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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 January 2016 (28.01.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/014061 A1

(51) International Patent Classification:
G06F 12/08 (2006.01)

(21) International Application Number:
PCT/US2014/048001

(22) International Filing Date:
24 July 2014 (24.07.2014)

(25) Filing Language: English

(26) Publication Language: English

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

(72) Inventor: **DEJONG, David Michael**; 4245 Technology Drive, Fremont, California 94538 (US).

(74) Agents: **ORTEGA, Arthur S.** et al.; Hewlett-Packard Company, Intellectual Property Administration, Mail Stop 35, 3404 E. Harmony Road, Fort Collins, Colorado 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, 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.

(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).

[Continued on next page]

(54) Title: STORING METADATA

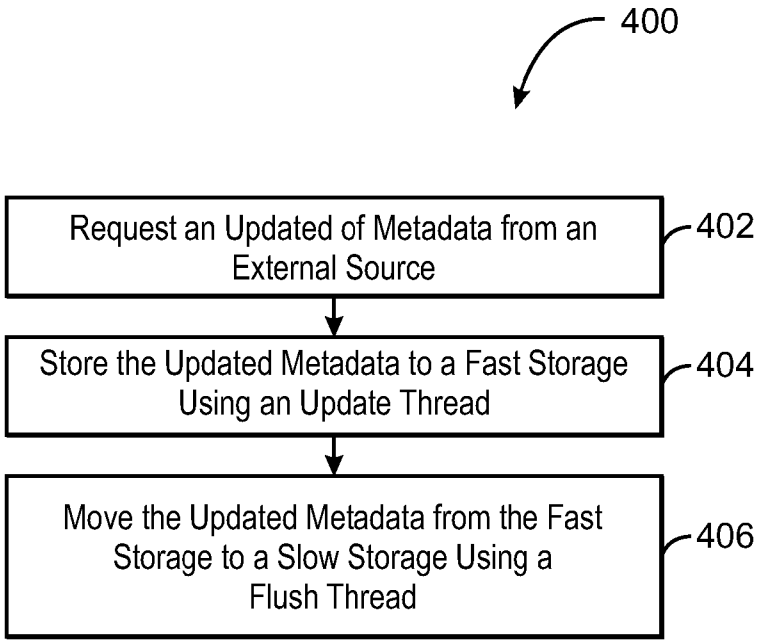

FIG. 4

(57) Abstract: A method and a system for storing metadata. The method includes requesting an update of metadata from an external source. The method includes storing updated metadata to a fast storage medium using an update thread. The method further includes moving the updated metadata from the fast storage medium to a slow storage medium using a flush thread.

WO 2016/014061 A1

Declarations under Rule 4.17:

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

— *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))*

STORING METADATA

BACKGROUND

[0001] In order to store metadata for a cluster system, updates to the metadata are synced to a slow storage disk before notifying a user about the success of the accepted changes. The process of retaining and storing the metadata may result in a delay of several seconds on a heavily loaded clustered system. Moreover, in the event that the cluster system fails before the metadata can be retained and stored, the integrity of the metadata may be compromised and lost during transmission.

DESCRIPTION OF THE DRAWINGS

[0002] The advantages of the present techniques are better understood by referring to the following detailed description and the attached drawings, in which:

[0003] Fig. 1 is a schematic of a computing system that stores updated metadata in a circular buffer within a fast storage medium, when a memory location is available in the circular buffer;

[0004] Fig. 2 is a schematic of a computing system that stores a checkpoint image within a checkpoint area of a slow storage medium, when a memory location is not available within a fast storage medium;

[0005] Fig. 3 is a block diagram of a computing system configured for storing metadata;

[0006] Fig. 4 is a method of storing metadata; and

[0007] Fig. 5 is a flow diagram including procedural steps to store metadata.

DETAILED DESCRIPTION OF SPECIFIC EXAMPLES

[0008] Metadata may be defined as data that describes other data, e.g., which provides underlying definitions or descriptions of the other data. By describing the contents and context of the other data, relevant information related to the other data is readily available to a user. For example, an image file may include metadata that describes date of creation, ownership, access permissions, among other information.

To be useful, a file system should record and store the updates related to the metadata in a fast and reliable manner while maintaining the integrity of the updated metadata in the event of an unpredictable system failure, e.g., power failure.

[0009] A file system, including a number of external sources (e.g. a plurality of nodes), may communicate with a slow, non-volatile storage to request storage of updated metadata. The slow, non-volatile storage may store and protect the updates by sequencing disk writes. With this approach, each individual on-disk metadata must be atomic, thus, forcing the updates to be fully contained by a single disk. However, before the file system can initiate a reply of a successful update, all updates must be successfully synced to the single disk. Further, if the single disk is selected from among several disks, a time consuming process of examining all of the disks may be initiated to manually select the single disk. As a result, a delay, which may be as long as several seconds on a heavily loaded cluster system, may create a latency issue.

[0010] Examples described herein describe a method of storing metadata that includes storing updated metadata to a fast storage medium before storing to a slow storage medium. This method may overcome latency issues associated with the slow storage medium by initially storing metadata to the fast storage medium, while protecting the metadata against system failure events.

[0011] Fig. 1 is a schematic of a computing system 100 that stores updated metadata in a circular buffer within a fast storage medium, when a memory location is available in the circular buffer. The computing system 100 may include an update thread 102, fast storage medium 104, a flush thread 106, and slow storage medium 108. The update thread 102 may be configured to communicate with an external source by receiving updates, determining the status of the updates, and storing the updates to the fast storage medium 104. The fast storage medium 104 may include random access memory (RAM), including dynamic random access memory (DRAM) and static random access memory (SRAM), and fast central processing unit (CPU) cache memory, among other volatile memory types. For example, updates related to the metadata may provide relevant information to a user by describing how, when, and by whom a particular set of data was collected, and how the data is formatted.

[0012] In examples described herein, the fast storage medium 104 may act as a holding place for the updated metadata 110, with little to no latency issues, before moving and storing it to the slow storage medium 108. The fast storage medium 104 includes memory that may need power to preserve the stored updated metadata 110. For example, if a device is switched to a powered-off position, the updated metadata 110 stored in the fast storage medium 104 may be lost.

[0013] The update thread 102 may initially determine if a memory location is available in the fast storage medium 104 to record the updates before generating and storing updated metadata 110 to that memory location. For example, if all memory locations in the circular buffer in the fast storage medium are currently filled with metadata that is waiting to be transferred to the slow storage medium, the update thread 102 may be blocked from storing the updates. This is discussed with respect to Fig. 2.

[0014] If a memory location exists, the update thread 102 may build up a log header, a log update header, and data for log updates to generate and to store the updated metadata 110. The log header may include a generation number of a log update, total size of a disk as used by the log update, and checkpoint generation numbers. The log update header may include a length of the log update and an offset as to the log update applied to the checkpoint generation numbers. A generation number may be defined as an increasing integer that updates by one each time the metadata is updated.

[0015] After the buildup of the updated metadata 110, it may be stored in the log area 112. In particular, the updated metadata 110 may be stored in a memory mirror of the log area 112. The log area 112 may be treated as a circular buffer so that existing metadata stored in the fast storage medium 104 may be overwritten by the updated metadata 110 when the log area 112 is full, assuming that the overwritten areas have been processed for storage.

[0016] Since the update thread 102 and the flush thread 106 may both attempt to access the updated metadata 110 simultaneously, a mutex may be utilized to protect against simultaneous access to the updated metadata 110. The mutex is a system object that allows multiple threads, e.g., update thread 102 and flush thread 106, to

share the same resource, such as file access to the locations being used for the storage of the updated metadata 110, without allowing simultaneous access. For example, once the update thread 102 begins to store the updated metadata 110 to the fast storage medium 104, a mutex may be created and used by the update thread 102. The mutex may prevent the flush thread 106 from accessing the updated metadata 110 during processing. Thus, the update thread 102 may have exclusive use of the mutex until the updated metadata 110 is fully stored to the fast storage medium 104. Thereafter, the update thread 102 may release the mutex, allowing the flush thread 106 to have access to the updated metadata 110.

[0017] After storing the updated metadata 110 to the fast storage medium 104, the update thread 102 may provide a signal to the flush thread 106 indicating that more metadata is available. In some examples, the flush thread may determine the updated metadata 110 is available by noting that a flag is set in a register corresponding to the updated metadata 110. The flush thread 106 may be configured to receive, store, and write the updated metadata 110 to the slow storage medium 108. The slow storage medium 108 includes memory that is saved regardless of the status of the power source, e.g., power on or power off. Examples of slow storage mediums 108 include a hard disk drive, flash memory, magnetic disk, magnetic tape, among other non-volatile storage types.

[0018] As shown in Fig. 1, the slow storage medium 108 may include a log area 114 and a checkpoint area 116. Upon receiving the signal that the updated metadata 110 is available, the flush thread 106 may acquire the mutex and may determine the capacity of the log area 114 for storage of the updated metadata 110.

[0019] In the event that a memory location is available, the flush thread 106 may copy a region of the log area 114 into a temporary memory buffer in an effort to ensure that the updated metadata 110 does not overwrite existing metadata. The copied log area may include the updated metadata 110 starting at a generation immediately after the updated metadata 110 currently present on the slow storage medium 108 and ending with the latest update. The flush thread 106 may dropped the mutex before

writing the temporary memory buffer to a location in the log area 114 of the slow storage medium 108.

[0020] Both the fast storage medium 104 and the slow storage medium 108 may require two different areas, a header area and the log area 112, 114, which is previously discussed. The updated metadata 110 may be written to the header area and the log area 112, 114 in alternation such that the most recently written area is not overwritten. Thus, writing to the areas may not be synced since there may be multiple updates to the fast storage medium104 in between multiple updates to the slow storage medium 108.

[0021] A latency issue for a read/write command may occur based on the time to initially log updates to the slow storage medium 108 and to receive an acknowledgment that the updated metadata has been written to the log area 114 of the slow storage medium 108. As discussed with respect to Fig. 1, the updated metadata 110 may be received and stored to the fast storage medium 104 before it is stored to the slow storage medium 108. Thus, a write command may be performed under a substantially reduced latency by utilizing the fast storage medium 104 to initially process and store the updated metadata.

[0022] Fig. 2 is a schematic of a computing system 100 that stores a checkpoint image within a checkpoint area of a slow storage medium, when a memory location is not available within a fast storage medium. Like numbers are as described with respect to Fig. 1.

[0023] When the update thread 102 receives a pending update for metadata, it may determine if an available memory location exists in a log area 112 of a fast storage medium 104 to record the updated metadata 110. If the update thread 102 determines that a memory location is not available in the log area 112, the update thread 102 may release a mutex, which prevents access to the updated metadata 110. The flush thread 106 may acquire the mutex and may store the current set of updated metadata 110, as a single image block, into a checkpoint area 116 located in a slow storage medium 108.

[0024] The checkpoint area 116 may include a header that describes a checkpoint, e.g., description of the updated metadata written to the checkpoint area and the actual updated metadata stored in the checkpoint area. The header may contain a generation

number of the checkpoint, the size used on a disk by the checkpoint, and the size of an uncompressed checkpoint.

[0025] Before actually storing the updated metadata 102 in the checkpoint area 116, the data may be compacted. The compaction includes scanning the checkpoint area 116 for areas marked to be skipped in order to remove data from that particular area and moving any data following the areas to be skipped back to eliminate empty regions. This process may remove any fragmentation within the updated metadata 110 and allow the flush thread 106 to keep track of both old and new positions of the updated metadata 110. The aforementioned checkpoint header and the compressed updated metadata 110 may be moved into a separate buffer. At that time, the mutex held by the flush thread 106 may be released and the flush thread 106 may write the separate buffer, which includes the updated metadata 110, to the checkpoint area 116 on the slow storage medium 108 to complete the process.

[0026] During the write to the checkpoint area 116, the update thread 102 may receive new updated metadata that may include an offset to be applied against an old checkpoint and an offset to be applied against a new checkpoint. The offset may be defined as the total size of the metadata and may be represented by a number ranging from zero to length-1, where the length refers to the total size of the metadata. This option allows for the log update to be reapplied against either checkpoint. If no new updated metadata is received during the write to the checkpoint area 116, a new log update may be automatically created that does not change the stored updated metadata. The new log update may simply reflect that the old checkpoint is not needed.

[0027] After storing the updated metadata to the checkpoint area 116 in the slow storage medium 108, a re-read may occur in order to move the updated metadata back into the fast storage medium 104. There may be two re-read processes, i.e., first process and second process, that may be used to re-read the updated metadata 110 back into either the log area 112 of the fast storage medium 104 or the log area 114 of the slow storage medium 108.

[0028] Since the second process may only be free of data loss if a clean shutdown occurs with all data flushed, it may be only used when an entire cluster is cleanly

rebooted or if the fast storage medium 104 is lost on all nodes. In all other cases, the first process may be utilized. The first process may include using the log area 112 from the fast storage medium 104 and the checkpoint area 116 from the slow storage 108. The second process may include using the log area 114 and the checkpoint area 116 from the slow storage 108.

[0029] Regardless of the process used, the re-read process may begin by reading the updated metadata 110 related to the most recent update from either the fast storage medium 104 or slow storage medium108. This may depend on whether the log area 112 of the fast storage medium 104 or the log area 114 of the slow storage medium 108 was previously used. The updated metadata 110 from the checkpoint area 116 may be uncompressed and a compression algorithm may be used to verify whether the write to the checkpoint 116 was completed and non-corrupted. If a checkpoint of a particular drive is not valid, the compression algorithm may continue to check until a valid checkpoint is found. Once a valid checkpoint is in a storage medium, the log area from the storage medium may be read into a buffer, where a first log header may be parsed in the buffer using a recorded start offset. If the generation of the log header is greater than the generation of the checkpoint, the log updates described by the log header may be applied using the offsets from the log update headers.

[0030] The re-read process may then proceed to a second log header where the aforementioned steps may be repeated. The re-read process may be considered complete when the log header for the generation recorded at the most recent updated is replayed. Upon completion, there may be uncertainty as to the state of the slow storage medium 108. To ensure a consistent slow storage medium 108, the re-read process may begin an immediate checkpoint of the updated metadata 110 until the re-read process proceeds until completion of the checkpoint.

[0031] Fig. 3 is a block diagram of a computing system 300 configured for storing metadata. The computing system 300 may include, for example, a server computer or a desktop computer, among others. In some embodiments, the computing system 300 may include a cluster system made up of a number of such devices, where each device may leave or rejoin the cluster system.

[0032] The computing system 300 may include a processor 302 that is adapted to execute stored instructions. The processor 302 can be a single core processor, a multi-core processor, a computing cluster, or any number of other appropriate configurations.

[0033] The processor 302 may be connected through a system bus 304, e.g., AMBA®, PCI®, PCI Express®, Hyper Transport®, Serial ATA, among others, to an input/output (I/O) device interface 306 adapted to connect the computing system 300 to one or more I/O devices 308. The I/O devices 308 may include, for example, a keyboard and a pointing device, wherein the pointing device may include a touchpad or a touchscreen, among others. The I/O devices 308 may be built-in components of the computing system 300, or may be devices that are externally connected to the computing system 300.

[0034] The processor 302 may be linked through the system bus 304 to a display device interface 310 adapted to connect the computing system 300 to display devices 312. The display devices 312 may include a display screen that is a built-in component of the computing system 300. The display devices 312 may also include computer monitors, televisions, or projectors, among others, that are externally connected to the computing system 300.

[0035] The processor 302 may be linked through the system bus 304 to a network interface controller (NIC) 314 adapted to connect the computing system 300 to a computer network including external nodes 316. The external nodes 316 may include computers, storage devices, servers, switches, routers, among others, that are externally connected to the computing system 300.

[0036] The processor 302 may also be linked through the system bus 304 to a memory device, e.g., fast storage medium 318. In some examples, the fast storage medium 318 can include random access memory, e.g., SRAM, DRAM, eDRAM, EDO RAM, DDR RAM, RRAM®, PRAM, among others. The fast storage medium 318 may contain an update thread 320, a flush thread 322, and a log area 324. The update thread 318 may embody software configured to communication with the external nodes 316 to receive an update request related to metadata and to implement storage of the updated metadata.

[0037] The log area 324 may be treated as a circular buffer for storing the updated metadata processed by the update thread 320. In particular, the log area 324 may include a fixed size of allocated memory that can be used to store the updated metadata. Thus, instead of allocating more memory, new metadata may overwrite existing metadata so that the fast storage medium 318 may be able to handle numerous updates within a particular time-frame, e.g., 50 to 100 nanoseconds (ns). Thus, storing the updated metadata to the fast storage medium 318 may result in little to no latency issues. The flush thread 322 may include software configured to move the updated metadata from the log area 324 to a slow storage medium 326.

[0038] The processor 302 may be linked through the system bus 304 to the slow storage medium 326. The slow storage medium 326 may be configured to receive the updated metadata from the flush thread 322 and to read and store the metadata for long-term persistent storage. The slow storage medium 326 may store the updated metadata in a checkpoint area 328 or a log area 330.

[0039] The checkpoint area 328 may contain a full description of all of the data at a given point in time. The log area 330 may describe the modifications made to the checkpoint area 328. To retrieve the most recent updated metadata, a user may access the checkpoint area 328. Thereafter, the user may retrieve updated metadata from either the log area 324 of the fast storage medium 318 or the log area 330 of the slow storage medium 326.

[0040] Fig. 4 is a method 400 of storing metadata. The method 400 provides a low latency storage process by initially storing metadata to a fast, volatile storage and subsequently storing and writing the metadata to a slow, non-volatile storage. At block 402, a request for an update to metadata from an external source is received by an update thread, which may result in the generation of updated metadata.

[0041] At block 404, the updated metadata is stored to a fast storage medium using the update thread. The fast storage medium may include volatile memory with little to no latency issues as the updated metadata can be quickly stored, for example, within 50 to 100 nanoseconds. The updated metadata to be stored can be treated as an arbitrary section of data where several operations may be conducted on the updated metadata.

For example, new updated metadata may be added to the end of the existing metadata, an existing segment of the existing metadata may be modified, and a segment of the existing metadata may be removed by modifying it such that it will be skipped during parsing of the metadata. At block 406, a flush thread moves the updated metadata from the fast storage medium to the slow storage medium for permanent storage.

[0042] Fig. 5 is a flow diagram 500 including process steps to store metadata. At block 502, updates may be sent to an update thread, which may generate updated metadata. The update thread may choose to store the updated metadata in a memory location of a buffer in a fast storage medium. At block 504, before storing the updated metadata, the update thread may determine if a memory location of a buffer is available in the fast storage medium. If a memory location of the buffer is available, at block 506, the update thread may store the updated metadata to the memory location of the buffer in the fast storage medium. The storage of the updated metadata initially to the fast storage medium may facilitate an efficient process, with little to no latency issues, since volatile memory is initially used for storage, as oppose to non-volatile memory.

[0043] Once the update thread stores the updated metadata to the fast storage medium, it may signal a flush thread as to the availability of the updated metadata. At block 508, the flush thread moves the updated metadata to a slow storage medium. Thereafter, at block 510, the flush thread may release the memory location of the buffer of the fast storage medium and await additional updated metadata from the update thread. When the update thread receives additional updates, the process steps may restart, i.e., at block 502.

[0044] If an unused memory location of the buffer is not available in the fast storage medium (i.e., at block 504), the flush thread may obtain the updated metadata to create a checkpoint image and to store the checkpoint image in the slow storage medium, at block 512. Accordingly, at block 514, the flush thread may process the checkpoint image from the slow storage medium to generate the log files. At block 516, the process determines if there are any unprocessed memory locations remaining in a checkpoint file. If unprocessed memory locations are available, the process returns to block 514. If there are no unprocessed memory locations remaining in the checkpoint

file, the process returns to block 502 where the update thread may generate updated metadata.

[0045] The method 500 described above may include additional steps not shown. For example, if a checkpoint has been stored in slow storage, the flush thread may copy the checkpoint data back to fast storage before processing the updates from fast storage. In some examples, not all of the steps shown may be needed. For example, the process shown in blocks 514 and 516 may not be used if the checkpoint data is moved from slow storage back to fast storage and processed from the fast storage.

[0046] While the present techniques may be susceptible to various modifications and alternative forms, the embodiments discussed above have been shown only by way of example. However, it should again be understood that the techniques is not intended to be limited to the particular embodiments disclosed herein. Indeed, the present techniques include all alternatives, modifications, and equivalents falling within the true spirit and scope of the appended claims.

CLAIMS

1. A method of storing metadata, comprising:
requesting an update of metadata from an external source;
storing updated metadata to a fast storage medium using an update thread; and
moving the updated metadata from the fast storage medium to a slow storage medium using a flush thread.

2. The method of claim 1, comprising determining if a memory location in a circular buffer is available to store updated metadata to the fast storage medium.

3. The method of claim 1, comprising storing the updated metadata in a circular buffer of the fast storage medium upon an indication of an available memory location in a circular buffer.

4. The method of claim 1, comprising signaling the availability of the updated metadata to the flush thread upon the completion of storing the updated metadata to the fast storage medium.

5. The method of claim 1, comprising storing the updated metadata to a temporary memory buffer within the slow storage medium using the flush thread.

6. The method of claim 1, comprising storing the updated metadata to a checkpoint within the slow storage medium using the flush thread upon determining a lack of an available memory location in a circular buffer of the fast storage medium.

7. A storage system for metadata, comprising
a processor;
a fast storage medium;
a slow storage medium; and

a computer-readable medium comprising:

an update thread configured to direct the processor to obtain metadata from a node and store the metadata in the fast storage medium; and

a flush thread configured to direct the processor to move the metadata from the fast storage medium to the slow storage medium.

8. The storage system of claim 7, wherein the slow storage medium comprises a header section, a checkpoint section, and a log section.

9. The storage system of claim 8, wherein the header section includes a generation number of a log update, total size of a disk used by the log update, and a checkpoint generation number.

10. The storage system of claim 8, wherein the checkpoint section comprises a header and compressed data.

11. The storage system of claim 8, wherein the log section is a circular buffer comprising a log header, a log update header, and data.

12. The storage system of claim 7, wherein the update thread determines if a memory location is available in a circular buffer of the fast storage medium to store the metadata.

13. The storage system of claim 7, comprising a temporary memory buffer in the slow storage medium to accept the metadata from the fast storage.

14. A non-transitory, computer readable medium comprising code configured to direct a processor to:

obtain metadata from a node and store the metadata in fast storage medium; and

move the metadata from the fast storage medium to a slow storage medium.

15. The non-transitory, computer readable medium of claim 14, comprising code configured to direct the processor to determine if a memory location is available to store the metadata in a circular buffer of the fast storage medium.

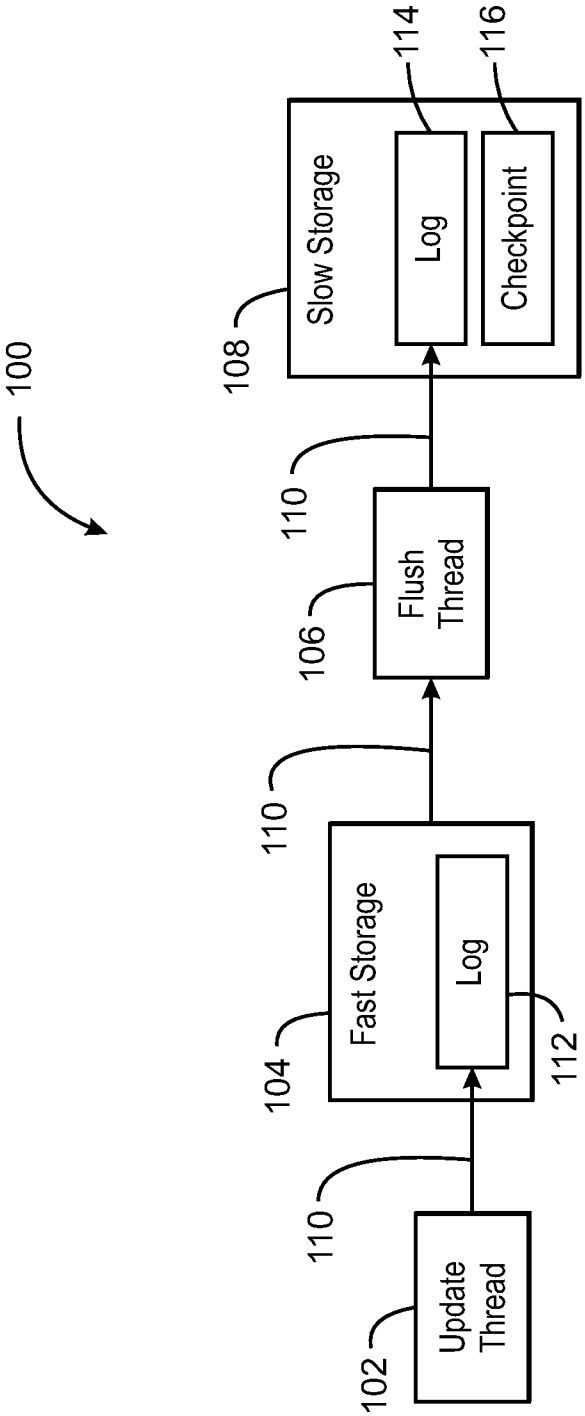
100
Update Thread
102
110
Fast Storage
104
Log
112
110
Flush Thread
106
110
Slow Storage
108
Log
114
Checkpoint
116

FIG. 1

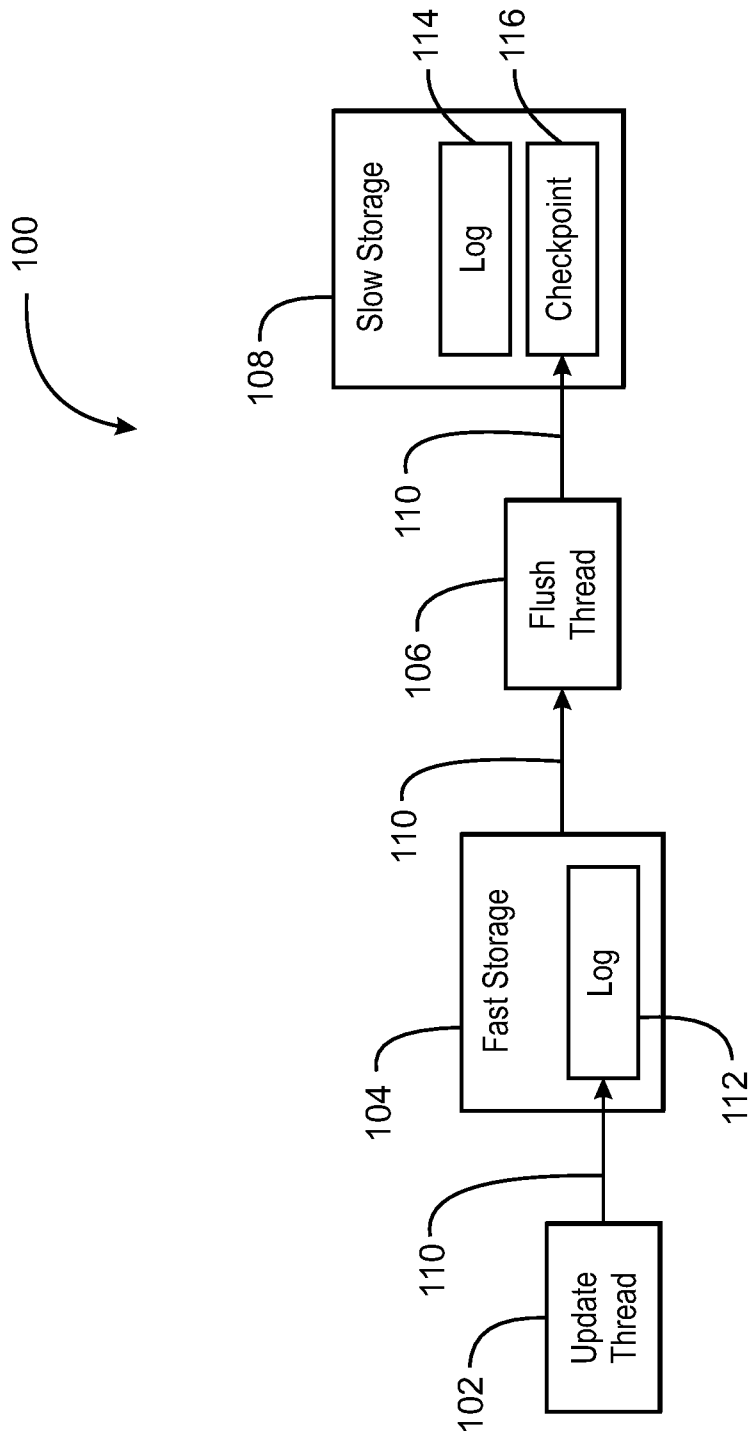
100
Update Thread
102
Fast Storage
Log
104
112
Flush Thread
106
Slow Storage
Log
Checkpoint
108
114
116
110
110
110

FIG. 2

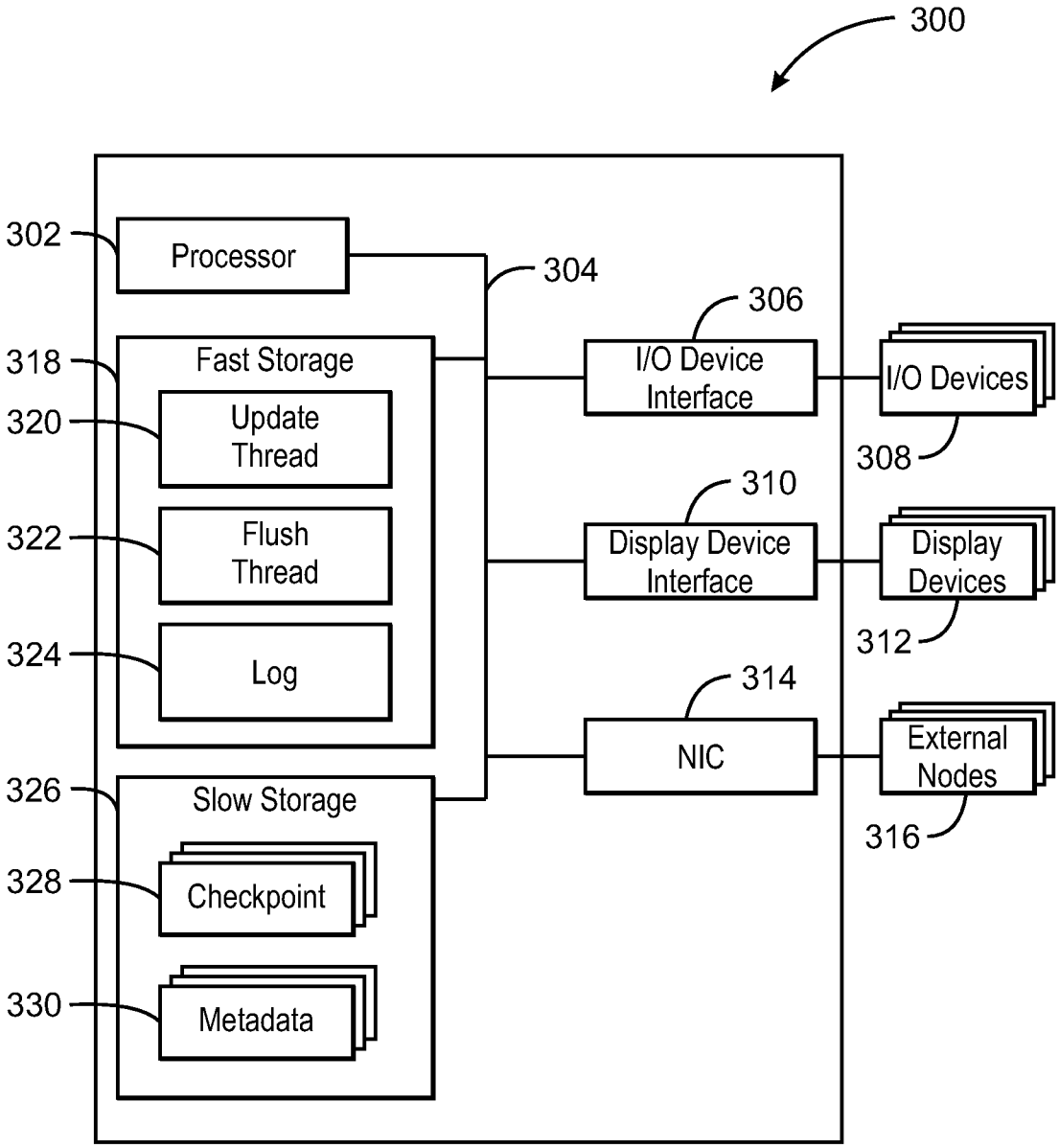
300
302
Processor
304
306
I/O Device Interface
I/O Devices
308
318
Fast Storage
320
Update Thread
322
Flush Thread
324
Log
310
Display Device Interface
Display Devices
312
314
NIC
External Nodes
316
326
Slow Storage
328
Checkpoint
330
Metadata

FIG. 3

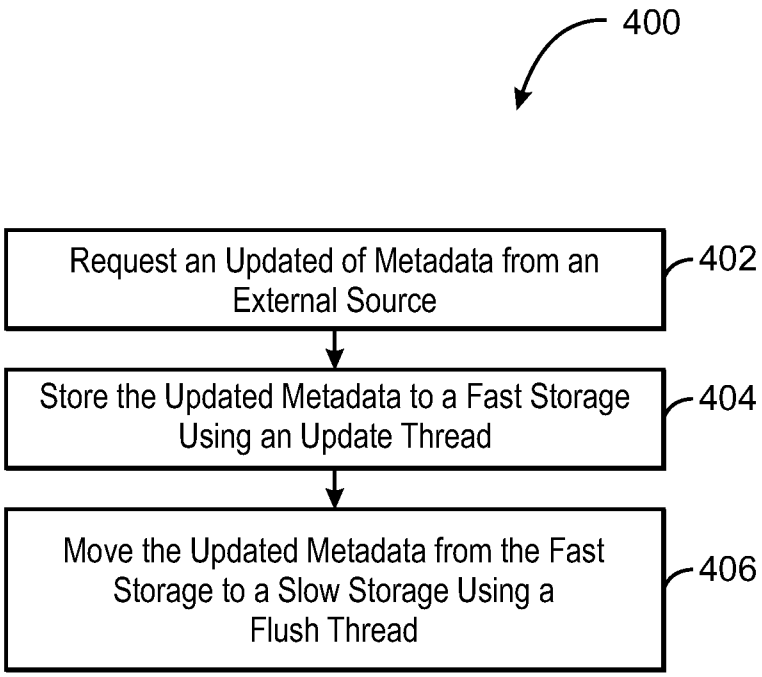
400
Request an Updated of Metadata from an External Source
402
Store the Updated Metadata to a Fast Storage Using an Update Thread
404
Move the Updated Metadata from the Fast Storage to a Slow Storage Using a Flush Thread
406

FIG. 4

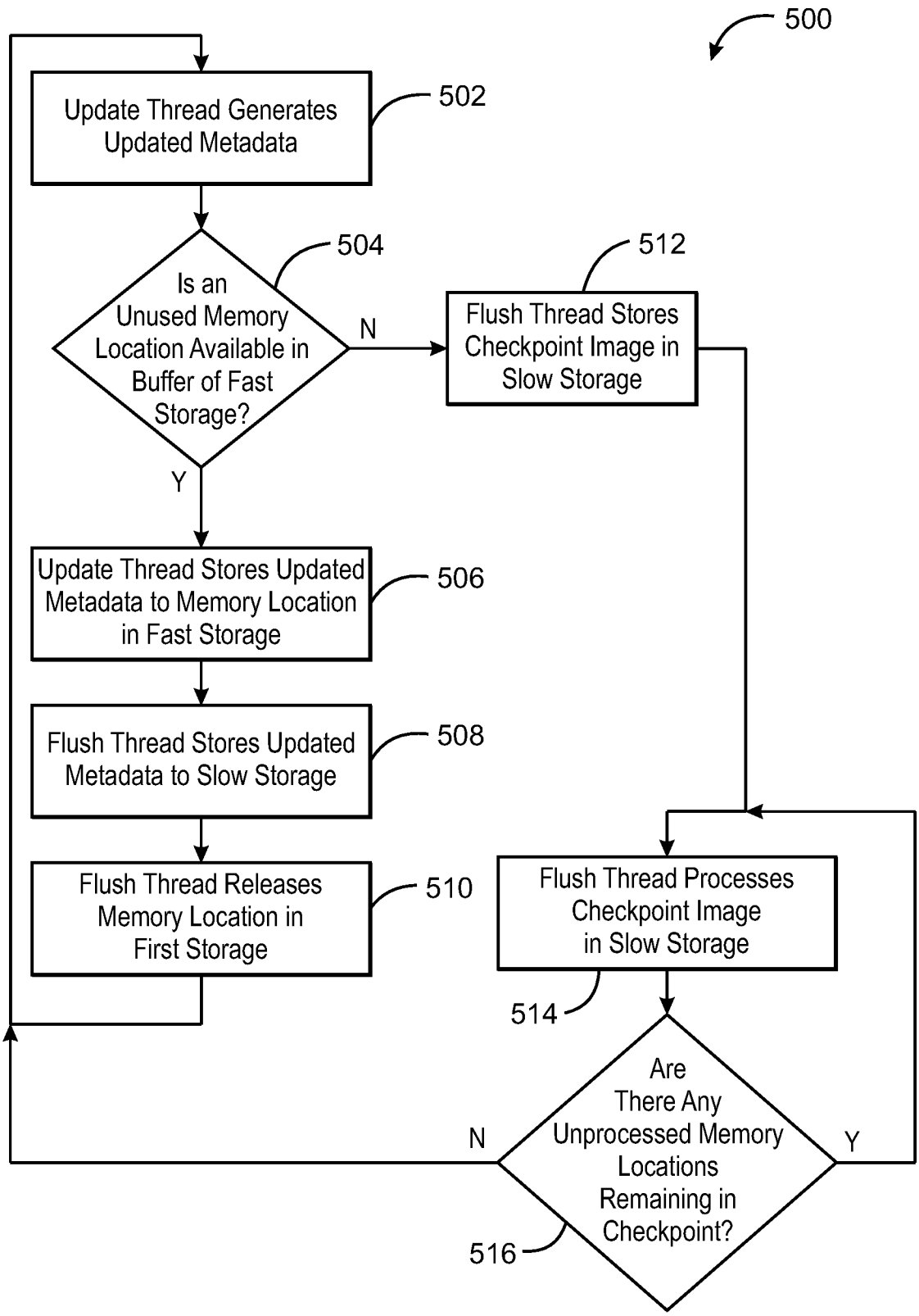
500
Update Thread Generates Updated Metadata
502
Is an Unused Memory Location Available in Buffer of Fast Storage?
504
N
Y
Flush Thread Stores Checkpoint Image in Slow Storage
512
Update Thread Stores Updated Metadata to Memory Location in Fast Storage
506
Flush Thread Stores Updated Metadata to Slow Storage
508
Flush Thread Releases Memory Location in First Storage
510
Flush Thread Processes Checkpoint Image in Slow Storage
514
Are There Any Unprocessed Memory Locations Remaining in Checkpoint?
516
N
Y

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/048001

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/08(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)
G06F 12/08; G06F 12/02; G06F 17/30; G06F 7/00; H04L 9/00; G06F 13/28

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: metadata, storing, moving, fast, slow, medium, thread, flush

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6732124 B1 (MICHIHIKO KOSEKI et al.) 04 May 2004 See column 1, lines 52-61; column 6, lines 47-50; column 9, lines 19-21; column 14, lines 58-59; column 25, lines 3-5; column 30, lines 9-53; column 33, lines 59-60; column 35, line 67; column 36 lines 1-2; column 37, lines 31-3; column 39, lines 51-52; claims 1-2, 4; and figures 1, 4.	1-7,12-15
Y		8-11
Y	US 2009-0222596 A1 (DAVID FLYNN et al.) 03 September 2009 See paragraphs [0020], [0076], [0095], [0125]; and figures 1, 3.	8-11
A	US 2012-0254120 A1 (RU FANG et al.) 04 October 2012 See paragraphs [0027]-[0028]; and figure 1.	1-15
A	US 2004-0015724 A1 (DUC PHAM et al.) 22 January 2004 See paragraphs [0014], [0016]; and figure 1.	1-15
A	EP 1840769 A1 (SUN MICROSYSTEMS, INC.) 03 October 2007 See paragraphs [0013], [0015]; and figure 1.	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
26 February 2015 (26.02.2015)	**26 February 2015 (26.02.2015)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. ++82 42 472 3473

Authorized officer
BYUN, Sung Cheal
Telephone No. +82-42-481-8262

Form PCT/ISA/210 (second sheet) (January 2015)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2014/048001

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6732124 B1	04/05/2004	JP 2000-284995 A	13/10/2000
		JP 3763992 B2	05/04/2006
US 2009-0222596 A1	03/09/2009	CN 102084331 A	01/06/2011
		EP 2286327 A1	23/02/2011
		JP 2011-521315 A	21/07/2011
		KR 10-2011-0048486 A	11/05/2011
		US 2008-0183882 A1	31/07/2008
		US 2011-0029496 A1	03/02/2011
		US 2012-005443 A1	05/01/2012
		US 2012-210021 A1	16/08/2012
		US 7836226 B2	16/11/2010
		US 8046500 B2	25/10/2011
		US 8205015 B2	19/06/2012
		WO 2009-126557 A1	15/10/2009
US 2012-0254120 A1	04/10/2012	None	
US 2004-0015724 A1	22/01/2004	AU 2003-281565 A1	09/02/2004
		US 7334124 B2	19/02/2008
		WO 2004-010630 A2	29/01/2004
		WO 2004-010630 A3	25/11/2004
EP 1840769 A1	03/10/2007	US 2007-0239797 A1	11/10/2007