshot is illustrated as a flat file data structure, although those of skill in the art will recognize that the snapshot may be embodied in a number of disparate types of data structures.

The snapshot 350 is used to associate the original cluster numbers from an information store with clusters on a storage device, which in the present embodiment is a magnetic tape. It should be appreciated by those of skill in the art that the present invention is not limited to magnetic tape, and that the systems and methods described herein may be applicable to using snapshots with other storage technologies, e.g., storing disk geometry data to identify the location of a cluster on a storage device, such as a hard disk drive.

The tape offsets 356 for the clusters 372 in the snapshot 370 are mapped to original disk cluster information 352. File and folder names 354 may be scanned from the information store's FAT and also mapped to the tape offsets 356. A file part column 358 in the snapshot tracks the clusters 372 for each file and folder where each file and folder contains an entry for the first cluster 372. For files or folders that are stored in more than one cluster, sometimes not in contiguous clusters, the offset table entry for each further cluster is numbered consecutively 358.

In order to identify the files and folders represented by the stored clusters 372, e.g., changed data, in the snapshot 370, the map may exclude data from columns relating to the original disc clusters 352 and last snapshot 360. In order to keep track of changed verses unchanged clusters, however, the original disk cluster information 352 is stored in the map 350. Other information may also be stored in the map 350, such as timestamps for last edit and creation dates of the files.

For each snapshot, even though only clusters that have been changed or created since a previous snapshot are tracked in a given snapshot after the initial snapshot t_0, the snapshot may be provided with the data from all previous snapshots to provide the latest snapshot with folder and file information such that an index of the entire information store is maintained concurrently each snapshot. Alternatively, this may be bypassed in favor of creating a snapshot that indexes all data at a given point in time in the information store and copying only changed data.

Entries from each snapshot 350 may also contain a last-snapshot field 360 that holds an identifier for the last snapshot containing the cluster indexed by the entry at the time the current snapshot was created. According to an alternative embodiment, e.g., for snapshots that do not store the information from the information store's FAT, the snapshot only tracks clusters stored in the information store with the clusters indexed by the snapshot. For those embodiments, the snapshot 350 contains neither file and folder information 345 nor file part information 358.

Returning to Fig. 3, once the first full snapshot t_0 has been taken, step 300, the storage manager may implement a rule, policy, or similar set of instructions that require snapshots to be taken at certain time intervals. Accordingly, at each time interval where a subsequent snapshot t_n is taken, the data agent works in conjunction with one or more of the media agents to perform and store snapshot and accompanying data that changed since the subsequent snapshot, t_{n-1}, loop 306.

For each snapshot, t_n, that is taken of the information store, a comparison is performed such that only the clusters which have changed or been created since the last snapshot, t_{n-1}, was taken of that volume are stored, step 310. For example, in some embodiments the data agent employs a block filter or similar construct known to those of skill in the art to compare snapshot t_n with t_{n-1} and thereby detect changed clusters on an information store. Alternatively, the data agent may use other techniques know in the art, such as Copy on Write ("COW"), to identify changed data on an information store. If a given cluster in the information store has changed since the last snapshot in which the cluster appears, or if the cluster from the information store was created subsequent to the last snapshot, then the cluster is read from information store and stored with the new snapshot being written to the storage device, step 314.

A determination is made regarding the given storage device to which the snapshot and changed data (which may also include newly created data) is to be written, step 316. Techniques such as those described in conjunction with storage of the initial snapshot, steps 302 and 304, may also be employed regarding storage of subsequent snapshots. Advantageously, the initial snapshot and any subsequent snapshot may written to any storage device available in the network. Furthermore, there is no limitation to the combination of devices used to store the snapshots for a given information store. For example, an initial snapshot may be written to storage device A, a second and third snapshots may be written to storage device B, and a fourth snapshot may be written to storage device C. Regardless of the storage device that is selected, step 316, the replication volume table is updated to reflect the location, step 318, allowing snapshots to be located when a user requests to restore the information store from which the snapshots were taken.

System administrators use stored snapshots, in conjunction with the changed data that the snapshot indexes or tracks, to recover lost or corrupted information. Fig. 5 presents a flow diagram illustrating one embodiment of a method for restoring an information store from one or more snapshots. If the user or a system process wants to restore an information store from one or more snapshots, an interface is presented to restore the snapshot, step 400. The interface may be, for example, a graphical user interface ("GUI"), and Application Programming Interface ("API") or other interface known to those of skill in the art. The storage manager scans the replication volume table to identify available snapshots for presentation in a menu that allows selection of an available snapshot, step 402.

When the user selects a snapshot, the storage manager performs a query of the replication volume table to identify all previous snapshots for an information store from which the selected snapshot was taken, step 404. This may be accomplished by performing a search on the replication volume table for all snapshots with the same PrimaryVolumeId or PhysicalVolumeId. Starting with the selected snapshot, for each snapshot in the query result, loop 406, the storage manager directs a given media agent, in conjunction with a given data agent, to read and restore all clusters of changed data not already restored from clusters indexed by a prior snapshot, e.g., the latest version of each cluster, step 408. According to one embodiment, this is accomplished by restoring the clusters indexed by each of the snapshots in the query result, starting with the original snapshot, and overwriting clusters indexed by the original snapshot with changed clusters indexed by subsequent snapshots up to the snapshot representing the point in time selected by the user or system process. As an alternative, the last snapshot field of the selected snapshot may be utilized to determine the snapshots that should be utilized in the restore operation. The latest version of each cluster, starting with those indexed by the selected snapshot, is then restored, step 408.

As discussed above, embodiments of the invention are contemplated wherein FAT information of the information store is stored in conjunction with a given snapshot, e.g. the file and folder information corresponding to the clusters of changed data indexed by a given snapshot. Accordingly, the storage manager may allow the user to select individual files and/or folders to be selected for restoration from a snapshot. With reference to Fig. 6, a flow diagram is presented illustrating one embodiment of a method for restoring individual files and/or folders indexed by a snapshot.

When the user desires to restore the information store to a given point in time, the user interface allows the user to view the files and folders indexed by a snapshot representing the point in time as if the user were viewing a folder structure on a storage device, step 500. The storage manager retrieves the file and folder information for changed data that is indexed by one or more snapshots for display. Once one or more files and/or folders are selected, step 502, the storage manager selects those snapshots that index the given version of the files and/or folders using the replication volume table, step 502. Each snapshot indexing data for the one or more files to be restored are opened serially, loop 506. The changed data for the selected files and folders that are indexed by the snapshots are restored from clusters indexed by each snapshot, step 508,but not overwriting clusters indexed by prior snapshots.

While the invention has been described and illustrated in connection with preferred embodiments, many variations and modifications as will be evident to those skilled in this art may be made without departing from the spirit and scope of the invention, and the invention is thus not to be limited to the precise details of methodology or construction set forth above as such variations and modification are intended to be included within the scope of the invention.

WHAT IS CLAIMED IS:

1. A method for tracking a plurality of snapshots of an information store, the method comprising:

performing a first snapshot of an information store that indexes the contents of the information store;

copying the contents of the information store to a first storage device, using the first snapshot;

updating a replication volume table indicating the storage of the contents of the first information store using the first snapshot on the first storage device;

performing a second snapshot of the information store that indexes the contents of the information store;

copying the contents of the information store to a second storage device using the second snapshot; and

updating the replication volume table indicating the storage of the contents of the first information store using the second snapshot on the second storage device.

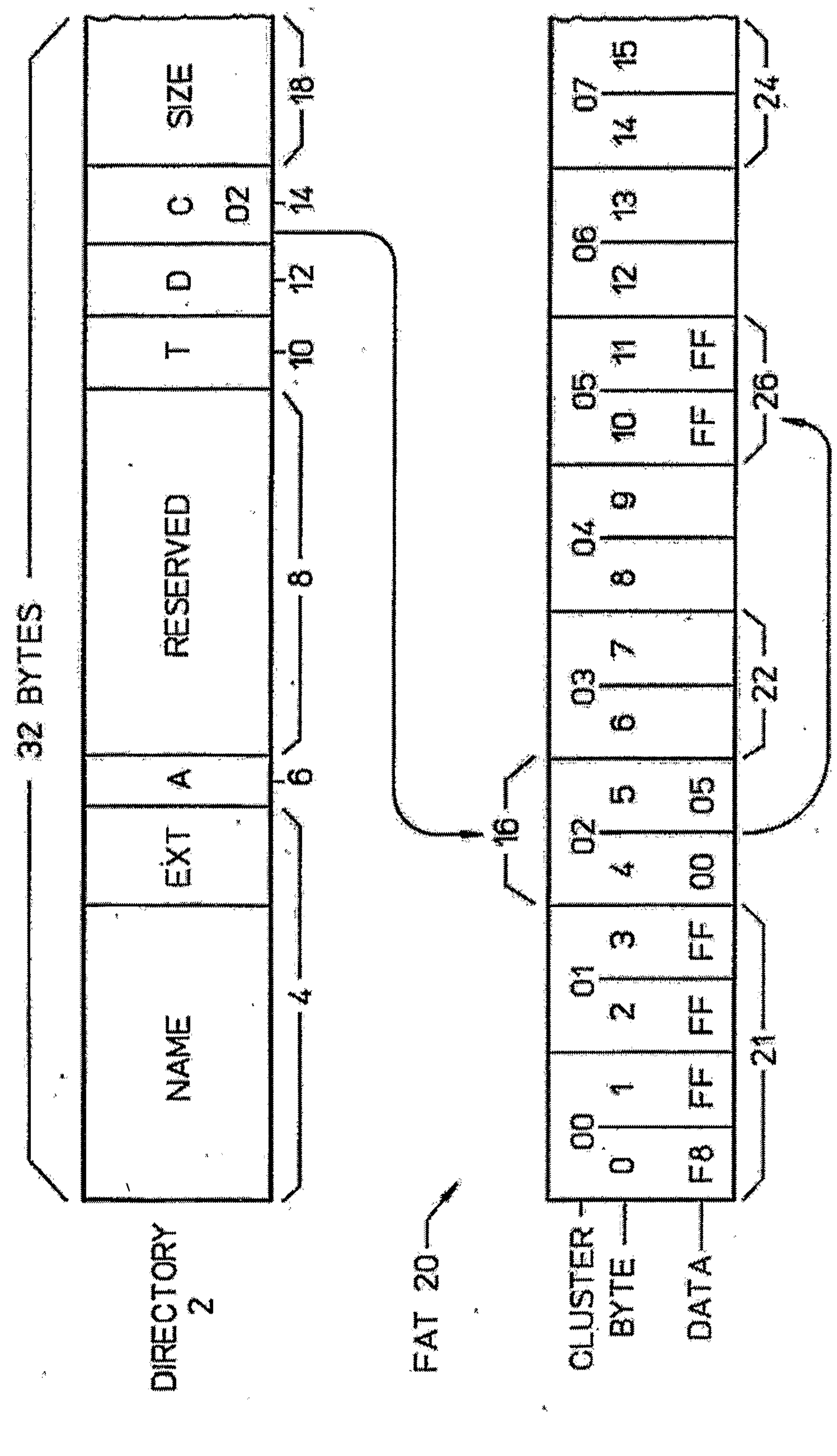


Fig. 1 (Prior Art)

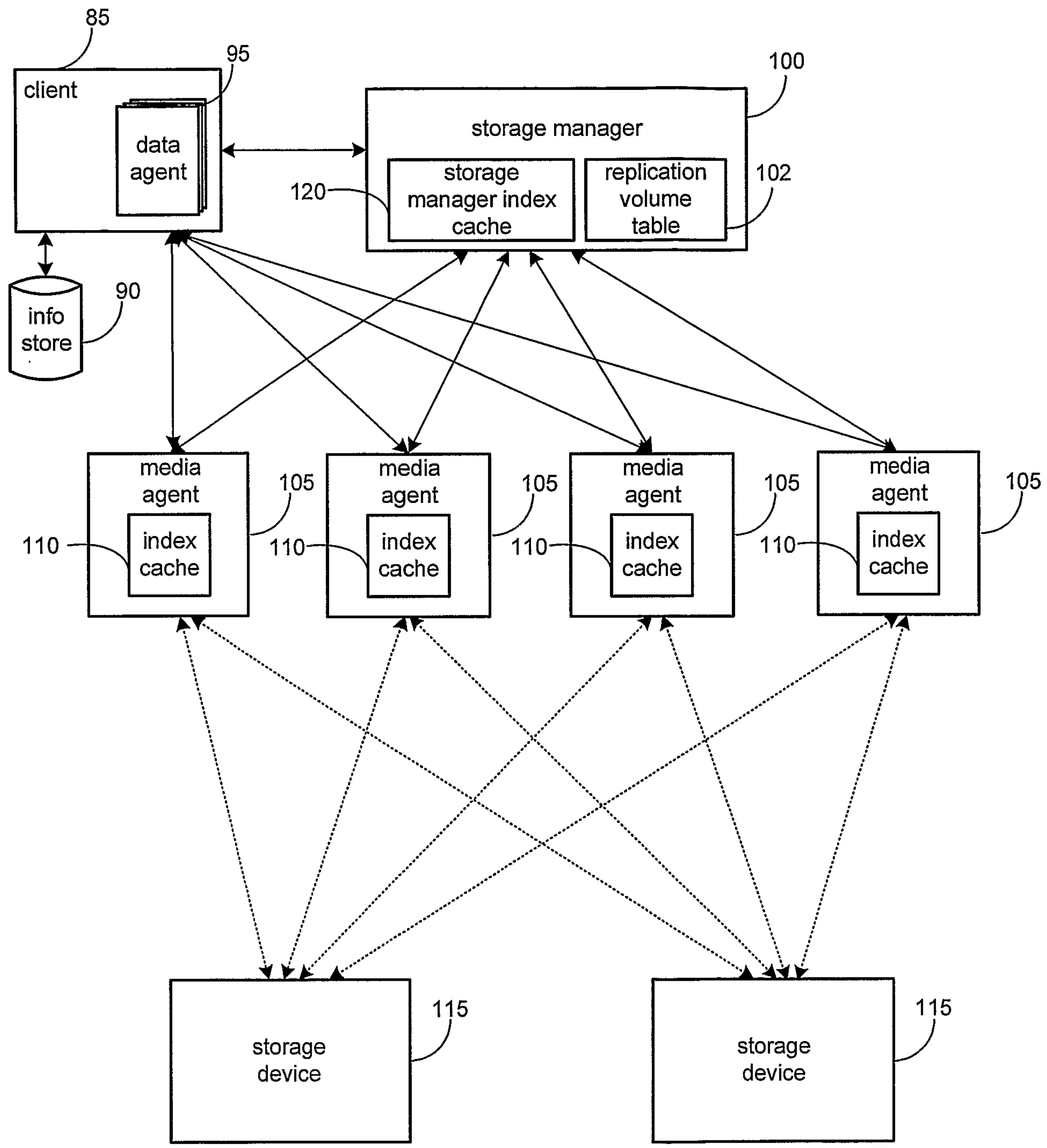

85
client
95
data agent
100
storage manager
120
storage manager index cache
replication volume table
102
info store
90
media agent
105
110
index cache
media agent
105
110
index cache
media agent
105
110
index cache
media agent
105
110
index cache
storage device
115
storage device
115

Fig. 2

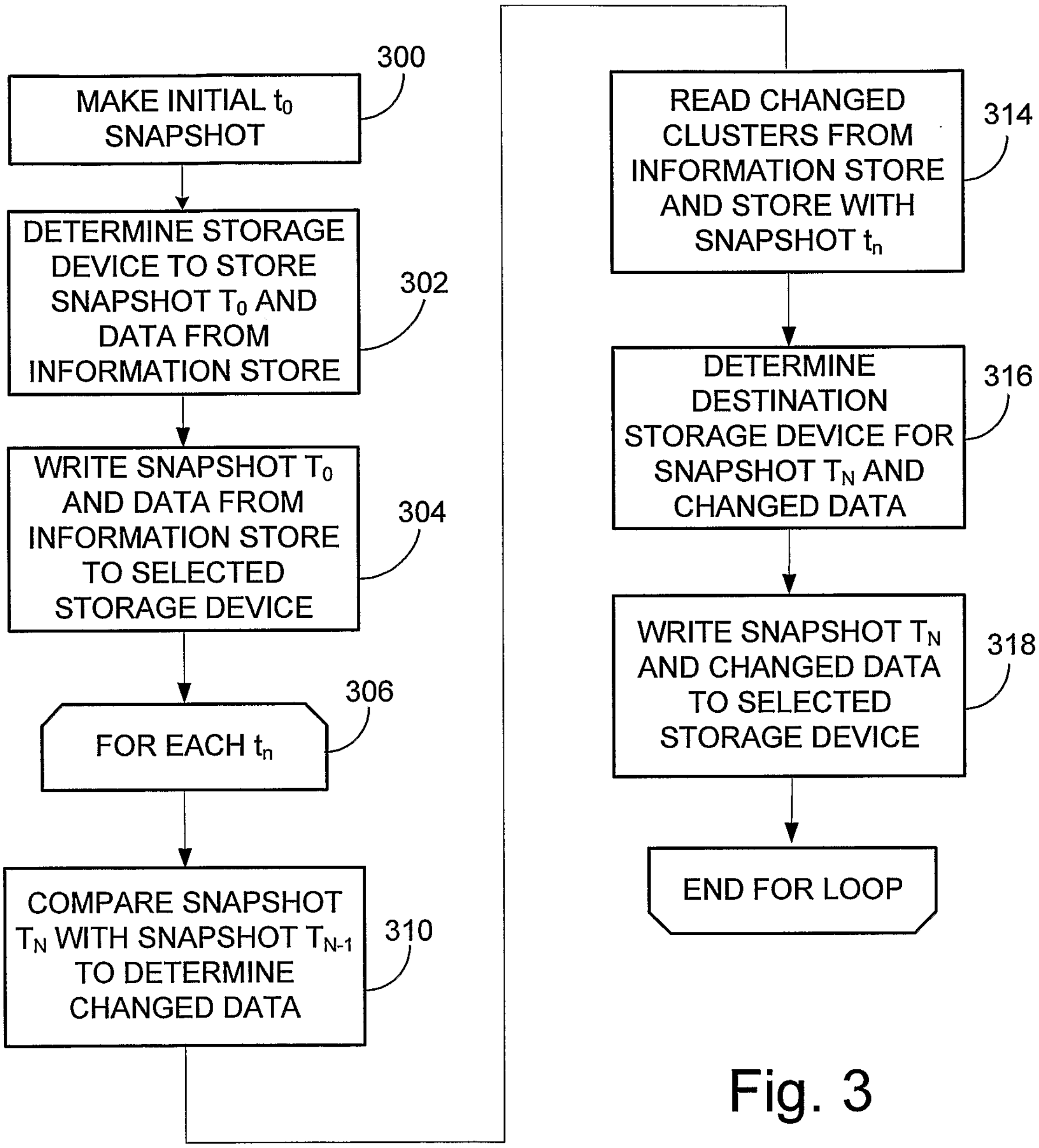
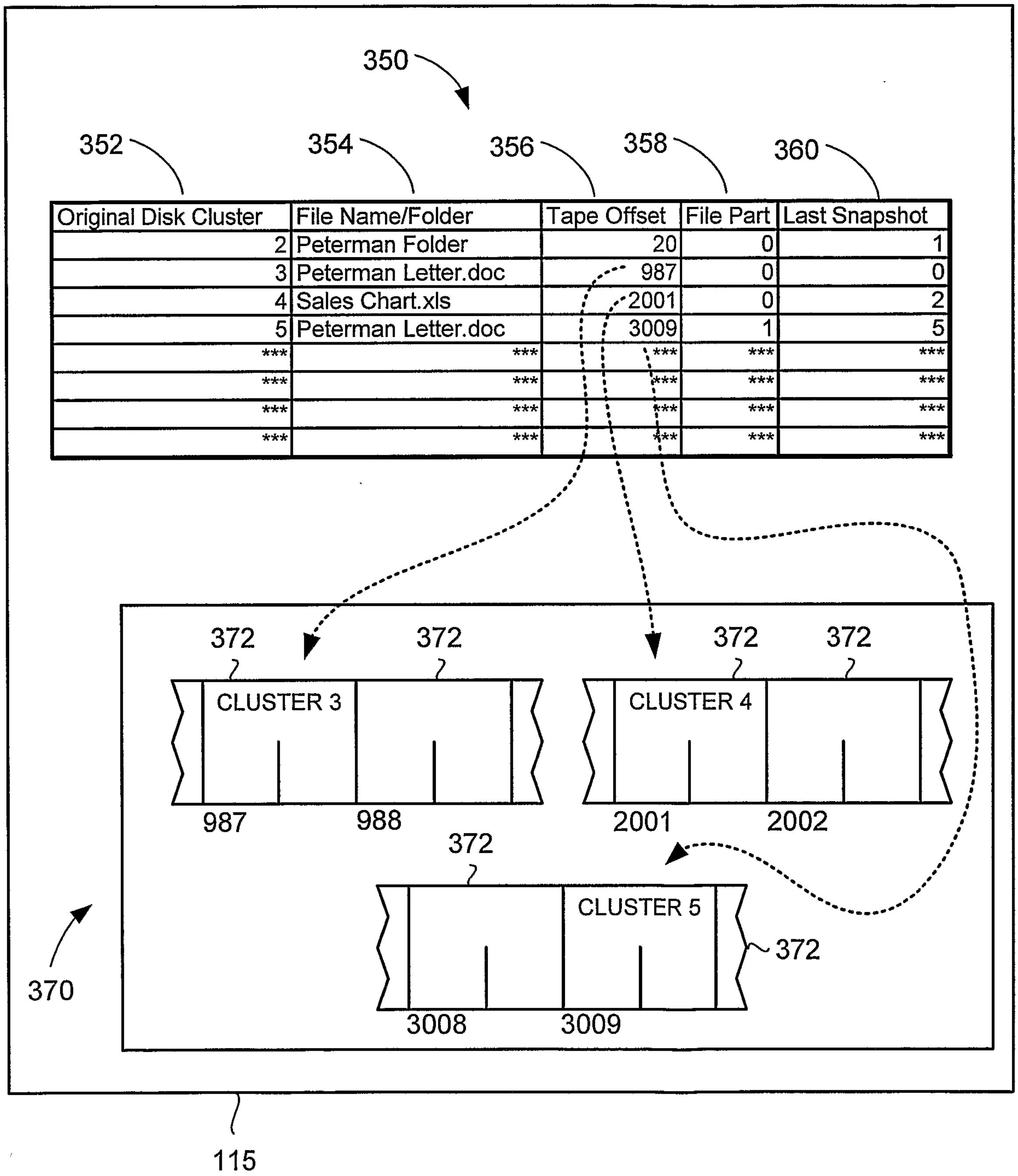

MAKE INITIAL t0 SNAPSHOT
300
DETERMINE STORAGE DEVICE TO STORE SNAPSHOT T0 AND DATA FROM INFORMATION STORE
302
WRITE SNAPSHOT T0 AND DATA FROM INFORMATION STORE TO SELECTED STORAGE DEVICE
304
FOR EACH tn
306
COMPARE SNAPSHOT TN WITH SNAPSHOT TN-1 TO DETERMINE CHANGED DATA
310
READ CHANGED CLUSTERS FROM INFORMATION STORE AND STORE WITH SNAPSHOT tn
314
DETERMINE DESTINATION STORAGE DEVICE FOR SNAPSHOT TN AND CHANGED DATA
316
WRITE SNAPSHOT TN AND CHANGED DATA TO SELECTED STORAGE DEVICE
318
END FOR LOOP

Fig. 3



Original Disk Cluster	File Name/Folder	Tape Offset	File Part	Last Snapshot
2	Peterman Folder	20	0	1
3	Peterman Letter.doc	987	0	0
4	Sales Chart.xls	2001	0	2
5	Peterman Letter.doc	3009	1	5
***	***	***	***	***
***	***	***	***	***
***	***	***	***	***
***	***	***	***	***

Fig. 4

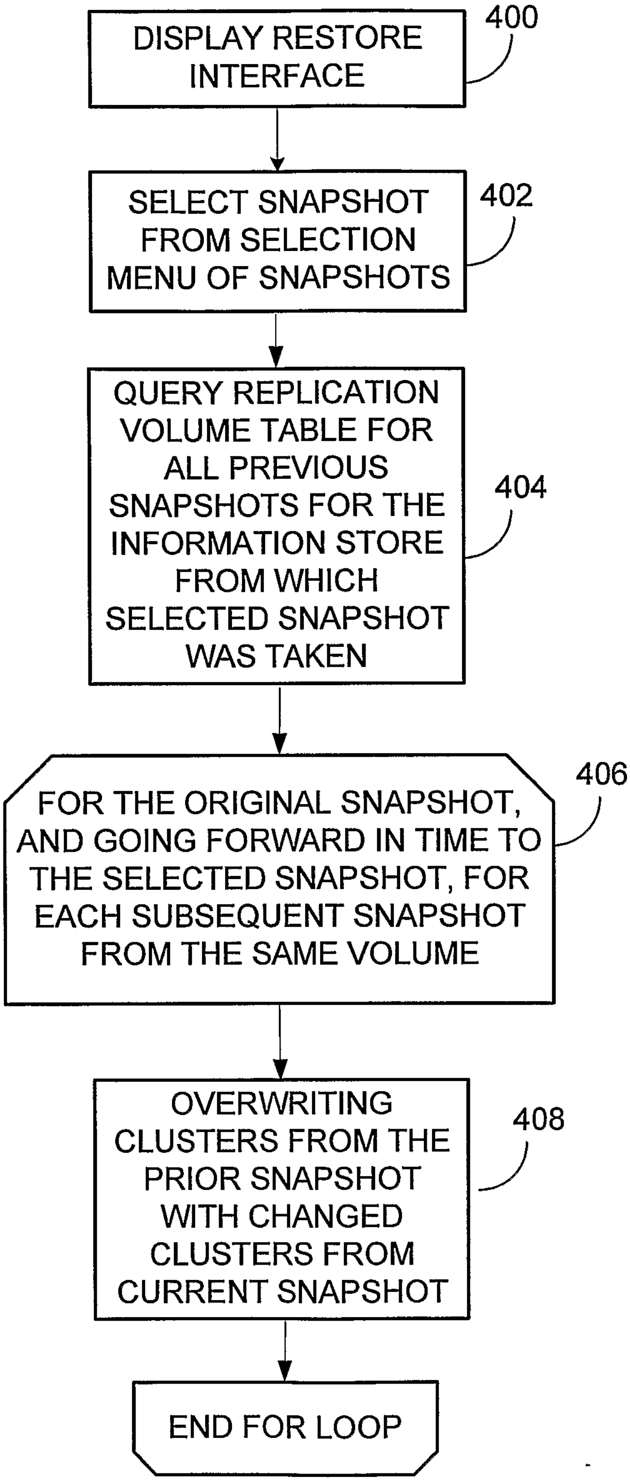

DISPLAY RESTORE INTERFACE
400
SELECT SNAPSHOT FROM SELECTION MENU OF SNAPSHOTS
402
QUERY REPLICATION VOLUME TABLE FOR ALL PREVIOUS SNAPSHOTS FOR THE INFORMATION STORE FROM WHICH SELECTED SNAPSHOT WAS TAKEN
404
FOR THE ORIGINAL SNAPSHOT, AND GOING FORWARD IN TIME TO THE SELECTED SNAPSHOT, FOR EACH SUBSEQUENT SNAPSHOT FROM THE SAME VOLUME
406
OVERWRITING CLUSTERS FROM THE PRIOR SNAPSHOT WITH CHANGED CLUSTERS FROM CURRENT SNAPSHOT
408
END FOR LOOP

Fig. 5

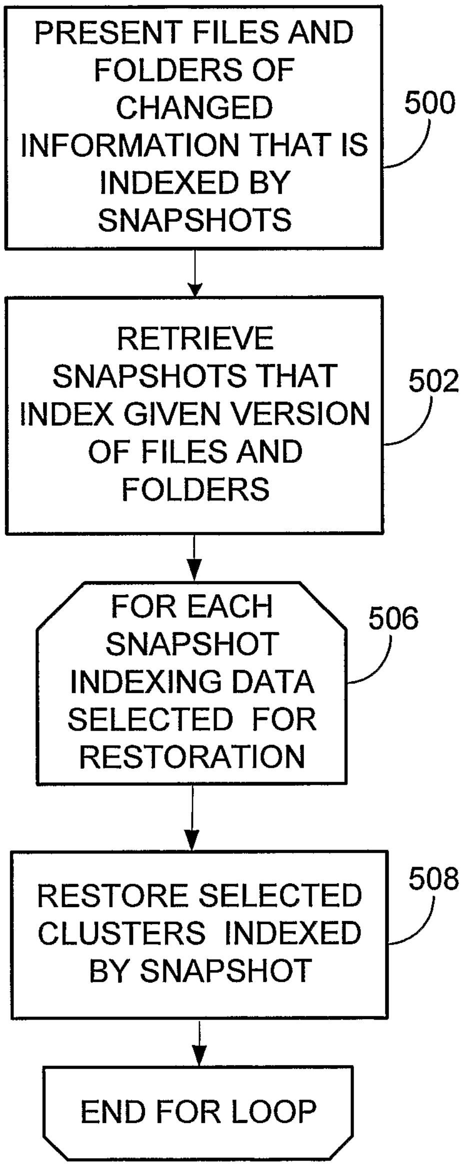

PRESENT FILES AND FOLDERS OF CHANGED INFORMATION THAT IS INDEXED BY SNAPSHOTS
500
RETRIEVE SNAPSHOTS THAT INDEX GIVEN VERSION OF FILES AND FOLDERS
502
FOR EACH SNAPSHOT INDEXING DATA SELECTED FOR RESTORATION
506
RESTORE SELECTED CLUSTERS INDEXED BY SNAPSHOT
508
END FOR LOOP

Fig. 6