



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

(21) International Application Number:

PCT/US2015/025969

(22) International Filing Date:

15 April 2015 (15.04.2015)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

719/CHE/2015 13 February 2015 (13.02.2015) IN

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

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

Published:

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

(54) Title: CONSISTENCY CHECK ON NAMESPACE OF AN ONLINE FILE SYSTEM

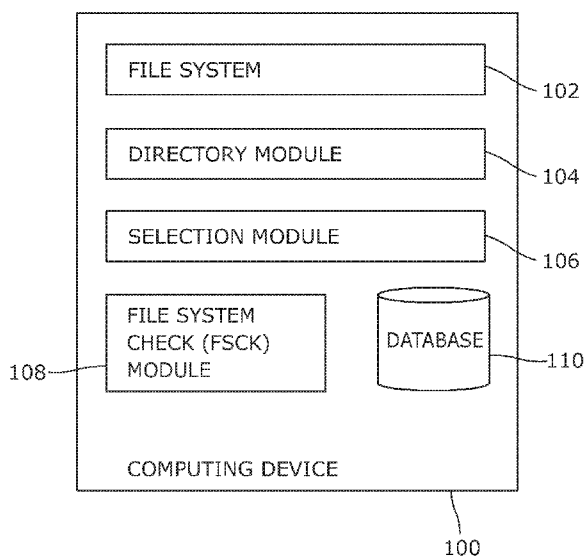


Fig. 1

(57) Abstract: Some examples described herein relate to performing a consistency check on namespace of an online file system. In an example, an inode table may be selected from an online file system. A directory inode may be selected in the inode table, and a consistency check may be performed on a dentry of a directory that maps to the selected directory inode. In response to identifying an inconsistency with the dentry, consequent to the consistency check, the dentry may be recorded in a database.



CONSISTENCY CHECK ON NAMESPACE OF AN ONLINE FILE SYSTEM

Background

[001] A file system is a means for organizing data on a storage device. In other words, a file system may be used to control how data is stored and retrieved from a storage system. Since data is typically stored in the form of a file, a file system provides the basic structure for organizing files in a storage device. The file system keeps track of file locations (physical or virtual) on a storage medium.

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 computing device for performing a consistency check on namespace of an online file system;

[004] FIG. 2 is a block diagram of an example computing environment for performing a consistency check on namespace of an online file system;

[005] FIG. 3 is a flowchart of an example method of performing a consistency check on namespace of an online file system; and

[006] FIG. 4 is a block diagram of an example system for performing a consistency check on namespace of an online file system.

Detailed Description

[007] A file system is an integral part of an operating system. It provides the underlying structure that a computing device uses to organize data on a storage medium. A computer file or “file” is the basic component of a 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.

[008] Since a large amount of data generated these days is stored in computer files, a modern file system should be able to expand to support a namespace that may grow to millions or billions of files. To ensure consistent data, a file system consistency check may be required to be performed at regular intervals. However, in many instances, a file system hosting a large amount of data may need to be online and serve user requests while consistency checks are carried out. In such case, an online consistency check performed on the namespace of a file system may take a few hours to multiple days depending on the size of the file system and number of file system objects. Further, during an online consistency check, the structure of a namespace tree may change at any time (for example, if a sub-tree is moved). Further still, in case any disconnect is identified in a namespace tree, it may become difficult to correct a sub-tree unless the disconnection is corrected. This may make a file system consistency check a stop-resume-stop model leading to an exponential increase in the time taken to complete the consistency checking since in order to fix any disconnect in the namespace, the entire inode table may need to be walked. In such scenarios, it may be challenging to verify the name space of a file system. In order to ensure that a name space consistency check is not affected by changes occurring in the structure of the name space, it may be desirable to have a mechanism of checking the name space consistency without walking the namespace tree of a file system.

[0009] The present disclosure describes various examples for performing a consistency check on namespace of an online file system. In an example, an inode table may be selected from an online file system. Further to selection of the inode table, a directory inode may be selected in the inode table, and a consistency check may be performed on a dentry of a directory that maps to the selected directory inode. In response to identifying an inconsistency with the dentry, consequent to the consistency check, the dentry may be recorded in a database.

[0010] FIG. 1 is a block diagram of an example computing device 100 for performing a consistency check on namespace of an online file system. Computing device 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, computing device 100 may be a file storage system.

[0011] In an example, computing device 100 may be a data storage device or medium. Computing device 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. Computing device 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. Computing device 100 may be a tertiary storage device such as, but not limited to, a tape library, an optical jukebox, and the like. In another example, storage device 102 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.

[0012] In the example of FIG. 1, computing device 100 may include a file system 102, a directory module 104, a selection module 106, a file system check (fsck) module 108, and a database 110. 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).

[0013] In general, file system 102 may be used for storage and retrieval of data from computing device 100. 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.

[0014] 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. Thus, each inode in the file system is a unique number, and the file system may locate the contents of a file by

its inode number. An inode may also store information related to file ownership and file access permissions.

[0015] As mentioned earlier, files in file system 102 may be organized by storing related files in a directory or sub-directory. Thus, a directory may include one or more files. In an example, each directory may be represented in the form of an inode table that maps directory and file names to inode numbers. In other words, a directory may store filenames and their respective inode numbers (i.e. (filename, inode number)), in an on-disk page. Each such page may be called as directory page. Directory page, thus, is a container which may hold a finite set of filenames and respective inode mapping i.e. {filename, inode number}. Apart from {filename, inode number}, a directory page may hold other metadata relevant to an inode or a file. The inodes of file system 102 may be placed in several tables (i.e. inode tables).

[0016] 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 directories that may contain a name-inode number mapping for that inode. If the link count in an inode is zero, it means that no directory 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.

[0017] In an example, a parent directory in the file system 102 may be represented by “. .” (dot dot), which maps to the inode of the parent directory. Likewise, a directory in the file system may be represented by “.” (dot), which maps to the inode of the directory. An empty directory in the file system 102 has a link count of two (i.e. 1+1): one link count for the parent directory (i.e. dot dot), and the other link count directory itself (i.e. dot). Each

subdirectory that is added to a directory increases the link count of the directory by one. For example, if a directory includes 4 subdirectories, its link count would be 6 (i.e. $2+4=6$).

[0018] 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.

[0019] Directory module 104 may select an inode table from an online file system (for example, 102). As mentioned earlier, a file system (for example, 102) may include one or more inode tables. In an example, the directory module 104 may randomly select an inode table from the file system 102 for performing a consistency check on the namespace of the file system 102. Likewise, the directory module 104 may select each inode table of the file system 102 for carrying out a consistency check on the namespace of the file system 102.

[0020] Selection module 106 may select a directory inode from the selected inode table. In other words, once the directory module 104 selects an inode table from the file system 102, the selection module 106 may randomly select a directory inode from the selected inode table for performing a consistency check on the namespace of the file system 102. Likewise, the selection module 106 may select each directory inode of a selected inode table.

[0021] As mentioned earlier, the file system 102 may include one or more directories. Each file name-inode number {i.e. filename, inode number}

mapping in a directory inode may be referred as directory entry or “dentry”. A file name in a directory entry may refer to name of a directory, a subdirectory, a file, or any other file object. For example, a mapping {*. (dot)*: 2}, where “*. (dot)*” is the file name of the parent directory and “2” is the inode number of the parent directory, may represent a dentry. In another example, a mapping {home : 222}, where “home” is file name of the home directory and “222” is the inode number of the home directory, may represent another dentry in the inode table. In a further example, a mapping {“rm”: 444}, where “rm” is file name of a file and “444” is the inode number of the file, may represent a dentry.

[0022] In an instance, a dentry may be construed a specific component in a file path. For instance, in a path “/abc/def”, both “abc” and “def” may represent files, wherein “abc” is a directory file and “def” is an ordinary file. In this case, “/”, “abc”, and “def” represent dentry objects. In other words, all components in a file path represent dentry objects (or dentries).

[0023] File system check (fsck) module 108 may be used for checking and repairing file system inconsistencies on the computing device 100. A file system (for example, 102) may become inconsistent, for example, due to power failure, a nonstandard shutdown, hardware failure, etc. This may cause inconsistencies and mismatched information relating to data blocks, free blocks, inodes, pointers, etc. in a file system. Fsck module 108 may perform a consistency check on a dentry of a directory that maps to the directory inode selected by the selection module 106. In an example, such consistency check may involve determining whether the selected dentry includes an orphan inode i.e. an inode without any dentry pointing to it. In an instance, such determination may be made by verifying whether the link between the dentry and inode is valid. This may be carried out by verifying whether the back pointer from the inode is correctly pointing to the directory inode. In an instance, the fsck module may perform aforementioned

consistency check for all dentries in the directory that maps to the selected directory inode.

[0024] In another example, a consistency check by fsck module 108 may involve determining whether the selected dentry is a dangling entry i.e. a directory entry that does not include a valid inode pointer, or includes a wrong pointer. In an instance, the fsck module may perform aforementioned consistency check for all dentries in the directory that maps to the selected directory inode. After the dentry-inode relation is verified, the fsck module may increment the link count maintained in-memory for the selected inode. At the end of checking all the inodes in the table, the fsck module 108 may build the in-memory link and apply the link counts on disk. After the inode table check is complete, if any inode having link count 0 is found, fsck module 108 may add the inode to the orphan inode list.

[0025] In an example, fsck module 108 may perform a consistency check related to a dentry along with a file metadata consistency check that may be carried out in parallel. In an instance, fsck module 108 may use a background thread, which is used for carrying out a file metadata consistency check, to perform a consistency check on a selected inode table. As part of the metadata consistency check, if an inode of type directory is encountered, the background thread may offload the thread to a directory checker thread after checking for the inode consistency. The directory checker thread may come from a thread pool that may verify all the dentries in the inode tables and verify the link to the inodes.

[0026] In an example, further to a consistency check, if the fsck module 108 identifies an inconsistency with the dentry, it may record such "corrupted" dentry in a database (for example, 110). Fsck module 108 may log a corrupted dentry along with its directory inode, to the database. In an instance, such database may be processed at the end of a consistency check, and dentries recorded therein may be corrected. For instance, if an

orphan inode (i.e. an inode with a link count of zero) is identified in the database, fsck module 108 may add a new dentry to the directory page that points to a back pointer pointing to the directory inode of the orphan inode. In another instance, for a corrupted dentry in the database, a corresponding inode with a link count of "0" (zero) may be identified in the database. If the parent directory inode number in the inode matches with the directory where an inconsistency is seen in its dentry, the dentry may be corrected to point to that inode and the link count on the inode may be corrected.

[0027] In an example, a connection between two directories in the file system 102 may be verified by setting a flag on the object state structure, which may be maintained to determine the consistency of the object, of an inode. Presence of such flag on a directory entry may indicate that a link is observed for this directory inode from a non-dot dot based dentry. In case the flag is not set for a directory inode, the directory may be added to an orphan inode list. Such directory may be corrected by adding a lost+found entry to the parent directory, which may be obtained through the back pointer found on the inode.

[0028] In an example, further to a consistency check carried out on a dentry of a directory that maps to the selected directory inode, if an inconsistency is identified with the dentry, such dentry may be recorded in database 110, for instance by , by fsck module. The database 110, therefore, may act as a repository of all those dentries that may include an inconsistency. The fsck module may also record the current directory of an inconsistent directory entry in the database 110. In an instance, fsck module 108 may correct the inconsistency with the directory entry that is recorded in the database 110.

[0029] FIG. 2 is a block diagram of an example computing environment 200 for performing a consistency check on namespace of an online file system. Computing environment 200 may include nodes 202, 204, 206, 208, and 210, and a server 212. 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 202, 204, 206, 208 and 210, and server 212 shown in FIG. 2 is for the purpose of illustration only and their number may vary in other implementations. In some examples, computing environment 200 may represent a file storage system wherein nodes 202, 204, 206, 208, and 210 may serve as file storage nodes. In an instance, said file storage system may be a scale-out file system.

[0030] Nodes 202, 204, 206, 208, and 210 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. Nodes 202, 204, 206, 208, and 210 may be a storage device. In an example, the 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, the 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, the 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.

[0031] Nodes 202, 204, 206, 208, and 210, may communicate with each other and server 212, for example, via a computer network 224. Computer network 224 may be a wireless or wired network. Computer network 224 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 224 may be a public network (for example, the Internet) or a private network (for example, an intranet).

[0032] In an example, server 212 may be analogous to computing device 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. 2 having a same or similarly described function in FIG. 1 are not being described in detail in connection with FIG. 2. Said components or reference numerals may be considered alike.

[0033] In an example, server 212 may include a file system 102, a directory module 104, a selection module 106, a file system check (fsck) module 108, and a database 110. In an example, one or more nodes (202, 204, 206, 208, and/or 210), may include a portion of the file system 102. In an instance, file system 102 is an online file system.

[0034] In an example, directory module 104 may select an inode table in each of a plurality of nodes hosting an online file system (for example, 102). For instance, directory module 104 may select an inode table in each of a plurality of nodes, such as 202, 204, and 208. Upon such selection, the selection module may select a directory inode in each of the selected inode tables. The file system check (fsck) module 108 may then perform, in parallel, a consistency check on a dentry in each of a directory that individually maps to each of the selected directory inode. Further to the consistency check, if the fsck module 108 identifies an inconsistency with the dentry in any of the selected directories, the fsck module 108 may log the dentry in a database (for example, 110).

[0035] FIG. 3 is a flowchart of an example method 300 for performing a consistency check on namespace of an online file system. The method 300,

which is described below, may at least partially be executed on a computing device 100 of FIG. 1 or server 212 of FIG. 2. However, other computing devices may be used as well. At block 302, an inode table may be selected (for example, by directory module 104), from an online file system (for example, 102). At block 304, a directory inode may be selected in the selected inode table (for example, by selection module 106). At block 306, a consistency check may be performed on a dentry (for example, by fsck module 108) of a directory that maps to the directory inode. At block 308, consequent to the consistency check by fsck module 108, if an inconsistency is identified with the dentry, fsck module 108 may record the dentry in a database (for example, 110).

[0036] FIG. 4 is a block diagram of an example system 400 for performing a consistency check on namespace of an online file system. System 400 includes a processor 402 and a machine-readable storage medium 404 communicatively coupled through a system bus. In an example, system 400 may be analogous to computing device 100 of FIG. 1 or server 212 of FIG. 2. Processor 402 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 404. Machine-readable storage medium 404 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 402. For example, machine-readable storage medium 404 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 404 may be a non-transitory machine-readable medium. Machine-readable storage medium 404 may store instructions 406, 408, and 410. In an example, instructions 406 may be executed by processor 402 to select a directory inode in an inode table of an online file system. Instructions 406 may be executed by processor 402 to perform a

consistency check on each directory entry of a directory that maps to the directory inode. Instructions 408 may be executed by processor 402 to record a directory entry in a database if consequent to the consistency check, an inconsistency is identified with the directory entry.

[0037] For the purpose of simplicity of explanation, the example method of FIG. 3 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, 2 and 4, and method of FIG. 3 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.

[0038] 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 performing a consistency check on namespace of an online file system, comprising:

selecting an inode table from an online file system;

selecting a directory inode in the inode table;

performing a consistency check on a dentry of a directory that maps to the selected directory inode; and

in response to identifying an inconsistency with the dentry, consequent to the consistency check, recording the dentry in a database.

2. The method of claim 1, wherein the performing comprises verifying whether a link between the dentry and an inode is valid.

3. The method of claim 2, wherein the verifying comprises verifying whether a back pointer from the inode is correctly pointing to directory inode of the dentry.

4. The method of claim 1, wherein the performing comprises determining whether the dentry maps to an invalid or incorrect inode.

5. The method of claim 1, wherein the consistency check on the dentry is performed in parallel to metadata consistency check on the inode table.

6. A system to perform a consistency check on namespace of an online file system, comprising:

a directory module to select an inode table in each of a plurality of nodes hosting an online file system;

a selection module to select a directory inode in each of the selected inode tables; and

a file system check (fsck) module to:

perform, in parallel, a consistency check on a dentry in each of a directory that individually maps to each of the selected directory inode; and

in response to identification of an inconsistency with the dentry, further to the consistency check, logging the dentry in a database.

7. The system of claim 6, wherein the inode table in each of a plurality of nodes is randomly selected.

8. The system of claim 6, wherein the file system is a scale-out file system.

9. The system of claim 6, wherein the dentry is logged along with directory inode of the dentry.

10. The system of claim 6, wherein the consistency check comprises the fsck module to:

- verify whether a link between the dentry and an inode is valid; and
- determine whether the dentry includes an invalid or incorrect inode.

11. A non-transitory machine-readable storage medium comprising instructions to perform a consistency check on namespace of an online file system, the instructions executable by a processor to:

- select a directory inode in an inode table of an online file system;
- perform a consistency check on each directory entry of a directory that maps to the directory inode; and

- in response to identification of an inconsistency with a directory entry, consequent to the consistency check, record the directory entry in a database.

12. The storage medium of claim 11, wherein the inconsistency includes presence of an orphan inode in the inode table.

13. The storage medium of claim 11, wherein the inconsistency includes presence of a dangling directory entry in the directory.

14. The storage medium of claim 11, further comprising instructions to correct the inconsistency with the directory entry that is recorded in the database.

15. The storage medium of claim 11, wherein the instructions to record the directory entry in the database include instructions to record current directory of the directory entry in the database.

1/4

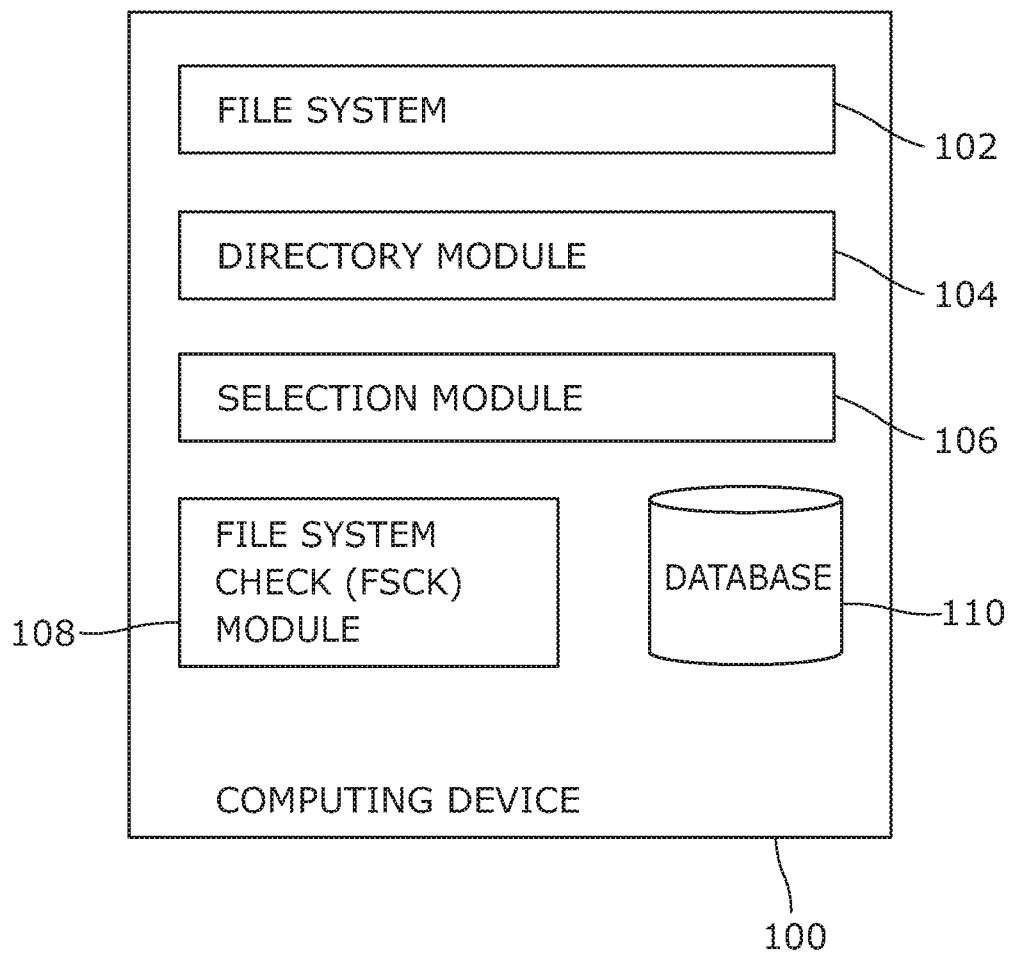


Fig. 1

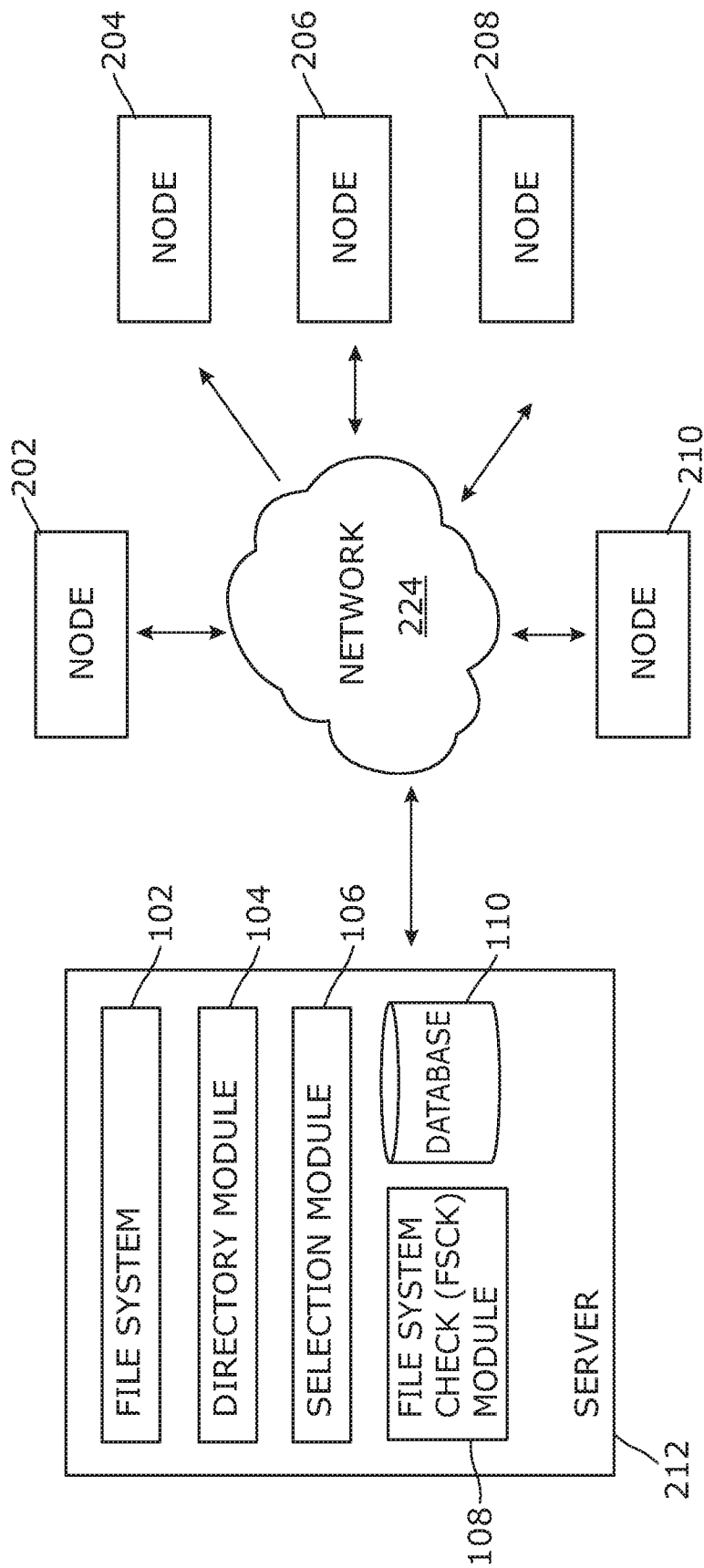


Fig. 2

200

3/4

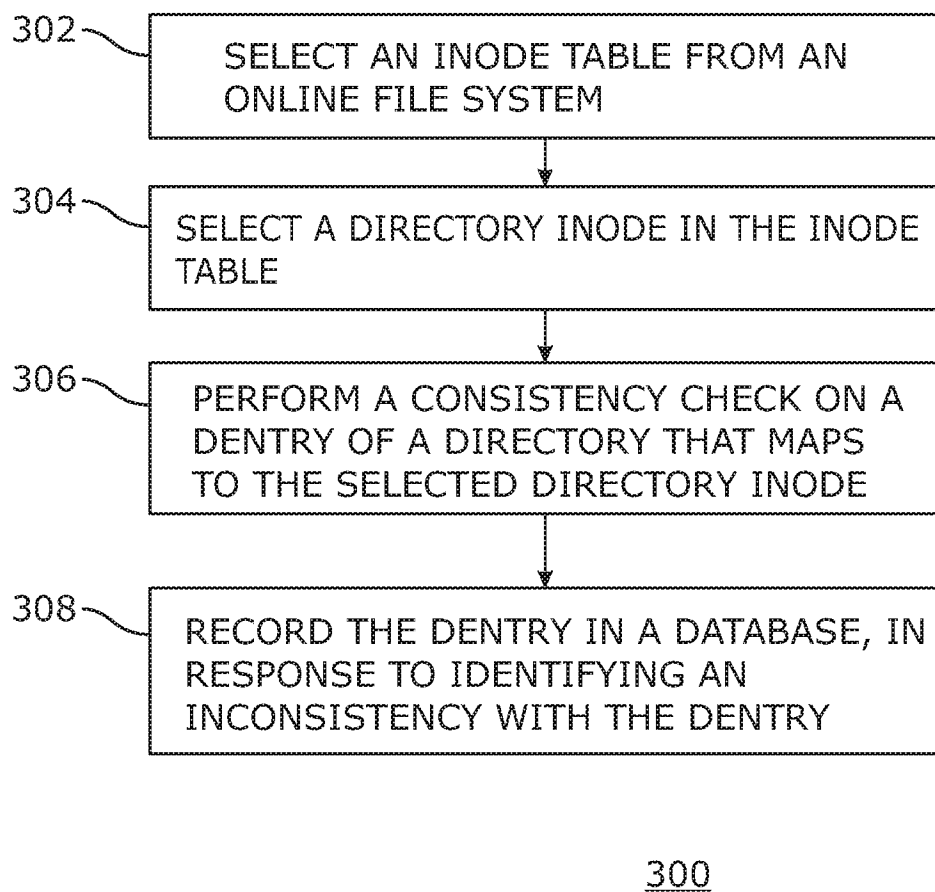


Fig. 3

4/4

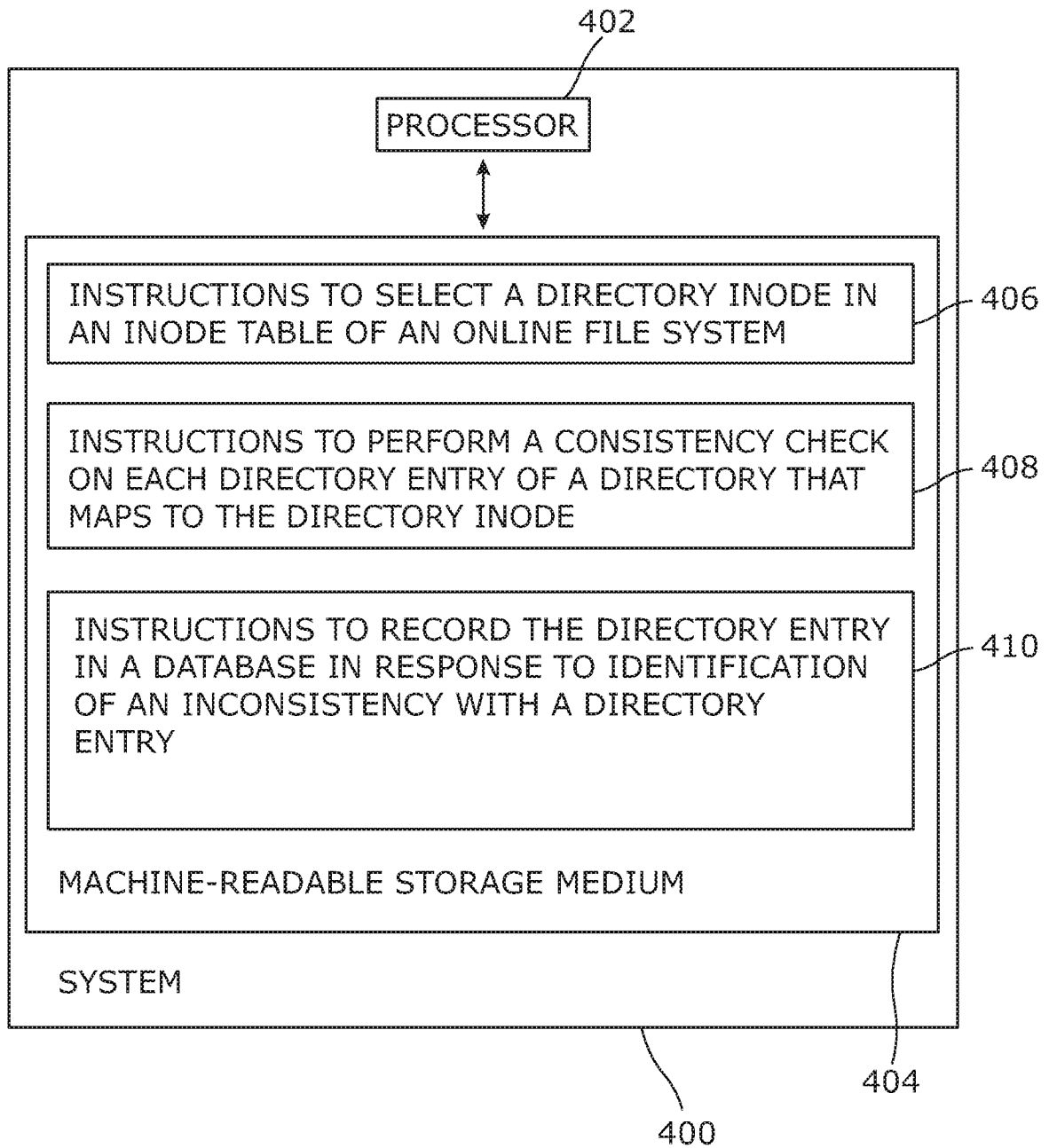


Fig. 4

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 12/00; G06F 15/167

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: inode, dentry, file system, directory, consistency check, 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.
Y	US 8131782 B1 (SUDHIR SRINIVASAN et al.) 06 March 2012 See column 2, lines 44-45; column 3, lines 14-16; column 6, lines 1-2 and 20-22; column 8, lines 33-37; column 11, lines 18-21; column 12, lines 19-32; column 16, lines 33-37 and 48-49; column 17, lines 23-28 and 40-44; and figures 5-6, 9, and 12.	1-15
Y	US 2012-0095970 A1 (NAGARAJ SHYAM et al.) 19 April 2012 See paragraphs [0010] and [0064]-[0067]; and figures 4-6 and 7.	1-15
A	WO 2010-050944 A1 (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 06 May 2010 See page 3, lines 10-27; page 8, lines 1-5; page 9, lines 26-29; and figure 3.	1-15
A	US 7,552,146 B1 (ANDY C. KAHN et al.) 23 June 2009 See column 5, lines 1-7, 39-42; and figure 4.	1-15
A	US 2008-0222152 A1 (SAURABH GODBOLE) 11 September 2008 See paragraphs [0021]-[0025] and figures 3-4.	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

16 December 2015 (16.12.2015)

Date of mailing of the international search report

16 December 2015 (16.12.2015)

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/US2015/025969

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 8131782 B1	06/03/2012	US 7430570 B1 US 7546319 B1 US 7890529 B1 US 8103639 B1 US 8316066 B1	30/09/2008 09/06/2009 15/02/2011 24/01/2012 20/11/2012
US 2012-0095970 A1	19/04/2012	US 8180811 B2	15/05/2012
WO 2010-050944 A1	06/05/2010	None	
US 7552146 B1	23/06/2009	None	
US 2008-0222152 A1	11/09/2008	US 7818302 B2	19/10/2010