



US 20160134696A1

(19) **United States**

(12) **Patent Application Publication**
Datuashvili et al.

(10) **Pub. No.: US 2016/0134696 A1**

(43) **Pub. Date: May 12, 2016**

(54) **ASYNCHRONOUS DISTRIBUTED
DE-DUPLICATION FOR REPLICATED
CONTENT ADDRESSABLE STORAGE
CLUSTERS**

(71) Applicant: **GOOGLE INC.**, MOUNTAIN VIEW,
CA (US)

(72) Inventors: **Gia Datuashvili**, Cupertino, CA (US);
Alexander Kesselman, San Jose, CA
(US); **Alexandre Drobychev**, San Jose,
CA (US)

(21) Appl. No.: **14/995,171**

(22) Filed: **Jan. 13, 2016**

Related U.S. Application Data

(63) Continuation of application No. 14/265,298, filed on
Apr. 29, 2014, now abandoned, which is a continuation
of application No. 12/644,693, filed on Dec. 22, 2009,
now Pat. No. 8,712,974.

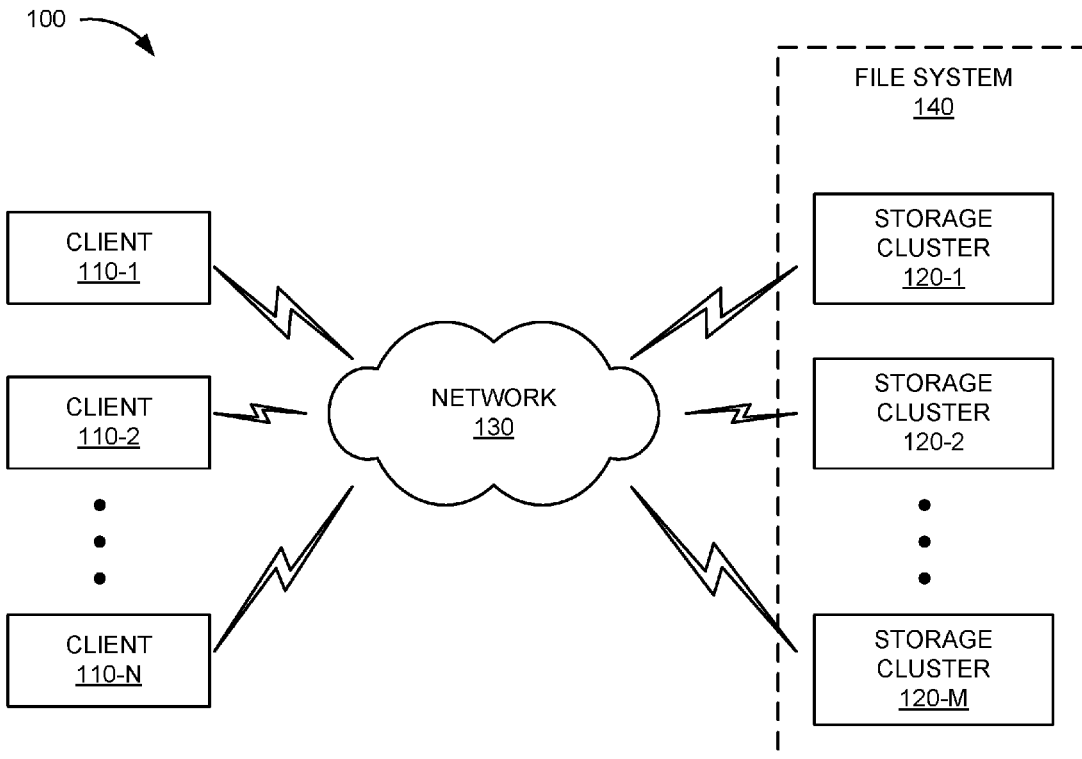
(60) Provisional application No. 61/139,857, filed on Dec.
22, 2008.

Publication Classification

(51) **Int. Cl.**
H04L 29/08 (2006.01)
G06F 17/30 (2006.01)
(52) **U.S. Cl.**
CPC **H04L 67/1095** (2013.01); **G06F 17/30575**
(2013.01); **G06F 17/30489** (2013.01); **G06F**
17/30371 (2013.01)

(57) **ABSTRACT**

A method is performed by a device of a group of devices in a distributed data replication system. The method includes storing an index of objects in the distributed data replication system, the index being replicated while the objects are stored locally by the plurality of devices in the distributed data replication system. The method also includes conducting a scan of at least a portion of the index and identifying a redundant replica(s) of the at least one of the objects based on the scan of the index. The method further includes de-duplicating the redundant replica(s), and updating the index to reflect the status of the redundant replica.



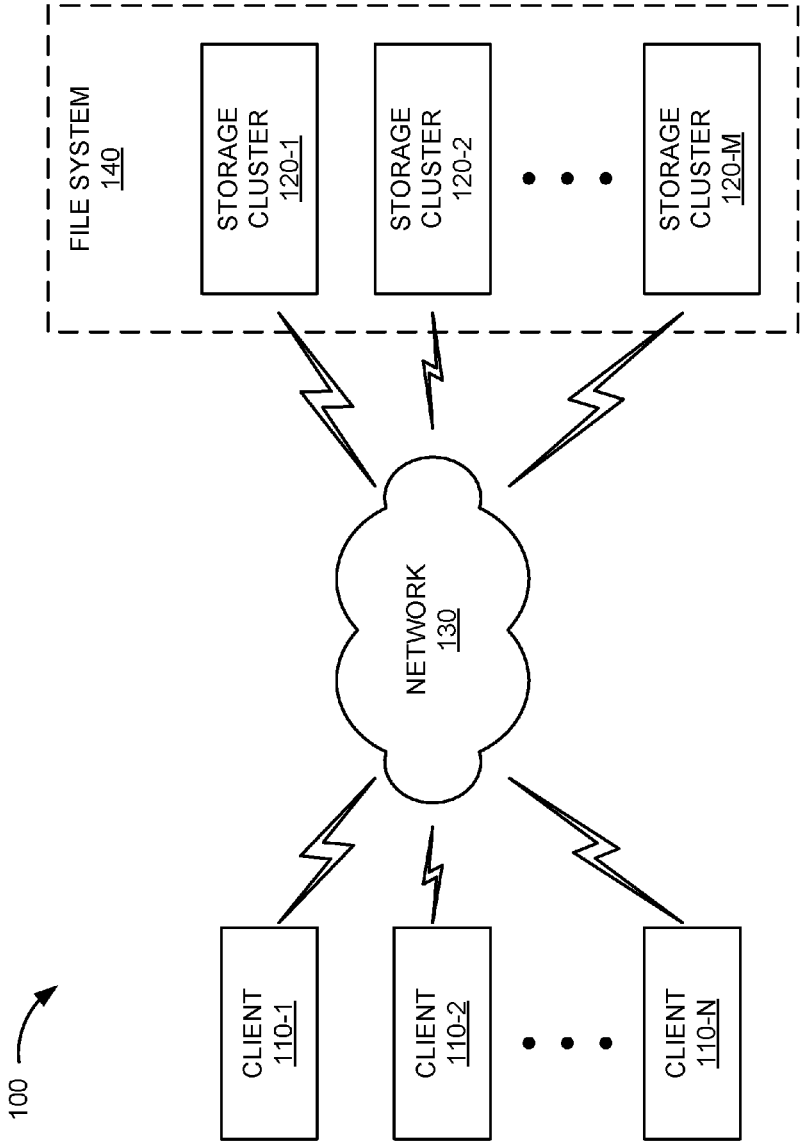


FIG. 1

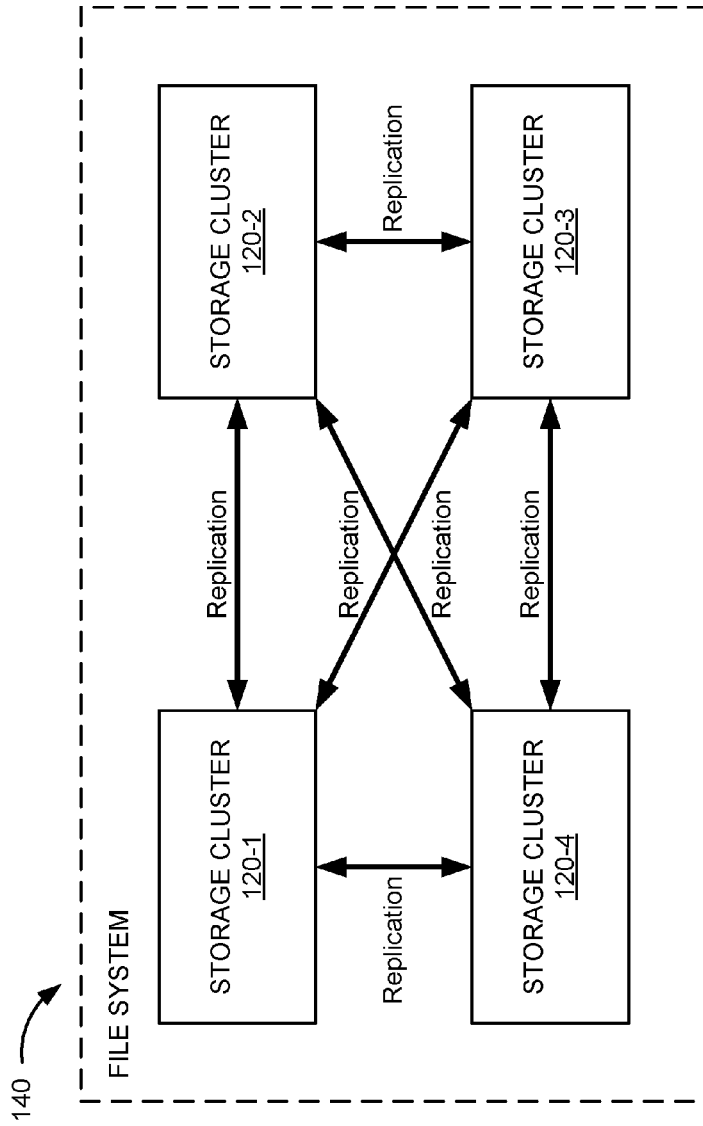


FIG. 2

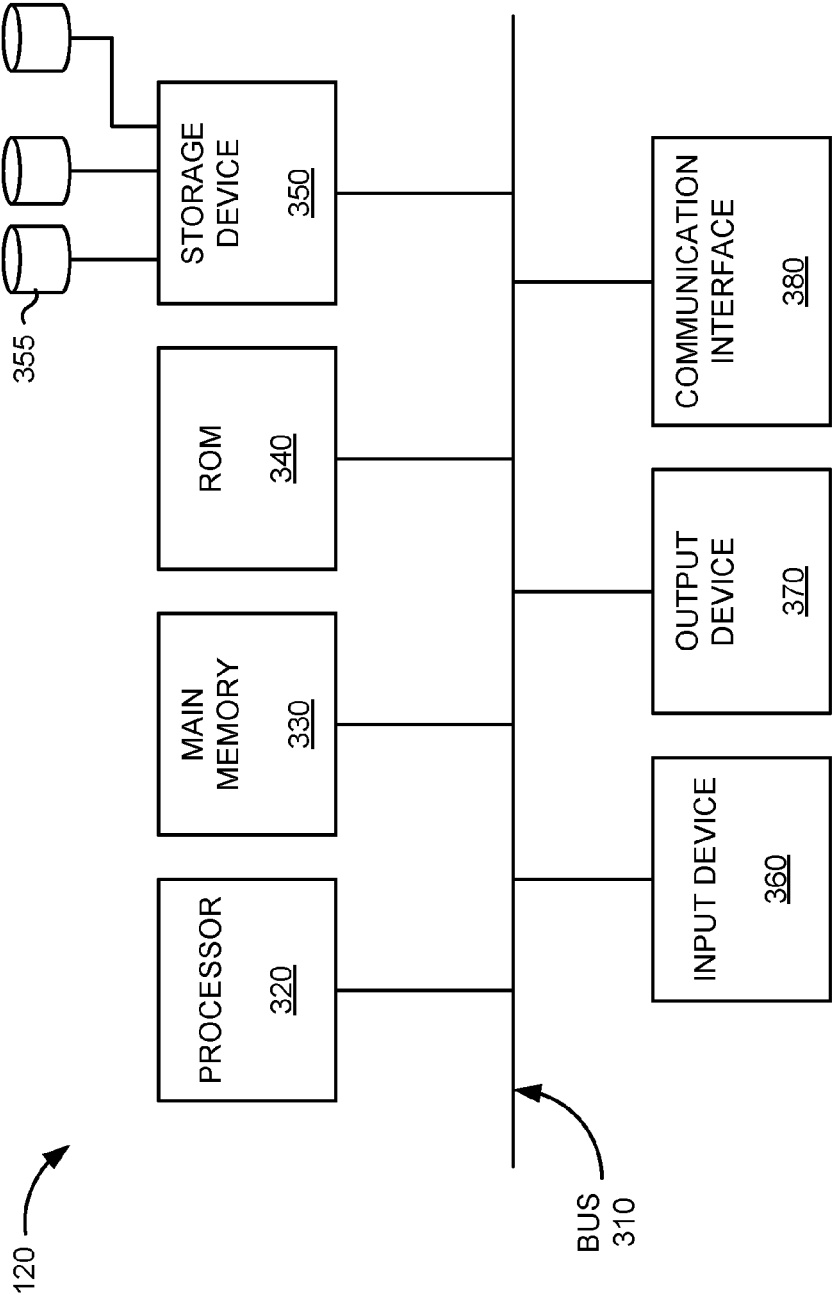


FIG. 3

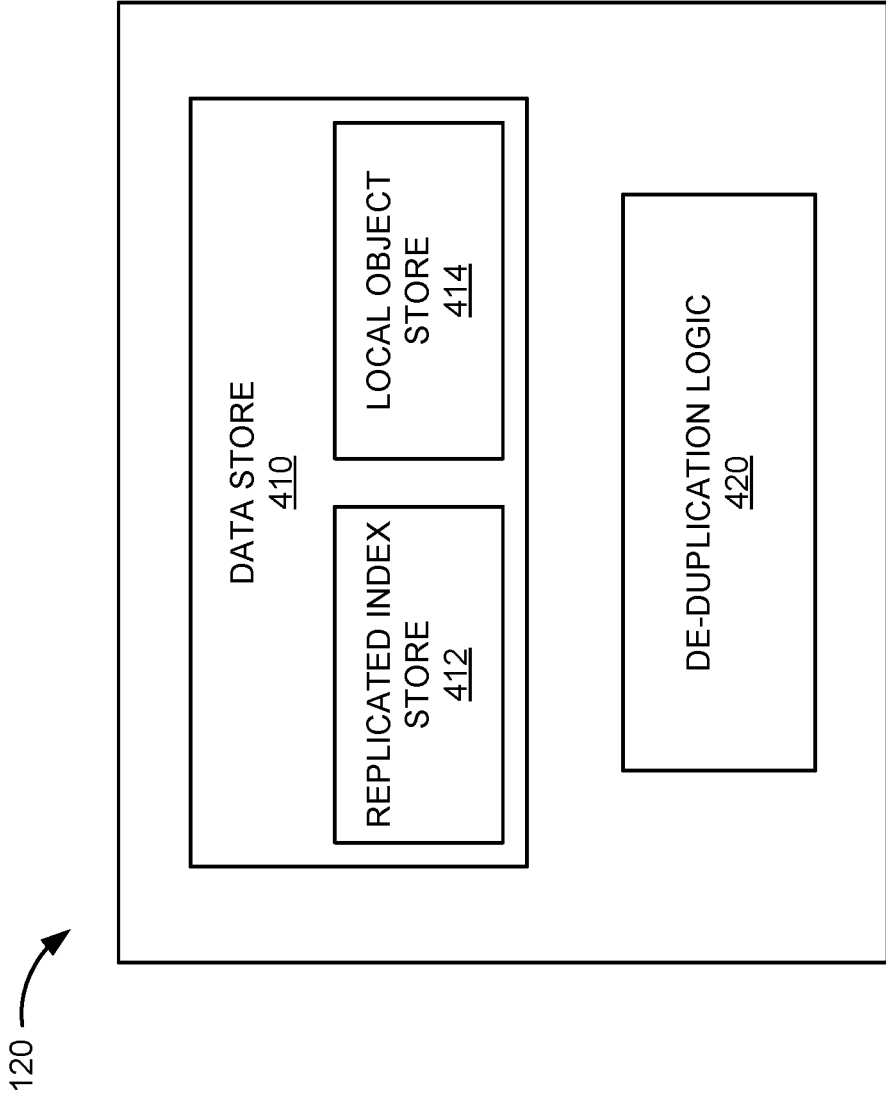


FIG. 4

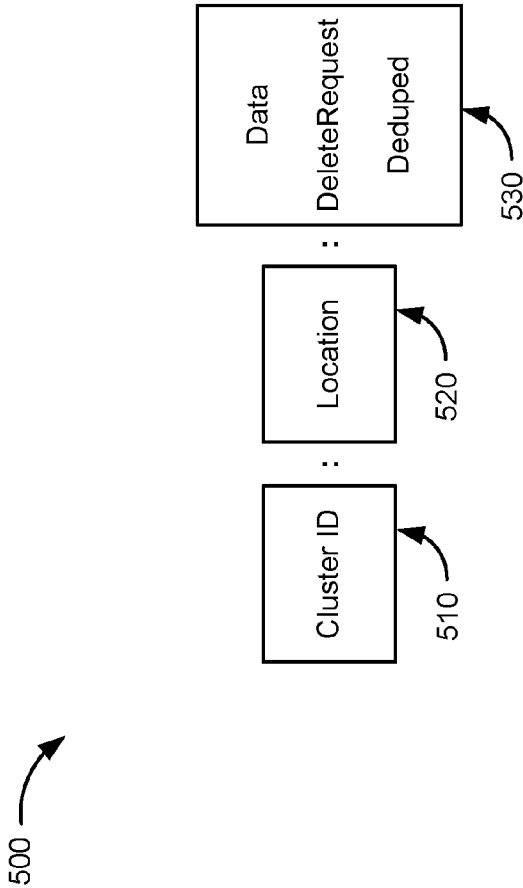


FIG. 5

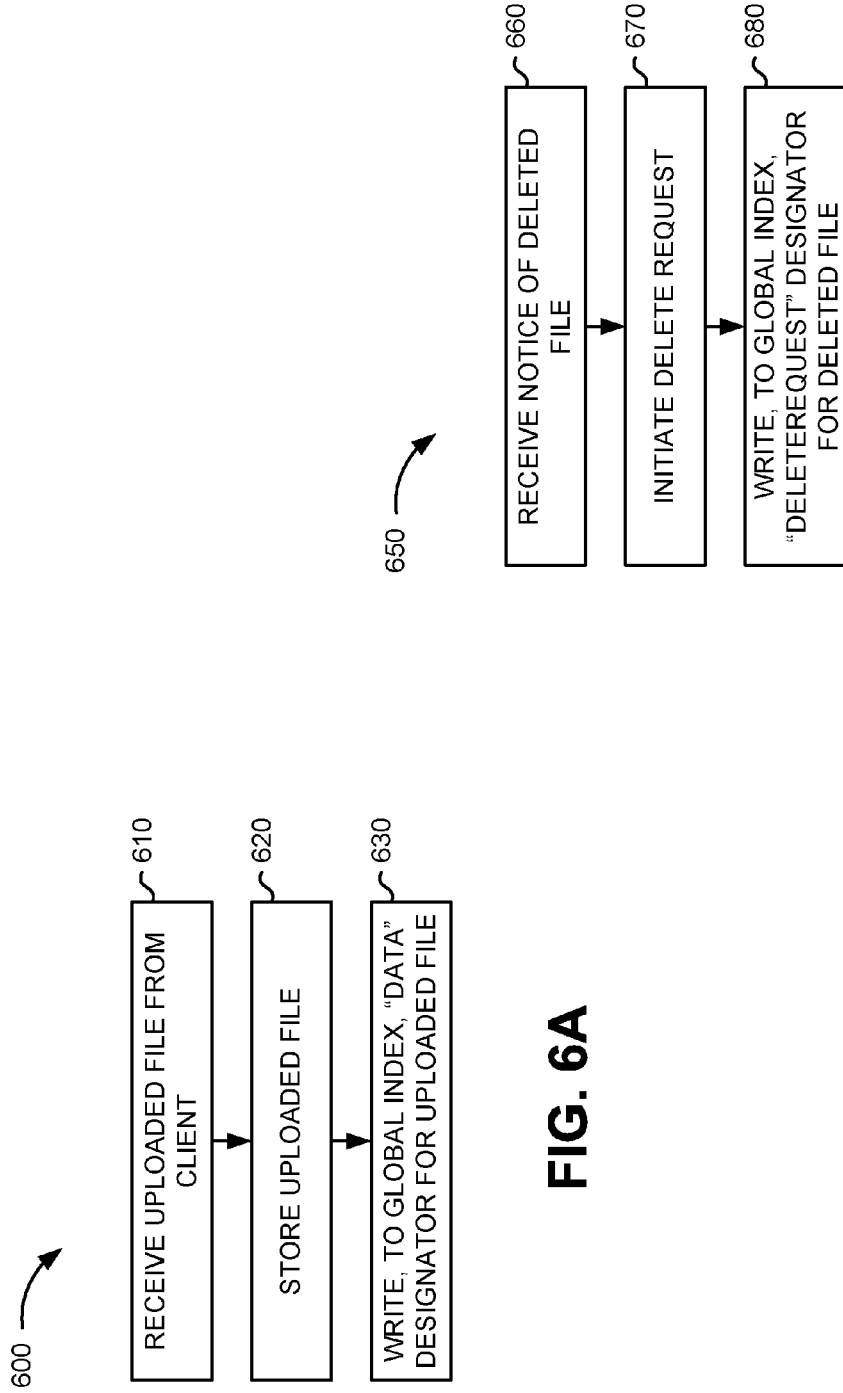


FIG. 6A

FIG. 6B

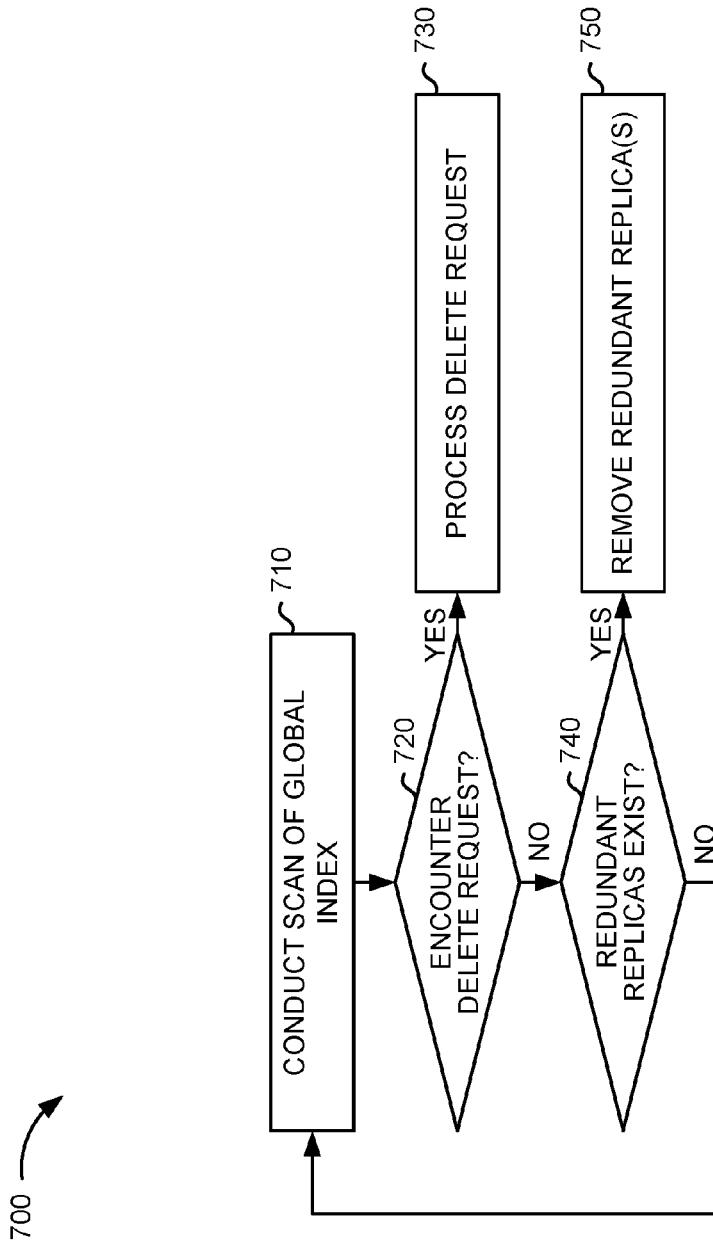


FIG. 7

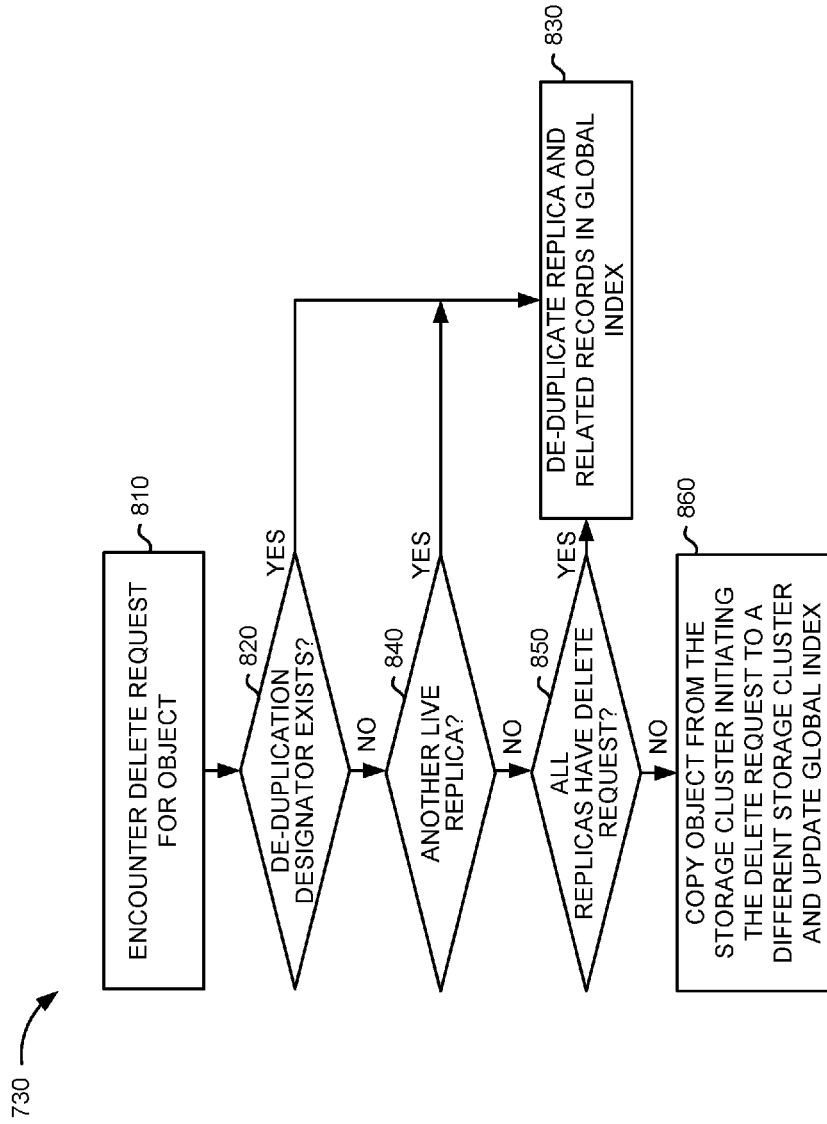


FIG. 8

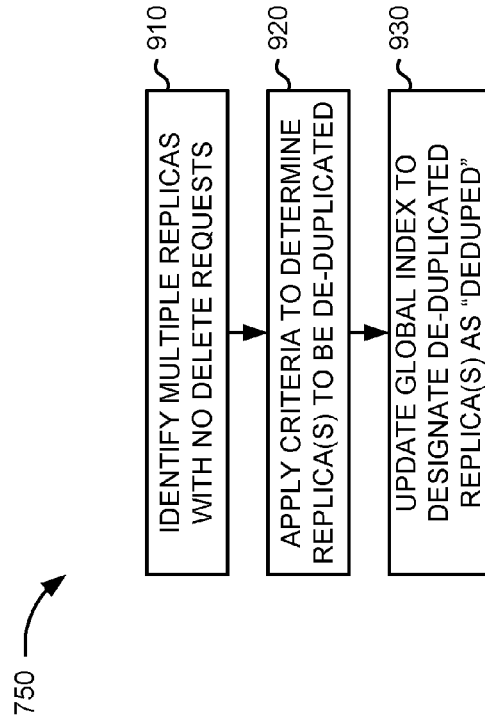


FIG. 9

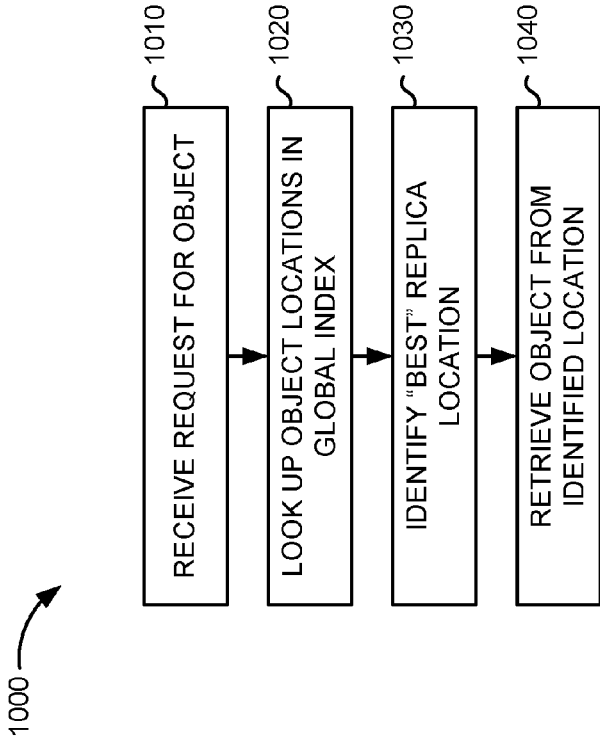


FIG. 10

1100 →

1110		1120	
Content Handle	...	De-duplication designation record	
Handle11	...	XX:Location01:Data, YY:Location01:Data	
Handle21	...	XX:Location02:Data, YY:Location02>DeleteRequest	
Handle31	...	YY:Location03:Data, YY:Location03>DeleteRequest, ZZ:Location01:Data, ZZ:Location01>DeleteRequest	
Handle41	...	YY:Location04:Data, YY:Location04>DeleteRequest, ZZ:Location02:Deduped	

FIG. 11

**ASYNCHRONOUS DISTRIBUTED
DE-DUPLICATION FOR REPLICATED
CONTENT ADDRESSABLE STORAGE
CLUSTERS**

RELATED APPLICATIONS

[0001] This application is a continuation of U.S. patent application Ser. No. 14/265,298 filed Apr. 29, 2014, which is a continuation of U.S. patent application Ser. No. 12/644,693 filed on Dec. 22, 2009 (now U.S. Pat. No. 8,712,974), which claims priority under 35 U.S.C. §119 based on U.S. Provisional Patent Application No. 61/139,857, filed Dec. 22, 2008, the disclosures of which are all incorporated by reference herein in their entirety.

BACKGROUND

[0002] The enterprise computing landscape has undergone a fundamental shift in storage architectures in that central-service architecture has given way to distributed storage clusters. As businesses seek ways to increase storage efficiency, storage clusters built from commodity computers can deliver high performance, availability and scalability for new data-intensive applications at a fraction of the cost compared to monolithic disk arrays. To unlock the full potential of storage clusters, the data is replicated across multiple geographical locations, thereby increasing availability and reducing network distance from clients.

[0003] Data de-duplication can identify duplicate objects and reduce required storage space by removing duplicates. As a result, data de-duplication is becoming increasingly important for a storage industry and is being driven by the needs of large-scale systems that can contain many duplicates.

SUMMARY

[0004] According to one implementation, a method may be performed by a device of a group of devices in a distributed data replication system. The method may include storing an index of objects in the distributed data replication system, the index being replicated while the replicas of objects are stored locally by the plurality of devices in the distributed data replication system. The method may also include conducting a scan of at least a portion of the index and identifying a redundant replica of the at least one of the objects based on the scan of the index. The method may further include de-duplicating the redundant replica by writing a de-duplication record to a portion of the index.

[0005] According to another implementation, a device, of a group of devices in a distributed data replication system, may include means for storing an index of objects in the distributed data replication system; means for writing changes to the index to designate a status of a replica of one of the objects; means for replicating the changes to the index to the plurality of devices in the distributed data replication system; means for conducting a scan of at least a portion of the index; means for identifying a redundant replica of the one of the objects based on the scan of the index; and means for de-duplicating the redundant replica.

[0006] According to yet another implementation, a system may include a memory to store instructions, a data store of objects and an index of the objects in the data store; and a processor. The processor may execute instructions in the memory to identify a status of an object in the data store, the status relating to whether the object has a replica and whether

a delete request is associated with the object, write a de-duplication designation record to the index based on the status of the object, replicate the index with the de-duplication designation record to one or more devices, and receive, from one of the one or more devices, other de-duplication designation records associated with the object, where the de-duplication designation record and the other de-duplication designation records provide a basis for deletion of one or more replicas of the object.

[0007] According to still another implementation, a method performed by one or more devices may include storing an index of objects in multiple devices within a distributed data replication system and replicating the index throughout the distributed data replication system while storing the objects locally, where each device is responsible for de-duplication of the objects within a particular subset of the index; conducting a scan of each of the subsets of the index to identify redundant replicas based on the scan; de-duplicating the redundant; and automatically copying an object from a device with a replica having an ongoing delete request to a device with a replica having been previously de-duplicated.

[0008] According to a further implementation, a computer-readable memory may include computer-executable instructions. The computer-readable memory may include one or more instructions to conduct a scan of a portion of an index of objects in a distributed data replication system; one or more instructions to identify a redundant replica of one of the objects based on the scan of the portion of the index; one or more instructions to de-duplicate the redundant replica.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate one or more embodiments described herein and, together with the description, explain these embodiments. In the drawings:

[0010] FIG. 1 is a diagram of an exemplary network in which systems and methods described herein may be implemented;

[0011] FIG. 2 is a diagram of an exemplary configuration of the file system of FIG. 1;

[0012] FIG. 3 is a diagram of exemplary components of a storage cluster of FIG. 1;

[0013] FIG. 4 is a functional block diagram of an exemplary storage cluster of FIG. 1;

[0014] FIG. 5 is a diagram of an exemplary record structure that may be used within an index of a distributed multi-master data replication system;

[0015] FIGS. 6A-6B are flowcharts of exemplary processes for managing client-initiated upload/delete operations;

[0016] FIG. 7 is a flowchart of exemplary process for performing de-duplication in a distributed multi-master data replication system;

[0017] FIG. 8 is a flowchart of exemplary process for managing a delete request;

[0018] FIG. 9 is a flowchart of exemplary process for removing duplicate replicas;

[0019] FIG. 10 is a flowchart of exemplary process for optimizing bandwidth consumption and reducing latency in a distributed multi-master data replication system; and

[0020] FIG. 11 is a diagram illustrating a portion of an exemplary global index according to an implementation described herein.

DETAILED DESCRIPTION

[0021] The following detailed description refers to the accompanying drawings. The same reference numbers in different drawings may identify the same or similar elements. Also, the following detailed description does not limit the invention.

[0022] Systems and/or methods described herein may provide an asynchronous distributed de-duplication algorithm for replicated storage clusters that provides availability, liveness and consistency guarantees for immutable objects. Implementations described herein may use the underlying replication layer of a distributed multi-master data replication system to replicate a content addressable index (also referred to herein as a “global index”) between different storage clusters. Each object of the global index may have a unique content handle (e.g., a hash value or digital signature). In implementations described herein, the removal process of redundant replicas may keep at least one replica alive.

Exemplary Network Configuration

[0023] FIG. 1 is a diagram of an exemplary system 100 in which systems and methods described herein may be implemented. System 100 may include clients 110-1 through 110-N (referred to collectively as clients 110, and individually as client 110) and storage clusters 120-1 through 120-M (referred to collectively as storage clusters 120, and individually as storage cluster 120) connected via a network 130. Storage clusters 120 may form a file system 140 (as shown by the dotted line in FIG. 1).

[0024] Network 130 may include one or more networks, such as a local area network (LAN), a wide area network (WAN), a telephone network (e.g., the Public Switched Telephone Network (PSTN)), an intranet, the Internet, a similar or dissimilar network, or a combination of networks. Clients 110 and storage clusters 120 may connect to network 130 via wired and/or wireless connections.

[0025] Clients 110 may include one or more types of devices, such as a personal computer, a wireless telephone, a personal digital assistant (PDA), a lap top, or another type of communication device, and/or a thread or process running on one of these devices. In one implementation, a client 110 includes, or is linked to, an application on whose behalf client 110 communicates with storage cluster 120 to read or modify (e.g., write) file data.

[0026] Storage cluster 120 may include one or more server devices, or other types of computation or communication devices, that may store, process, search, and/or provide information in a manner described herein. In one implementation, storage cluster 120 may include one or more servers (e.g., computer systems and/or applications) capable of maintaining a large-scale, random read/write-access data store for files. The data store of storage cluster 120 may permit an indexing system to quickly update portions of an index if a change occurs. The data store of storage cluster 120 may include one or more tables (e.g., a document table that may include one row per uniform resource locator (URL), auxiliary tables keyed by values other than URLs, etc.). In one example, storage cluster 120 may be included in a distributed storage system (e.g., a “Bigtable” as set forth in Chang et al., “Bigtable: A Distributed Storage System for Structured Data,” Proc. of the 7th OSDI, pp. 205-218 (November 2006)) for managing structured data (e.g., a random-access storage

cluster of documents) that may be designed to scale to a very large size (e.g., petabytes of data across thousands of servers).

[0027] Although not shown in FIG. 1, system 100 may include a variety of other components, such as one or more dedicated consumer servers or hubs. A consumer server, for example, may store a read-only copy of a data store from one or more storage clusters 120 for access by clients 110. A hub, for example, may store a read-only copy of a data store from one or more storage clusters 120 for distribution to one or more consumer servers.

Exemplary Storage Cluster Configuration

[0028] FIG. 2 is a diagram of an exemplary configuration of the file system 140. As shown in FIG. 2, file system 140 may include storage clusters 120-1, 120-2, 120-3, and 120-4. In one implementation, file system 140 may be a distributed multi-master data replication system, where each of storage clusters 120-1, 120-2, 120-3, and 120-4 may act as a master server for the other storage clusters. In file system 140, data may be replicated across storage clusters 120-1, 120-2, 120-3, and 120-4 (e.g., in multiple geographical locations) to increase data availability and reduce network distance from clients (e.g., clients 110). Generally, distributed objects and references may be dynamically created, mutated, cloned and deleted in different storage clusters 120 and an underlying data replication layer (not shown) maintains the write-order fidelity to ensure that all storage clusters 120 will end up with the same version of data. Thus, the data replication layer respects the order of writes to the same replica for a single object.

[0029] A global index of all of the objects in the distributed multi-master data replication system may be associated with each storage cluster 120. Each stored object may be listed by a unique content handle (such as a hash value, digital signature, etc.) in the global index. Selected storage clusters may each be assigned to be responsible for a distinct range of the content handles in the global index. For example, a single storage cluster 120 may be responsible for de-duplication of objects associated with particular content handles. Changes to the global index made by one storage cluster may be replicated to other storage clusters.

[0030] Although FIG. 2 shows exemplary functional components of file system 140, in other implementations, file system 140 may contain fewer, additional, different, or differently arranged components than depicted in FIG. 2. In still other implementations, one or more components of file system 140 may perform one or more tasks described as being performed by one or more other components of file system 140.

[0031] FIG. 3 is a diagram of exemplary components of storage cluster 120. Storage cluster 120 may include a bus 310, a processor 320, a main memory 330, a read-only memory (ROM) 340, a storage device 350, an input device 360, an output device 370, and a communication interface 380. Bus 310 may include one or more conductors that permit communication among the components of storage cluster 120.

[0032] Processor 320 may include any type of processor or microprocessor that may interpret and execute instructions. Main memory 330 may include a random access memory (RAM) or another type of dynamic storage device that may store information and instructions for execution by processor 320. ROM 340 may include a ROM device or another type of static storage device that may store static information and

instructions for use by processor 320. Storage device 350 may include a magnetic and/or optical recording medium and its corresponding drive. For example, storage device 350 may include one or more local disks 355 that provide persistent storage. In one implementation, storage cluster 120 may maintain metadata, for objects stored in file system 140, within one or more computer-readable mediums, such as main memory 330 and/or storage device 350. For example, storage cluster 120 may store a global index within storage device 350 for all the objects stored within a distributed multi-master data replication system.

[0033] Input device 360 may include one or more mechanisms that permit an operator to input information to storage cluster 120, such as a keyboard, a keypad, a button, a mouse, a pen, etc. Output device 370 may include one or more mechanisms that output information to the operator, including a display, a light emitting diode (LED), etc. Communication interface 380 may include any transceiver-like mechanism that enables storage cluster 120 to communicate with other devices and/or systems. For example, communication interface 380 may include mechanisms for communicating with other storage clusters 120 and/or clients 110.

[0034] FIG. 4 illustrates a functional block diagram of storage cluster 120. As shown in FIG. 4, storage cluster 120 may include data store 410 and de-duplication logic 420. In one implementation, as illustrated in FIG. 4, data store 410 may be provided within storage cluster 120. In other implementations, some or all of data store 410 may be stored within one or more other devices of system 100 in communication with storage cluster 120, such as external memory devices or devices associated with an indexing system (not shown).

[0035] Data store 410 may include a replicated index store 412 and a local object store 414. Replicated index store 412 may be included as part of the replication layer of the distributed multi-master data replication system. Replicated index store 412 may store information associated with the global index. At least a portion of replicated index store 412 may be replicated on multiple storage clusters 120. The number of replicas for each replicated index store 412 may be user-configurable. Local object store 414 may store objects locally within storage cluster 120. Local object store 414 may include files, such as images or videos uploaded by clients (e.g., clients 110).

[0036] De-duplication logic 420 may include logic to remove redundant replicas from storage clusters within the distributed multi-master data replication system (e.g., storage clusters 120-1, 120-2, 120-3, and 120-4). De-duplication logic 420 for each participating storage cluster may be assigned to be responsible for a particular section of the global index. For example, de-duplication logic 420 may be assigned to a particular range of content handles for the global index. Thus, only one storage cluster within the distributed multi-master data replication system may be able to perform destructive operations (e.g., deletion of replicas) on a replicated object within the system.

[0037] To facilitate de-duplication, records may be generated by de-duplication logic 420 and appended to a portion of the global index associated with a particular content handle. Records may include, for example, a "Data" designator for initiating a live replica, a "DeleteRequest" designator for indicating an ongoing delete request for a replica, and a "Deduped" designator for indicating a replica that has been selected for de-duplication. Record formats and uses are described in more detail below.

[0038] Although FIG. 4 shows exemplary functional components of storage cluster 120, in other implementations, storage cluster 120 may contain fewer, additional, different, or differently arranged functional components than depicted in FIG. 4. In still other implementations, one or more functional components of storage cluster 120 may perform one or more other tasks described as being performed by one or more other functional components.

Exemplary Record Structure

[0039] FIG. 5 provides an illustration of an exemplary record structure 500 for a de-duplication designation record that may be written to the global index in an exemplary implementation. The de-duplication designation record may be associated in the global index with a particular content handle of an object replica. As shown in FIG. 5, record structure 500 may include storage cluster identifier ("ID") section 510, a storage location section 520, and designation section 530. Storage cluster identification section 510 may include a unique identification (e.g., "Cluster ID") for the storage cluster 120 that is storing the object replica for which the record is being written. Location section 520 may include an address for the location of the replica within storage cluster 120 that is identified by storage cluster identification section 510. Designation section 530 may include, for example, a "Data" designator, a "DeleteRequest" designator, or a "Deduped" designator.

[0040] Record structure 500 may be listed in the form of "ClusterID:Location:Designation." For example, a record for a replica may be added to the global index by storage cluster 120-1 with the record "01:234523/2000:DeleteRequest," where "01" is the cluster ID for storage cluster 120-1, "234523/2000" is the location, within storage cluster 120-1 at which the replica is stored, and "DeleteRequest" is the designator. A record for another replica of the same object in storage cluster 120-2 may be "02:234544/1000:Data," where "02" is the cluster ID for storage cluster 120-2, "234544/1000" is the location within storage cluster 120-2, and "Data" is the designator.

Exemplary Process Flows

[0041] FIGS. 6A and 6B are flowcharts of exemplary processes for managing client-initiated upload/delete operations. FIG. 6A depicts a flowchart for an exemplary process 600 of uploading an object from a client. FIG. 6B depicts a flowchart for an exemplary process 650 of removing an object deleted by a client. In one implementation, processes 600 and 650 may be performed by one of storage clusters 120. Processes 600 and 650 may be implemented in response to client (e.g., client 110) activities. For particular examples of processes 600 and 650 described below, reference may be made to storage cluster 120-1 of file system 140, where storage cluster 120-1 includes a cluster ID of "01."

[0042] Referring to FIG. 6A, process 600 may begin when an uploaded file is received from a client (block 610). For example, storage cluster 120-1 may receive a new file from one of clients 110. The uploaded file may be stored (block 620) and a "Data" designator for the uploaded file may be written to the global index (block 630). For example, storage cluster 120-1 may store the uploaded file in a memory (e.g., storage device 350) and add a content handle for the object to the global index. Storage cluster 120-1 may also write a data

record (e.g., “01:Location:Data”) to the replicated global index addressed by the content handle of the object.

[0043] Referring to FIG. 6B, process 650 may begin when a notice of a deleted file is received (block 660). For example, storage cluster 120-1 may receive an indication that one of clients 110 has deleted a file. A delete request may be initiated (block 670) and a “DeleteRequest” designator for the deleted file may be written to the global index (block 680). For example, storage cluster 120-1 may initiate a delete request to asynchronously remove the delete file from file system 140. Storage device 120-1 may also write a “DeleteRequest” record (e.g., “01:Location:DeleteRequest”) to the replicated global index addressed by the content handle of the object.

[0044] FIG. 7 is a flowchart of an exemplary process 700 for performing de-duplication in a distributed multi-master data replication system (e.g., file system 140). In one implementation, process 700 may be performed by one of storage clusters 120. In another implementation, some or all of process 700 may be performed by another device or a group of devices, including or excluding storage cluster 120. Process 700 may be implemented periodically in each storage cluster 120 and may include a scan of all or a portion of the objects in the storage cluster 120. For particular examples of process 700 described below, reference may be made to storage clusters 120-1 and 120-2 of file system 140, where storage cluster 120-1 includes a cluster ID of “01” and storage cluster 120-2 includes a cluster ID of “02.”

[0045] As illustrated in FIG. 7, process 700 may begin with conducting a scan of the global index (block 710). For example, storage cluster 120-1 (using, e.g., de-duplication logic 420) may conduct a scan of all or a portion of the objects listed in the global index. The scan may identify, for example, multiple replicas and/or objects marked for deletion.

[0046] It may be determined if a delete request is encountered (block 720). For example, storage cluster 120-1 may encounter an object in the global index that includes a delete request designator (e.g., “02:Location:DeleteRequest”) from another storage cluster (e.g., from storage cluster 120-2). If it is determined that a delete request is encountered (block 720-YES), then the delete request may be processed (block 730). For example, storage cluster 120-1 may process the delete request as described in more detail with respect to FIG. 8.

[0047] If it is determined that a delete request is not encountered (block 720-NO), then it may be determined if redundant replicas exist (block 740). Redundant replicas may be replicated objects in different locations that have no outstanding delete requests for the object. For example, storage cluster 120-1 may identify multiple replicas for the same object that correspond to a content handle for which storage cluster 120-1 is responsible. The multiple replicas may be stored, for example, in different storage clusters (e.g., storage cluster 120-1 and storage cluster 120-2) or in different locations within the same storage cluster.

[0048] If it is determined that redundant replicas exist (block 740-YES), then the redundant replicas(s) may be removed (block 750). For example, storage cluster 120-1 may remove the redundant replica(s) as described in more detail with respect to FIG. 9. If it is determined that redundant replicas do not exist (block 740-NO), then the process may return to block 710, where another scan of the global index may be conducted (block 710).

[0049] FIG. 8 illustrates exemplary operations associated with the processing of a delete request of block 730 of FIG. 7.

A delete request may be encountered for an object (block 810). For example, a scan being conducted by storage cluster 120-1 may identify a content handle in the global index with a delete request designator previously written by storage cluster 120-1 to delete a replica in a certain storage cluster (e.g., “02:Location:DeleteRequest”). Assuming that storage cluster 120-1 is responsible for the content handle, storage cluster 120-1 may apply operations to determine if the replica can now be de-duplicated.

[0050] It may be determined if a de-duplication designator exists (block 820). For example, storage cluster 120-1 may review other records in the global index associated with the content handle to determine if a de-duplication designator exists (e.g., 02:Location:Deduped”). If it is determined that a de-duplication designator exists (block 820—YES), then the replica and the related records in the global index may be de-duplicated (block 830). For example, storage cluster 120-1 may initiate a delete request to delete the replica in storage cluster 120-2 (if any) and delete any records (e.g., “02:Location:*”, where “*” may be any designator) from the global index that relate to the content handle for the deleted replica.

[0051] If it is determined that a de-duplication designator does not exist (block 820—NO), then it may be determined if another live replica exists (block 840). For example, storage cluster 120-1 may review the content handle for the global index to determine whether another live replica exists for the object. The global index may include, for example, a data record for that content handle from another storage cluster (e.g., “03:Location:Data”).

[0052] If another live replica exists (block 840—YES), then the replica may be de-duplicated as described above with respect to block 830. If another live replica does not exist (block 840—NO), then it may be determined if all replicas have delete requests (block 850). For example, storage cluster 120-1 may review the content handle for the global index to determine whether all the replicas associated with the content handle have an outstanding delete request (e.g., “*:*:DeleteRequest”, where “*” may be any ClusterID and any location, respectively).

[0053] If it is determined that all replicas have delete requests (block 850—YES), then the replica may be de-duplicated as described above with respect to block 830. If it is determined that all replicas do not have delete requests (block 850—NO), then the object may be copied from a storage cluster that initiated a delete request to a different storage cluster and the global index may be updated (block 860). For example, in response to the record “02:Location:DeleteRequest,” storage cluster 120-1 may copy the object from storage cluster 120-2 to another storage cluster 120-3 for which there is a de-duplication record (e.g., “03:Location:Deduped”) and no outstanding delete request. Storage cluster 120-1 may delete the previous de-duplication record (e.g., “03:Location:Deduped”) associated with the replica and write a data designator (e.g., “03:Location:Data”) to the corresponding content handle of the object in the global index.

[0054] FIG. 9 illustrates exemplary operations associated with the removing of duplicate references of block 750 of FIG. 7. Multiple replicas with no delete requests may be identified (block 910). For example, storage cluster 120-1 may review the global index and identify two or more replicas that have no outstanding delete requests corresponding to a content handle for which storage cluster 120-01 is responsible.

[0055] Criteria to determine replica(s) to be de-duplicated may be applied (block **920**). For example, storage cluster **120-1** may apply criteria to de-duplicate the redundant replica that may be stored within storage cluster **120-1**. The criteria to de-duplicate redundant replicas may be based on a variety of factors, such as geographic proximity of the replicas, available storage capacity at a storage cluster, or other factors. Storage cluster **120-1** (e.g., using de-duplication logic **420**) may apply the criteria to the two or more replicas that have no outstanding delete requests identified above. In some implementations, multiple replicas may be identified to be de-duplicated. In other implementations, storage cluster **120-1** may leave more than one live replica (e.g., a replica not marked for de-duplication).

[0056] The global index may be updated to designate de-duplicated replica(s) as “Deduped” (block **930**). For example, for each de-duplicated replica, storage cluster **120-1** may delete the previous data record (e.g., “02:Location:Data”) associated with the replica and write a de-duplication designator (e.g., “02:Location:Deduped”) to the corresponding content handle in the global index.

[0057] De-duplication of the redundant replicas may be accomplished using de-duplication messages that are replicated as a part of the global index. The replicas marked for de-duplication may be stored within storage cluster **120-1** or within another storage cluster (e.g., storage cluster **120-2**, **120-3**, **120-4**, etc.). In one implementation, storage cluster **120-1** may delete locally-stored replicas and the corresponding “01:Location:Data” record from the global index and add “01:Location:Deduped” to the global index. Storage cluster **120-1** may also initiate delete messages, using the replicated global index, to delete replicas stored in other clusters.

[0058] FIG. **10** provides a flowchart of an exemplary process **1000** for optimizing bandwidth consumption and reducing latency in a distributed multi-master data replication system (e.g., file system **140**). In one implementation, process **1000** may be performed by one of storage clusters **120**. In another implementation, some or all of process **1000** may be performed by another device or group of devices, including or excluding storage cluster **120**. For particular examples of process **1000** described below, reference may be made to storage cluster **120-1** of file system **140**, where the storage cluster **120-1** includes a cluster ID of “01.”

[0059] As illustrated in FIG. **1000**, process **1000** may begin with receiving a request for an object (block **1010**). For example, storage cluster **120-1** may receive a request from a client (e.g., client **110-1**) to obtain an object.

[0060] Object locations may be looked up in the global index (block **1020**). For example, storage cluster **120-1** may look up the replica location(s) for the object in the replicated global index using the content handle of the object.

[0061] The “best” replica location may be identified (block **1030**). For example, assuming that more than one replica is available, storage cluster **120-1** may determine the “best” replica to retrieve to minimize network resources. For example, the “best” replica may be the replica that has the closest geographic location to storage cluster **120-1**. In other implementations, the “best” replica may be based on a combination of available network connectivity, geographic location, and/or other criteria. Thus, in some implementations, the “best” replica for the object may be stored locally within storage cluster **120-1**.

[0062] The object may be retrieved from the identified location (block **1040**). For example, storage cluster **120-1** may

request the “best” replica from the closest available storage cluster and receive the replica to satisfy the client request. Storage cluster **120-1** may then send the replica to the client.

EXAMPLES

[0063] FIG. **11** provides a portion **1100** of an exemplary global index according to an implementation described herein. The index may include, among other information, a content handle column **1110** and a De-duplication designation record column **1120**. Assume, in exemplary index portion **1100**, a distributed multi-master data replication system includes three storage clusters, XX, YY, and ZZ. A de-duplication algorithm may run periodically in each of storage clusters XX, YY, and ZZ and may scan all or a portion of the global index. Also, records (e.g., Data, DeleteRequest, and Deduped) may be written by one of storage clusters XX, YY, or ZZ to the global index associated with a particular object content handle. Modifications to the global index may be replicated to all other participating clusters (e.g., the remaining of storage clusters XX, YY, and ZZ).

[0064] As shown in FIG. **11**, index portion **1100** includes content handles and associated delete designation records for four objects. “Handle11” has records indicating replicas are stored at storage cluster XX (“XX:Location01:Data”) and storage cluster YY (“YY:Location01:Data”), respectively. “Handle21” has a record indicating a replica is stored at storage cluster XX (“XX:Location02:Data”) and another replica at storage cluster YY has an ongoing delete request (“YY:Location:02:DeleteRequest”). “Handle31” has records indicating replicas are stored at storage cluster YY (“XX:Location03:Data”) and storage cluster ZZ (“ZZ:Location01:Data”), respectively. “Handle31” also has two records indicating the replicas have ongoing delete requests at storage cluster YY (“YY:Location03:DeleteRequest”) and storage cluster ZZ (“ZZ:Location01:DeleteRequest”). “Handle41” has records indicating a replica is stored at storage cluster YY (“XX:Location04:Data”) and a record indicating the replica with an ongoing delete request at storage cluster YY (“YY:Location04:DeleteRequest”). Handle41 also has one record indicating de-duplication of a replica has occurred (“ZZ:Location02:Deduped”). The de-duplication algorithm used by the storage clusters can operate using guidelines consistent with the principles described herein. Assume storage cluster XX is assigned responsibility for the portion of the global index including “Handle11,” “Handle21,” “Handle31,” and “Handle41.”

[0065] When an object is fully uploaded in a storage cluster, the storage cluster may write a data record (e.g., “ClusterID:Location:Data”) to the replicated global index addressed by the content handle of the object. For example, “XX:Location01:Data” and “YY:Location01:Data” illustrate data records for replicas of “Handle11.” Also, “XX:Location02:Data” illustrates a data record for a replica of “Handle21.” Similar data records can be seen for “Handle31” and “Handle41.”

[0066] When an object is requested in a storage cluster, the storage cluster may look up the replica locations in the replicated global index using the content handle of the object and fetch the replica from the “best” (e.g., closest) cluster. For example, assuming an object corresponding to “Handle11” is requested at storage cluster ZZ and that storage cluster YY is closer to storage cluster ZZ than is storage cluster XX, storage cluster ZZ may request the object replica corresponding to “Handle11” from storage cluster YY.

[0067] When an object is deleted in a storage cluster, the storage cluster may write “ClusterID:Location:DeleteRequest” to the replicated global index addressed by the content handle of the object. For example, “YY:Location02:DeleteRequest” illustrates a record for a deleted replica of “Handle21” in storage cluster YY. Similarly, “YY:Location03:DeleteRequest” and “ZZ:Location01:DeleteRequest” illustrate records for deleted replicas of “Handle31” for storage clusters YY and ZZ, respectively.

[0068] If the scan in a storage cluster encounters multiple replicas that have no outstanding delete requests corresponding to a content handle the storage cluster is responsible for, the storage cluster may delete redundant replicas of the object (possibly leaving more than one live replica). For each deleted replica in another storage cluster, the storage cluster may delete the data record and write a de-duplication record. For example, the scan in storage cluster XX may identify that “Handle11” has records indicating replicas are stored at storage cluster XX (“XX:Location01:Data”) and storage cluster YY (“YY:Location01:Data”), respectively. Based on criteria provided for removing redundant references, storage cluster XX may initiate deletion of the replica at storage cluster YY. Storage cluster XX may delete the record “YY:Location01:Data” shown in FIG. 11 and write “YY:Location01:Deduped” instead.

[0069] If the scan in storage cluster XX encounters a delete request (e.g., “ClusterID:Location:DeleteRequest”) for a replica in another storage cluster (e.g., storage cluster YY or ZZ) corresponding to a content handle that storage cluster XX is responsible for, storage cluster XX may apply the following analysis. If there is a “Deduped” record for the same storage cluster and location as the delete request, if there exists another live replica of the object, or if all replicas have outstanding delete requests, the storage cluster XX can delete the replica of the object in storage cluster YY or ZZ (if any) and delete the records “YY:Location:*” or “ZZ:Location:*”. For example, the replica for “Handle21” in storage cluster YY and the record “YY:Location02:DeleteRequest” may be deleted by storage cluster XX since another live object (indicated by the record “XX:Location02:Data”) exists. Similarly, the replica for “Handle31” in storage cluster YY and the record “YY:Location03:DeleteRequest” may be deleted by storage cluster XX since both replicas in storage cluster YY and storage cluster ZZ have outstanding delete requests.

[0070] If storage cluster XX cannot delete the replica of the object in storage cluster YY or ZZ (e.g., there is not a “Deduped” record or another live replica of the object, and all replicas do not have outstanding delete requests), storage cluster XX can copy the object from YY or ZZ to another storage cluster for which there is a de-duplication record and no outstanding delete request, deleting the de-duplication record and writing a data record. For example, the replica for “Handle41” in storage cluster YY (“YY:Location04:DeleteRequest”) may trigger storage cluster XX to copy the object associated with “Handle41” to storage cluster ZZ. Storage cluster XX may update the global index to change “ZZ:Location02:Deduped” to “ZZ:Location02:Data.”

[0071] The correctness of the algorithm is straightforward as all deletion operations on the object are performed only by the scan process in the storage cluster responsible for its content handle. The algorithm also transparently deals with multiple object replicas in the same cluster that have different locations (e.g. XX:Location1 and XX:Location2).

CONCLUSION

[0072] Systems and/or methods described herein may store a global index of objects in a distributed data replication system and replicate the global index and some of the objects throughout the distributed data replication system. A storage cluster may be assigned as the responsible entity for de-duplication within a particular subset of the global index. The storage cluster may conduct a scan of the subset of the global index and identify redundant replicas based on the scan. The storage cluster may de-duplicate the redundant replicas stored locally or in a remote storage cluster.

[0073] The foregoing description of implementations provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention.

[0074] For example, in another implementation a synchronous version of the de-duplication algorithm may be used in which different storage clusters communicate directly rather than using the replication layer within a distributed data replication system.

[0075] Also, while series of blocks have been described with regard to FIGS. 6A-10, the order of the blocks may be modified in other implementations. Further, non-dependent blocks may be performed in parallel.

[0076] It will be apparent that embodiments, as described herein, may be implemented in many different forms of software, firmware, and hardware in the implementations illustrated in the figures. The actual software code or specialized control hardware used to implement embodiments described herein is not limiting of the invention. Thus, the operation and behavior of the embodiments were described without reference to the specific software code—it being understood that software and control hardware may be designed to implement the embodiments based on the description herein.

[0077] Further, certain implementations described herein may be implemented as “logic” or a “component” that performs one or more functions. This logic or component may include hardware, such as a processor, microprocessor, an application specific integrated circuit or a field programmable gate array, or a combination of hardware and software (e.g., software executed by a processor).

[0078] It should be emphasized that the term “comprises” and/or “comprising” when used in this specification is taken to specify the presence of stated features, integers, steps, or components, but does not preclude the presence or addition of one or more other features, integers, steps, components, or groups thereof.

[0079] Even though particular combinations of features are recited in the claims and/or disclosed in the specification, these combinations are not intended to limit the disclosure of the invention. In fact, many of these features may be combined in ways not specifically recited in the claims and/or disclosed in the specification.

[0080] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article “a” is intended to include one or more items. Where only one item is intended, the term “one” or similar language is used. Further, the phrase “based on,” as used herein is intended to mean “based, at least in part, on” unless explicitly stated otherwise.

What is claimed is:

1. A computer-implemented method comprising:
 - at a first device of a plurality of devices of a distributed storage system, wherein the first device includes one or more processors and memory storing programs for execution by the one or more processors:
 - accessing a global index of objects stored in the distributed storage system, wherein the global index specifies which of the plurality of devices store replicas of each object stored in the distributed storage system;
 - scanning a first portion of the global index;
 - in response to the scanning, identifying a redundant replica of one of the objects, wherein the redundant replica is at a second device of the plurality of devices;
 - writing a new record to the first portion of the global index corresponding to the redundant replica, wherein the new record designates the redundant replica for de-duplication; and
 - subsequent to the writing, replicating the first portion of the global index to the plurality of devices for de-duplication of the redundant replica at the second device.
 - 2. The method of claim 1, wherein the first device is designated as the sole device of the plurality of devices for selecting redundant replicas of objects identified in the first portion of the global index.
 - 3. The method of claim 2, wherein a second device is designated as the sole device of the plurality of devices for selecting redundant replicas of objects identified in a second portion of the global index, wherein the second portion is distinct from the first portion.
 - 4. The method of claim 1, wherein identifying the redundant replica comprises identifying two or more replicas marked for deletion and selecting one of the records marked for deletion as the redundant replica.
 - 5. The method of claim 1, wherein identifying the redundant replica comprises identifying two or more replicas not marked for deletion and selecting one of the records as the redundant replica based on storage capacity at devices where the replicas are stored.
 - 6. The method of claim 1, wherein the global index is a content addressable index.
 - 7. A distributed storage system, comprising a plurality of devices, each device including:
 - one or more processors;
 - memory; and
 - one or more programs stored in the memory configured for execution by the one or more processors, the one or more programs comprising instructions for:
 - at a first device of the plurality of devices:
 - accessing a global index of objects stored in the distributed storage system, wherein the global index specifies which of the plurality of devices store replicas of each object stored in the distributed storage system;
 - scanning a first portion of the global index;
 - in response to the scanning, identifying a redundant replica of one of the objects, wherein the redundant replica is at a second device of the plurality of devices;
 - writing a new record to the first portion of the global index corresponding to the redundant replica, wherein the new record designates the redundant replica for de-duplication; and
 - subsequent to the writing, replicating the first portion of the global index to the plurality of devices for de-duplication of the redundant replica at the second device.
 - 8. The distributed storage system of claim 7, wherein the first device is designated as the sole device of the plurality of devices for selecting redundant replicas of objects identified in the first portion of the global index.
 - 9. The distributed storage system of claim 8, wherein a second device is designated as the sole device of the plurality of devices for selecting redundant replicas of objects identified in a second portion of the global index, wherein the second portion is distinct from the first portion.
 - 10. The distributed storage system of claim 7, wherein the instructions for identifying the redundant replica comprise instructions for identifying two or more replicas marked for deletion and selecting one of the records marked for deletion as the redundant replica.
 - 11. The distributed storage system of claim 7, wherein the instructions for identifying the redundant replica comprise instructions for identifying two or more replicas not marked for deletion and selecting one of the records as the redundant replica based on storage capacity at devices where the replicas are stored.
 - 12. The distributed storage system of claim 7, wherein the global index is a content addressable index.
 - 13. A non-transitory computer readable storage medium storing one or more programs configured for execution by a plurality of devices in a distributed storage system, each having one or more processors and memory, the one or more programs comprising instructions for:
 - at a first device of the plurality of devices:
 - accessing a global index of objects stored in the distributed storage system, wherein the global index specifies which of the plurality of devices store replicas of each object stored in the distributed storage system;
 - scanning a first portion of the global index;
 - in response to the scanning, identifying a redundant replica of one of the objects, wherein the redundant replica is at a second device of the plurality of devices;
 - writing a new record to the first portion of the global index corresponding to the redundant replica, wherein the new record designates the redundant replica for de-duplication; and
 - subsequent to the writing, replicating the first portion of the global index to the plurality of devices for de-duplication of the redundant replica at the second device.
 - 14. The computer readable storage medium of claim 13, wherein the first device is designated as the sole device of the plurality of devices for selecting redundant replicas of objects identified in the first portion of the global index.
 - 15. The computer readable storage medium of claim 14, wherein a second device is designated as the sole device of the plurality of devices for selecting redundant replicas of objects identified in a second portion of the global index, wherein the second portion is distinct from the first portion.
 - 16. The computer readable storage medium of claim 13, wherein the instructions for identifying the redundant replica comprise instructions for identifying two or more replicas marked for deletion and selecting one of the records marked for deletion as the redundant replica.
 - 17. The computer readable storage medium of claim 13, wherein the instructions for identifying the redundant replica

comprise instructions for identifying two or more replicas not marked for deletion and selecting one of the records as the redundant replica based on storage capacity at devices where the replicas are stored.

18. The computer readable storage medium of claim **13**, wherein the global index is a content addressable index.

* * * * *