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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 September 2017 (21.09.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/160318 A1

(51) International Patent Classification:
G06F 17/40 (2006.01) *G06F 15/16* (2006.01)
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2016/023260

(22) International Filing Date:
18 March 2016 (18.03.2016)

(25) Filing Language: English

(26) Publication Language: English

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

(72) Inventor: **JONES, Kevin Lloyd**; Longdown Avenue Stoke Gifford, Gifford Bristol BS34 8QZ (GB).

(74) Agents: **CREERON, Kerry T.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, 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, 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).

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))*

[Continued on next page]

(54) Title: DEDUPLICATING BLOCKS OF DATA

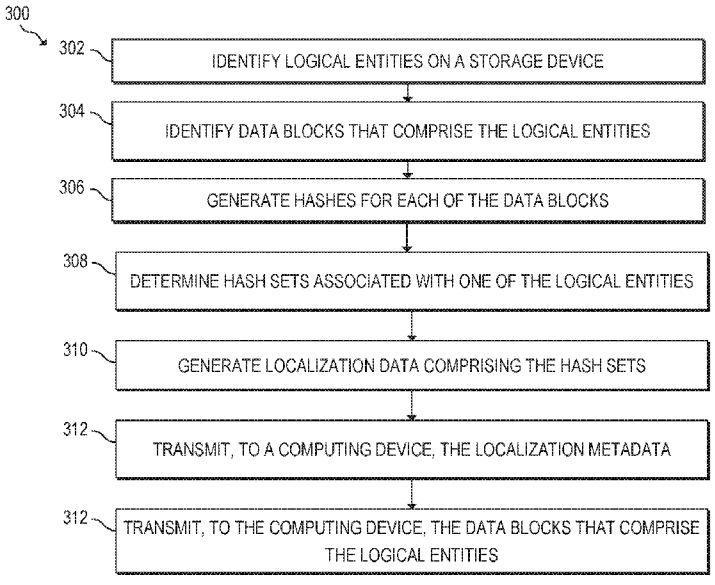

FIG. 3

(57) Abstract: Some examples provide a system including a processor and a non-transitory medium. The non-transitory medium comprises instructions stored thereon that, when executed, cause the processor to: receive a stream of data blocks, and receive localization metadata that indicates information about logical entities of the stream. The instructions, when executed, may further cause the processor to: reconstitute the logical entities from the stream of data blocks based on localization metadata and hashes derived from the data blocks, and deduplicate at least some of the data blocks of the reconstituted logical entities.

WO 2017/160318 A1

Published:

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

Deduplicating Blocks of Data

BACKGROUND

[0001] A computing device may store data blocks on a storage device. Some of the data blocks may be duplicates of each other.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Certain examples are described in the following detailed description and in reference to the drawings, in which:

[0003] FIG. 1 is a diagram of an example computing system that may perform deduplication of data blocks;

[0004] FIG. 2 is another diagram of an example computing system that may perform deduplication of data blocks;

[0005] FIG. 3 is a flowchart of an example method for performing deduplication of data blocks;

[0006] FIG. 4 is a flowchart of an example method for performing deduplication of data blocks; and

[0007] FIG. 5 is a block diagram of an example for performing deduplication of data blocks.

DETAILED DESCRIPTION

[0008] Datacenters may comprise numerous computing devices. In some examples, each computing device may store a copy of a same file. In some examples, a computing device may repeatedly store different evolutions or versions of a same file, however the file versions may share many common data blocks. That is, there may exist a duplicate data blocks across multiple file versions, and/or duplicate data blocks of a same file on multiple computing devices. For example, a file common to a particular operating system file may be duplicated among the file systems of computing devices running a same operating system (OS) version. As another example, computing devices repeatedly change (e.g. append) and store a file. However, many data blocks of the repeatedly changing file may be identical.

[0009] A computing device may support deduplication capabilities, which may reduce the storage of duplicate blocks of data. More particularly a device such as a deduplication appliance receives storage requests to write data to an address of a storage device. The deduplication appliance determines whether the blocks of data to be stored (data blocks) are already stored on the storage device.

[0010] If the data blocks are already present on the storage device, the deduplication appliance determines that the data blocks do not need to be written. Instead, the deduplication device maintains a single copy of the data blocks and creates a record that maps to the address of the single copy of the data blocks already present on the storage device. In this manner, the deduplication appliance avoids storing duplicate copies of data blocks, thereby reducing storage requirements. If the deduplication appliance receives a read request for the stored data block, the deduplication appliance reads the address stored in the pointer and retrieves the data blocks at the location referenced by the pointer.

[0011] However, a deduplication appliance may not be readily able to identify data blocks from virtual machine (VM) backups for deduplication. As an example, a backup agent that executes on a virtual machine may send a stream of data representing a VM backup. However, the stream may not identify particular files, data blocks or block boundaries as they are stored on the virtual machine. Additionally, the backup agent may send data blocks and files in an unordered fashion that does not correspond to a sequential ordering of blocks on the VM. Because the block sequencing of the backup stream may differ from the block sequencing of the deduplication store (i.e. storage coupled with a deduplication appliance), the deduplication appliance may fail to detect duplicate blocks because the block matching algorithm of the deduplication appliance degrades when blocks are mis-sequenced with respect to their storage sequence within the deduplication store.

[0012] The techniques described herein improve deduplication of such unordered backup streams. A backup agent executing on a source device (i.e. a computing device, such as a VM, to be backed up) generates and stores

localization metadata. The localization metadata comprises a plurality of hash sets. Each hash set comprises a plurality of hash values that are generated in a sequence. The sequence of the hash values is based on each of the blocks of a logical entity (e.g. a file) of the source machines' file system.

[0013] During a backup, the backup agent executing on the source device may transmit the localization metadata to the deduplication appliance discretely, or embedded within the backup stream. The localization metadata may have an embedded signature (e.g. magic data) to facilitate identification, extraction and reconstitution by the deduplication appliance. The backup agent also transmits the unordered stream comprised of the source machine's file system blocks to the deduplication appliance. The deduplication appliance reconstitutes the logical entities of the source device based on the localization metadata to detect duplicate blocks and to sequentially store non-duplicate blocks.

[0014] FIG. 1 is a conceptual diagram of an example computing system that may perform deduplication of data blocks. Computing system 100 is illustrated in FIG. 1. Computing system 100 comprises a device 102, which may be a deduplication appliance in some examples. Device 102 comprises a processor 104 and a non-transitory medium containing instructions stored thereon that, when executed, cause the processor to perform certain functionality.

[0015] Processor 104 may comprise a central processing unit (CPU), graphics processing unit (GPU), application specific integrated circuit (ASIC), digital signal processor (DSP), field programmable gate array (FPGA) or the like. Processor 104 may comprise any combination of the aforementioned. Processor 104 may also comprise one or more virtual devices, such as virtual processors of one or more virtual machines. Medium 106 may comprise software, firmware, non-volatile memory, or the like. Medium 106 may also be any combination of the aforementioned types of media. Processor 104 executes the instructions on medium 106.

[0016] Processor 104 receives localization metadata 118 and a stream 108. Stream 108 comprises hashes data block 116 and hashes 110. A

computing device transmitting stream 108 generates hashes 110 using a hashing algorithm for data blocks 116. In some examples, there may be a hash for each of data blocks 116. In some examples, one of hashes 110 may correspond to a plurality of data blocks 116.

[0017] Processor 104 may receive stream 108 in combination or discretely from localization metadata 106. Localization metadata 118 comprises information about logical entities in stream 108. Logical entities 112 may comprise files, data blocks, or the like. Based on localization metadata 118, hashes 110 and data blocks 116, processor 104 determines reconstituted logical entities 112.

[0018] Processor 104 determines deduplicated blocks 114 based on reconstituted logical entities 112. Processor 104 may determine deduplicated block 114 by comparing hashes or bit sequences of reconstituted logical entities 112 with hashes or bit sequences of a deduplication store.

[0019] FIG. 2 is another conceptual diagram of an example computing system that may perform deduplication of data blocks. FIG. 2 illustrates a computing system 200. Computing system 200 comprises a device 102, localization metadata 118, and stream 108 as in FIG. 1.

[0020] In the example of FIG. 2, backup agent 216 of virtual machine /virtual snapshot 214 may generate stream 108. Stream 108 comprises blocks 204A–204E. Blocks 204A–204C correspond to hashes 206A–206C. Blocks 204D–204E correspond to partial hashes 206D–206E. That is, in some examples, backup agent 216 may generate full hashes, but may store partial (e.g. truncated hashes) in data blocks of virtual machine/virtual snapshot 214 to reduce the disk space used to store the hashes.

[0021] In the example of FIG. 2, processor 104 receives localization metadata 118 and stream 108. Localization metadata 118 comprises hash sets 202. Hash sets 202 comprise a first hash set, “Set 1”, and a second hash set, “Set 2.” Set 1 indicates an ordering of blocks of stream that comprise a logical entity. Set 1 indicates that the reconstituted entity comprises the hashes 206C and 206A in that order.

[0022] Processor 104 determines and reconstitutes logical entities 112. Processor 104 attempts to deduplicate data blocks 114 based on whether data blocks 114 correspond to deduplicated block 114 of deduplication store 224. Deduplication store 224 may comprise a storage appliance, a SAN, or the like. Deduplicated blocks 226 are blocks that are already present on deduplication store 224. Processor 104 determines whether data blocks 114 match deduplicated blocks 226 based on a block matching algorithm. If a match exists, processor 104 attempts to deduplicate data blocks 226 by storing a reference to deduplicated blocks 226 rather than making a new copy of data blocks 114.

[0023] As indicated above, stream 108 may not be ordered based on logical entities of virtual machine / virtual snapshot 214. Some logical entities may be present in stream 108 that are not indicated by hash sets 202. In some examples, processor 104 may determine a logical entity that is not associated with hash sets 202, e.g. non-associated logical entity 212. Processor 104 may determine data blocks 228 of non-associated logical entity 212, and attempt to deduplicate data blocks 228.

[0024] Processor 104 receives set 1 and based on set 1, reconstitutes a logical entity by examining stream 108 for hashes 206C and 206A indicated by set 1. Based on hashes 206C and 206A, processor 104 identifies the corresponding blocks 204A and 204C. Processor 104 may reorder and/or concatenate blocks 206A and 206C if necessary to construct at least one of reconstituted logical entities 112.

[0025] Processor 104 performs a similar process with respect to set 2 of hash sets 202. Processor 104 matches hash 206B, partial hash 206D and partial hash 206E with the corresponding hashes of stream 108 to determine the ordering of blocks 204B, 204D, and 204E. Processor 104 then reconstitutes a logical entity based on the ordering indicated in set 2 of hash sets 202.

[0026] FIG. 3 is a flowchart of an example method for assembling operating system volumes. FIG. 3 comprises method 300. Method 300 may be described below as being executed or performed by a system, for example, computing system 100 (FIG. 1), or computing system 200 (FIG. 2). In various

examples, method 300 may be performed by hardware, software, firmware, or any combination thereof. Other suitable systems and/or computing devices may be used as well. Method 300 may be implemented in the form of executable instructions stored on at least one machine-readable storage medium of the system and executed by at least one processor of the system. In various examples, the machine-readable storage medium is non-transitory. Alternatively or in addition, method 300 may be implemented in the form of electronic circuitry (e.g., hardware). In alternate examples of the present disclosure, one or more blocks of method 300 may be executed substantially concurrently or in a different order than shown in FIG. 3. In alternate examples of the present disclosure, method 300 may include more or fewer blocks than are shown in FIG. 3. In some examples, one or more of the blocks of method 300 may, at certain times, be ongoing and/or may repeat.

[0027] Method 300 may start at block 302 at which point a computing device, such as a backup agent executing on virtual machine/virtual machine snapshot 214, may identify logical entities stored on a storage device, e.g. a storage device of the VM/snapshot. At block 304, virtual machine/VM snapshot 214 may identify blocks of data (e.g. data blocks 116 or data blocks 204A–204E) that comprise the logical entities. At block 306, virtual machine/VM snapshot 214 may generate hashes (e.g. hashes 206A–206C and partial hashes 206D–206E) for each of data blocks 204A–204E.

[0028] At block 308, virtual machine/VM snapshot 214 may determine hash sets (e.g. hash sets 202) that comprise at least some of the hashes. Each of each of the hash sets may be associated with one of the logical entities. Each of hash sets 202 may indicate data blocks of the associated logical entity. At block 310, virtual machine/VM snapshot 214 may generate localization metadata, e.g. localization metadata 118. The localization metadata may comprise hash sets 202.

[0029] At block 312, virtual machine/VM snapshot 214 may transmit, to a computing device, such as device 102, the localization metadata 118. At block 314, virtual machine/VM snapshot 214 may transmit, to the computing device, data blocks that comprise the logical entities.

[0030] FIG. 4 is a flowchart of an example method for assembling operating system volumes. FIG. 4 comprises method 400. Method 400 may begin at block 402. At block 402, a computing device, such as virtual machine/VM snapshot 214, may execute a backup agent, such as backup agent 216 (illustrated in FIG. 2). At block 404, virtual machine/VM snapshot 214 may identify logical entities stored on a storage device. At block 406, virtual machine/VM snapshot 214 may identify blocks of data that comprise the logical entities.

[0031] At block 408, virtual machine/VM snapshot 214 may generate hashes for each of the blocks of data. In some examples, at least some of the hashes, e.g. hashes 204A–204E, may be partial hashes. In some examples a backup agent may generate the hashes. At block 410, virtual machine/VM snapshot 214 may determine hash sets (e.g. hash sets 202) that comprise at least some of the hashes. The hash sets may indicate a sequence of the associated blocks of the logical entity.

[0032] At block 412, virtual machine/VM snapshot 214 may generate localization metadata, e.g. localization metadata 118. The localization metadata comprises magic data, wherein the magic data comprises an identifier value of the localization metadata. In some examples, the backup agent may generate the localization metadata. The localization metadata may comprise hash sets 202.

[0033] At block 414, virtual machine/VM snapshot 214 may transmit, to a computing device, such as device 102, the localization metadata 118. In some examples, virtual machine/VM snapshot 214 transmit the localization metadata separately from stream 108. For example, virtual machine/virtual machine snapshot 214 may transmit localization metadata 118 using a network socket, another type of separate network connection, or any other transport mechanism separate from that used to transmitting the data blocks. At block 416, virtual machine/VM snapshot 214 may transmit, to the computing device, data blocks that comprise the logical entities.

[0034] FIG. 5 is a block diagram of an example for deduplicating data blocks. In the example of FIG. 5, system 500 includes a processor 510 and a

machine-readable storage medium 520. Although the following descriptions refer to a single processor and a single machine-readable storage medium, the descriptions may also apply to a system with multiple processors and multiple machine-readable storage mediums. In such examples, the instructions may be distributed (e.g., stored) across multiple machine-readable storage mediums and the instructions may be distributed (e.g., executed by) across multiple processors.

[0035] Processor 510 may be one or more central processing units (CPUs), microprocessors, and/or other hardware devices suitable for retrieval and execution of instructions stored in machine-readable storage medium 520. In the particular example shown in FIG. 5, processor 510 may fetch, decode, and execute instructions 522, 524, 525, 528, 530, 532 to perform deduplication of data blocks.

[0036] As an alternative or in addition to retrieving and executing instructions, processor 510 may include one or more electronic circuits comprising a number of electronic components for performing the functionality of one or more of the instructions in machine-readable storage medium 520. With respect to the executable instruction representations (e.g., boxes) described and shown herein, it should be understood that part or all of the executable instructions and/or electronic circuits included within one box may, in alternate examples, be included in a different box shown in the figures or in a different box not shown.

[0037] Machine-readable storage medium 520 may be any electronic, magnetic, optical, or other physical storage device that stores executable instructions. Thus, machine-readable storage medium 520 may be, for example, Random Access Memory (RAM), an Electrically-Erasable Programmable Read-Only Memory (EEPROM), non-volatile memory, a storage drive, an optical disc, and the like. Machine-readable storage medium 520 may be disposed within system 500, as shown in FIG. 5. Machine-readable medium 520 is non-transitory in various examples. In this situation, the executable instructions may be "installed" on the system 500. Alternatively, machine-readable storage medium 520 may be a portable, external or remote storage

medium, for example, that allows system 500 to download the instructions from the portable/external/remote storage medium.

[0038] Referring to FIG. 5, stream receiving instructions 522 when executed by a processor, e.g. processor 510, may cause processor 510 to receive a stream of data blocks comprising hashes associated with the data blocks. Localization metadata receiving instructions 524 when executed, may cause processor 510 to receive localization metadata comprising sets of the hashes.

[0039] Data block identification instructions 526, when executed, may cause processor 510 to identify the data blocks based on the hashes of the stream. Logical entity identification instructions 528, when executed, cause processor 510 to identify data blocks of logical entities based on the hash sets.

[0040] Reconstitution instructions 530, when executed, may cause processor 510 to reconstitute the logical entities from the stream based on the hashes and the hash sets. Deduplication instructions 532 when executed, may cause processor 510 to deduplicate at least some of the data blocks of the reconstituted logical entities.

CLAIMS

1. A deduplicating method comprising:

identifying logical entities stored on a storage device;

identifying blocks of data that comprise the logical entities;

generating hashes for each of the data blocks;

determining hash sets comprising at least some of the hashes, wherein each of the hash sets are associated with one of the logical entities,

wherein each of the hash sets indicates blocks of the associated logical entity;

generating localization metadata which comprises the hash sets;

transmitting, to a computing device, the localization metadata; and

transmitting, to the computing device, the blocks of data that comprise the logical entities.

2 The method of claim 1, comprising:

transmitting, to the computing device, the localization metadata separately from the blocks of data.

3. The method of claim 1, wherein generating the hashes comprises generating the hashes by a backup agent executing on a virtual machine.

4. The method of claim 1, wherein the method is performed by a virtual machine, the method comprising:

executing, by the virtual machine, a backup agent; and

generating, by the backup agent, the localization metadata.

5. The method of claim 1,

wherein each of the hash sets indicates a sequence of the associated blocks of the logical entity.

6. The method of claim 1, wherein the localization metadata comprises magic data, wherein the magic data comprises an identifier value of the localization metadata.

7. The method of claim 1, wherein at least some of the hashes are partial hashes.

8. A device comprising:

a processor; and

a non-transitory medium storing instructions thereon that, when executed, cause the processor to:

receive a stream of data blocks;

receive localization metadata that indicates information about logical entities of the stream;

reconstitute the logical entities from the stream of data blocks based on localization metadata and hashes derived from the data blocks; and

deduplicate at least some of the data blocks of the reconstituted logical entities.

9. The device of claim 8, wherein the stream comprises data blocks of a virtual machine or a snapshot of a virtual machine, and

wherein the stream is not ordered based on logical entities.

10. The device of claim 8, wherein at least some of the hashes are partial hashes.

11. The device of claim 8,

wherein the medium comprises instructions stored thereon that, when executed, cause the processor to:

determine a logical entity that is not associated with any of the hash sets; and

deduplicate data blocks of the determined logical entity that are not associated with any of the hash sets.

12. The device of claim 8, further comprising a deduplication store,

wherein the localization metadata comprises hash sets, wherein the hash sets indicate an ordering of data blocks of the logical entities,

the medium further comprising instructions stored thereon that, when executed, cause the processor to:

deduplicate the at least some of the data blocks of the reconstituted logical entities responsive to determining that the at least some of the reconstituted logical entities are stored in the deduplication store.

13. A non-transitory machine-readable storage medium encoded with instructions, the instructions that, when executed, cause a processor to:

receive a stream of data blocks comprising hashes associated with the data blocks;

receive localization metadata comprising sets of the hashes;

identify the data blocks based on the hashes of the stream;

identify data blocks of logical entities based on the hash sets;

reconstitute the logical entities from the stream based on the hashes and the hash sets; and

deduplicate at least some of the data blocks of the reconstituted logical entities.

14. The non-transitory machine-readable storage medium of claim 13, wherein the processor to: receive the localization metadata prior to receiving the stream of blocks.

15. The non-transitory machine-readable storage medium of claim 13, wherein the localization metadata indicates an ordering of blocks of the logical entities of the stream.

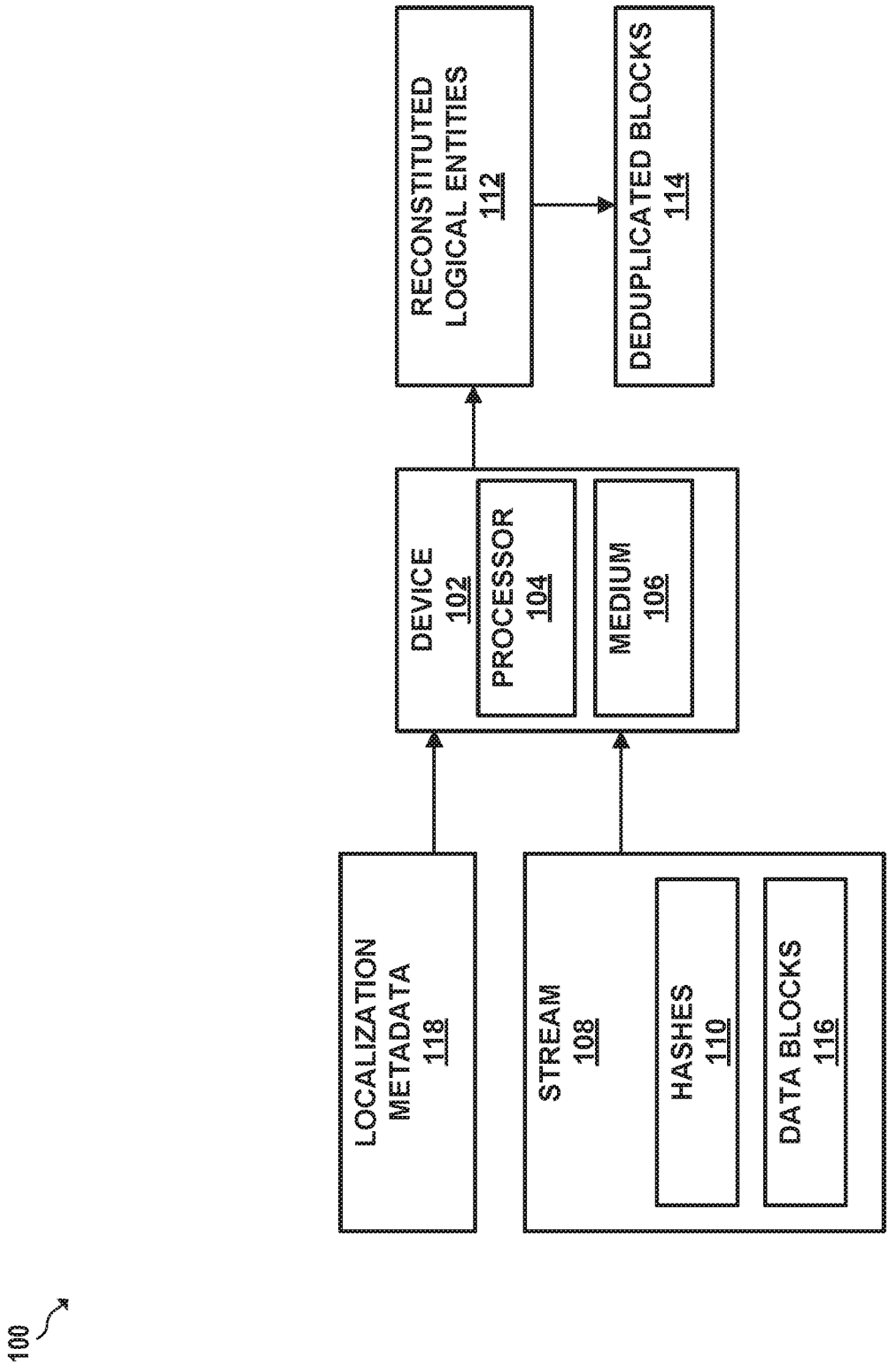
RECONSTITUTED LOGICAL ENTITIES 112
DEDUPLICATED BLOCKS 114
DEVICE 102
PROCESSOR 104
MEDIUM 106
LOCALIZATION METADATA 118
STREAM 108
HASHES 110
DATA BLOCKS 116
100

FIG. 1

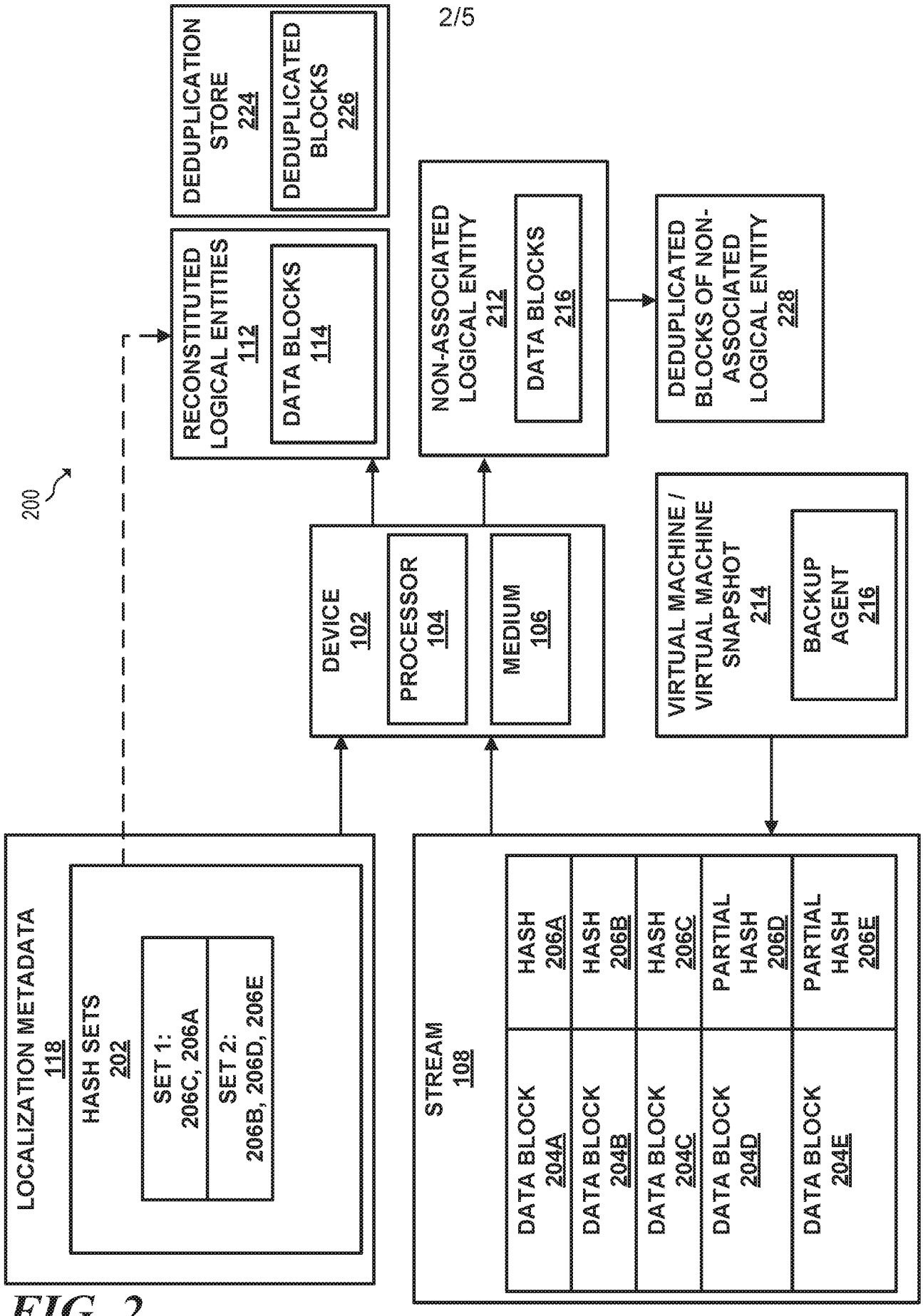
200
LOCALIZATION METADATA 118
HASH SETS 202
SET 1: 206C, 206A
SET 2: 206B, 206D, 206E
STREAM 108
DATA BLOCK 204A
DATA BLOCK 204B
DATA BLOCK 204C
DATA BLOCK 204D
DATA BLOCK 204E
HASH 206A
HASH 206B
HASH 206C
PARTIAL HASH 206D
PARTIAL HASH 206E
DEVICE 102
PROCESSOR 104
MEDIUM 106
VIRTUAL MACHINE / VIRTUAL MACHINE SNAPSHOT 214
BACKUP AGENT 216
RECONSTITUTED LOGICAL ENTITIES 112
DATA BLOCKS 114
DEDUPLICATION STORE 224
DEDUPLICATED BLOCKS 226
NON-ASSOCIATED LOGICAL ENTITY 212
DATA BLOCKS 216
DEDUPLICATED BLOCKS OF NON-ASSOCIATED LOGICAL ENTITY 228

FIG. 2

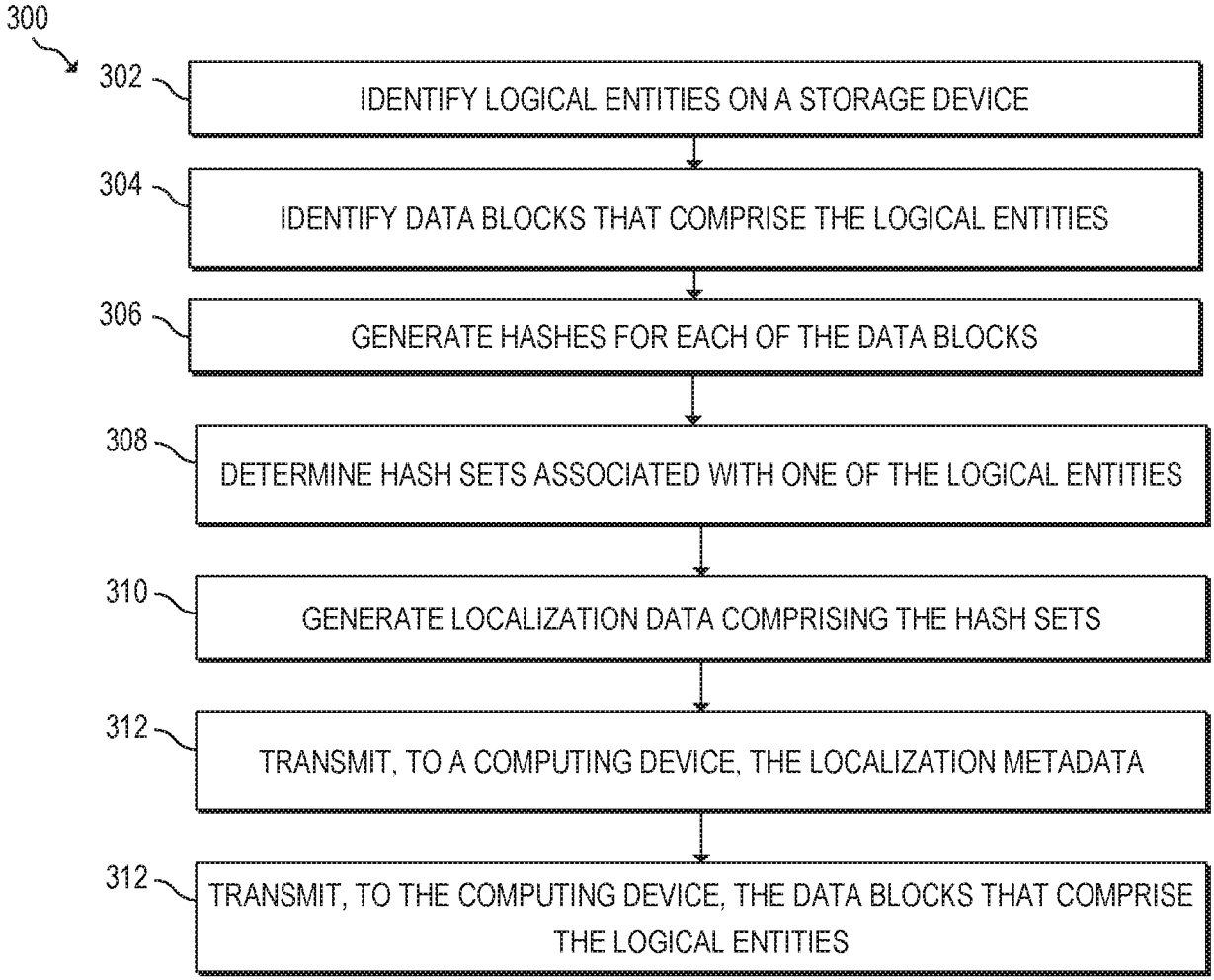
300
302
IDENTIFY LOGICAL ENTITIES ON A STORAGE DEVICE
304
IDENTIFY DATA BLOCKS THAT COMPRISE THE LOGICAL ENTITIES
306
GENERATE HASHES FOR EACH OF THE DATA BLOCKS
308
DETERMINE HASH SETS ASSOCIATED WITH ONE OF THE LOGICAL ENTITIES
310
GENERATE LOCALIZATION DATA COMPRISING THE HASH SETS
312
TRANSMIT, TO A COMPUTING DEVICE, THE LOCALIZATION METADATA
312
TRANSMIT, TO THE COMPUTING DEVICE, THE DATA BLOCKS THAT COMPRISE THE LOGICAL ENTITIES

FIG. 3

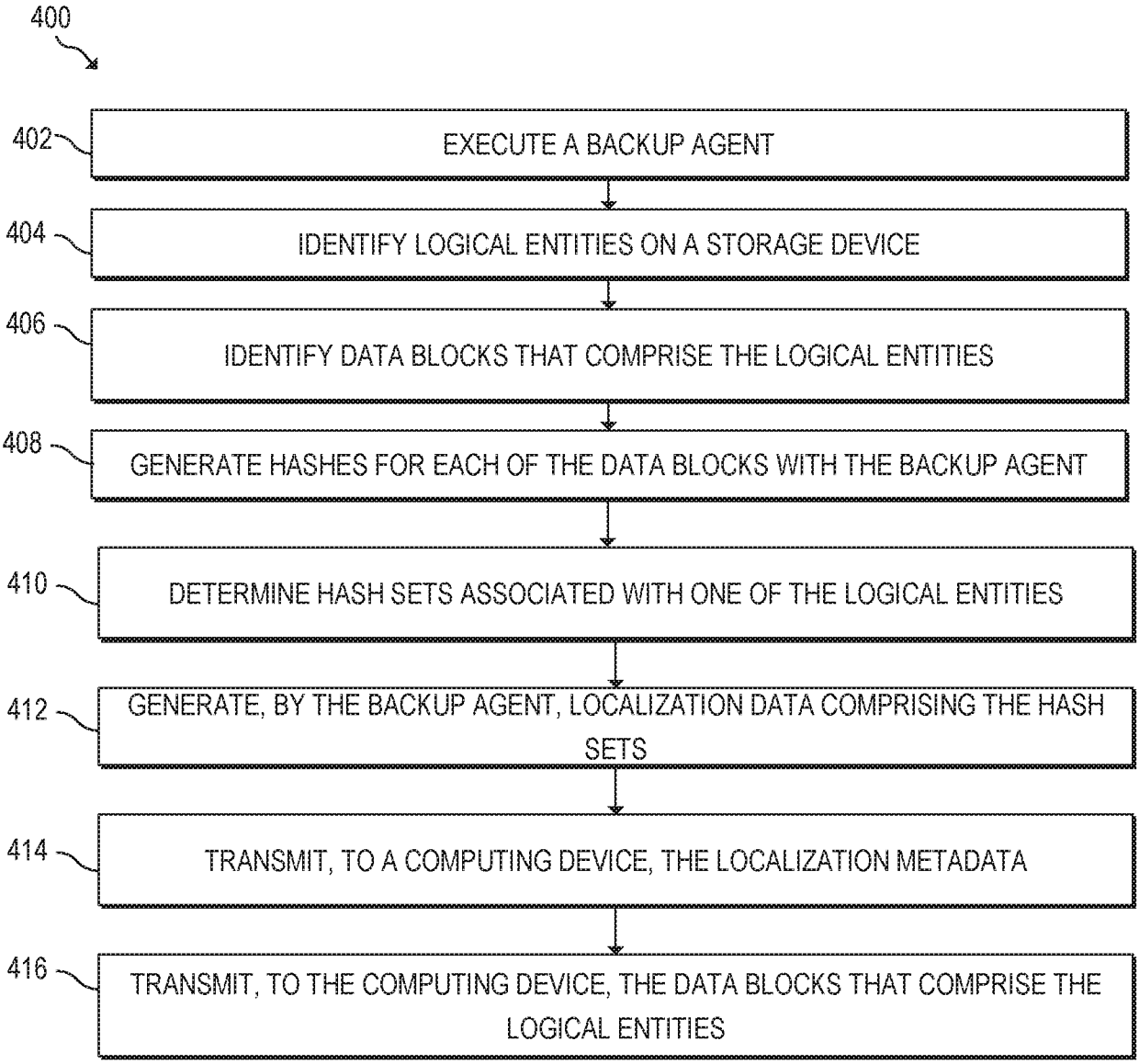
400
402
EXECUTE A BACKUP AGENT
404
IDENTIFY LOGICAL ENTITIES ON A STORAGE DEVICE
406
IDENTIFY DATA BLOCKS THAT COMPRISE THE LOGICAL ENTITIES
408
GENERATE HASHES FOR EACH OF THE DATA BLOCKS WITH THE BACKUP AGENT
410
DETERMINE HASH SETS ASSOCIATED WITH ONE OF THE LOGICAL ENTITIES
412
GENERATE, BY THE BACKUP AGENT, LOCALIZATION DATA COMPRISING THE HASH SETS
414
TRANSMIT, TO A COMPUTING DEVICE, THE LOCALIZATION METADATA
416
TRANSMIT, TO THE COMPUTING DEVICE, THE DATA BLOCKS THAT COMPRISE THE LOGICAL ENTITIES

FIG. 4

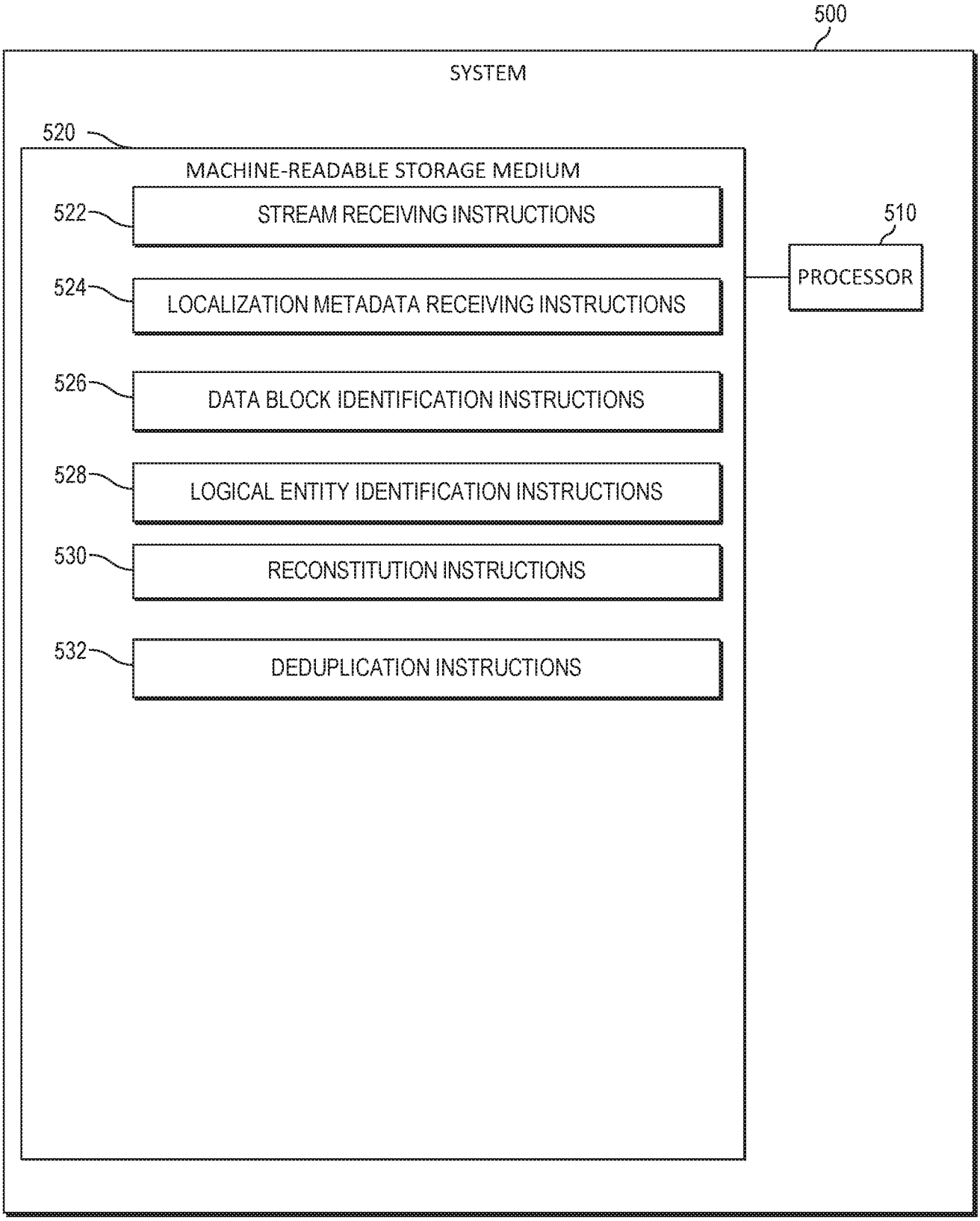
500
SYSTEM
520
MACHINE-READABLE STORAGE MEDIUM
522
STREAM RECEIVING INSTRUCTIONS
524
LOCALIZATION METADATA RECEIVING INSTRUCTIONS
526
DATA BLOCK IDENTIFICATION INSTRUCTIONS
528
LOGICAL ENTITY IDENTIFICATION INSTRUCTIONS
530
RECONSTITUTION INSTRUCTIONS
532
DEDUPLICATION INSTRUCTIONS
510
PROCESSOR

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/023260

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/40(2006.01)i, G06F 17/30(2006.01)i, G06F 15/16(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 17/40; G06F 12/02; G06F 17/30; G06F 7/00; G06F 12/00; G06F 9/06; G06F 15/16

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:deduplicate, identify, logical entity, generate, hash, localization metadata, 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.
X	US 2013-0018855 A1 (KAVE ESHGHI et al.) 17 January 2013 See paragraphs [0011], [0015]-[0019], [0033], and [0042]-[0055]; claims 1 and 15; and figures 2A-3.	1-15
A	US 2012-0047324 A1 (RODERICK B. WIDEMAN) 23 February 2012 See paragraphs [0013]-[0016]; and figure 1.	1-15
A	US 9152500 B1 (STORAGECRAFT TECHNOLOGY CORPORATION) 06 October 2015 See column 2, line 36 - column 3, line 5; and figure 4.	1-15
A	WO 2013-165388 A1 (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 07 November 2013 See paragraphs [0015]-[0021]; and figure 1A.	1-15
A	US 2014-0297603 A1 (ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE) 02 October 2014 See paragraphs [0011]-[0026] and figure 2.	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
29 December 2016 (29.12.2016)	**30 December 2016 (30.12.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2016/023260

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013-0018855 A1	17/01/2013	CN 102934097 A	13/02/2013
		CN 102934097 B	20/04/2016
		EP 2583183 A1	24/04/2013
		US 8799238 B2	05/08/2014
		WO 2011-159322 A1	22/12/2011
US 2012-0047324 A1	23/02/2012	GB 2496804 A	22/05/2013
		WO 2012-027165 A1	01/03/2012
US 9152500 B1	06/10/2015	US 2016-085630 A1	24/03/2016
WO 2013-165388 A1	07/11/2013	CN 104246718 A	24/12/2014
		EP 2845107 A1	11/03/2015
		EP 2845107 A4	23/12/2015
		US 2015-0066877 A1	05/03/2015
US 2014-0297603 A1	02/10/2014	KR 10-2014-0117994 A	08/10/2014