



(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2015/026898

(22) International Filing Date:
21 April 2015 (21.04.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
930/CHE/2015 26 February 2015 (26.02.2015) IN

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **GANJIHAL, Anand Andaneppa**; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN). **MONDAL, Santigopal**; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN). **RAVEENDRAN, Anoop**

Kumar; Sy.No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN).

(74) Agents: **ORTEGA, Arthur S.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU,

[Continued on next page]

(54) Title: FILE LEVEL SNAPSHOTS IN A FILE SYSTEM

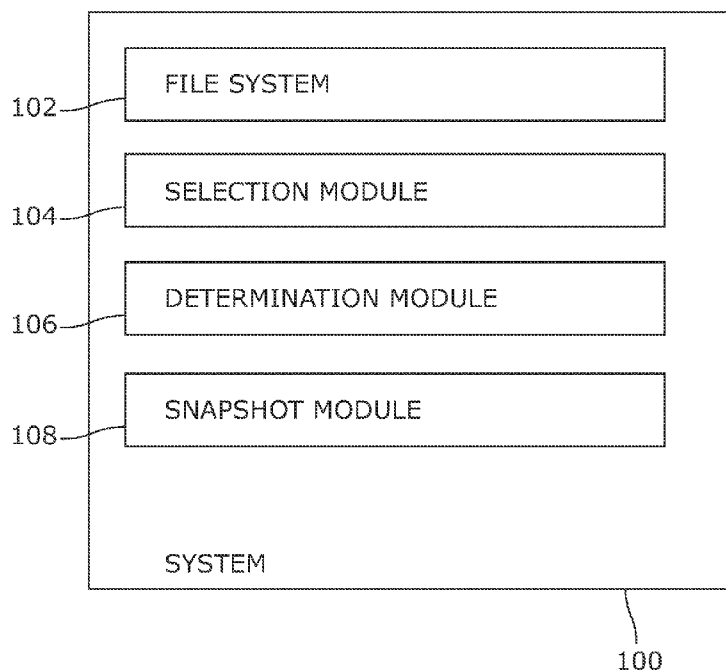


Fig. 1

(57) Abstract: Some examples described herein relate to implementing file level snapshots in a file system. In an example, a file may be selected for generating a snapshot file of the file. A determination may be made whether the snapshot file to be generated is a first snapshot of the file. If the snapshot file to be generated is the first snapshot of the file, a master snapshot file of the file may be created, and block-map table from the file may be copied to the master snapshot file. A snapshot file may be generated, and block-map table, both in the file and the snapshot file, may be replaced with an offset-map table.



TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

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

Published:

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

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

FILE LEVEL SNAPSHOTS IN A FILE SYSTEM

Background

[001] As organizations increasingly rely on digital storage systems to store their business data, effective data protection technologies become more and more relevant. One such technology involves creating a snapshot of a system at a point of time. Snapshots provide a means of data protection, data cloning, and data mining. Snapshot files offer a reliable way of backing-up a storage device having large volumes of data, quick data recovery, better application availability, and lower data management costs.

Brief Description of the Drawings

[002] For a better understanding of the solution, embodiments will now be described, purely by way of example, with reference to the accompanying drawings, in which:

[003] FIG. 1 is a block diagram of an example system for implementing a file level snapshot in a file system;

[004] FIGS. 2A and 2B illustrate generation of a snapshot of a file, by way of an example;

[005] FIGS. 3A and 3B illustrate performing a Redirect-on-Write (ROW) on a snapshot of a file, by way of an example;

[006] FIGS. 4A and 4B illustrate deletion of a snapshot of a file, by way of an example;

[007] FIG. 5 is a block diagram of an example computing environment for implementing a file level snapshot in a file system;

[008] FIG. 6 is a flowchart of an example method of implementing a file level snapshot in a file system; and

[009] FIG. 7 is a block diagram of an example system for implementing a file level snapshot in a file system.

Detailed Description

[0010] Snapshot may be defined as a mechanism to record the state of a storage system at a given point of time. A snapshot creates a point-in-time copy of the data (for example of a file system). A snapshot copy may be used for a variety of purposes, for example, to restore a failed storage system.

[0011] A snapshot may be created using different methodologies. One of the approaches for creating a snapshot of data is called “Copy-on-write”. In copy-on-write, a snapshot of a storage volume is created in pre-designated storage space. When an application writes to a block, the contents of the original block that is to be overwritten are moved to a new location in the snapshot file. Thus, during copy-on-write, the original data block is moved to the snapshot storage volume, prior to the write onto the block. A subsequent read request is directed to the original volume. If the request relates to changed data, it is directed to blocks in the snapshot file. On the other hand, if the request pertains to unchanged data, it is served by the original volume. A copy-on-write snapshot thus involves three steps: “read, rewrite and write”, since the previously written block of data first has to be read, re-written to another block of storage and then the new data written where the other data previously resided.

[0012] In another snapshot creation approach called as “Redirect-on-write”, a new write to the original data is redirected to another location earmarked for snapshot. In other words, in redirect-on-write, the original data is “frozen” and

any write changes to the original data i.e. “changed data” is written elsewhere on the disk. This eliminates the three step “read, rewrite and write” process associated with copy-on-write and reduces snapshots to just one “write” step.

[0013] The present disclosure describes various examples for implementing a file level snapshot in a file system. In an example, a file may be selected for generating a snapshot file of the file. Upon selection, a determination may be made whether the snapshot file to be generated is a first snapshot of the file. If it is determined that the snapshot file to be generated is the first snapshot of the file, a master snapshot file of the file may be created, and the block-map table in the file may be copied to the master snapshot file. Further to such copying, the page reference count in block-map table of the master snapshot file may be set to two. A snapshot file of the file may then be generated, and the block-map table, both in the file and the snapshot file, may be replaced with an offset-map table. The offset-map table includes information related to mapping of file pages between the file and the master snapshot file, and between the snapshot file and the master snapshot file, respectively.

[0014] The present disclosure does not depend on maintenance of a huge reference count per volume and per block (Logical Block Number), which may be required by some of the existing implementations. Considering volume sizes are increasing to the tune of tens of terabytes, maintaining a reference count per volume and per LBN causes a huge burden on a storage system since it results into a huge metadata file. Further, such file may also weaken a file system by turning into a single point of failure. The proposed solution discounts reduces the need for a huge per volume per block reference count file thereby decreasing disk usage and the risk of a single point of failure.

[0015] FIG. 1 is a block diagram of an example system 100 for implementing a file level snapshot in a file system. In an example, system 100 may represent any type of computing system capable of reading machine-executable instructions. Examples of computing device 100 may include, without

limitation, a server, a desktop computer, a notebook computer, a tablet computer, a thin client, a mobile device, a personal digital assistant (PDA), a phablet, and the like. In an example, system 100 may be a file server system or file storage system.

[0016] In another example, system 100 may be a data storage device or medium. System 100 may be a primary storage device such as, but not limited to, random access memory (RAM), read only memory (ROM), processor cache, or another type of dynamic storage device that may store information and machine-readable instructions that may be executed by a processor. For example, Synchronous DRAM (SDRAM), Double Data Rate (DDR), Rambus DRAM (RDRAM), Rambus RAM, etc. System 100 may be a secondary storage device such as, but not limited to, a floppy disk, a hard disk, a CD-ROM, a DVD, a pen drive, a flash memory (e.g. USB flash drives or keys), a paper tape, an Iomega Zip drive, and the like. System 100 may be a tertiary storage device such as, but not limited to, a tape library, a magnetic tape drive, an optical jukebox, and the like. In another example, system 100 may be a Direct Attached Storage (DAS) device, a Network Attached Storage (NAS) device, a Redundant Array of Inexpensive Disks (RAID), a data archival storage system, or a combination of these devices.

[0017] In the example of FIG. 1, system 100 may include a file system, a selection system, a determination module, and a snapshot module. The term “module” may refer to a software component (machine readable instructions), a hardware component or a combination thereof. A module may include, by way of example, components, such as software components, processes, tasks, co-routines, functions, attributes, procedures, drivers, firmware, data, databases, data structures, Application Specific Integrated Circuits (ASIC) and other computing devices. A module may reside on a volatile or non-volatile storage medium and configured to interact with a processor of a computing device (e.g. 100).

[0018] In general, file system 102 may be used for storage and retrieval of data from system 100. In an example, a computer file or “file” is the basic component of file system. Each piece of data on a storage device may be called a “file”. A file may contain data, such as text files, image files, video files, and the like, or it may be an executable file or program. Files in file system 102 may be organized by storing related files in a directory or sub-directory. A directory or sub-directory is also a file. The term “directory” (or “file directory”), as used herein, may include a file that contains references (for example, names) to other files. Thus, a directory may be considered as a container for files.

[0019] In an example, an inode may be associated with each file of the file system 102. An inode is a data structure, which is used to represent a file system object (for example, directory, file, etc.). Each file or directory may be associated with an inode, which is identified by an integer number (i.e. an inode number). An inode may store the attributes and disk block location(s) of a file system object's data. In other words, an inode may store information about data blocks associated with a file or directory, or it may point to a data block map that points to the data blocks. In an instance, an “inode block map” may hold page to disk block mapping for a file or directory. In other words, an inode block map may provide mapping information between a page of a file (or directory) and a Logical Block Number (LBN) on a storage system.

[0020] In an example, a file in the file system 102 may be represented by using multiple file names that may map to the same inode number. Any of these file names may be used to identify the inode number of the file. These file names may be called as “links” (or pointers) to the file. In an instance, for each inode in the file system 102, a “link count” in the inode may be used to track the number of files that may contain a name-inode number mapping for that inode. If the link count in an inode is zero, it means that no file points to the inode. If the link count in an inode is one, it means that the inode has only one

name-inode number mapping. Likewise, if the inode has two name-inode number maps, its link count is considered to be two.

[0021] Selection module 104 may select a file for generating a snapshot file (or “snapshot”) of the file. In an example, the selection module may select the file from file system 102. File system 102 may be a local file system or a scale-out file system such as a shared file system or a network file system. Examples of a shared file system may include a Storage Area Network (SAN) file system or a cluster file system. Examples of a network file system may include a distributed file system or a distributed parallel file system. Some non-limiting examples of file systems that may be used on storage device (example, 102) may include FAT (FAT12, FAT16, FAT32), NTFS, HFS and HFS+, HPFS, UFS, ext2, ext3, and ext4. In an example, file system 102 is an online file system.

[0022] In an example, file system 102 may be associated with a volume, which is, generally, a single accessible storage area on a single partition of a storage medium, such as a hard disk, optical disc, NAND flash memory, etc. A volume may be referred to by using a Logical Unit Number (LUN). A LUN may refer to an entire physical disk, or a subset of a physical disk or disk volume. The physical disk may be an entire single disk drive, a subset of a single disk drive, or a disk volume from a RAID controller. LUNs are used to identify individual or collections of physical disk devices for address by a protocol associated with a SCSI, iSCSI, Fibre Channel (FC).

[0023] A volume may be divided into blocks, which is a sequence of bits or bytes. In an instance, file system 102 may use blocks to store file and directories. A file or directory may be stored over several blocks that may be located at various places on a volume. In other words, a file or directory may be spread over different physical areas of a storage medium.

[0024] Once selection module 104 selects a file for creating a snapshot of the file, determination module 106 may determine if the snapshot file to be generated is a first snapshot of the file. In other words, determination module 106 may determine whether any previous snapshot(s) of the file exist. If no previous snapshots of the file are found in the file system, and the snapshot file to be generated is going to be the first snapshot of the file, snapshot module 108 may create a master snapshot file of the file. The master snapshot file of the file is a point-in-time copy of data in the file. Once a master snapshot of the file is created, snapshot module 108 may copy block-map table information from the file to the master snapshot file. The block-map table in the file includes a page-to-block mapping for all the pages in the file. Snapshot module 108 may copy this table from the file to the master snapshot file of the file. Snapshot module 108 may also set page reference count in block-map table of the master snapshot file to two. Thus, the master snapshot file may include a page-to-block mapping for all the pages in the file, and a reference count for each of the extent maps in the file.

[0025] Once the master snapshot file of the file is created, snapshot module 108 may generate a first snapshot file (i.e. snapshot) of the file. The first snapshot file of the file is a point-in-time copy of data in the file. Snapshot module 108 may replace the block-map table in the file with an offset-map table. The offset-map table may include information related to mapping of file pages between the file and the master snapshot file. Snapshot module 108 may also replace the block-map table in the first snapshot file with an offset-map table. In this case, the offset-map table may include information related to mapping of file pages between the snapshot file and the master snapshot file. In an example, the initial page-to-page mapping in the offset-map table may be 1-to-1 i.e. data on page 0 of the file or the first snapshot file may be on page 0 of the master snapshot file. Snapshot module 108 may add file inode number of the master snapshot file to the file and the first snapshot file inodes. Snapshot module 108 may also set link count of the master snapshot file to two.

[0026] FIGS. 2A and 2B illustrate generation of a snapshot of a file, by way of an example. Referring to FIG. 2A, in an example, a file "foo.txt" may be selected (for example, by selection module 104) for creating a snapshot file of the file. An extent map 204 of the file "foo.txt", which provides a mapping between a page of the file "foo.txt" and a Logical Block Number (LBN) on a storage volume 1 (206), is also illustrated in FIG. 2A. In an example, in response to a determination (for example, by determination module 106) that the snapshot file to be generated is a first snapshot of the file, selection module 106 may generate a master snapshot file ("Snap.master File") of the file "foo.txt". Snapshot module 108 may then copy block-map table 210 from the file "foo.txt" to the master snapshot file ("Snap.master File"), and set page reference count in block-map table of the master snapshot file to two. Snapshot module 108 may generate a first snapshot file of the file. Once the first snapshot file is generated, snapshot module 108 may replace block-map table, both in the file and the snapshot file, with an offset-map table 212 and 214. The offset-map table may include information related to mapping of file pages between the file and the master snapshot file (212), and between the snapshot file and the master snapshot file (214), respectively.

[0027] If the determination module 106 determines that the snapshot file to be generated is not a first snapshot file of the file i.e. a previous snapshot(s) of the file exists, snapshot module may generate another snapshot file of the file. For example, if the determination module determines that a first snapshot file of the file is present in the file system, snapshot module 108 may generate a second snapshot file of the file. Likewise, if the determination module 106 determines that two snapshots files of the file are present in the file system, snapshot module 108 may generate a third snapshot file of the file. In any case, further to generation of another snapshot file of the file, snapshot module 108 may copy offset-map table from the file to the another snapshot file, and update, for all pages that are pointed by the offset-map table in the snapshot file, page reference count in block-map table of master snapshot file of the file by one. Snapshot module 108 may add file inode number of the

master snapshot file to inode of another snapshot file. Snapshot module may also increment link count of the master snapshot file by one.

[0028] Snapshot module 108 may perform a Redirect-on-Write (ROW) on a snapshot file of the file. A Redirect-on-Write (ROW) may be performed on any version of the snapshot. For example, a Redirect-on-Write (ROW) may be carried out on the first snapshot file of the file, the second snapshot of the file, the third snapshot of the file, or on any other version of the file. In an example, to perform a Redirect-on-Write (ROW) on a snapshot file of the file, snapshot module 108 may identify a page of the snapshot file where the modified data may be written. In an instance, such page may be identified through a page ID of the page. Snapshot module 108 may then determine inode number of the master snapshot file from inode number of the snapshot file, and disk block location and page reference count of the page from block-map table of the master snapshot file. If it is determined that the page reference count is more than one (i.e. the page in the master snapshot file is referenced by other snapshot files), snapshot module 108 may append a new page to the master snapshot file, and update block-map table of the master snapshot file to reflect the location of the new page. Snapshot module 108 may then proceed to write the "modified data" to newly allocated page in the master snapshot file. Subsequent to data write on the newly allocated page, snapshot module 108 may update offset-map table of the snapshot file to point to the new page in the master snapshot file, and decrement reference count of old page in the block-map table of the master snapshot file to indicate that the snapshot file is not sharing the old page anymore. Snapshot module 108 may also set the page reference count of the new page to one. On the other hand, if it is determined that the page reference is one (i.e. the page in the master snapshot file is not referenced by other snapshot files), snapshot module 108 may allow the snapshot file to overwrite data on to master snapshot file page which is being referenced by the file page of the snapshot file. In other words, in such case, the snapshot file may be allowed to overwrite the master snapshot file page pointed by snapshot page ID.

[0029] FIGS. 3A and 3B illustrate performing a Redirect-on-Write (ROW) on a snapshot of a file, by way of an example. In an instance, to perform a Redirect-on-Write (ROW) on a snapshot file of a file, snapshot module 108 may identify a page of the snapshot file where the modified data may be written. Referring to FIG. 3A, in an example, a snapshot file (i.e. snapshot - 1) of the file "foo.txt" may be selected for performing a ROW. An offset map 304 of the snapshot file "snapshot - 1", which provides a page-to-page mapping between the snapshot file and the master snapshot file ("Snap.master File"), is illustrated in FIG. 3A. FIG. 3A also illustrates block-map table 308 of the master snapshot file ("Snap.master File") of the file "foo.txt". The block-map table includes a page-to-block mapping for all the pages in the file along with a page reference count. Referring to FIG. 3B, let's consider a scenario, by way an example, where data on page 1 of the file "foo.txt" may need to be modified (for instance, by an application or a user). In such case, snapshot module may perform a Redirect-on-Write by determining page reference count for page 1 from block-map table of the snapshot master file ("Snap.master File"). If it is determined that the page reference count is more than one (in the present example, the page reference count for page 1 is two (310), as indicated in FIG. 3A), snapshot module 108 may append a new page to the master snapshot file ("Snap.master File"), and update block-map table of the master snapshot file ("Snap.master File") to reflect the location of the new page. In the present case, a new page is allocated from volume 1 (306) at LBN 'n' and appended to the master snapshot file as page 4 (312). File offset-map table of "foo.txt" is updated to point 1 to page 4 of the master snapshot file and modified data is written to page 4 of the master snapshot file. Page 1 of the "snapshot -1" file continues to point to page 1 of the master snapshot file. Snapshot module 108 may decrement reference count of old page in the block-map table of the master snapshot file to indicate that the snapshot file is not sharing the old page anymore. Snapshot module 108 may also set the page reference count of old page to one (314).

[0030] Snapshot module 108 may delete a snapshot file of the file. Snapshot module 108 may delete any version of a snapshot. For example, snapshot module 108 may delete the first snapshot file of the file, the second snapshot file of the file, the third snapshot file of the file, and the like. In an example, to delete a snapshot file of the file, snapshot module 108 may walk the offset-map table in the snapshot file inode. Snapshot module 108 may determine, for a page ID of the snapshot file that is present in the offset-map table of the snapshot file, the page ID of the master snapshot file corresponding to the page ID of the snapshot file. Snapshot module 108 may use the page ID of the master snapshot file to determine, from block-map table of the master snapshot file, page reference count for the page ID of the snapshot file. If the page reference count is more than one (i.e. if the page in the master snapshot file is referenced by other snapshot files), snapshot module 108 may decrement the page reference count by one in the block-map table of the master snapshot file. On the other hand, if the page reference is one (i.e. if the page in the master snapshot file is not referenced by other snapshot files), snapshot module may delete storage allocated to the page ID of the master snapshot file, and mark the disk block address to -1 (to represent a hole). Snapshot module 108 may perform the aforementioned determination for each page in the snapshot file. In an example, snapshot module 108 may determine the page reference counts for all pages in the master snapshot file. If it is determined that page reference counts for all pages in the master snapshot file is one (i.e. none of the pages in the master snapshot file is referenced by other snapshot files), snapshot module 108 may delete the snapshot file, and decrement the link count in the master snapshot file by one. In an example, snapshot module 108 may determine the link count of the master snapshot file. If it is determined that link count of the master snapshot file is zero, snapshot module 108 may delete the master snapshot file.

[0031] FIGS. 4A and 4B illustrate deletion of a snapshot of a file, by way of an example. FIG. 4A illustrates offset map (406) of a file "foo.txt" along with offset maps (402 and 404) of two snapshot files of the file "foo.txt" (i.e. snapshot - 1

and snapshot - 2). A block map (408) of master snapshot file of the file "foo.txt" is also illustrated. In an example, snapshot module 108 may delete snapshot - 1 file of the file "foo.txt". This is illustrated in FIG. 4B. Further to deletion of snapshot - 1 file, corresponding references in the master snapshot file of "foo.txt" may be deleted by reducing the respective page reference count in the block-map table (408). If the page reference count becomes zero, "-1" is placed in place of the LBN in the block map table of the master snapshot file to indicate a hole (412).

[0032] FIG. 5 is a block diagram of an example computing environment for implementing a file level snapshot in a file system. Computing environment 500 may include nodes 502, 504, 506, 508, and 510, and a server 512. As used herein, a "node" may be a computing device (i.e. includes at least one processor), a storage device, a network device, or any combination thereof. The number of nodes 502, 504, 506, 508 and 510, and server 512 shown in FIG. 5 is for the purpose of illustration only and their number may vary in other implementations. In some examples, computing environment 500 may represent a file storage system wherein nodes 502, 504, 506, 508, and 510 may serve as file storage nodes. In an instance, said file storage system may be a scale-out file system.

[0033] Nodes 502, 504, 506, 508, and 510 may each be a computing device such as a desktop computer, a notebook computer, a tablet computer, a mobile phone, personal digital assistant (PDA), a server, and the like. In another example, one or more of the nodes 502, 504, 506, 508, and 510 may each be a storage device. In an example, such storage device may be a primary storage device such as, but not limited to, random access memory (RAM), read only memory (ROM), processor cache, or another type of dynamic storage device that may store information and machine-readable instructions that may be executed by a processor. For example, Synchronous DRAM (SDRAM), Double Data Rate (DDR), Rambus DRAM (RDRAM), Rambus RAM, etc. In another example, such storage device may be a secondary

storage device such as, but not limited to, a floppy disk, a hard disk, a CD-ROM, a DVD, a pen drive, a flash memory (e.g. USB flash drives or keys), a paper tape, an Iomega Zip drive, and the like. In another example, such storage device may be a tertiary storage device such as, but not limited to, a tape library, an optical jukebox, and the like. In another example, the storage device may be a Direct Attached Storage (DAS) device, a Network Attached Storage (NAS) device, a tape drive, a magnetic tape drive, a data archival storage system, or a combination of these devices.

[0034] Nodes 502, 504, 506, 508, and 510, may communicate with each other and server 512, for example, via a computer network 524. Computer network 524 may be a wireless or wired network. Computer network 524 may include, for example, a Local Area Network (LAN), a Wireless Local Area Network (WAN), a Metropolitan Area Network (MAN), a Storage Area Network (SAN), a Campus Area Network (CAN), or the like. Further, computer network 524 may be a public network (for example, the Internet) or a private network (for example, an intranet).

[0035] In an example, server 512 may be analogous to system 100, in which like reference numerals correspond to the same or similar, though perhaps not identical, components. For the sake of brevity, components or reference numerals of FIG. 5 having a same or similarly described function in FIG. 1 are not being described in detail in connection with FIG. 5. Said components or reference numerals may be considered alike.

[0036] In an example, server 512 may include a file system 102, a selection system 104, a determination module 106, and a snapshot module 108. In an example, one or more nodes (502, 504, 506, 508, and 510), may include a portion of the file system 502. In an instance, server 512 may be a file server. In another instance, server 512 may be a storage server that may provide one or more nodes block level access to stored data.

[0037] FIG. 6 is a flowchart of an example method 600 for implementing a file level snapshot in a file system. The method 600, which is described below, may at least partially be executed on a computing device 100 of FIG. 1 or server 512 of FIG. 5. However, other computing devices may be used as well. At block 602, a file may be selected for generating a snapshot file of the file. At block 604, a determination may be made whether the snapshot file to be generated is a first snapshot of the file. At block 606, a master snapshot file of the file may be created, in response to the determination that the snapshot file to be generated is the first snapshot of the file. At block 608, block-map table may be copied from the file to the master snapshot file. At block 610, the snapshot file may be generated. At block 612, block-map table, both in the file and the snapshot file, may be replaced with an offset-map table. The offset-map table includes information related to mapping of file pages between the file and the master snapshot file, and between the snapshot file and the master snapshot file, respectively.

[0038] FIG. 7 is a block diagram of an example system 700 for implementing a file level snapshot in a file system. System 700 includes a processor 702 and a machine-readable storage medium 704 communicatively coupled through a system bus. In an example, system 700 may be analogous to system 100 of FIG. 1 or server 512 of FIG. 5. Processor 702 may be any type of Central Processing Unit (CPU), microprocessor, or processing logic that interprets and executes machine-readable instructions stored in machine-readable storage medium 704. Machine-readable storage medium 704 may be a random access memory (RAM) or another type of dynamic storage device that may store information and machine-readable instructions that may be executed by processor 702. For example, machine-readable storage medium 704 may be Synchronous DRAM (SDRAM), Double Data Rate (DDR), Rambus DRAM (RDRAM), Rambus RAM, etc. or a storage memory media such as a floppy disk, a hard disk, a CD-ROM, a DVD, a pen drive, and the like. In an example, machine-readable storage medium 704 may be a non-transitory machine-readable medium. Machine-readable storage medium 704

may store instructions 706, 708, 710, 712, 714, and 716. In an example, instructions 706 may be executed by processor 702 to select a file for generating a snapshot file of the file. Instructions 708 may be executed by processor 702 to determine if the snapshot file to be generated is a first snapshot of the file. Instructions 710 may be executed by processor 702 to create a master snapshot file of the file, if the snapshot file to be generated is the first snapshot of the file. Instructions 712 may be executed by processor 702 to copy block-map table from the file to the master snapshot file. Instructions 714 may be executed by processor 702 to generate the snapshot file. Instructions 716 may be executed by processor 702 to replace block-map table, both in the file and the snapshot file, with an offset-map table, wherein the offset-map table includes information related to mapping of file pages between the file and the master snapshot file, and between the snapshot file and the master snapshot file, respectively.

[0039] For the purpose of simplicity of explanation, the example method of FIG. 6 is shown as executing serially, however it is to be understood and appreciated that the present and other examples are not limited by the illustrated order. The example systems of FIGS. 1, 5 and 7, and method of FIG. 6 may be implemented in the form of a computer program product including computer-executable instructions, such as program code, which may be run on any suitable computing device in conjunction with a suitable operating system (for example, Microsoft Windows, Linux, UNIX, and the like). Embodiments within the scope of the present solution may also include program products comprising non-transitory computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer. By way of example, such computer-readable media can comprise RAM, ROM, EPROM, EEPROM, CD-ROM, magnetic disk storage or other storage devices, or any other medium which can be used to carry or store desired program code in the form of computer-executable instructions and which can be accessed by a general

purpose or special purpose computer. The computer readable instructions can also be accessed from memory and executed by a processor.

[0040] It may be noted that the above-described examples of the present solution is for the purpose of illustration only. Although the solution has been described in conjunction with a specific embodiment thereof, numerous modifications may be possible without materially departing from the teachings and advantages of the subject matter described herein. Other substitutions, modifications and changes may be made without departing from the spirit of the present solution. All of the features disclosed in this specification (including any accompanying claims, abstract and drawings), and/or all of the steps of any method or process so disclosed, may be combined in any combination, except combinations where at least some of such features and/or steps are mutually exclusive.

Claims:

1. A method of implementing file level snapshots in a file system, comprising:
 - selecting a file for generating a snapshot file of the file;
 - determining if the snapshot file to be generated is a first snapshot of the file;
 - if the snapshot file to be generated is the first snapshot of the file, creating a master snapshot file of the file;
 - copying block-map table from the file to the master snapshot file;
 - generating the snapshot file; and
 - replacing block-map table, both in the file and the snapshot file, with an offset-map table, wherein the offset-map table includes information related to mapping of file pages between the file and the master snapshot file, and between the snapshot file and the master snapshot file, respectively.
2. The method of claim 1, further comprising:
 - if the snapshot file to be generated is not the first snapshot of the file:
 - generating the snapshot file;
 - copying offset-map table from the file to the snapshot file; and
 - updating, for all pages that are pointed by the offset-map table in the snapshot file, page reference count in block-map table of master snapshot file of the file by one.
3. The method of claim 2, further comprising adding inode number of the master snapshot file to an inode table in the snapshot file.
4. The method of claim 2, further comprising incrementing link count of the master snapshot file by one.
5. The method of claim 1, further comprising performing a Redirect-on-Write (ROW) on the snapshot file, wherein the performing comprises:
 - identifying a page of the snapshot file for writing modified data;

determining inode number of the master snapshot file from inode number of the snapshot file;

determining disk block location and page reference count of the page, from block-map table of the master snapshot file;

if the page reference count is more than one, appending a new page to the snapshot master file, and updating block-map table of the snapshot master file to reflect location of the new page;

writing the modified data to the new page in the snapshot master file; and

updating offset-map table of the snapshot file to point to the new page in the master snapshot file.

6. The method of claim 5, further comprising:

if the page reference count is one:

identifying file page of the master snapshot file that is referenced by the page of the snapshot file; and

overwriting data on the file page of the master snapshot file with the modified data,

7. A system to implement file level snapshots in a file system, comprising:

a selection module to select a file for generating a snapshot file of the file;

a determination module to determine if the snapshot file to be generated is a first snapshot of the file; and

a snapshot module to:

create a master snapshot file of the file, if the snapshot file to be generated is the first snapshot of the file;

copy block-map table from the file to the master snapshot file;

set page reference count in block-map table of the master snapshot file to two;

generate the snapshot file; and

replace block-map table, both in the file and the snapshot file, with an offset-map table, wherein the offset-map table includes information related to

mapping of file pages between the file and the master snapshot file, and between the snapshot file and the master snapshot file, respectively.

8. The system of claim 7, wherein the snapshot module to migrate the snapshot master file from a current storage volume to another storage volume on a storage system.

9. The system of claim 8, wherein the snapshot module to update the block-map table of the master snapshot file to reflect the migration of the snapshot master file from the current storage volume to the another storage volume on the storage system.

10. The system of claim 8, wherein the snapshot module to add inode of the master snapshot file both to inode of the file and inode of the snapshot file.

11. The system of claim 8, wherein the snapshot module to set link count of the master snapshot file to two.

12. A non-transitory machine-readable storage medium comprising instructions to implement file level snapshots in a file system, the instructions executable by a processor to:

- select a file to generate a snapshot file of the file;

- determine if the snapshot file to be generated is a first snapshot of the file;

and

- create a master snapshot file of the file, if the snapshot file to be generated is the first snapshot of the file;

- copy block-map table from the file to the master snapshot file;

- generate the snapshot file; and

- replace block-map table, both in the file and the snapshot file, with an offset-map table, wherein the offset-map table includes information related to mapping of file pages between the file and the master snapshot file, and between the snapshot file and the master snapshot file, respectively.

13. The storage medium of claim 12, further comprising instructions to delete the snapshot file, wherein the instructions to delete comprise instructions to:

- determine, for a page ID of the snapshot file that is present in the offset-map table of the snapshot file, page ID of the snapshot master file corresponding to the page ID of the snapshot file;

- use the page ID of the snapshot master file to determine, from block-map table of the snapshot master file, page reference count for the page ID of the snapshot file;

- if the page reference count is more than one, decrement the page reference count by one in the block-map table of the master snapshot file; and

- if the page reference count is one, delete storage allocated to the page ID of the master snapshot file.

14. The storage medium of claim 13, wherein the instructions to delete further comprise instructions to:

- determine if page reference count for each page in the master snapshot file is one; and

- delete the snapshot file, if the page reference count for each page in the master snapshot file is one.

15. The storage medium of claim 13, wherein the instructions to delete further comprise instructions to:

- decrement link count of the master snapshot file by one; and

- in response to a determination that the link count of the master snapshot file is zero, delete the master snapshot file.

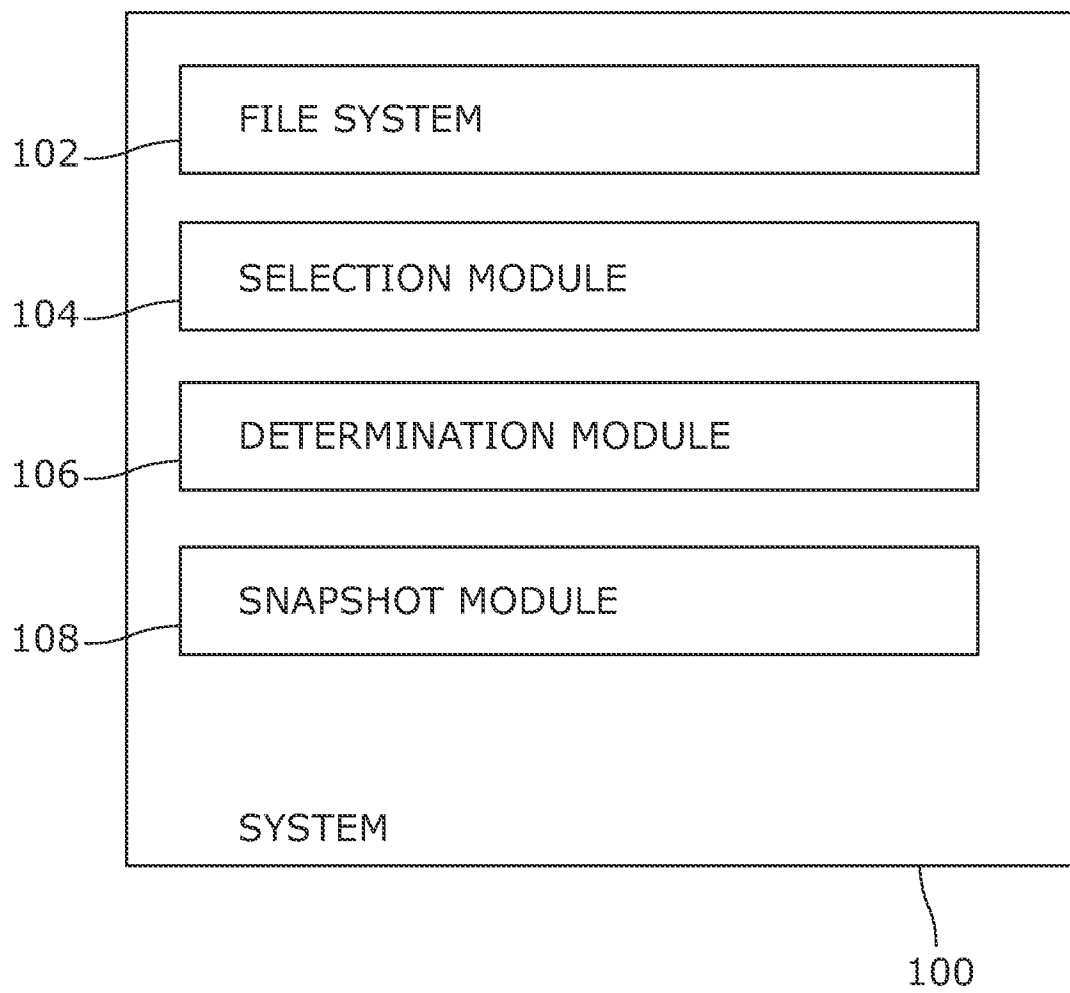


Fig. 1

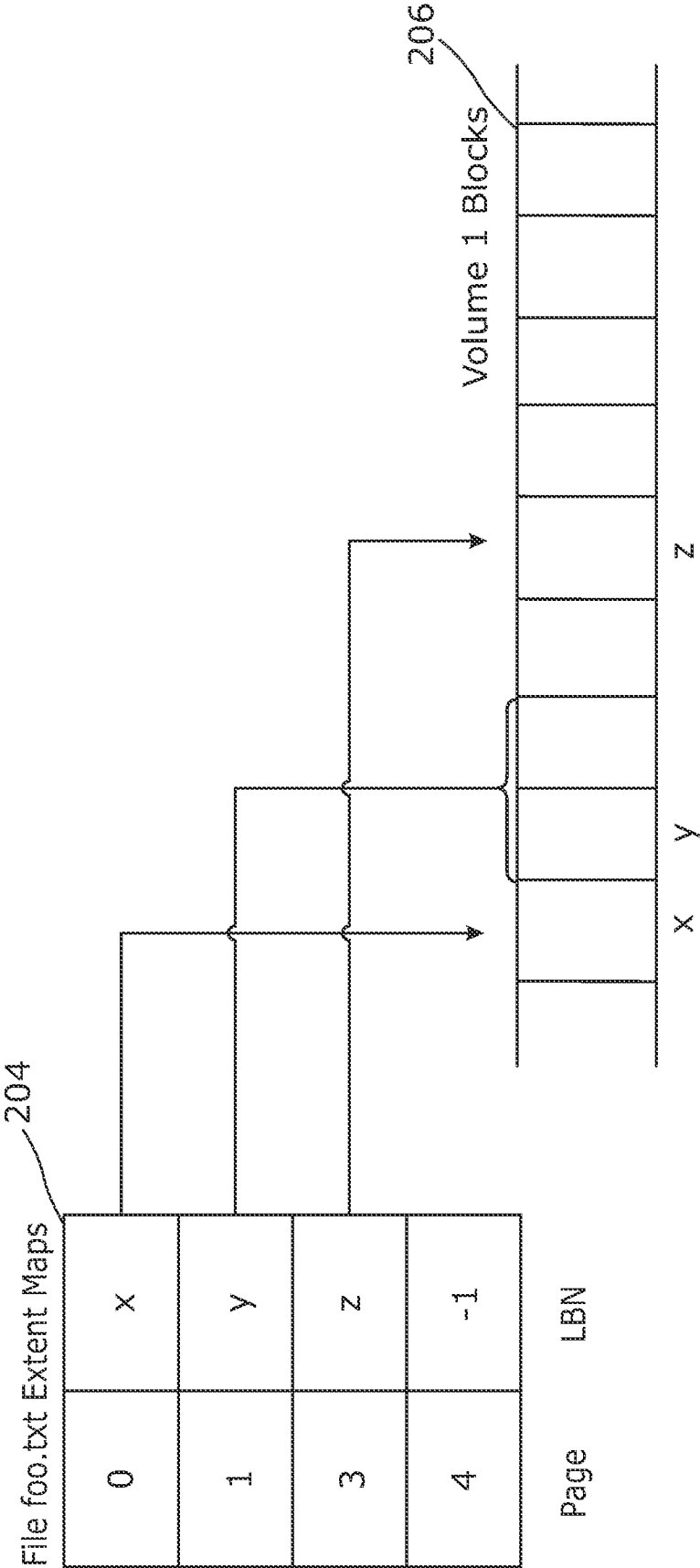
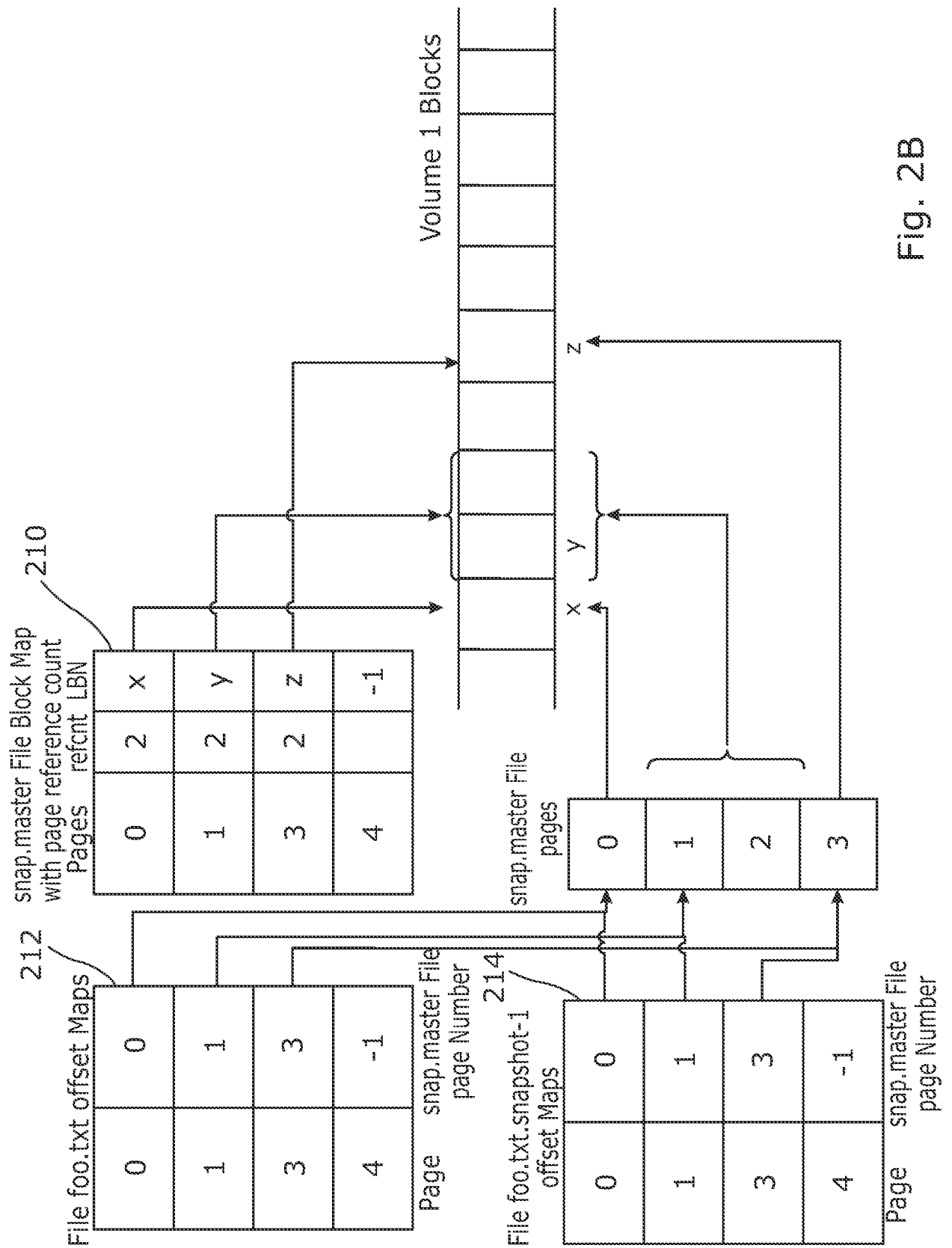


Fig. 2A



3204

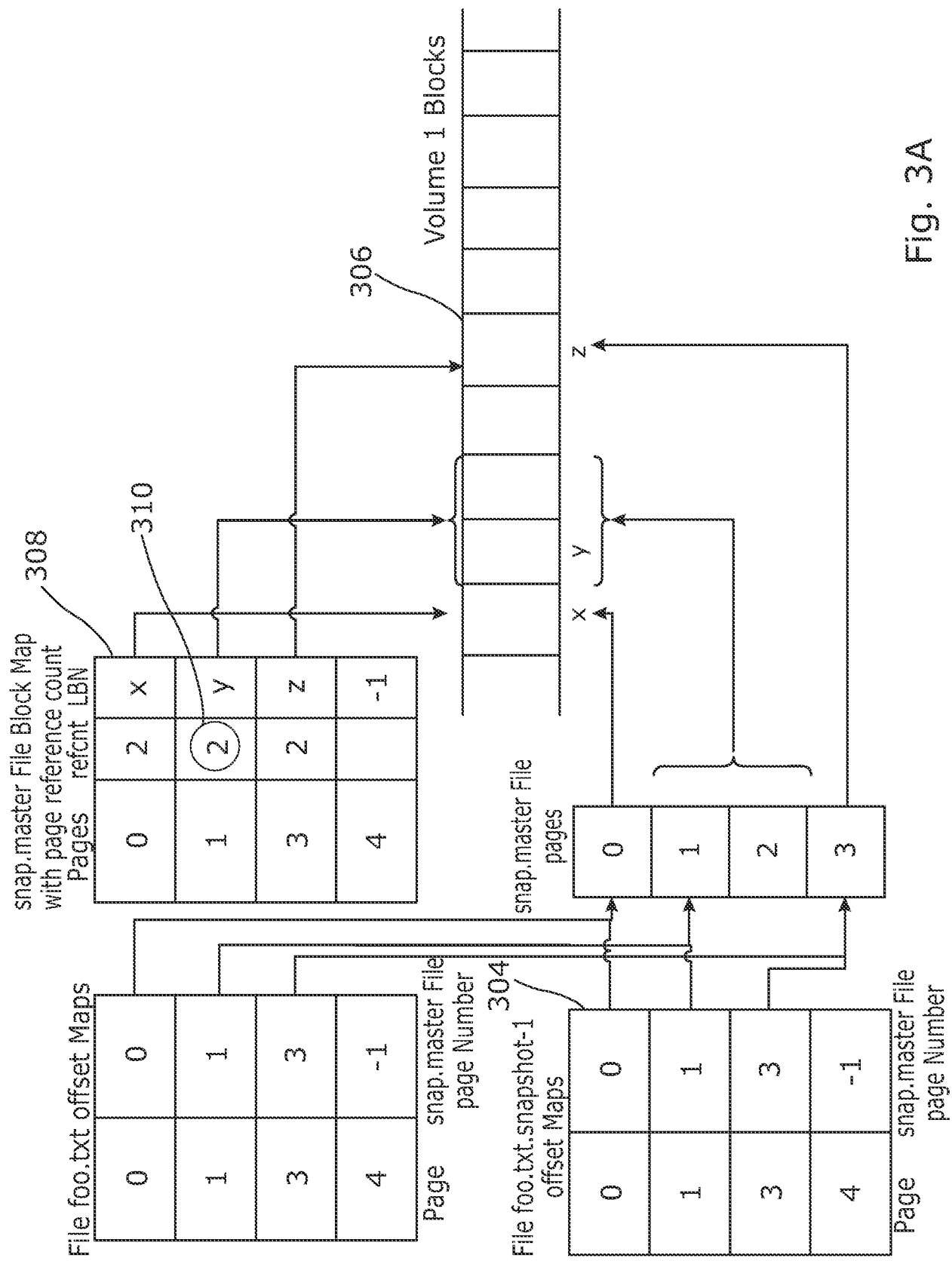
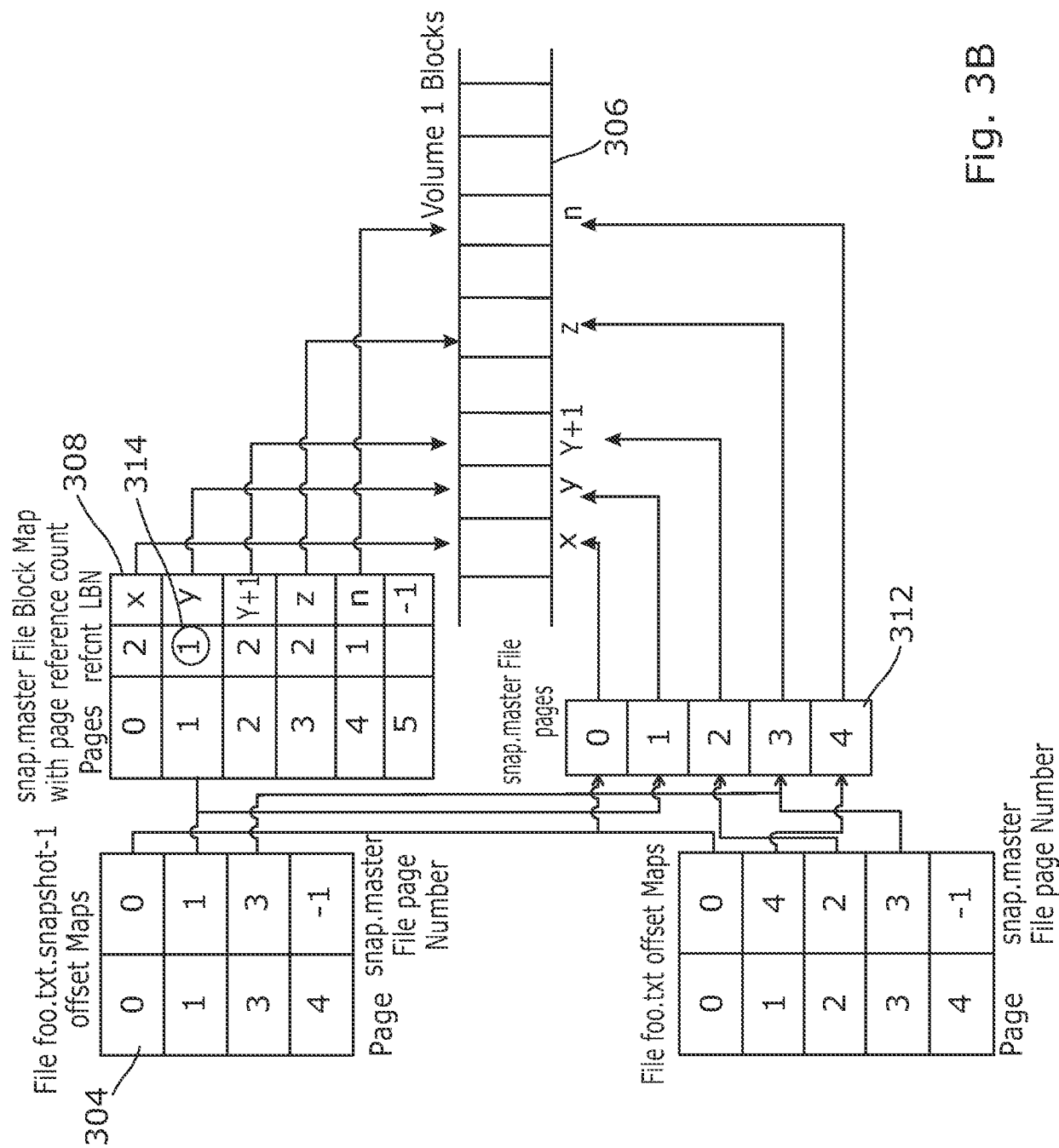


Fig. 3A



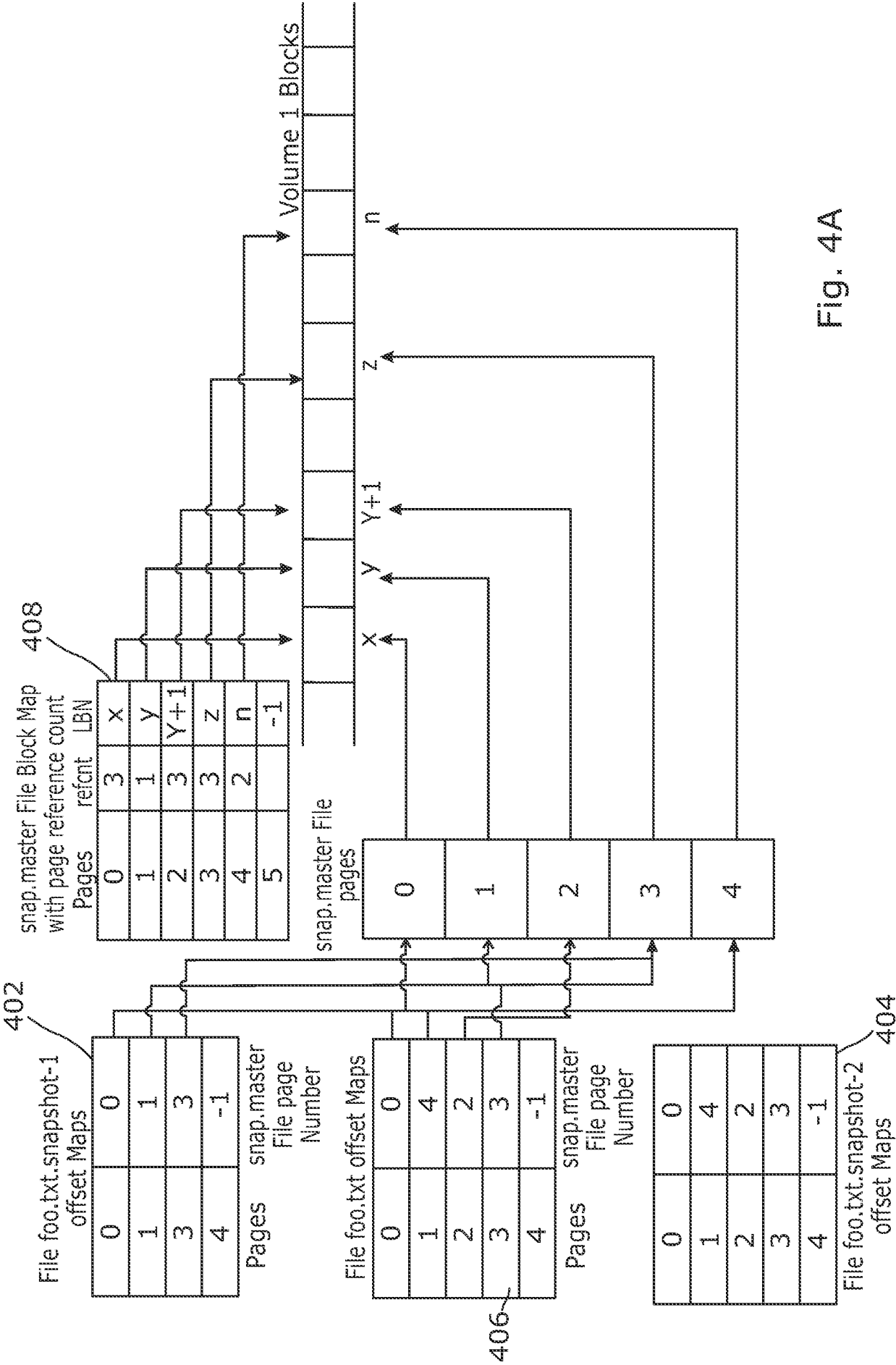


Fig. 4A

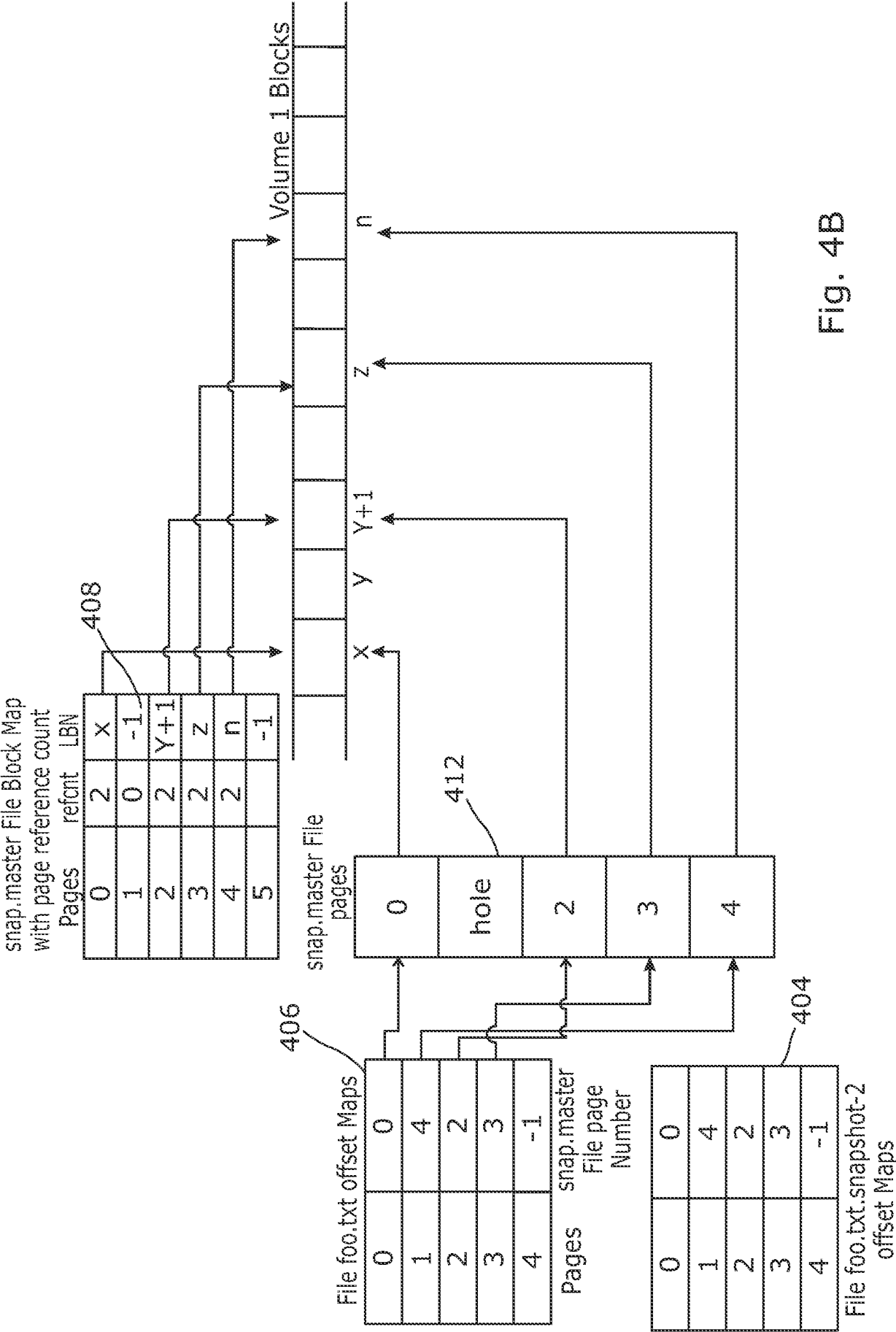
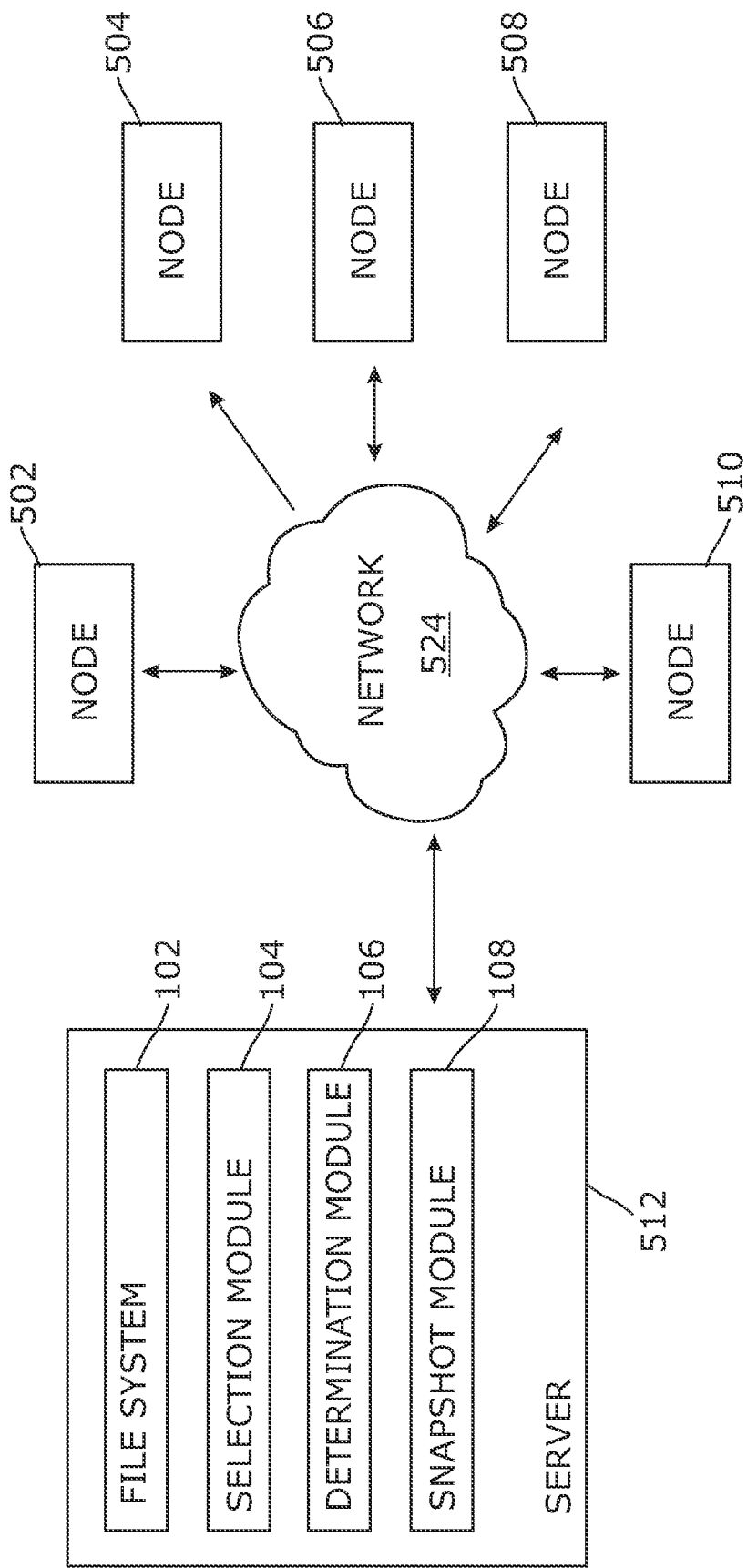


Fig. 4B



500

Fig. 5

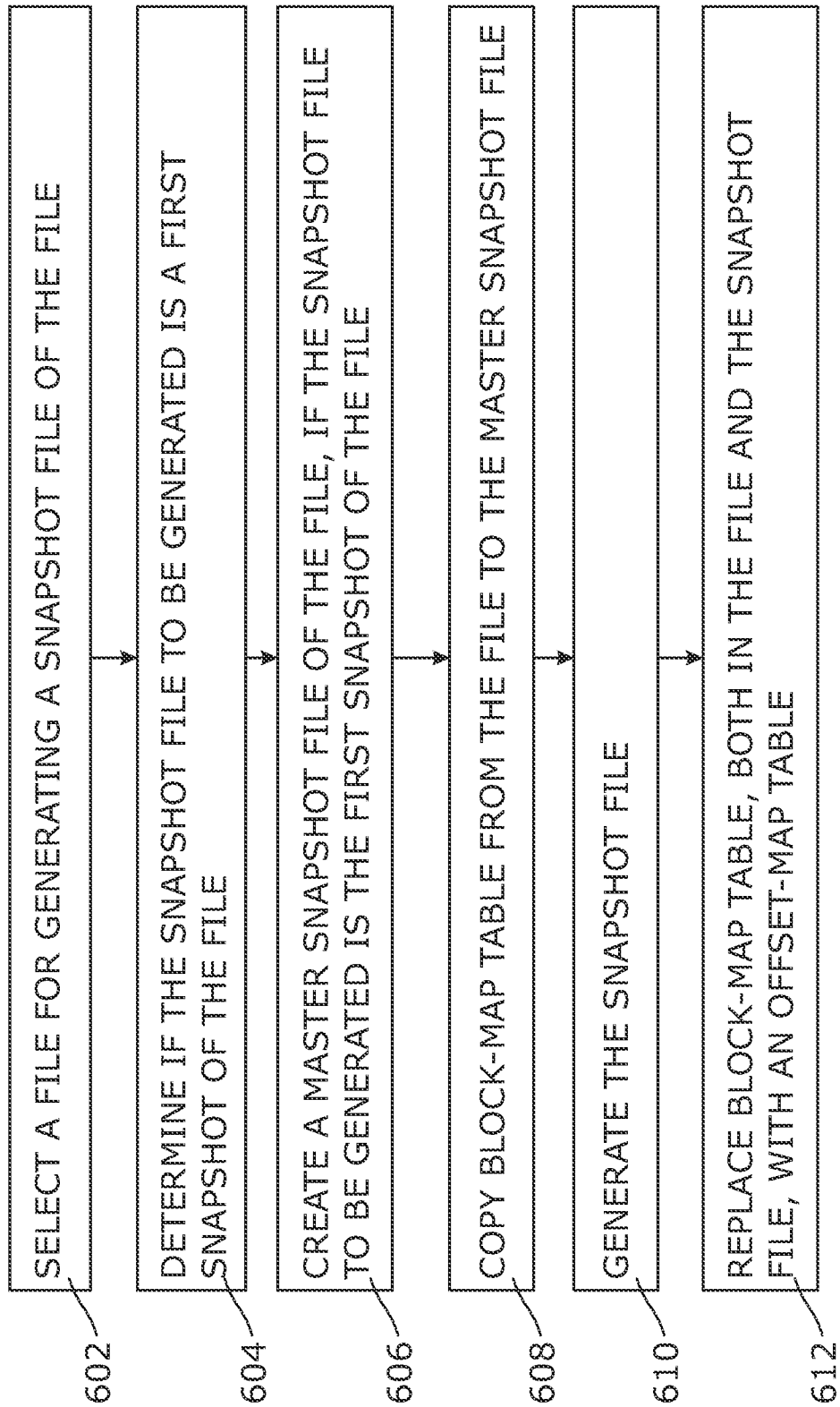
600

Fig. 6

10/10

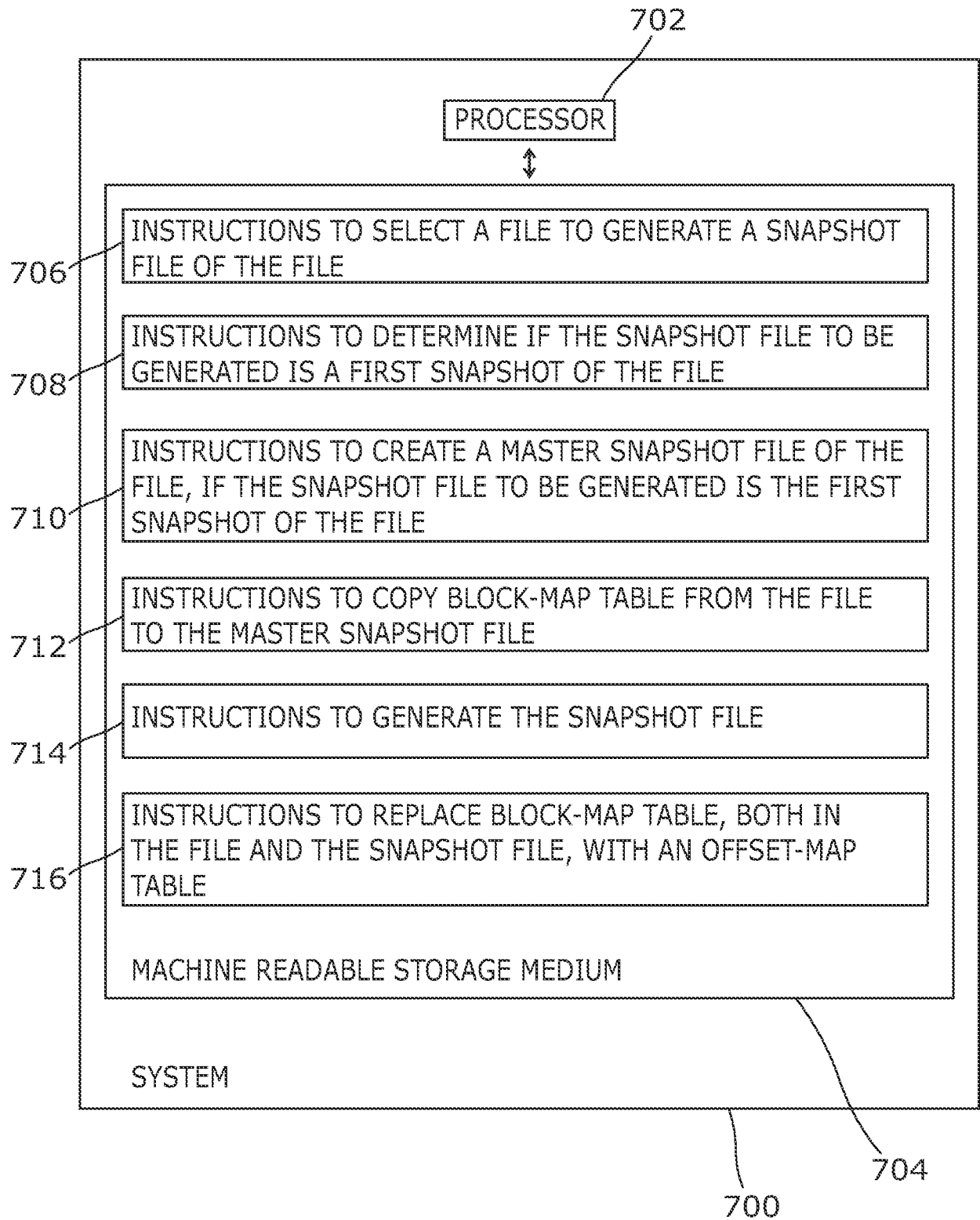


Fig. 7

A. CLASSIFICATION OF SUBJECT MATTER**G06F 17/30(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 17/30; G06F 12/02; G06F 12/16; G06F 17/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: file system, first snapshot, master file snapshot, block-map table, replace, offset-map table, and similar terms.**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 8,954,383 B1 (SRINIVASA RAO VEMPATI et al.) 10 February 2015 See column 12, line 51 - column 13, line 13; column 13, line 63 - column 14, line 23; column 14, line 62 - column 16, line 41; and figures 5, 7, and 9-11.	1-15
A	US 2009-0070535 A1 (JANET ELIZABETH ADKINS et al.) 12 March 2009 See paragraphs [0032]-[0040] and figure 3.	1-15
A	US 7,991,973 B2 (JOEL JACOBSON et al.) 02 August 2011 See column 10, lines 1-33; and figure 4.	1-15
A	US 8,412,688 B1 (PHILIPPE ARMANGAU et al.) 02 April 2013 See column 9, lines 52-62; column 20, line 39 - column 21, line 64; and figures 5 and 30-32.	1-15
A	US 2014-0250076 A1 (COMVAULT SYSTEMS, INC.) 04 September 2014 See paragraphs [0017] and [0044]-[0047]; claim 1; and figures 7-8.	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

25 November 2015 (25.11.2015)

Date of mailing of the international search report

25 November 2015 (25.11.2015)

Name and mailing address of the ISA/KR

International Application Division
Korean Intellectual Property Office
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan City, 35208,
Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/026898

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 8954383 B1	10/02/2015	None	
US 2009-0070535 A1	12/03/2009	US 7958325 B2	07/06/2011
US 7991973 B2	02/08/2011	US 2009-0276593 A1	05/11/2009
US 8412688 B1	02/04/2013	None	
US 2014-0250076 A1	04/09/2014	US 2010-0076932 A1	25/03/2010
		US 8725688 B2	13/05/2014