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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 October 2015 (01.10.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/145454 A1

(51) International Patent Classification:
H04L 29/06 (2006.01) *H04L 29/08* (2006.01)
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/IN2014/000199

(22) International Filing Date:
28 March 2014 (28.03.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive West, Houston, Texas TX 77070 (US).

(72) Inventors; and
(71) Applicants *(for US only)*: **AVASTHI, Vinay** [IN/IN]; Sy. No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka, 560048 (IN). **MANOHAR BAGEWADI, Sanjeev** [IN/IN]; Sy. No.192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka, 560048 (IN).

(74) Agent: **ALANKI, Pradeep Kumar, N.V.**; Global IP Services, 198F, 27th Cross, 3rd Block, Jayanagar, Bangalore, Karnataka, 560011 (IN).

(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, LT, 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.

[Continued on next page]

(54) Title: DATA FILE HOARDING

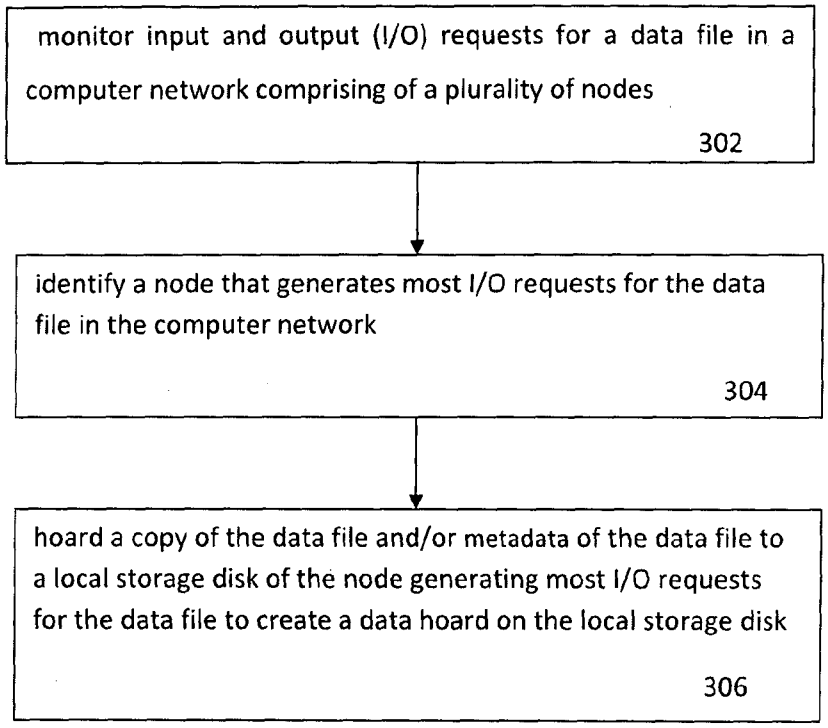

FIG. 3

[Continued on next page]

WO 2015/145454 A1

(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, 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).

Declarations under Rule 4.17:

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

— *of inventorship (Rule 4.17(iv))*

Published:

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

(57) Abstract: Some examples describe a method for hoarding a data file. Input and output (I/O) requests for a data file in a computer network comprising of a plurality of nodes are monitored, and a node generating most I/O requests for the data file in the computer network is identified. A copy of the data file and/or metadata related to the data file is hoarded to a local storage disk of the node generating most I/O requests for the data file to create a data hoard.

DATA FILE HOARDING

Background

[001] Data storage requirement of organizations has increased over the years. To support this increased requirement, file systems have also evolved over time. From earlier local file systems to more recent scale-out file systems such as a Storage Area Network (SAN) or Network Attached Storage (NAS)-based system, file systems have tried to keep pace not only with the continuous need of the organizations for additional storage but also more complex requirements such as remote file access, data sharing, data security, etc. According to an estimate, over a thousand types of file systems are currently available.

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 environment that facilitates hoarding of a data file;

[004] FIG. 2 is a block diagram of an example computing environment that facilitates hoarding of a data file;

[005] FIG. 3 is a flowchart of an example method for hoarding a data file;

[006] FIG. 4 is a flowchart of an example method for hoarding a data file;

[007] FIG. 5 is a flowchart of an example method for opening a data file from a data hoard;

[008] FIG. 6 is a flowchart of an example method for reading a data block from a data hoard; and

[009] FIG. 7 is a block diagram of an example system that facilitates hoarding of a data file.

Detailed Description of the Invention

[0010] The demand for large-scale storage systems for unstructured data has increased enormously over the years. Enterprises are demanding storage systems that can store billions of files (with file sizes ranging from a few kilobytes to petabytes) along with a file-system that can handle such storage systems with minimum of costs and complexity. Earlier file systems, which are still used generally by small organizations, were local file systems that were co-hosted with an application(s) on a server. In case additional storage capacity was needed, a local file system was typically scaled-up by adding one or more disks to the existing system. It was soon realized that although this solved the requirement of additional storage initially, it eventually resulted in a lowering the performance of a storage system since it placed an additional load on the existing processing and bandwidth resources that were required to access data stored on the additional storage space. Thus, a scale-out file system was designed to obviate the limitations of a scale-up file system.

[0011] Scale-out file systems or shared file systems can overcome constraints of a scale-up file system by providing, for instance, shared storage devices to multiple clients. In a scale-out file system, storage capacity of a system may be expanded by including additional storage resources (for example, additional disks) according to a user's requirement. Each storage resource in a scale-out storage system may include its own processing and bandwidth resource. This eases the burden on similar pre-existing resources present in a system. Scale-out file systems allow sharing of data on storage device(s) among multiple client or server systems, thus allowing more efficient use of hardware resources of an enterprise.

[0012] A scale-out file system should be designed to serve data blocks residing on remote nodes seamlessly to any user. In one instance, it may achieve this objective by retrieving data requested by a client system from a remote node on a network if it does not find the requested data on a local storage of the client system. However, whenever a data file is required to be retrieved from a remote node, it may result in a very high latency for that request. To overcome this high latency, some scale-out file systems employ read-ahead strategies where they predict the future data blocks that may be requested and retrieve them before an actual request is generated by a node in the system. However, such approaches are limited in their utility, for example, in a typical NAS-based system since it may be very difficult to obtain an accurate prediction of data that may get requested in future. Additionally, present scale-out file systems use a local cache of the node to cache and serve the data it had requested. Since operating system caches are highly valuable resources that are limited in size such an approach has many shortcomings. First, since a first access to a data is performed through a remote call, in the event a data is not accessed repeatedly, the caching may not occur at all. Second, since to guard against data loss in case of a failure, a write operation is typically required to be written to a remote node, caching of data does not help in case of an archive storage system where the primary workload is to write data. Third, a read operation on a remote node may copy all the dirty pages to be flushed to a local disk. In case of remote files, the read may have to wait until the flush is completed on the remote system.

[0013] The present disclosure describes a hoarding mechanism for a data file or metadata. The hoarding mechanism may utilize a portion of a local disk of a node to store data file from a remote node based on a pre-defined criterion. Once the data file or metadata is stored on the local disk of a node, a subsequent request for the data file or metadata from the same node may be serviced from the stored data on the local disk. In an example, input and output (I/O) requests for a data file in a computer network comprising of a plurality of nodes may be monitored to identify a

node that generates the most I/O requests for the data file in the computer network. Upon identification, a copy of the data file or metadata related to the data file may be hoarded to a local disk of the node generating most I/O requests for the data file to create a data hoard.

[0014] FIG. 1 is a block diagram of an example computing environment 100 that facilitates hoarding of a data file. Computing environment 100 may include nodes 102, 104, 106, 108, and 110, and a server 112. 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. In the example of FIG. 1, each of the nodes 102, 104, 106, 108, and 110 includes a storage unit 114, 116, 118, 120, and 122 respectively. The number of nodes 102, 104, 106, 108 and 110, and server 112 shown in FIG. 1 is for the purpose of illustration only and their number may vary in other implementations. For instance, nodes 102, 104, 106, 108, and 110 may include more than one storage unit. In some examples, computing environment 100 may represent a file storage system wherein nodes 102, 104, 106, 108, and 110 may serve as file storage nodes. In an instance, said file storage system may be a scale-out file system.

[0015] Nodes 102, 104, 106, 108, and 110 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 102, 104, 106, 108, and 110 may each be a network device such a router, a switch, and the like. Nodes 102, 104, 106, 108, and 110 may be a storage system such as a Direct Attached Storage (DAS) device, a Network Attached Storage (NAS) device, a server, a tape storage device, or any other storage device.

[0016] Nodes 102, 104, 106, 108, and 110, may communicate with each other and server 112 via a computer network 124. Computer network 124 may be a wireless or wired network. Computer network 124 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 124 may be a public network (for example, the Internet) or a private network (for example, an intranet).

[0017] Storage units 114, 116, 118, 120, and 122 may each include a non-transitory machine- readable storage medium that may store machine executable instructions, data file, metadata related to a data file, etc. Some non-limiting examples of a non-transitory machine-readable storage medium may include a hard disk, a storage disc (for example, a CD-ROM, a DVD, etc.), a disk array, a storage tape, solid state drive, and the like. In an example, some or all of storage units 114, 116, 118, 120, and 122 may be present external to nodes 102, 104, 106, 108, and 110 and may communicate with them via a communication interface.

[0018] Server 112 may include a monitoring module 126, an identification module 128, and a hoarding module 130. The term "module" may refer to a software component (machine executable 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. The module may reside on a volatile or non-volatile storage medium and configured to interact with a processor of a computing device.

[0019] Monitoring module 126 may monitor input and output (I/O) requests for a data file in a computer network (for example, 124) comprising of a plurality of nodes (for example, 102, 104, 106, 108, and 110). Identification module 126 may identify, from the plurality of nodes, a node that generates most I/O requests for the data file in the computer network. Upon identification, hoarding module 126 may hoard a copy of the data file and/or metadata related to the data file to a local storage disk of the node generating most I/O requests for the data file to create a data hoard on the local storage disk. A "data hoard" may be defined as a repository on a local storage

disk of a node that stores a copy of a data file (or data files) and/or metadata related to the data file (or data files) that was/were most frequently accessed or requested by the node, amongst a plurality of nodes, on a computer network.

[0020] FIG. 2 is a block diagram of an example computing environment 200 that facilitates hoarding of a data file. Computing environment 200 is analogous to computing environment 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 connection with FIG. 2. Said components or reference numerals may be considered alike.

[0021] Computing environment 200 may include nodes 102, 104, 106, 108, and 110, and a server 112. Server 112 may include a monitoring module 126, an identification module 128, a hoarding module 130, a hoarding policy management module 132, and an intelligent dependency module 134.

[0022] In an example, hoarding module 130 may hoard a copy of the data file and/or the metadata of the data file based on a pre-defined criterion or user policy (or policies) instructions. Pre-defined policy instructions may be stored in hoarding policy management module 132. Some non-limiting examples of a pre-defined criterion or user-defined policy may include a file name, a file name extension, a directory name, a location of the data file, a geographical area, and a time period. Hoarding module 130 may receive or retrieve pre-defined policy (or policies) instructions from hoarding policy management module 132 and apply the criterion defined therein to hoard a copy of the data file and/or metadata related to the data file to a local storage disk of the node generating most I/O requests for the data file.

[0023] Monitoring module 126 may service an I/O request for a data file and/or metadata related to a data file from a data hoard. For instance, when an I/O request for a data file and/or metadata related to a data file is generated in a computer network, monitoring module 126 may determine whether a copy of the data file

and/or metadata related to a data file is present in a data hoard on the node generating said request. If aforesaid data is present, monitoring module 126 may service the I/O request for the data file and/or metadata related to the data file from the data hoard itself. This helps save time in servicing a data file (or metadata) request that originates from a node since the data file would not required to be retrieved from another node (for example, a remote node) on the network.

[0024] Monitoring module 126 may update the copy of a data file and/or the metadata related to a data file in a data hoard after it services an I/O request for the data file and/or metadata related to the data file from the data hoard. In other words, if a request for a data file and/or the metadata related to a data file from a node has been successfully serviced from a data hoard present on the node, monitoring module 126 may update any changes made to the data file and/or metadata in the data hoard. In this manner, a data hoard would have an updated copy of the data file and/or metadata in the data hoard. Also, further to a successful I/O service request for a data file and/or the metadata related to a data file from a data hoard present on a node, monitoring module 126 may update the original data file(s) and/or metadata related to a data file(s) on a node that hosts said files or metadata for any changes made to a copy of the data file and/or metadata in the data hoard. Said differently, monitoring module 126 may synchronize a copy of a data file present in a data hoard with its original version which may be present on another node on a network.

[0025] Monitoring module 126 may determine when a file open request for the data file originates from a node on a network whether the data file is present in a data hoard on the node. If the data file is present in the data hoard, monitoring module 126 may service the file open request from the data hoard. If the data file is not present in the data hoard, monitoring module 126 may initiate a hoarding request to a node that stores the data file for a copy of the data file. Subsequent to the hoarding request, hoarding module 130 may acquire a copy of the data file from the node

that stores the data file and hoard the acquired copy of the data file to a data hoard on a local disk of the requesting node.

[0026] Monitoring module 126 may determine when a request for reading a data block from a data file originates from a node on a network whether the data block is present in a data hoard on the node. If the data block is present in the data hoard, monitoring module 126 may service the data block read request from the data hoard. If the data block is not present in the data hoard, monitoring module 126 may initiate a hoarding request to a node that stores the data block for a copy of the data block. Subsequent to the hoarding request, hoarding module 130 may acquire a copy of the data block from the node that stores the data block and hoard the acquired copy of the data block to a data hoard on a local disk of the requesting node.

[0027] Intelligent dependency module 134 may analyze the input and output (I/O) requests for a data file in a computer network to determine and hoard a copy of an associated data file to a local disk of a node that generates most I/O requests for the data file. Intelligent dependency module 134 may be an artificial intelligence engine that analyses the input and output (I/O) requests for data files in a computer network to determine dependent or related data files. For example, a database file may always open along with an index file. In another example, an XML file may require an XSD file. In a yet another example, intelligent dependency module 134 may analyze that after every I/O request for a file with an extension .tbl, a file with same name and .idx extension is also read. In such non-limiting example scenarios, intelligent dependency module 134 may acquire, from another node on the network, a copy of an associated data file and hoard it to a local disk of a node that generates most I/O requests for the data file.

[0028] FIG. 3 is a flowchart of an example method 300 for hoarding a data file. The method 200, which is described below, may be executed on a computing device such as servers of FIG. 1 and 2. However, other computing devices may be used as

well. At block 302, a server (for example, 112) may monitor input and output (I/O) requests for a data file in a computer network comprising of a plurality of nodes. At block 304, the server may identify a node that generates most I/O requests for the data file in the computer network. At block 306, the server may hoard a copy of the data file and/or metadata of the data file to a local storage disk of the node generating most I/O requests for the data file to create a data hoard on the local storage disk. In some examples, the server may hoard the copy of the data file and/or the metadata of the data file based on a pre-defined policy. Some non-limiting examples of a pre-defined policy may include a file name, a file name extension, a directory name, a location of the data file, a time period (for example, the time when a data file was created, modified, etc.), a geographical area, or a combination thereof. To provide an illustration, if the pre-defined policy is to retrieve the metadata and data blocks of a data file based on a directory structure then a copy of all the files in the directory may be retrieved and hoarded to a local disk of the node that generates most I/O requests for the data file. To provide another illustration, if the pre-defined policy is to retrieve the metadata and the data blocks of a data file based on a particular time and geographical area (e.g. between 8 AM and 5 PM, USA) then a copy of the metadata and the data blocks of a data file may be hoarded for the specified time period in a node located in the USA that generates most I/O requests for the data file in the network. Thus, based on one or more pre-defined policies, the hoarding of a copy of the data file and/or the metadata of the data file may take place.

[0029] In some examples, the server may hoard a copy of an associated data file to a local disk of the node generating most I/O requests for the data file. There may be certain type of data files that are dependent or associated with another data file. For instance, there are data files that may be executed in conjunction with other data files. For example, a database file may always open along with an index file. In another example, an XML file may require an XSD file. In such scenarios, the server

may hoard, in addition to a copy of the data file, a copy of a related data file to a local disk of the node generating most I/O requests for the data file.

[0030] FIG. 4 is a flowchart of an example method 400 for hoarding a data file. The method 400, which is described below, may be executed on a computing device such as servers of FIG. 1 and 2. However, other computing devices may be used as well. At block 402, a server (for example, 112) may monitor input and output (I/O) requests for a data file in a computer network comprising of a plurality of nodes. At block 404, the server may identify a node that generates most I/O requests for the data file in the computer network. At block 406, the server may hoard a copy of the data file and/or metadata of the data file to a local storage disk of the node generating most I/O requests for the data file to create a data hoard on the local storage disk. In some examples, the server may hoard the copy of the data file and/or the metadata of the data file based on a pre-defined policy. Some non-limiting examples of a pre-defined policy may include a file name, a file name extension, a directory name, a location of the data file, a time period (for example, the time when a data file was created, modified, etc.), a geographical area, or a combination thereof.

[0031] At block 408, determine if an I/O request for the data file originates from the same node (i.e. the node that had generated the most I/O requests for the data file), a search for the data file may be performed in the data hoard on the local disk of the node and if the data file is found to be present, the I/O request may be serviced from the data hoard. At block 410, upon servicing the I/O request for the data file from the data hoard, the data hoard may be updated. In other words, the copy of the data file and/or metadata of the data file may be updated in the local storage disk of the node.

[0032] FIG. 5 is a flowchart of an example method 500 for opening a data file from a data hoard. The method 500, which is described below, may be executed on a computing device such as servers of FIGS. 1 and 2. However, other computing

[0034] FIG. 7 is a block diagram of an example system 700 that facilitates hoarding of a data file. 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 server 112 of FIG. 1 or FIG. 2. 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 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 may be a non-transitory machine-readable medium. Machine-readable storage medium 704 may store monitoring instructions 706, identification instructions 708, and hoarding instructions 710. In an example, monitoring instructions 706 may be executed by processor 702 to monitor input and output (I/O) requests for a file in a file storage area system comprising of a plurality of file storage nodes. Identification instructions 708 may be executed by processor 702 to identify, amongst the plurality of file storage nodes, a file storage node generating most I/O requests for the file in the file storage system. Hoarding instructions 710 may be executed by processor 702 to hoard a copy of the file from a remote node to a local disk of the node generating most I/O requests for the file based on a pre-defined criterion.

[0035] For the purpose of simplicity of explanation, the example methods of FIGS. 3 - 6 are 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 7, and methods of FIGS. 3 - 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.

[0036] 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.

Claims:

1. A method, comprising:

monitoring input and output (I/O) requests for a data file in a computer network comprising of a plurality of nodes;
identifying, from the plurality of nodes, a node generating most of the I/O requests for the data file in the computer network; and
hoarding a copy of the data file and/or metadata related to the data file to a local storage disk of the node generating the most I/O requests for the data file to create a data hoard.

2. The method of claim 1, further comprising:

in response to determining generation of an I/O request for the data file and/or the metadata related to the data file from the node generating the most I/O requests for the data file, servicing the I/O request for the data file and/or the metadata related to the data file from the data hoard.

3. The method of claim 2, further comprising updating the copy of the data file and/or the metadata related to the data file in the data hoard further to servicing the I/O request for the data file and/or the metadata related to the data file from the data hoard.

4. The method of claim 1, wherein the hoarding of the copy of the data file and/or the metadata of the data file is based on a pre-defined policy, wherein the pre-defined policy includes one of a file name, a file name extension, a directory name, a location of the data file, a geographical area, and a time period.

5. The method of claim 1, further comprising:

determining when a file open request for the data file originates from the node generating the most I/O requests for the data file whether the data file is present in the data hoard; and
if the data file is present in the data hoard, servicing the file open request from the data hoard.

6. The method of claim 5, further comprising:

if the data file is not present in the data hoard, initiating a hoarding request to a node that stores the data file for a copy of the data file;
acquiring the copy of the data file from the node that stores the data file; and
hoarding the copy of the data file to the data hoard.

7. The method of claim 1, further comprising:

determining when a request for reading a data block from a data file originates from the node generating most I/O requests for the data file whether the data block is present in the data hoard; and
if the data block is present in the data hoard, servicing the request for reading the data block from the data hoard.

8. The method of claim 7, further comprising:

if the data block is not present in the data hoard, initiating a hoarding request to a node that stores the data block for a copy of the data block;
acquiring the copy of the data block from the node that stores the data block; and
hoarding the copy of the data block in the data hoard.

9. A system, comprising:

a monitoring module to:

monitor input and output (I/O) requests for a data file in a network comprising of a plurality of nodes;
identify, from the plurality of nodes, a node generating most I/O requests for the data file in the network; and
hoard a copy of the data file and/or metadata related to the data file to a local storage disk of the node generating most I/O requests for the data file to create a data hoard on the local storage disk.

10. The system of claim 9, further comprising:

a hoarding policy management module, wherein the hoarding policy management module includes a user-defined policy that defines a criterion to hoard the data file and/or the metadata related to the data file to the local disk of the node generating most I/O requests.

11. The system of claim 9, further comprising:

an intelligent dependency module that analyzes the input and output (I/O) requests for the data file in the network to determine and hoard a copy of an associated data file to the local disk of the node generating most I/O requests for the data file.

12. The system of claim 9, wherein the data file is present on a node remote from the node generating most I/O requests for the data file in the network.

13. The system of claim 12, wherein the data file present on the remote node is updated when an I/O request for the data file from the node generating most I/O requests for the data file is serviced from the data hoard.

14. A non-transitory machine-readable storage medium, the non-transitory machine-readable storage medium comprising machine executable instructions, the machine executable instructions when executed by a processor causes the processor to:

monitor input and output (I/O) requests for a file in a file storage area system comprising of a plurality of file storage nodes;
identify, amongst the plurality of file storage nodes, a file storage node generating most of the I/O requests for the file in the file storage system; and
hoard a copy of the file from a remote node to a local disk of the node generating the most I/O requests for the file based on a pre-defined criterion.

15. The non-transitory machine-readable storage medium of claim 14, wherein the file storage system is a scale-out file system.

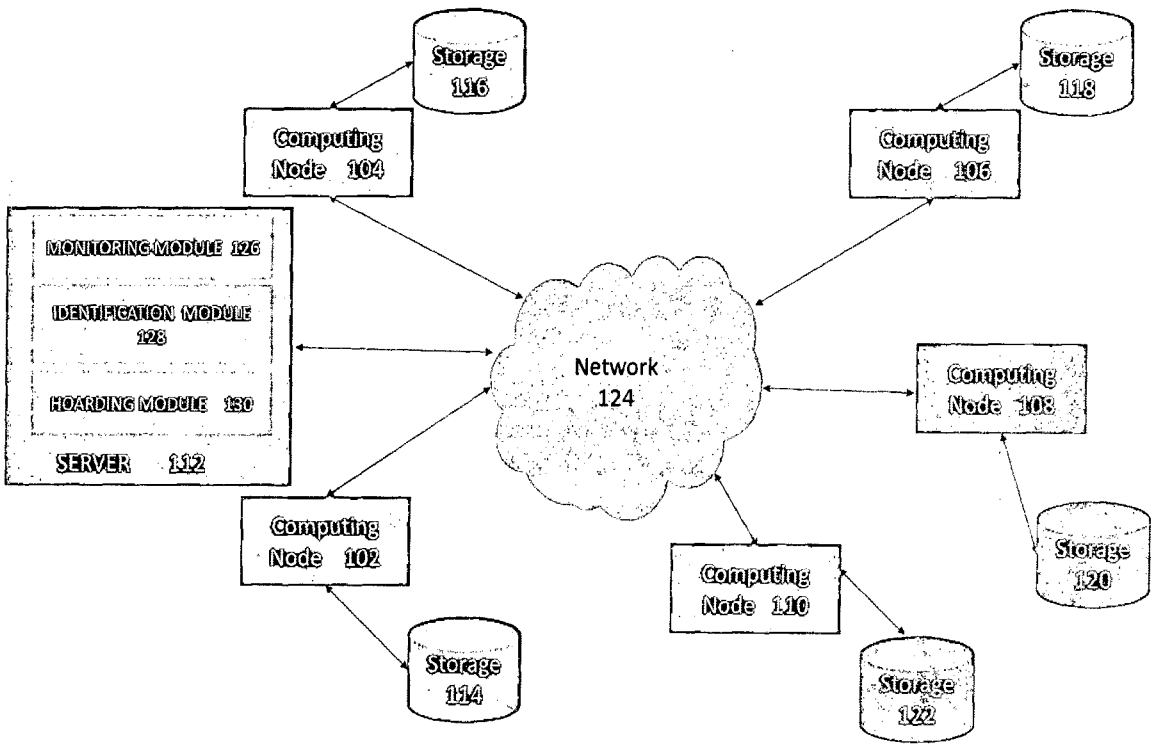
Storage 116
Storage 118
Computing Node 104
Computing Node 106
MONITORING MODULE 126
IDENTIFICATION MODULE 128
HOARDING MODULE 130
SERVER 112
Network 124
Computing Node 108
Computing Node 102
Storage 120
Computing Node 110
Storage 114
Storage 122

FIG. 1 100

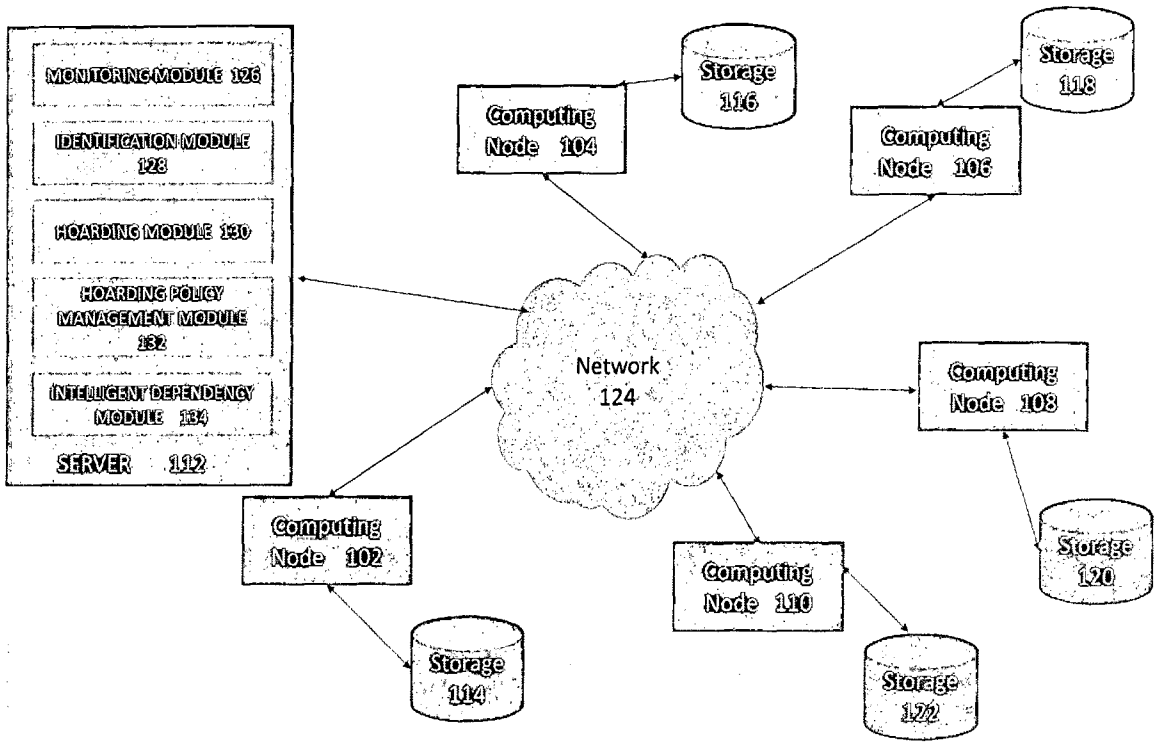
MONITORING MODULE 126
IDENTIFICATION MODULE 128
HOARDING MODULE 130
HOARDING POLICY MANAGEMENT MODULE 132
INTELLIGENT DEPENDENCY MODULE 134
SERVER 112
Computing Node 104
Storage 116
Computing Node 106
Storage 118
Network 124
Computing Node 108
Storage 120
Computing Node 102
Storage 114
Computing Node 110
Storage 122

FIG. 2 200

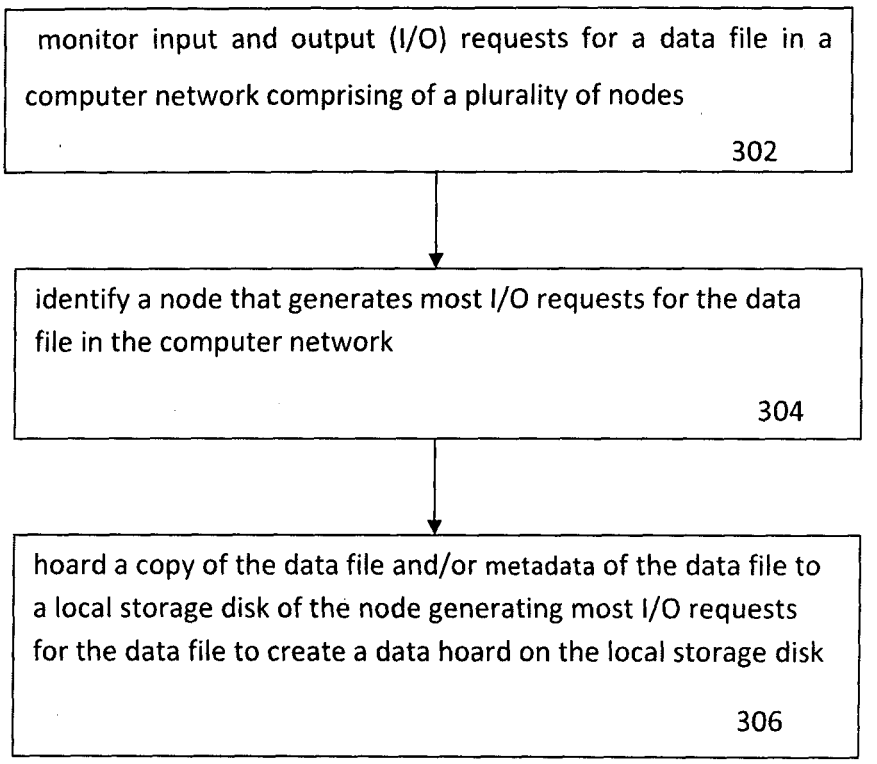
monitor input and output (I/O) requests for a data file in a computer network comprising of a plurality of nodes
302
identify a node that generates most I/O requests for the data file in the computer network
304
hoard a copy of the data file and/or metadata of the data file to a local storage disk of the node generating most I/O requests for the data file to create a data hoard on the local storage disk
306

FIG. 3

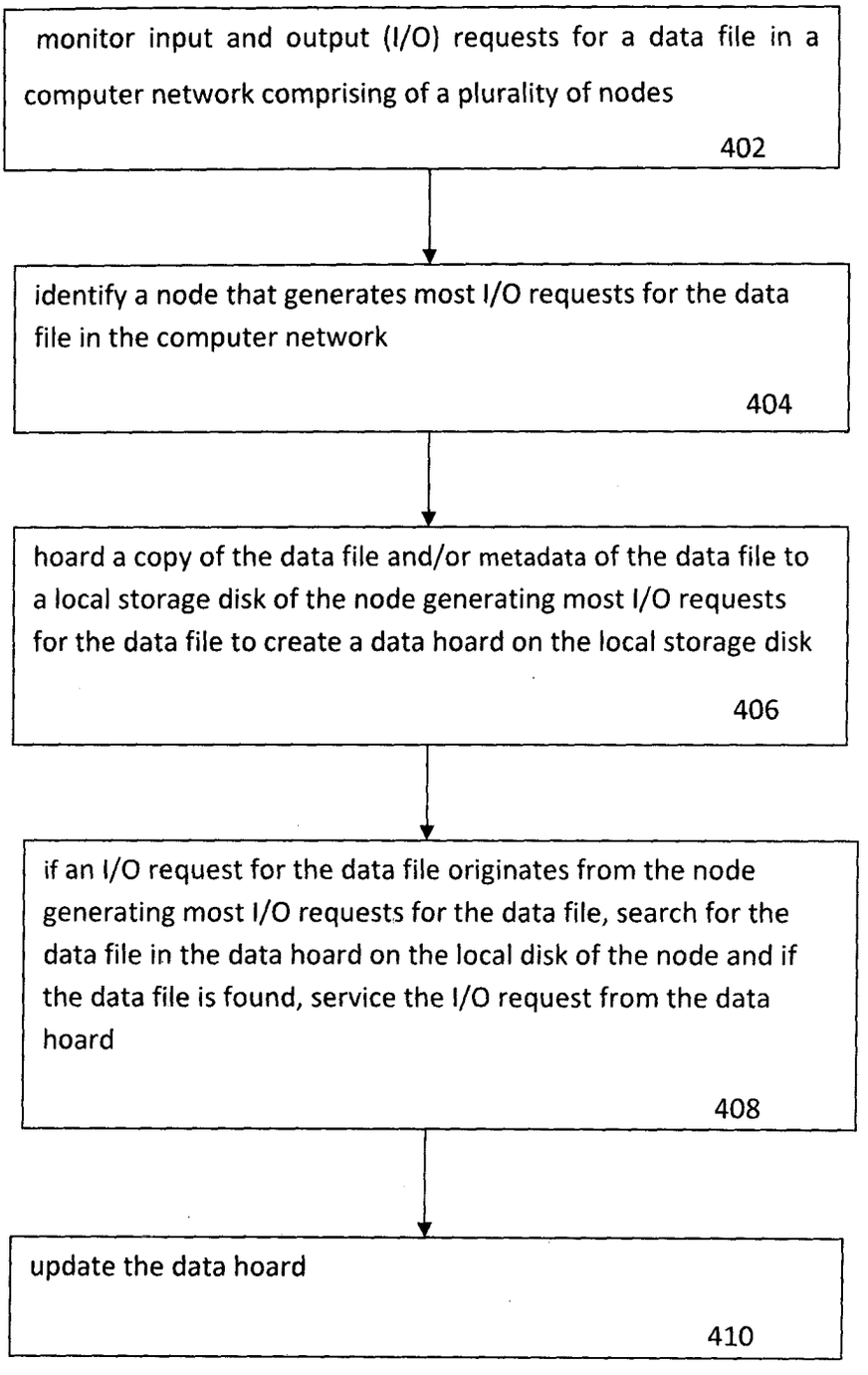
monitor input and output (I/O) requests for a data file in a computer network comprising of a plurality of nodes
402
identify a node that generates most I/O requests for the data file in the computer network
404
hoard a copy of the data file and/or metadata of the data file to a local storage disk of the node generating most I/O requests for the data file to create a data hoard on the local storage disk
406
if an I/O request for the data file originates from the node generating most I/O requests for the data file, search for the data file in the data hoard on the local disk of the node and if the data file is found, service the I/O request from the data hoard
408
update the data hoard
410

FIG. 4

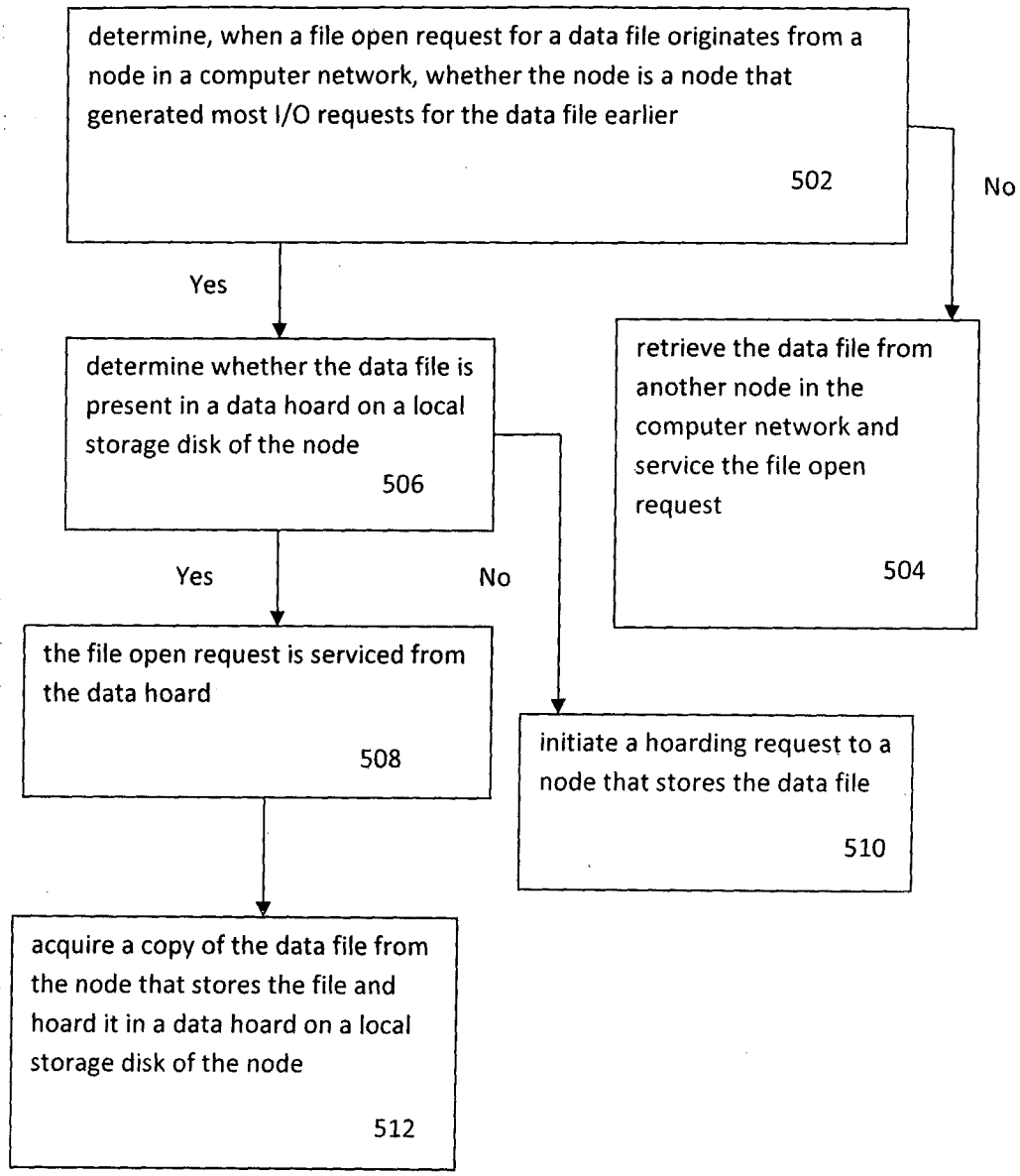
determine, when a file open request for a data file originates from a node in a computer network, whether the node is a node that generated most I/O requests for the data file earlier
502
No
Yes
determine whether the data file is present in a data hoard on a local storage disk of the node
506
retrieve the data file from another node in the computer network and service the file open request
504
Yes
No
the file open request is serviced from the data hoard
508
initiate a hoarding request to a node that stores the data file
510
acquire a copy of the data file from the node that stores the file and hoard it in a data hoard on a local storage disk of the node
512

FIG. 5

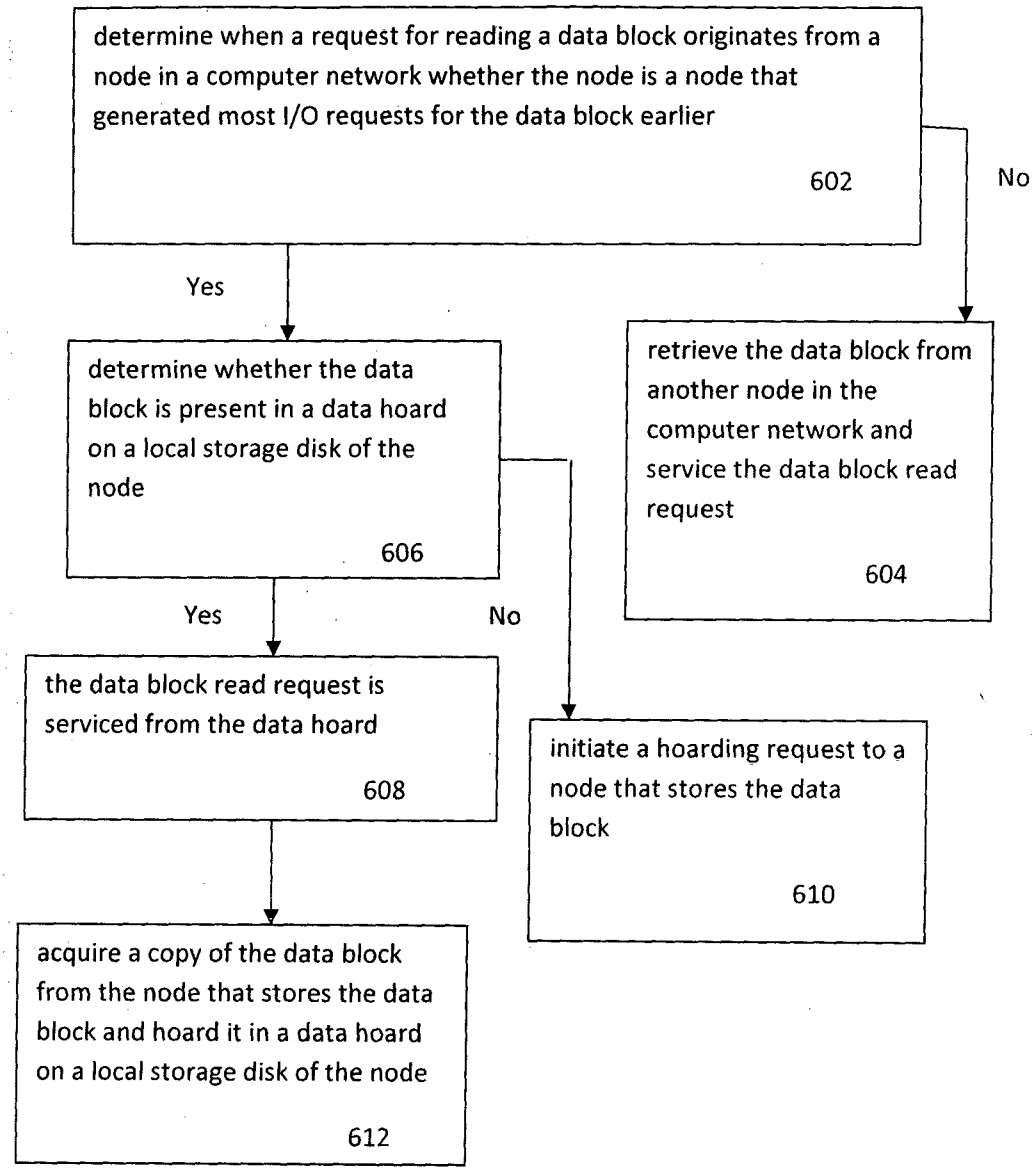
determine when a request for reading a data block originates from a node in a computer network whether the node is a node that generated most I/O requests for the data block earlier
602
No
Yes
retrieve the data block from another node in the computer network and service the data block read request
604
determine whether the data block is present in a data hoard on a local storage disk of the node
606
Yes
No
the data block read request is serviced from the data hoard
608
initiate a hoarding request to a node that stores the data block
610
acquire a copy of the data block from the node that stores the data block and hoard it in a data hoard on a local storage disk of the node
612

FIG. 6

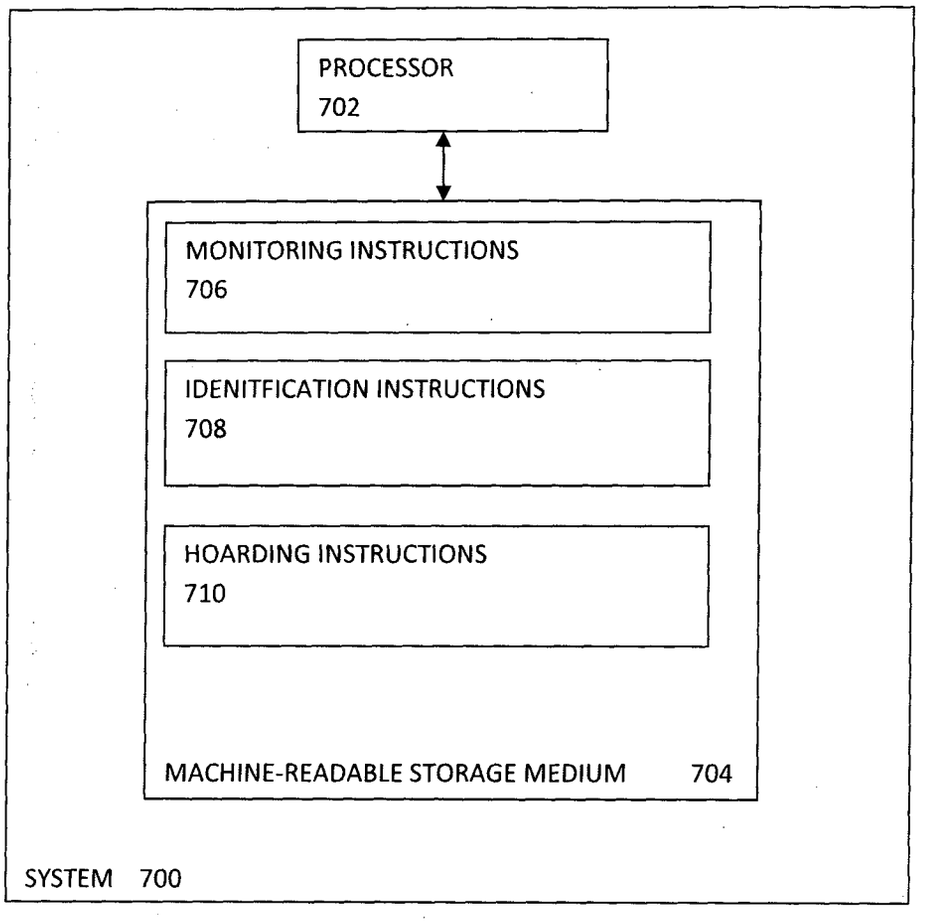
PROCESSOR
702
MONITORING INSTRUCTIONS
706
IDENITFICATION INSTRUCTIONS
708
HOARDING INSTRUCTIONS
710
MACHINE-READABLE STORAGE MEDIUM 704
SYSTEM 700

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IN2014/000199

A. CLASSIFICATION OF SUBJECT MATTER

H04L29/06(2006.01)i; G06F 17/30(2006.01)i; H04L 29/08(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)

H04L; G06F

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

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

CNABS; CNTXT; CNKI; VEN:I/O, input, output, read, access, frequency, high, large, great, most, hot, hotspot, local, locally, copy, store, save, hoard

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 101510219 A (CHENGDU HUAWEI SYMANTEC TECHNOLOGIES CO LTD) 19 August 2009 (2009-08-19) description, page 5, line 22 to page 12, line 21, and figure 10	1-15
A	CN 101370030 A (UNIV SOUTHEAST) 18 February 2009 (2009-02-18) the whole document	1-15
A	CN 103455284 A (BEIJING TEAMSUN TECHNOLOGY CO LTD) 18 December 2013 (2013-12-18) the whole document	1-15
A	CN 103455577 A (CHINESE ACAD SCI COMPUTER NETWORK INFORMATION CENTER ET AL.) 18 December 2013 (2013-12-18) the whole document	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	Date of mailing of the international search report
09 December 2014	**31 December 2014**

Name and mailing address of the ISA/CN	Authorized officer
STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA(ISA/CN) **6,Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088 China**	**LI,Meili**
Facsimile No. **(86-10)62019451**	Telephone No. **(86-10)62089957**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IN2014/000199

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 101510219 A	19 August 2009	CN 101510219 B	14 September 2011
CN 101370030 A	18 February 2009	CN 101370030 B	16 March 2011
CN 103455284 A	18 December 2013	Non e	
CN 103455577 A	18 December 2013	Non e	