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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 June 2016 (16.06.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/091282 A1

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

(21) International Application Number:
PCT/EP2014/076947

(22) International Filing Date:
9 December 2014 (09.12.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building Bantian Longgang District, Shenzhen, Guangdong 518129 (CN).

(72) Inventor; and
(71) Applicant *(for US only)*: **TOAFF, Yair** [IL/DE]; c/o Huawei Technologies Duesseldorf GmbH Riesstr. 25, 80992 Munich (DE).

(74) Agent: **KREUZ, Georg**; Huawei Technologies Duesseldorf GmbH Messerschmittstr. 4, 80992 Munich (DE).

(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).

[Continued on next page]

(54) Title: APPARATUS AND METHOD FOR DE-DUPLICATION OF DATA

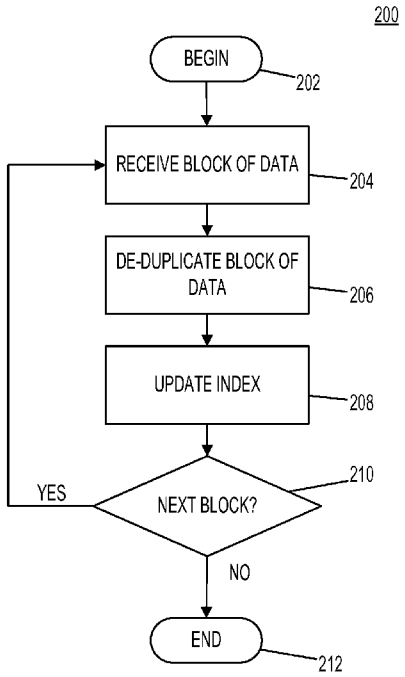

Fig. 2

(57) Abstract: An apparatus and a method for de-duplicating data are described. The apparatus may comprise an interface configured to receive a block of data and a de-duplication engine coupled to the interface and configured to de-duplicate the block of data based on a query of an index using a plurality of values calculated from the block of data and insert a plurality of new entries into the index, each entry corresponding to one of the plurality of values, wherein to insert the plurality of new entries into the index, the de-duplication engine is further configured to determine, for each new entry, a number of entries in the index corresponding to the same value, and if the number of entries exceeds a threshold, replace one of the entries with the new entry.

WO 2016/091282 A1

Published:

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

APPARATUS AND METHOD FOR DE-DUPLICATION OF DATA

FIELD OF THE INVENTION

The disclosure generally relates to an apparatus and method for de-duplicating data and, more particularly, to updating of an index used for de-duplication.

BACKGROUND

Data de-duplication is a data compression technique for eliminating duplicate copies of repeating data, which can be applied in various situations handling possibly repeating data, such as data storage, backup processes, data transmission and others. Data de-duplication is typically applied to reduce the size of data by replacing currently handled data with references to identical previously processed data. For example, in data storage, parts of data or byte patterns are identified and stored. As the analysis continues, other parts of data are compared to the stored copies and whenever a match occurs, the redundant parts of data may be replaced with references to the stored data.

In order to perform de-duplication, an index is typically used, which includes characteristics of and references to the stored data. Since the index has to be queried for each de-duplication of new data, a naïve implementation and maintenance of the index may significantly slow down processing. For example, if the size of the index becomes too large, the index cannot be maintained in local memory or cache and has to be at least partially stored on and retrieved from a larger, yet slower storage, a phenomenon which is also known as the disk lookup bottleneck. Accordingly, naïve approaches to de-duplication typically have excessive resource and throughput requirements and, therefore, often cannot be implemented in end-user products.

Other approaches for de-duplication attempt to reduce the size of the index. However, these approaches typically have a decreased de-duplication ratio since the index may not hold enough data to represent previously stored data, leading to a potential duplicate storage of data.

SUMMARY

In view of the above, one object of the present invention is to provide a fast and efficient de-duplication approach, both in view of utilization of resources and de-duplication ratio. The foregoing and other objects are achieved by the features of the independent claims. Further implementation forms are apparent from the dependent claims, the description and the Figures.

According to a first aspect of the present disclosure, an apparatus is provided which comprises an interface configured to receive a block of data. The apparatus further comprises a de-duplication engine coupled to the interface and configured to de-duplicate the block of data based on a query of an index using a plurality of values calculated from the block of data, and insert a plurality of new entries into the index, each entry corresponding to one of the plurality of values, wherein to insert the plurality of new entries into the index, the de-duplication engine is further configured to determine, for each new entry, a number of entries in the index corresponding to the same value, and if the number of entries exceeds a threshold, replace one of the entries with the new entry.

The de-duplication is based on an index query, wherein the index may be updated with a plurality of new entries that are derived from the plurality of values characterizing the block of data. Each entry of the index may include a key and a value, wherein the key of the entry may correspond to one of the plurality of values and the value of the entry may refer to the block of data. Accordingly, the index may be updated with entries, which may all refer to the block of data or at least parts of the block of data in order to enable subsequent de-duplication of data against the block of data.

The index may store several entries corresponding to the same value. These entries may refer to different blocks of data or at least different parts of blocks of data, which can be used to determine several candidate blocks of data for a given value for subsequent de-duplication. The index may hold a number K of entries referring to the same value. For example, the most recent K copies of entries with the same value may be kept regardless of a similarity between the data or versions of the data that they represent. This allows for finding of previously processed data from up to K versions backwards. This advantageously leads to a more efficient de-duplication since the block of data can be de-duplicated against most suitable blocks of data. However, since the number of entries corresponding to the same value is limited by the threshold, the size of the index can be directly controlled. This advantageously enables an efficient handling and storing of the index in memories with a limited size, such as local caches. In particular, the size

of the index can be directly controlled by setting of a suitable threshold and can be adapted to available resources and current requirements of the apparatus, even at runtime.

Hence, the threshold-based configuration of the index avoids storing and loading of large indices from potentially slow storage devices and, therefore, decreases the disk lookup bottleneck, while enabling lookup of a variety of previously processed data in order to find suitable hints for de-duplication. Furthermore, an implementation of the index update does not require detailed analysis of the index and complex processing of new entries in order to keep the index small enough to reside in local memory.

The de-duplication engine may be implemented in hardware, as software or as a combination of hardware and software. In particular, the de-duplication engine may be implemented as a dedicated processing module or a dedicated processor, which may be configured to execute the de-duplication approach according to aspects and implementations of the present disclosure.

As used throughout this description, the term "or" is understood as an inclusive conjunction or alternation, unless otherwise stated. Hence, the expression "A or B" means either A or B, or A and B, which can also be denoted as "A and/or B".

In a first possible implementation of the apparatus according to the first aspect as such, to de-duplicate the block of data, the de-duplication engine is further configured to query the index using the plurality of values, and if said querying yields a matching entry, retrieve characteristics of previously processed data according to the matching entry and compare the retrieved characteristics with the characteristics of the block of data. Each matching entry, which may be an entry that corresponds to at least one of the plurality of values of the block of data, may identify a previously processed block of data. Since the corresponding value of the matching entry may be calculated from a part of the previously processed block of data, a subsequent comparison of the characteristics of the previously processed block of data with the characteristics of the current block of data may be used to determine the similarity of further parts of the previously processed block of data with the current block of data. This has the advantage that, in a first stage, the index may be used to determine a suitable set of candidate blocks for de-duplication in order to enable a fast lookup of candidate blocks, and a comparison can be limited to a suitable set of candidate blocks in a second stage, thereby reducing the number of accesses to characteristics of previously processed blocks.

According to a second possible implementation of the apparatus according to the first aspect as such or according to the first implementation form of the first aspect, previously processed data is stored according to one or more versions, each entry of the index referring to one of said versions of the previously processed data. Accordingly, the previously processed data may be previously stored data. Even though the index can be kept relatively small, at the same time, index searches may result in different versions of the previously stored data. The threshold K can be used to directly control the size of the index in order to provide an optimal solution for the trade-off between index size and variety of indexed data. For example, current backup systems typically generate an initial full backup or snapshot, which may contain all data, and multiple incremental backups or snapshots, which may only include differences with regard to the initial full backup or a previous incremental backup. Hence, incremental backups may only contain a subset of a snapshot's content. In order to access the full snapshot's content, data from both the snapshot and previous snapshots or the initial full backup may be retrieved. Accordingly, backed up data can be retrieved according to several versions of data, which may be represented by one or more snapshots. Previous versions of data may also be present in primary storage systems and other storage approaches. By storing entries in the index referring to more than one version of data, the de-duplication ratio can be significantly improved.

De-duplication can be used in a variety of application scenarios, requiring efficient handling and compression of larger amounts of data, such as for storage of data, in backup systems and for data transmission or communication. It is to be understood, that even though aspects and implementations according to the present disclosure may refer to de-duplication for a particular application scenario, de-duplication is not restricted to this application scenario only and can be applied in various other application areas in order to exploit redundancy of previously processed data.

In a third possible implementation of the apparatus according to the first aspect as such or according to any of the preceding implementation forms of the first aspect, the de-duplication engine is further configured to determine whether the block of data is to be de-duplicated based on said query and if at least a part of the block of data is to be de-duplicated, replace the at least a part of the block of data with a reference to the matching data. The de-duplication process may use a container or a similar data structure for holding metadata for the block of data. The container may also include references to previously processed matching data as well as references to parts of the block of data which are not de-duplicated since no matching data has been processed previously. For example, the block of data or individual parts of the block of

data may be de-duplicated and stored according to results of the de-duplication in respective containers. The parts of the block of data that are not to be de-duplicated may be treated as new data. The new data may be added to an open container and further processed, such as written to a storage or transmitted via a network. The block metadata may be stored and used in subsequent de-duplication of further data blocks.

In a fourth possible implementation of the apparatus according to the first aspect as such or according to any of the preceding implementation forms of the first aspect, said index is a sparse index, each entry of the sparse index referring to a block of data. Accordingly, the index does not comprise entries for each small data unit, but refers to larger units of data, such as blocks of data, using keys which may be derived from parts of the data that may, however, correspond to the small data units. Hence, the value derived from a part of the block of data may be used to refer to the entire block of data. This is advantageous, since the size of the index may be kept small in order to avoid the disk lookup bottleneck. However, due to data locality, which describes a tendency of parts of data in a data stream or block of data to re-occur together in subsequent data streams or blocks, a sparse index leads to efficient retrieval of matching blocks of data with a sufficiently high hit ratio.

According to a fifth possible implementation of the apparatus according to the first aspect as such or according to any of the preceding implementation forms of the first aspect, the de-duplication engine is further configured to divide the block of data into a plurality of segments. The segments may have a fixed size or a variable size. Fixed size segment de-duplication may involve setting a segment size and segmenting the block of data into respective segments. Processing of fixed size segments may be advantageous, since the fixed size avoids variations in the total size of data thereby allowing for an easier organization and allocation of resources for caching or storage of the segments or blocks of data.

Preferably, the segments of the block of data have a variable size. For example, a 4 MB block of data may be divided into segments of 4 to 8 kB. However, it is to be understood that any other size of blocks or segments may be used and the present invention is not restricted to a particular size of blocks or segments. The block of data may be divided into variably sized segments based on results of an algorithm that may analyze the data of the block. Even though handling of variably sized segments may require a more complex organization of the individual segments, de-duplication based on variable segments may be more efficient since flexible segment boundaries may better reflect small changes in individual segments. For example, an inclusion of even a small amount of data into a block of data may lead to a shift of segment

boundaries and therefore to changes in all subsequent segments if segments with a fixed size are used. In contrast, segment boundaries may be independent of local changes in approaches using variably sized segments since segment boundaries may be determined based on characteristics of the data in the block of data, such as particular data values or functions applied to the data.

Preferably, the block of data may be divided into a plurality of segments using a rolling hash function on the block of data. A rolling hash function is a hash function where the input is hashed in a window that moves through the block of data. For example, Rabin-Karp rolling hash or cyclic polynomial hashing can be used as a rolling hash function. Preferably, cyclic polynomial hashing is used, since it has better performance than Rabin-Karp. Cyclic polynomial hashing is a form of tabulation hashing based on a hash function h. The hash function h may be an array or a hash table mapping characters to random integers. The rolling hash function may be applied on every byte of the block until the hash value modulus the expected average size of segments is equal to a certain value. Hence, as soon as the output of the rolling hash function reaches a certain value, a segment boundary may be determined and a respective segment may be cut. For example, in order to determine segment boundaries, the rolling hash may be calculated mod 4k and as soon as the rolling hash is equal to 0 a hash boundary can be set. It is to be understood that any other value can be used as long as it is smaller than or corresponds to an expected size of segments.

In a sixth possible implementation of the apparatus according to the fifth implementation form of the first aspect, for a plurality of blocks of data to be de-duplicated, each block of data comprises the same number of segments. Additionally or as an alternative at least some blocks of data may comprise a different number of segments per block. If the de-duplication engine is configured to divide blocks of data into a plurality of segments with a fixed size, each block of data having the same size will have the same number of segments. However, if the de-duplication engine is configured to divide blocks of data into segments of a variable size, each block of data may comprise a different number of segments per block. However, even if segments with a variable size are used, the de-duplication engine may be configured to generate blocks of data with a same number of segments, for example, by adding virtual segments at the beginning, inside, or at the end of each block. The virtual segments may be empty or have a size of zero. This may be advantageous for processing of individual blocks, since processing of blocks with the same number of segments may simplify organization and handling of blocks and respective segments.

According to a seventh possible implementation form of the fifths or sixth implementation form of the apparatus according to the first aspect, the de-duplication engine is further configured to calculate the plurality of values based on hash values of the segments. Accordingly, the hash values of the segments may be used as the values for performing the index query in order to determine suitable candidates for de-duplication of the block of data. The algorithm for calculating the hash values may be selected with regard to a probability of false positives produced by the algorithm, which should be extremely unlikely. For example, the SHA1 hash algorithm can be used to calculate the hash values. However, it is to be understood that any other hash algorithm with desired properties of false positives may be applied to calculate the hash values of individual segments.

In an eighth possible implementation form of the seventh implementation form of the apparatus according to the first aspect, the de-duplication engine is further configured to select a number of said hash values as the plurality of values. For example, the hash values may be compared with a threshold, or the value or bit characteristics of the hash values may be further analyzed. For example, hash values having a number of most significant or least significant bits equal to 0 may be selected as the plurality of values. Preferably, the selected number of hash values may be further processed to generate the plurality of values. For example, only a number of most significant or least significant bits of the hash values may be used as the plurality of values. According to an example, the hash value of SHA1 may be 20 bytes. In order to use the values for the index, the hash value may be treated as two parts of 4 bytes + 16 bytes. Out of the list of hash values of the segments in the block, values may be chosen such that the 4 bytes of the hash values are local maxima. For instance, four values may be chosen, one on each 1 MB part of a 4 MB block. Of these hash values, the remaining 16 bytes may be taken and used as the key in the index. It is to be understood that the split and selection of the values into 4+16 bytes is an example only and any other split of the hash value may serve to select and process a number of hash values as the plurality of values. However, the part that is used for choosing is preferably not the most significant part of the key in order to avoid uniformity of the values in the index. The local maxima can be replaced by any other function that gives an ordered output, such as min, taking into account the locality of the data. Otherwise, a small change in the data could affect the selected representatives or hash values in the index if all of them are from segments from the same area of the block.

Hence, for each block, a set of hash values or fingerprints corresponding to the plurality of values may be used to query the index. If the values or fingerprints in the index are equal, this

may serve as an indication that the previously processed block of data may be similar to the current block of data. Accordingly, the determination of similar blocks of data may be performed according to the plurality of values in order to determine suitable candidates that are subsequently compared to the current block of data in order to determine matching blocks of data. With regard to the above example, four fingerprints or hash values may be selected per block as the plurality of values and if all fingerprints are identical with the previously processed block, it may be assumed that both blocks are at least partially identical. Any changes of data in a current block may be reflected in only a reduced number of fingerprints being identical with fingerprints of a previously processed block, such as two or three fingerprints. Sampling of hash values to produce the plurality of selected values is advantageous since the number of queries for each individual block of data is reduced to the number of selected values.

According to a ninth possible implementation form of the apparatus according to the first aspect as such or according to any of the preceding implementation forms of the first aspect, the entries of the index are replaced according to a replacement strategy. During index update, the plurality of values, which may correspond to individual hash values of the segments of the block of data, are inserted into the index as new entries. Each entry may contain a key and a value, wherein the key of the entry may be the value used in de-duplication, such as the hash value of a segment, and the value of the entry may refer to the block of data. If the index does not comprise any entry having a key with the same value, the new entry may be included into the index. If the index already includes an entry having a key with the same value, the de-duplication engine may determine the number of entries in the index having a key with the same value. If the number of entries is below the threshold, the new entry may be included into the index and, optionally, a reference count can be increased. If the number of entries exceeds or equals to the threshold, the de-duplication engine may select one of the entries according to the replacement strategy.

Preferably, the replacement strategy includes replacing the oldest entry. The de-duplication engine may be configured to determine the age of each entry having a key with the same value, such as by retrieving respective time stamps of the entries or time stamps or versions of the corresponding blocks of data that are being referred to by the individual entries. After determining the oldest entry or the entry referring to the oldest data or version of data, the oldest entry may be replaced with the new entry.

In a tenth possible implementation form of the apparatus according to the first aspect as such or according to any of the preceding implementation forms of the first aspect, said query yields

an ordered list of entries of the index, wherein said entries are ordered according to a rank. Preferably, the rank is calculated based on a number of matching values of the plurality of values. The plurality of values, such as the hash values of all or a selected number of segments, may be searched for in the index and an ordered list of entries having at least one matching entry may be returned. The list may include references to the previously processed blocks referred to by the respective entries. The list may be ordered with regard to a probability of a good match of the entire previously processed block with the current block of data. The rank may also take into account which of the blocks in the list was most recently used as a tiebreaker during de-duplication. The rank may also take into account the age of each entry, such that most recent entries may be prioritized. This is advantageous, since in most cases changes in processed data accumulate and it is more likely that blocks referred to by most recent entries will be similar to the currently processed block of data. The list may also be sorted according to a first characteristic, such as a number of matching entries, and subsequently according to a second, third and any further characteristic, such as the age of the entries. Furthermore, the list can be truncated to a number of entries and either all entries in the list may be used as hints for de-duplication or the truncated number of entries can be used as hints for de-duplication. For example, an index search of four keys may result in two sets of three values of respective entries and the rank may be calculated of each value according to its age, wherein the most recent may have the highest rank. The rank of the sets may be the sum of the ranks of its values. For example, three values may be kept for each key in respective entries and the first may have a rank of 1000, the second a rank of 900 and the third a rank of 800. Hence, the number of values in the set may be the most significant aspect and the age of the value may be the tiebreaker between sets of the same size. Time stamps for individual entries in the index may be kept in respective values in order to determine the age. Additionally or as an alternative, the values of the entries may be kept in ordered lists according to the age and when a new one is inserted into the index, the oldest may be thrown.

According to an eleventh possible implementation form of the apparatus according to the first aspect as such or according to any of the preceding implementation forms of the first aspect, said interface is configured to receive the block of data in a block write command. The interface and the de-duplication engine may have a basic I/O scope of blocks. The size of a block may be preferably in the range of 4 MB to 16 MB. However, it is to be understood that any block size may be used and the present invention is not limited to a particular size of blocks.

According to a second aspect, a user equipment is provided which comprises an apparatus according to one implementation form. In particular, the user equipment may comprise an apparatus including an interface configured to receive a block of data and a de-duplication engine coupled to the interface and configured to de-duplicate the block of data based on a query of an index using a plurality of values calculated from the block of data, and insert a plurality of new entries into the index, each entry corresponding to one of the plurality of values, wherein to insert a plurality of new entries into the index, the de-duplication engine is further configured to determine, for each new entry, a number of entries in the index corresponding to the same value, and if the number of entries exceeds a threshold, replace one of the entries with a new entry. It is to be understood that the user equipment according to the second aspect may include any combination of features of an apparatus according to one or more implementation forms of the first aspect.

In a first possible implementation form of the user equipment according to the second aspect as such, the user equipment is one of a storage or network device with de-duplication functionality. For example, the user equipment may be a network attached storage (NAS) or a secondary storage device. The user equipment may also be a communication device.

In a third aspect, a method for de-duplicating data is provided comprising the steps of receiving a block of data, de-duplicating the block of data based on a query of an index using a plurality of values calculated from the block of data, and inserting a plurality of new entries into the index, each entry corresponding to one of the plurality of values, wherein said inserting the plurality of new entries into the index includes determining, for each new entry, a number of entries in the index corresponding to the same value, and if the number of entries exceeds a threshold, replacing one of the entries with the new entry. The method according to the third aspect provides for an efficient de-duplication using an index of a reasonable size in order to avoid the disk lookup bottleneck while maintaining the ability of finding several references to previously processed data in order to choose best suitable blocks of data for de-duplication.

In a first possible implementation form of the method according to the third aspect as such, said de-duplicating the block of data includes querying the index using the plurality of values and if said querying yields a matching entry, retrieving characteristics of previously processed data according to the matching entry and comparing the retrieved characteristics with the characteristics of the block of data.

According to a second possible implementation of the method according to the third aspect as such or according to the first implementation form of the third aspect, said method further comprises determining whether the block of data is to be de-duplicated based on said query, and if at least a part of the block of data is to be de-duplicated, replacing the at least a part of the block of data with a reference to the matching data.

In a third possible implementation of the method according to the third aspect as such or according to any of the preceding implementation forms of the third aspect, the method further comprises dividing the block of data into a plurality of segments.

In a fourth possible implementation of the method according to the third aspect as such or according to any of the preceding implementation forms of the third aspect, the method further comprises calculating the plurality of values based on hash values of the segments.

According to a fifth possible implementation of the method according to the third aspect as such or according to any of the preceding implementation forms of the third aspect, the method further comprises selecting a number of said hash values as the plurality of values.

According to a sixth possible implementation of the method according to the third aspect as such or according to any of the preceding implementation forms of the third aspect, the method further comprises replacing the entries of the index according to a replacement strategy, including replacing the oldest entry.

In further possible implementation forms of the method according to the third aspect as such or according to any of the preceding implementation forms of the third aspect, the method may comprise steps directed at functionality of one or more implementation forms of the apparatus according to the first aspect or the user equipment according to the second aspect in any combination.

According to a fourth aspect, a computer program with a program code for performing a method according to one implementation form of the third aspect is provided, wherein the method is performed when the computer program runs on a computer.

According to a fifth aspect, a computer program product is provided, which comprises a readable storage medium storing program codes thereon for use by a user equipment, the program code comprising instructions for performing individual method steps according to possible implementation forms of the method according to the third aspect.

Implementation forms of the invention can thus provide an efficient de-duplication approach, which uses an optimized index that enables access to a number of previously processed data in order to determine suitable data for de-duplication.

BRIEF DESCRIPTION OF THE DRAWINGS

The specific features, aspects and advantages of the present disclosure will be better understood with regard to the following description and accompanying drawings.

Fig. 1 is a high level system diagram including a plurality of apparatuses in accordance with one or more embodiments of the invention;

Fig. 2 is a flow chart of a method for de-duplication according to one or more embodiments of the present invention; and

Fig. 3 is another flow chart of a method for de-duplication according to one or more embodiments of a present invention.

DETAILED DESCRIPTION

Fig. 1 shows a high level system diagram including a plurality of apparatuses according to one or more embodiments of the present disclosure. In the system 100, apparatuses 102a, 102b, and 102c may perform de-duplication operations on data in order to compress the data or reduce the amount of data for subsequent processing based on redundancy in previously processed data, in a variety of application areas.

Apparatus 102a may be a storage device which may include a de-duplication engine 104 and storage 106 including one or more storage devices, such as an array of hard disk drives, solid-state drives, flash memories and others in any combination. The apparatus 102a may be coupled to a processing device 108, such as a laptop or a personal computer. The processing device 108 may provide data to the apparatus 102a for storage or backup. For example, apparatus 102a may be a secondary storage, a network attached storage or any other suitable storage device configured to store or backup data. Computing device 108 may access apparatus 102a via a wireless or wired link or network, as indicated by the arrows between apparatus 102a and processing device 108.

The de-duplication engine 104 may provide de-duplication functionality, which may be regarded as a specialized data compression technique for eliminating duplicate copies of replicate data in the data stored in storage 106. The de-duplication engine 104 may implement de-duplication by storing data in containers. The incoming data may be provided to the de-duplication engine 104 as one or more blocks of data. The de-duplication engine 104 may divide each block of data into a sequence of segments. For each block, the de-duplication engine 104 may determine a suitable set of blocks that have been previously stored in storage 106 in order to de-duplicate the current block of data. The determination of suitable segments or blocks of data is mainly based on data locality properties of the underlying data, which assumes that a similar sequence of data will reappear in subsequent blocks of data.

In order to find a best block of data or segment of data to de-duplicate against, the de-duplication engine 104 may use an index. The index may be stored in a memory or cache (not shown) in order to enable a quick access to the index. The index may be either a full index or a sparse index. The full index may contain fingerprints for every small piece or unit of data of the block of data that entered the apparatus 102a. A sparse index may contain a hint for every larger piece or unit of data that entered the apparatus 102a, such as respective blocks of data. Hence, the sparse index may contain hints allowing for a determination of best candidate areas for searching for matches based on data locality. When such a hint is found, the characteristics or fingerprints of that area may be loaded from storage 106 and compared against the currently processed block of data. Furthermore, the sparse index may include several entries corresponding to the same hint. This allows for consideration of a plurality of versions of the data irrespective of a similarity with a most recent version of the data, which may be still valid in the storage 106.

After de-duplication of the block of data and respective storage of either new data in the storage 106 or references to older versions of the data in storage 106, the de-duplication engine 104 may update the index by inserting a plurality of new entries into the index, each entry corresponding to value used for querying the index during de-duplication. New entries may be inserted into the index by determining a number of entries in the index corresponding to the same value. If the number of entries exceeds a threshold, one of the entries is replaced with the new entry.

Apparatus 102a may address the de-duplication engine 104 in the context of a block write command. The de-duplication engine 104 may analyze the new block of data and generate keys that should be searched for in the index. According to results of the search in the index,

fingerprints of suitable areas of previously stored blocks of data may be loaded from storage 106 and the new block of data may be de-duplicated against the previously stored blocks of data. Hints may be generated for the new block of data and inserted into the index. In order to keep the index small while still enabling access to older versions of the data, the index may keep up to K hints or entries with the same value, while the K+1 copy may be removed according to any suitable replacement strategy, such as LRU, date, or any other logical policy. The K copies of the same hint or entries for the same value enable to find a best candidate when the same data is seen again by the apparatus 102a and also find older versions of the data, which may be needed for cases where an old snapshot was restored and written again, for example.

Apparatus 102b may comprise de-duplication functionality similar to the apparatus 102a. However, the apparatus 102b may apply de-duplication on data which are to be transmitted over a network 110. For example, the processing device 108 may submit data to apparatus 102b in order to transmit the data to a destination via the network 110. The apparatus 102b may be connected to the network 110 using any kind of link or connection technology, such as using a wireless or a wired connection. The apparatus 102b may comprise a de-duplication engine, which may be similar to the de-duplication engine 104 of apparatus 102a. The apparatus 102b may further include a local storage (not shown) in order to temporarily buffer the data. De-duplication of a current block of data in apparatus 102b may be performed against previously transmitted data using an index, which may be used, maintained and updated in a similar way as the index in apparatus 102a. The apparatus may de-duplicate the data and send respective compressed data via the network 110. Accordingly, if redundant data is found in a current block of data, apparatus 102b may replace the redundant data with a reference to previously transmitted data to reduce the amount of data transmitted over the network. The destination may restore the compressed data by restoring references to previously received data.

Apparatus 102c may be any kind of user equipment, such as a communication or mobile device, tablet device or any other portable or stationary computing device, which may provide de-duplication functionality in order to exploit redundancy of data in larger data sets. For example, apparatus 102c may de-duplicate data for storage or communication purposes. The apparatus 102b may also perform reciprocal functionality. For example, apparatus 102b may receive compressed de-duplicated data from apparatus 102b via network 110 and may restore the received compressed de-duplicated data by replacing references to previously received data with the previously received data.

Fig. 2 shows a flow chart of a method according to one embodiment of the present disclosure. The method 200 may be a computer-implemented method for de-duplicating data. The method 200 may start in item 202 and proceed with item 204, wherein a block of data may be received. For example, the block of data may be received by an interface of an apparatus for de-duplication, such as one of the apparatuses 102a, 102b, and 102c shown in Fig. 1.

The method 200 may proceed with item 206, wherein the block of data may be de-duplicated based on a query of an index using a plurality of values calculated from the block of data. The de-duplication of the block of data may comprise dividing the block of data into a plurality of segments, which may be of a variable length, for example by using a rolling hash function, and calculating hash values for the individual segments of the block of data. Based on the hash values or a selected number of hash values, the index may be queried to provide candidate entries of the index, which may refer to previously processed blocks of data having one or more matching segments, which may represent suitable candidates for de-duplication. The current block of data may be de-duplicated against most suitable candidate blocks by replacing matching segments in the block of data with references to the matching previously processed data.

The method 200 may proceed with item 208, wherein the index may be updated by inserting a plurality of new entries into the index, each entry corresponding to one of the plurality of values. The plurality of new entries may be inserted into the index by determining, for each new entry, a number of entries in the index corresponding to the same value. If no entry in the index corresponding to the same value is found, the new entry may be directly included into the index. If the number of entries corresponding to the same value is below a threshold, the new entry may be added to the index and a reference count for entries with the same value may be increased. If the number of entries exceeds a threshold, one of the entries may be replaced with the new entry, for example, by following a replacement strategy, such as replacing the oldest entry in the index.

After update of the index in item 208, the method 200 may proceed with item 210 in order to determine whether a next block of data is to be processed. If a next block of data is to be processed, the method may re-iterate item 204. If no further block is to be processed, the method may end in item 212.

Fig. 3 is a flow chart of a method according to one embodiment of the present disclosure. The method 300 may be a computer-implemented method. The method 300 may start with item 302

and may proceed with item 304, where a write command may be received including a logical block of data to be de-duplicated.

In item 306, the block may be split into segments of variable or fixed length which may serve as a basis for calculation of hash values characterizing the block of data. The hash values may be evaluated in item 308 in order to determine a suitable block for de-duplication in item 310, such as by querying an index referring to previously processed blocks of data. Based on the selected blocks and a comparison of characteristics of the current block of data and the previously processed blocks of data, the block of data may be de-duplicated in item 312.

After de-duplication 312, the method 300 may update the index with new entries referring to the current block of data, in item 314, and the method 300 may proceed with item 316 in order to determine whether a next block of data is to be processed. If processing continues, the method 300 may re-iterate item 304 to receive a logical block in a next write command. If no further blocks of data are to be processed, the method 300 may end in item 318.

The method 200 of Fig. 2 and method 300 of Fig. 3 may be implemented by a de-duplication engine, such as the de-duplication engine 104 of apparatus 102a shown in Fig. 1. Even though individual method steps of methods 200 and 300 of Figs. 2 and 3, respectively, have been shown in a sequential order of processing steps, it is to be understood that individual method steps may be performed concurrently and independently from each other in the same or in a different order. Furthermore, individual method steps can be omitted and further method steps can be added according to requirements of embodiments of the present disclosure.

Although the present invention and its advantages have been described in detail, it should be understood that various changes, substitutions and alternations can be made herein without departing from the scope of the invention as defined by the appended claims.

CLAIMS

1. An apparatus comprising:

an interface configured to receive a block of data; and

a de-duplication engine (104) coupled to the interface and configured to de-duplicate the block of data based on a query of an index using a plurality of values calculated from the block of data, and insert a plurality of new entries into the index, each entry corresponding to one of the plurality of values, wherein to insert the plurality of new entries into the index, the de-duplication engine (104) is further configured to determine, for each new entry, a number of entries in the index corresponding to the same value, and if the number of entries exceeds a threshold, replace one of the entries with the new entry.

2. The apparatus according to claim 1, wherein to de-duplicate the block of data, the de-duplication engine (104) is further configured to query the index using the plurality of values, and if said querying yields a matching entry, retrieve characteristics of previously processed data according to the matching entry and compare the retrieved characteristics with the characteristics of the block of data.

3. The apparatus according to claim 1 or 2, wherein previously processed data is stored according to one or more versions, each entry of the index referring to one of said versions of the previously processed data.

4. The apparatus according to one of the preceding claims, wherein the de-duplication engine (104) is further configured to determine whether the block of data is to be de-duplicated based on said query, and if at least a part of the block of data is to be de-duplicated, replace the at least a part of the block of data with a reference to the matching data.

5. The apparatus according to one of the preceding claims, wherein said index is a sparse index, each entry of the sparse index referring to a block of data.

6. The apparatus according to one of the preceding claims, wherein the de-duplication engine (104) is further configured to divide the block of data into a plurality of segments.

7. The apparatus according to claim 6, wherein for a plurality of blocks of data to be de-duplicated, each block of data comprises the same number of segments, or wherein at least some blocks of data comprise a different number of segments per block.

8. The apparatus according to claim 6 or 7, wherein the de-duplication engine (104) is further configured to calculate the plurality of values based on hash values of the segments.

9. The apparatus according to claim 8, wherein the de-duplication engine (104) is further configured to select a number of said hash values as the plurality of values.

10. The apparatus according to one of the preceding claims, wherein the entries of the index are replaced according to a replacement strategy including replacing the oldest entry.

11. The apparatus according to one of the preceding claims, wherein said query yields an ordered list of entries of the index, wherein said entries are ordered according to a rank.

12. The apparatus according to claim 11, wherein the rank is calculated based on a number of matching values of the plurality of values or based on an age of the entry.

13. The apparatus according to one of the preceding claims, wherein said interface is configured to receive the block of data in a block write command.

14. A user equipment comprising an apparatus according to one of the preceding claims.

15. A method for de-duplicating data, comprising:

receiving (204; 304) a block of data;

de-duplicating (206; 312) the block of data based on a query of an index using a plurality of values calculated from the block of data; and

inserting a plurality of new entries into the index, each entry corresponding to one of the plurality of values, wherein said inserting the plurality of new entries into the index includes determining, for each new entry, a number of entries in the index corresponding to the same value, and if the number of entries exceeds a threshold, replacing one of the entries with the new entry.

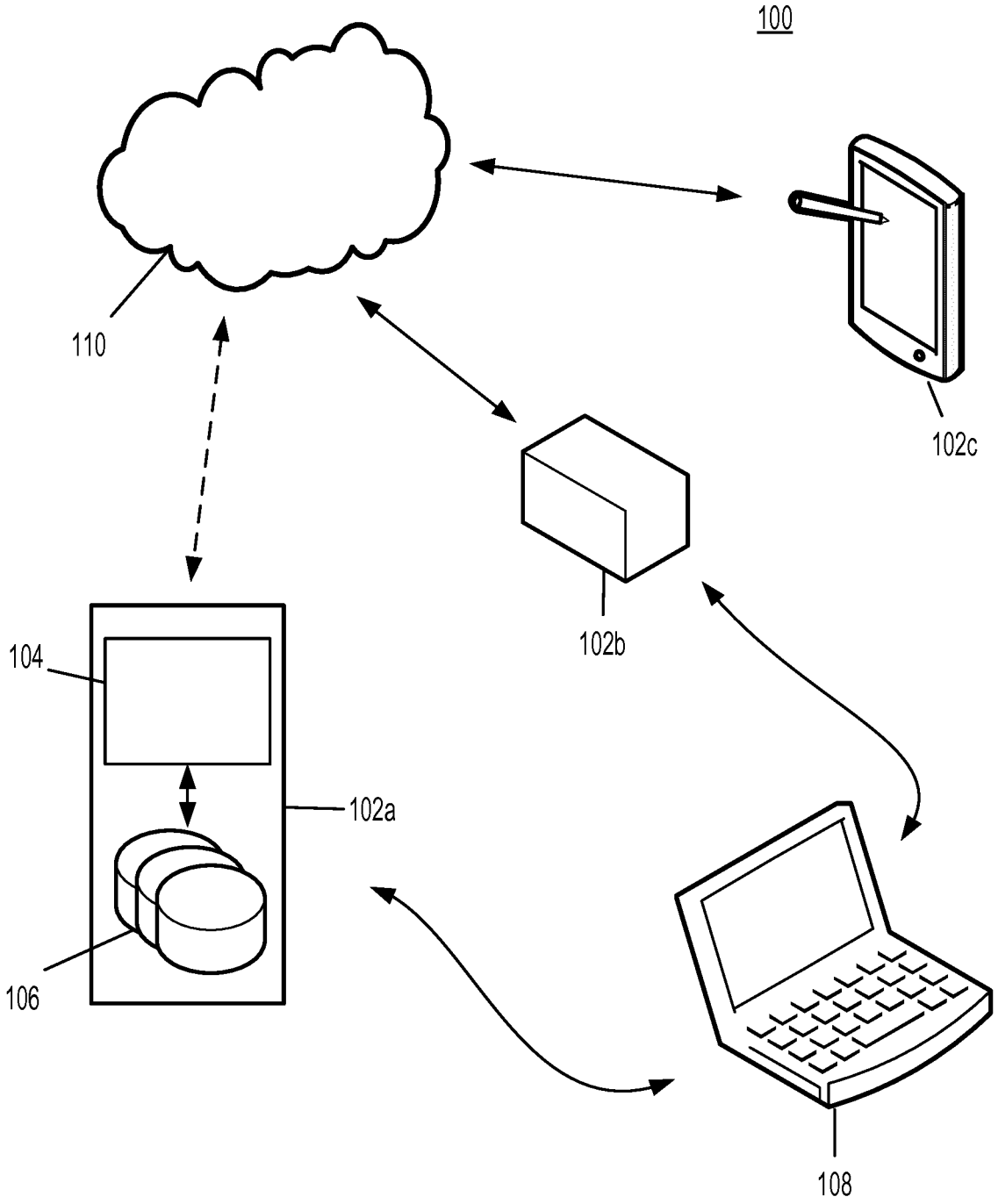

FIG. 1

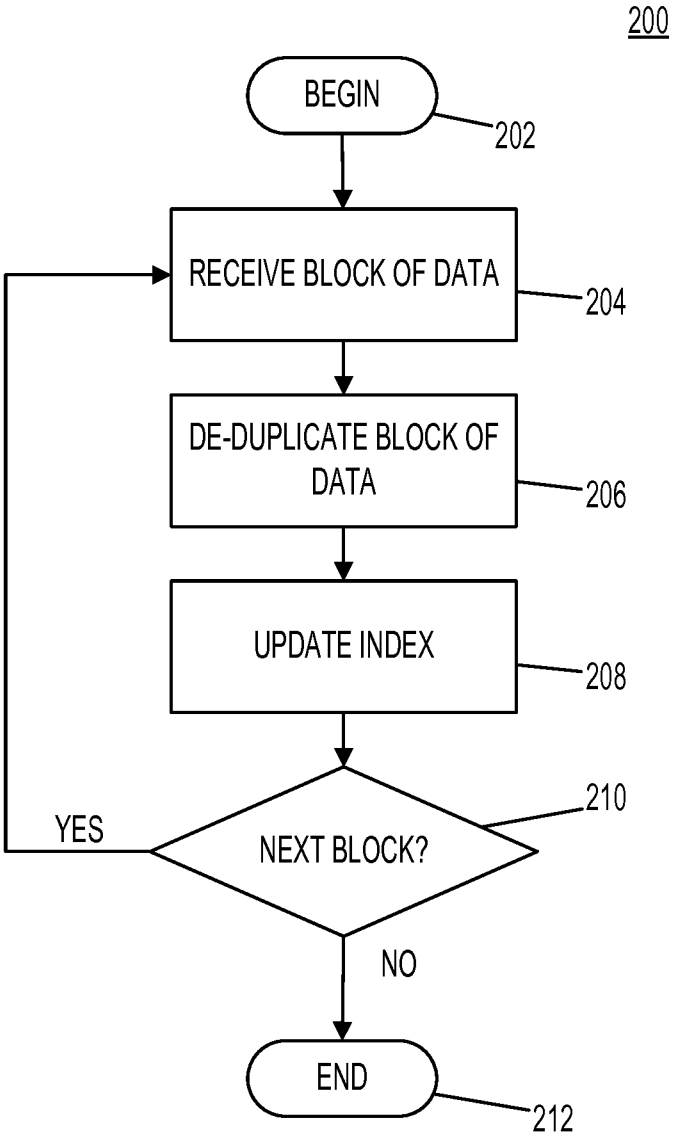
200
BEGIN
202
RECEIVE BLOCK OF DATA
204
DE-DUPLICATE BLOCK OF DATA
206
UPDATE INDEX
208
NEXT BLOCK?
210
YES
NO
END
212

Fig. 2

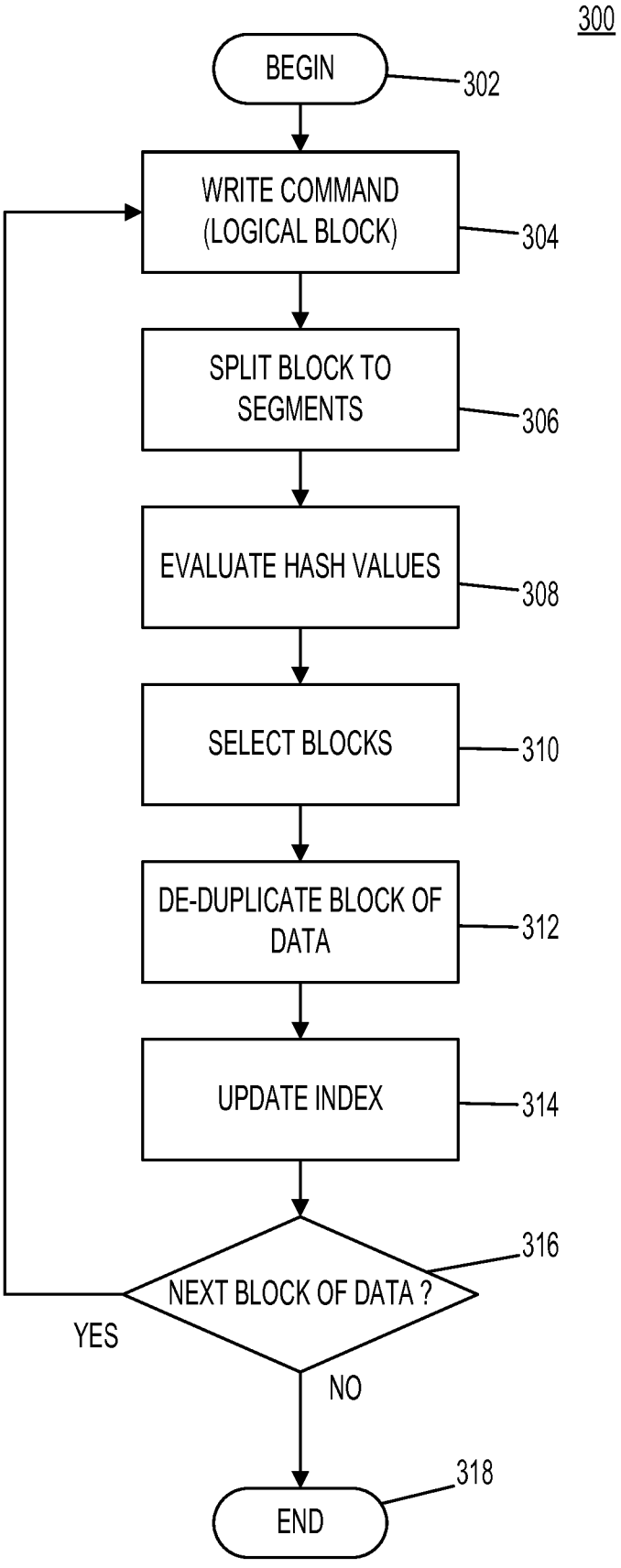
300
BEGIN
302
WRITE COMMAND (LOGICAL BLOCK)
304
SPLIT BLOCK TO SEGMENTS
306
EVALUATE HASH VALUES
308
SELECT BLOCKS
310
DE-DUPLICATE BLOCK OF DATA
312
UPDATE INDEX
314
NEXT BLOCK OF DATA ?
316
YES
NO
END
318

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/076947

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30
ADD.

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

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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MARK LILLIBRIDGE ET AL: "Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality", 7TH USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES; FAST 2009, USENIX ASSOCIATION, USA; SAN FRANCISCO, CA, USA , 1 January 2009 (2009-01-01), pages 111-123, XP008150451, Retrieved from the Internet: URL:http://static.usenix.org/events/fast09/tech/full_papers/lillibridge/lillibridge.pdf sections 3, 3.2, 3.3 ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C.

[X] 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 July 2015	05/08/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Correia Martins, F

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2014/076947

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 8 463 742 B1 (FLOYD JERED J [US] ET AL) 11 June 2013 (2013-06-11) column 5, line 58 - column 6, line 45 column 7, line 20 - line 31 -----	1-15
X	Aaron Brown ET AL: "Block-level Inline Data Deduplication in ext3", , 23 December 2010 (2010-12-23), XP055205164, Retrieved from the Internet: URL:http://pages.cs.wisc.edu/~kosmatka/dedupfs/paper.pdf [retrieved on 2015-07-29] section 3 -----	1-15
X	US 2011/246741 A1 (RAYMOND ROBERT MICHAEL [US] ET AL) 6 October 2011 (2011-10-06) paragraph [0011] - paragraph [0014] paragraph [0022] paragraph [0044] - paragraph [0047] -----	1-15

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2014/076947

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 8463742 B1	11-06-2013	US 8463742 B1	11-06-2013
		US 8898107 B1	25-11-2014
US 2011246741 A1	06-10-2011	NONE	