US 20160085633A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2016/0085633 A1

(43) Pub. Date: Mar. 24, 2016

Factor et al.

(54) **AUTOMATED DATA RECOVERY FROM REMOTE DATA OBJECT REPLICAS**

(71) Applicant: **International Business Machines Corporation**, Armonk, NY (US)

(72) Inventors: **Michael E. Factor**, Haifa (IL); **David Hadas**, Zichron Yaakov (IL); **Elliot K. Kolodner**, Haifa (IL)

(21) Appl. No.: **14/964,050**

(22) Filed: **Dec. 9, 2015**

Related U.S. Application Data

(63) Continuation of application No. 14/058,293, filed on Oct. 21, 2013.

Publication Classification

(51) **Int. Cl.**

G06F 11/14 (2006.01)

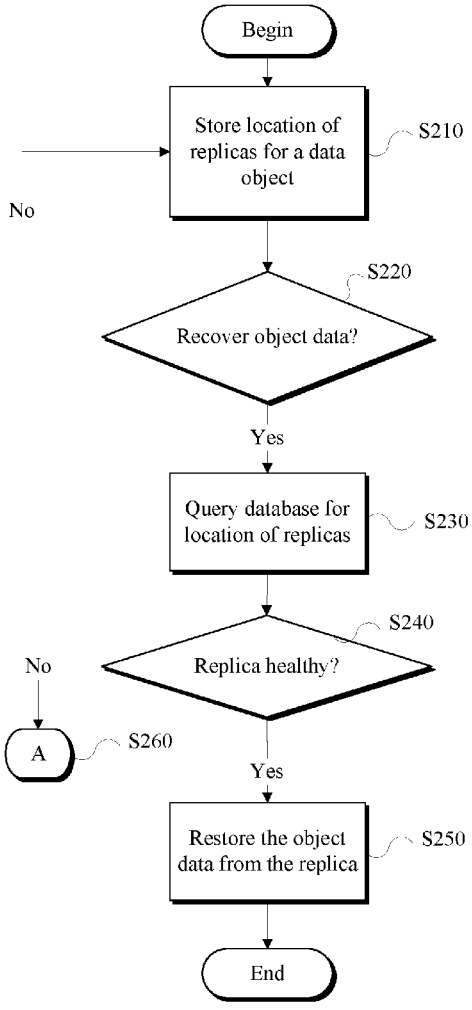
G06F 17/30 (2006.01)

(52) **U.S. Cl.**

CPC ***G06F 11/1451*** (2013.01); ***G06F 17/30424*** (2013.01)

(57) **ABSTRACT**

Machines, systems and methods for recovering data objects in a distributed data storage system, the method comprising storing one or more replicas of a first data object on one or more clusters in one or more data centers connected over a data communications network; recording health information about said one or more replicas, wherein the health information comprises data about availability of a replica to participate in a restoration process; calculating a query-priority for the first data object; querying, based on the calculated query-priority, the health information for the one or more replicas to determine which of the one or more replicas is available for restoration of the object data; calculating a restoration-priority for the first data object based on the health information for the one or more replicas; and restoring the first data object from the one or more of the available replicas, based on the calculated restoration-priority.



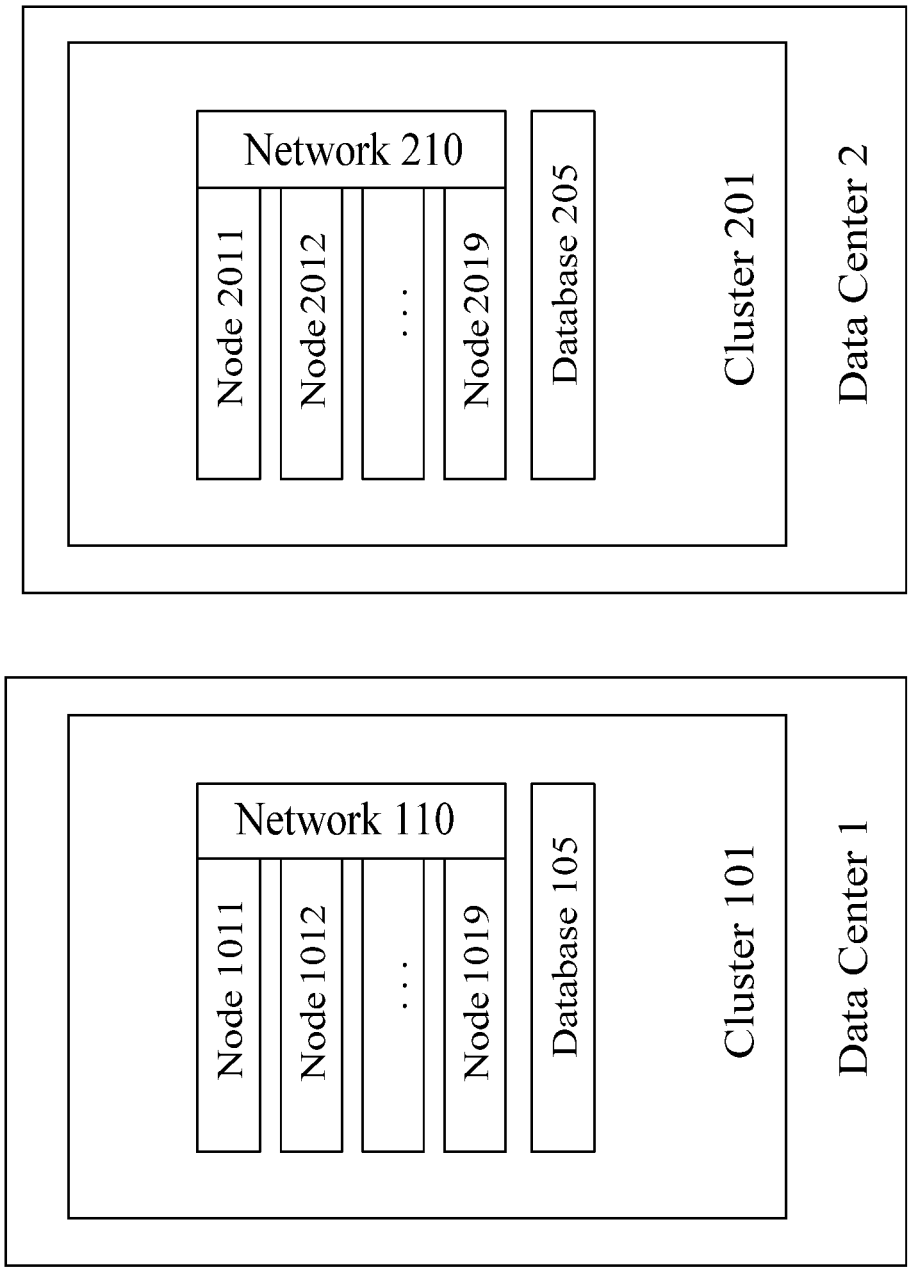

Data Center 1
Cluster 101
Network 110
Node 1011
Node 1012
. . .
Node 1019
Database 105
Data Center 2
Cluster 201
Network 210
Node 2011
Node 2012
. . .
Node 2019
Database 205

FIG. 1A

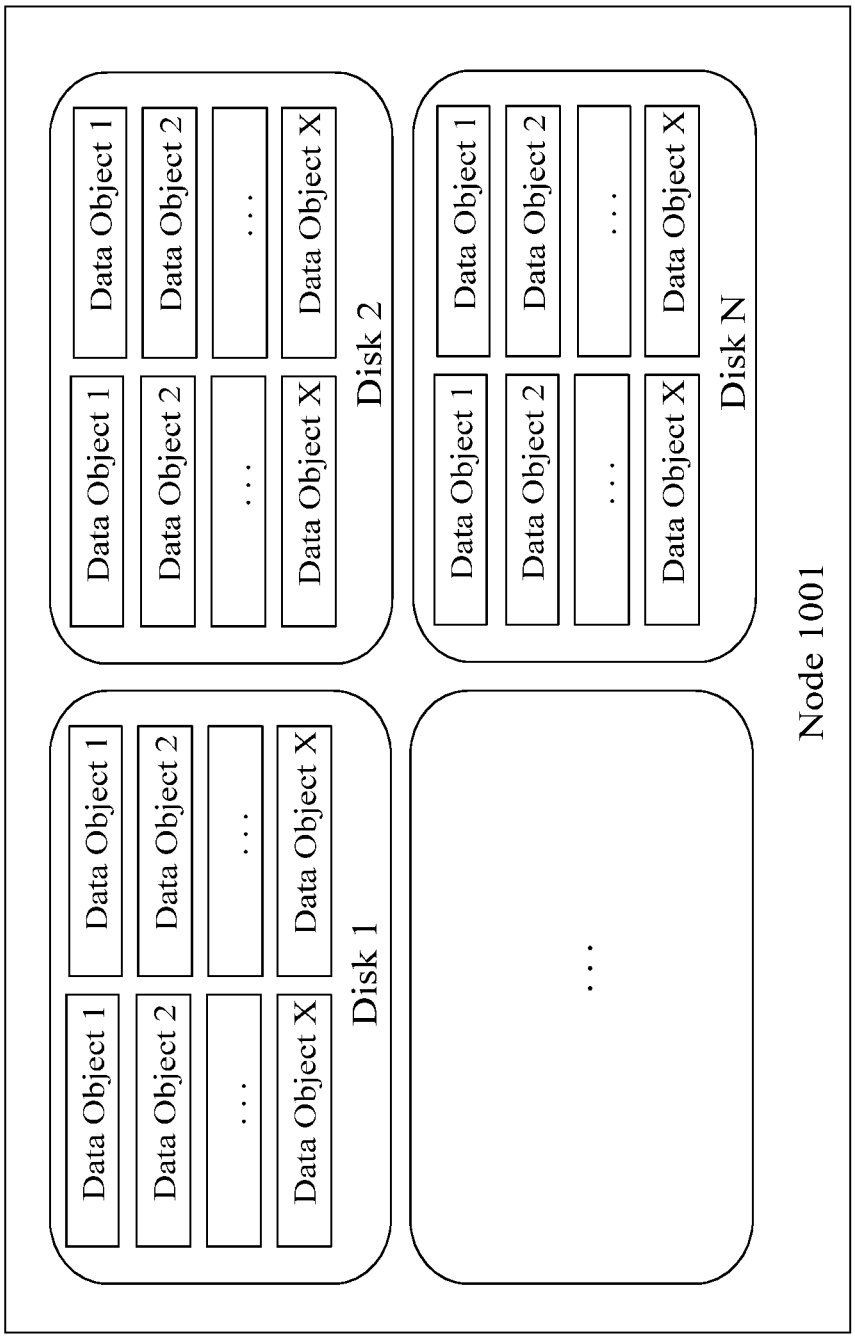

Data Object 1
Data Object 2
. . .
Data Object X
Disk 1
Disk 2
Disk N
Node 1001

FIG. 1B

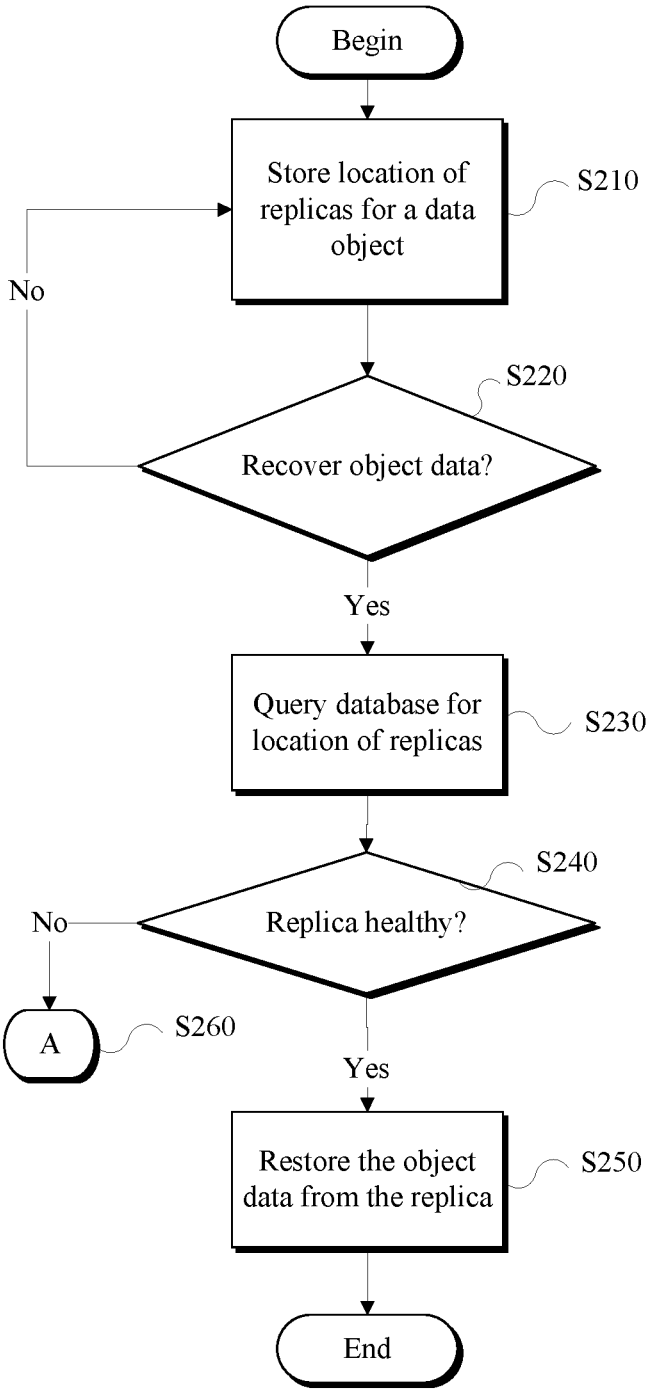

Begin
Store location of replicas for a data object
S210
No
S220
Recover object data?
Yes
Query database for location of replicas
S230
S240
No
Replica healthy?
A
S260
Yes
Restore the object data from the replica
S250
End

FIG. 2

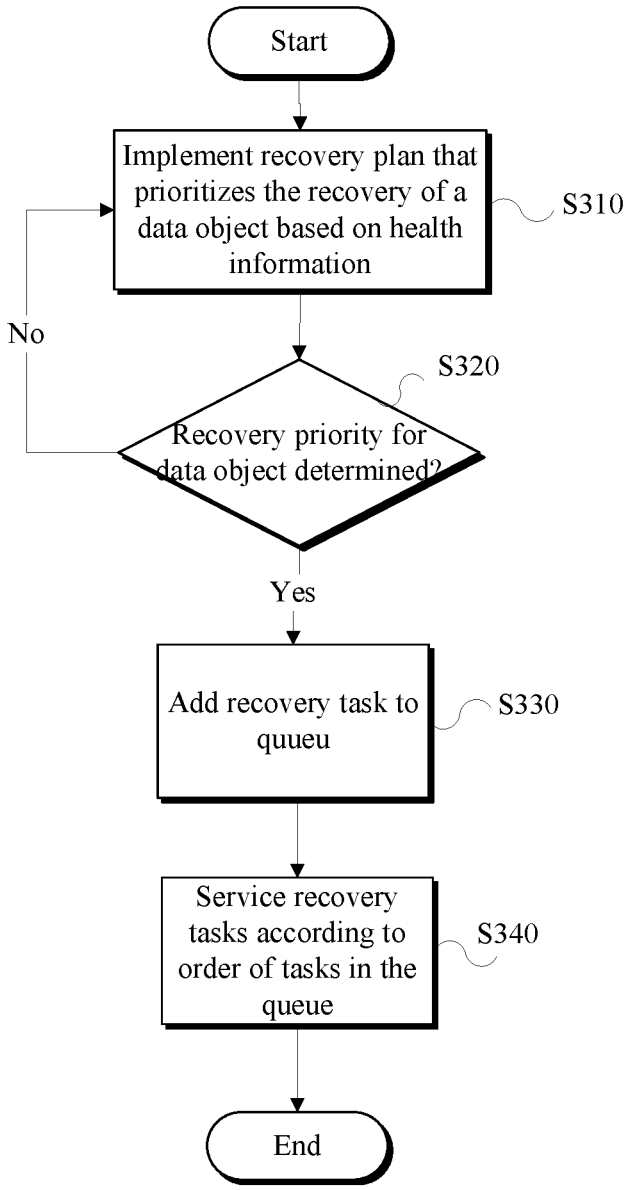

Start
Implement recovery plan that prioritizes the recovery of a data object based on health information
S310
No
S320
Recovery priority for data object determined?
Yes
Add recovery task to quueu
S330
Service recovery tasks according to order of tasks in the queue
S340
End

FIG. 3

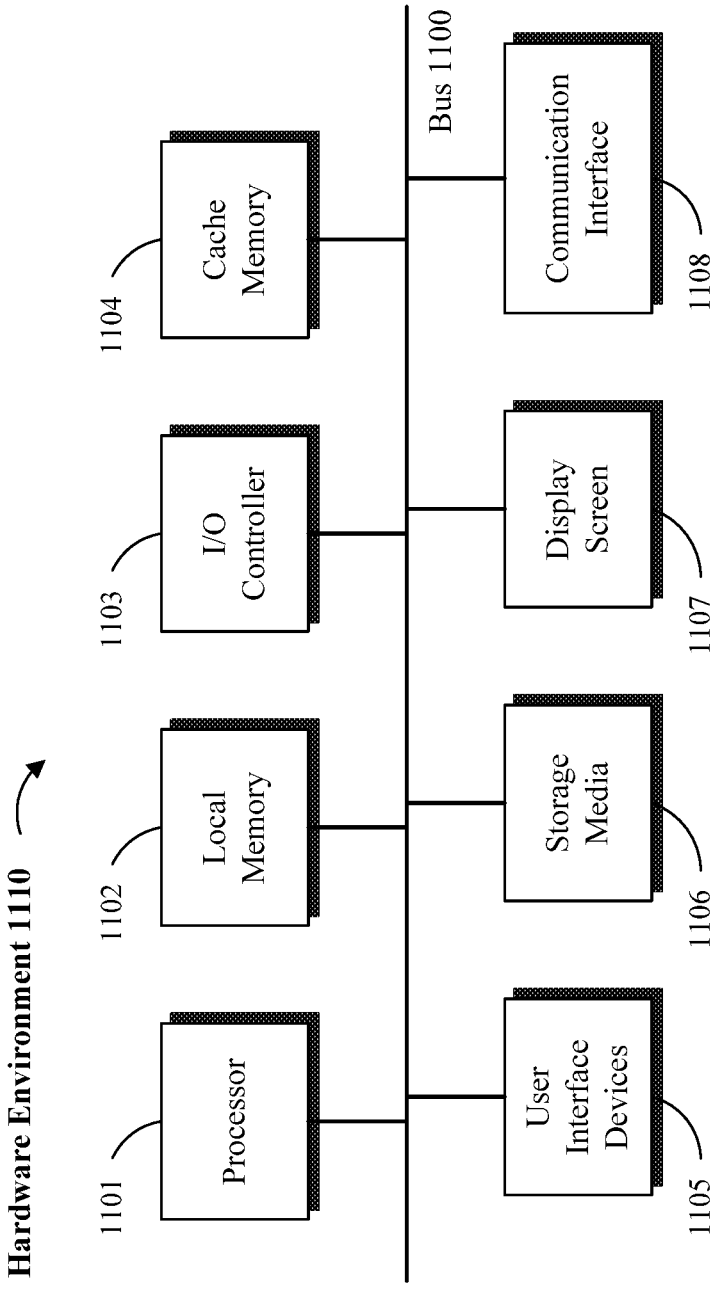

Hardware Environment 1110
1101
Processor
1102
Local Memory
1103
I/O Controller
1104
Cache Memory
Bus 1100
1105
User Interface Devices
1106
Storage Media
1107
Display Screen
1108
Communication Interface

FIG. 4A

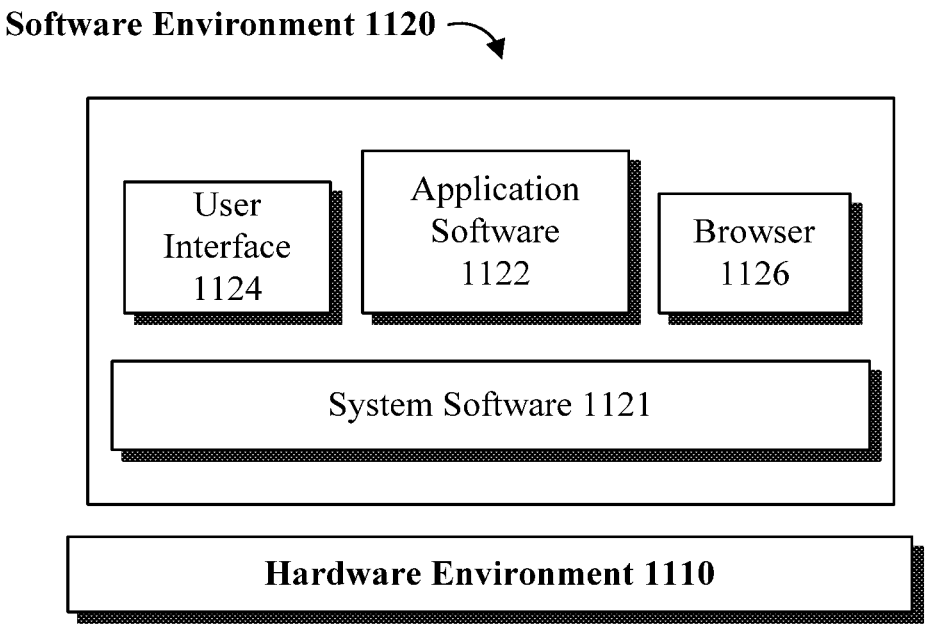

Software Environment 1120
User Interface 1124
Application Software 1122
Browser 1126
System Software 1121
Hardware Environment 1110

FIG. 4B

AUTOMATED DATA RECOVERY FROM REMOTE DATA OBJECT REPLICAS

COPYRIGHT & TRADEMARK NOTICES

[0001] A portion of the disclosure of this patent document may contain material, which is subject to copyright protection. The owner has no objection to the facsimile reproduction by any one of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyrights whatsoever.

[0002] Certain marks referenced herein may be common law or registered trademarks of the applicant, the assignee or third parties affiliated or unaffiliated with the applicant or the assignee. Use of these marks is for providing an enabling disclosure by way of example and shall not be construed to exclusively limit the scope of the disclosed subject matter to material associated with such marks.

TECHNICAL FIELD

[0003] The disclosed subject matter relates generally to data recovery from remotely stored data replicas and, more particularly, to a system and method for dispersing and storing data across a plurality of data centers for the purpose of data recovery.

BACKGROUND

[0004] Mass-scale data storage systems are desirable for their sheer size and ability to process and store vast amounts of data. Generally, these mass-scale systems are built to include a plurality of data centers located remotely from one another. Each data center is usually comprised of a plurality of independent clusters wherein each cluster has a plurality of nodes. These nodes are coupled to each other by way of a data network infrastructure and are ideally independent from each other insofar as the data is stored on storage media that are separately maintained.

[0005] The purpose for keeping the data centers at remote sites is to mitigate the risk of damage to multiple data centers in case of a catastrophic event (e.g., flood, earthquake, or tornado). In the event that a data center experiences the loss of a data object or a replica, the recovery process requires locating a surviving replica from which to replicate or restore the lost data. Accordingly, data centers are typically built with multiple local replicas per cluster. This feature allows a cluster to self-recover from a fault using the surviving local replicas.

[0006] If all local replicas are lost, the data may be recovered from one or more remotely stored replicas. Such remote recovery is currently accomplished via a manual process, often requiring administrative assistance. Due to the size and distribution of resources in a mass-scale data storage system, information regarding the location and availability of replicas across the plurality of data centers and clusters is not readily scalable and, in turn, is unavailable to all nodes across the plurality of data centers.

SUMMARY

[0007] For purposes of summarizing, certain aspects, advantages, and novel features have been described herein. It is to be understood that not all such advantages may be achieved in accordance with any one particular embodiment. Thus, the disclosed subject matter may be embodied or carried out in a manner that achieves or optimizes one advantage or group of advantages without achieving all advantages as may be taught or suggested herein.

[0008] Machines, systems and methods for recovering data objects in a distributed data storage system are provided. The method comprises storing one or more replicas of a first data object on one or more clusters in one or more data centers connected over a data communications network; recording health information about said one or more replicas, wherein the health information comprises data about availability of a replica to participate in a restoration process; calculating a query-priority for the first data object; querying, based on the calculated query-priority, the health information for the one or more replicas to determine which of the one or more replicas is available for restoration of the object data; calculating a restoration-priority for the first data object based on the health information for the one or more replicas; and restoring the first data object from the one or more of the available replicas, based on the calculated restoration-priority.

[0009] In accordance with one or more embodiments, a system comprising one or more logic units is provided. The one or more logic units are configured to perform the functions and operations associated with the above-disclosed methods. In yet another embodiment, a computer program product comprising a computer readable storage medium having a computer readable program is provided. The computer readable program when executed on a computer causes the computer to perform the functions and operations associated with the above-disclosed methods.

[0010] One or more of the above-disclosed embodiments in addition to certain alternatives are provided in further detail below with reference to the attached figures. The disclosed subject matter is not, however, limited to any particular embodiment disclosed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The disclosed embodiments may be better understood by referring to the figures in the attached drawings, as provided below.

[0012] FIGS. **1**A and **1**B illustrate block diagrams of exemplary system infrastructure for storing and recovering data in a distributed data communications network, in accordance with one or more embodiments.

[0013] FIGS. **2** and **3** are flow diagrams illustrating exemplary methods for automated data recovery from remotely stored replicas of lost data objects, in accordance with one embodiment.

[0014] FIGS. **4**A and **4**B are block diagrams of hardware and software environments in which the disclosed systems and methods may operate, in accordance with one or more embodiments.

[0015] Features, elements, and aspects that are referenced by the same numerals in different figures represent the same, equivalent, or similar features, elements, or aspects, in accordance with one or more embodiments.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS

[0016] In the following, numerous specific details are set forth to provide a thorough description of various embodiments. Certain embodiments may be practiced without these specific details or with some variations in detail. In some instances, certain features are described in less detail so as not

to obscure other aspects. The level of detail associated with each of the elements or features should not be construed to qualify the novelty or importance of one feature over the others.

[0017] Referring to FIG. **1**A, a computing environment is illustrated that enables data recovery in a data storage system that replicates copies of data objects (i.e., replicas) across multiple data centers. As shown, a plurality of data centers (e.g., data center **1**, data center **2**, etc.) may be implemented, where a data center is comprised of one or more clusters (e.g., cluster **101**, cluster **201**, etc.). A data center may also comprise one or more networks (e.g., network **110**, network **210**, etc.) having one or more nodes (e.g., nodes **2011**, **2012**, etc.). A remote or local database (e.g., database **105**, database **205**, etc.) in addition to power supplies and cooling systems (not shown) may be also included in a data center.

[0018] Referring to FIG. **1**B, a network node **1001** is illustrated which includes a plurality of storage media (e.g., disks **1** through N) on which multiple data objects **1** through X may be stored. In accordance with one embodiment, a robust, reliable and efficient data recovery system is implemented in which the overall number of data object replicas stored on a single node in the network is limited to a minimum where possible by dispersing and storing data objects on remote nodes or clusters in a distributed data storage system. In this manner, if a fault results in loss, damage, or otherwise unavailability of a replica at a local cluster, the likelihood of recovering the data is higher if one or more replicas of the data are stored in remote clusters, because the probability of the remote cluster having been affected by the same local fault is minimal.

[0019] Referring to FIG. **2**, in accordance with one implementation, a database may be configured to communicate with a local cluster to store and provide the placement of data object replicas for one or more data objects (S**210**), where the data objects are stored on one or more network nodes in the local cluster. For example, a database may store information about data objects to indicate that a data object **1** is stored on disk **1**, in node **1001**, in cluster **101** of data center **1**. Further, additional information may be stored in the database in association with a data object to indicate the locations of one or more replicas of said data object within the local data center and across remote data centers in a distributed storage network. Thus, the database may include a record that associates a local data object with its corresponding remote replicas in clusters **1** through N.

[0020] A data object may be deemed faulty if at least one replica of the data object is lost or damaged. If a faulty data object is to be restored or recovered(S**220**), the database record for the faulty data object may be queried to determine the list of clusters **1** through N that hold replicas of the faulty data object (S**230**). Once the location or locations of the replicas are determined, optionally, a plurality of parallel processes **1** through N may be initiated in order to communicate with the corresponding clusters on which the remote replicas **1** through N are stored. The purpose of such query may be to, for example, determine health data about the availability of a replica for the purposes of recovery (S**240**). Health data may be in form of an indicator (e.g., a logical flag, such as a Boolean operand) associated with the replica for indicating whether the replica is available on the corresponding disk, node or cluster, whether local or remote.

[0021] Accordingly, health data for a replica may be utilized to determine whether the faulty data object may be successfully recovered from that replica. If a healthy replica is available, then the data object may be restored from that replica (S**250**), otherwise remedial measures may be taken (S**260**). For example, if a remote node on which a replica of a faulty data object is stored is conducting a replica recovery process for the same faulty data object, then the health data for the target replica would indicate that the target replica is not available for the purpose of the replica recovery. Health data for a replica in various distributed data centers may also, optionally, include information about the number of replicas available for a certain data object. The health data may be analyzed to calculate a recovery plan based on recovery criteria, which may include the risk of loss based on the number or the location of available replicas on local or remote clusters.

[0022] For example, to define a recovery criteria or plan, a priority function may be used to determine the priority of recovering a first faulty data object over a second faulty data object based on the probability of losing the first faulty data object and the cost associated with losing or recovering the first faulty data object as compared to the second faulty data object. Depending on implementation, for example, it may be desirable to recover faulty data objects with fewer numbers of local duplicates first, over faulty data objects that have a higher number of local duplicates. As another example, it may be desirable to recover faulty data objects that are rated as being stored on less secure or less reliable data storage media first. Other recovery plans with alternative priority schemes may be also implemented in accordance with factors that may suggest or assign higher importance or cost with the recovery of certain data objects.

[0023] In accordance with one embodiment, an exemplary priority function P(D)=Func(N(D),C(D),n) may be used for recovery of a faulty data object from one or more replicas in a distributed network, where:

[0024] D represents a data object with multiple replicas in multiple clusters;

[0025] L(D) represents the local replica of D;

[0026] $R(D)_i$, i=1 . . . n, where "i" and "n" are natural numbers, with a remote replica indexed i of D out of "n" remote replicas;

[0027] H(D)i represents the health of $R(D)_i$ at the time a recovery query is run, wherein ad-hoc information provided by a remote cluster may indicate that a replica is "Available" or "Unavailable", for example;

[0028] N(D) represents the number of remote replicas for which $H(D)_i$="Available", for example;

[0029] C(D) represents the cost of losing N replicas of D;

[0030] P(D) represents the priority given by the system for the recovery operation of D; and

[0031] S(D) represents a source of the remote replica of D that has H(D)i ="Available", for example, and was chosen by the system from which to copy from.

[0032] Referring to FIG. **3**, in accordance with one embodiment, the recovery plan may be implemented using a recovery priority queue, where the relative priority of recovery of a data object is determined from the analysis of the health information associated with the data object and other factors noted above (S**310**). Once the priority of recovery for a data object is determined (S**320**), for example, based on a priority function such as the one noted above, then a corresponding recovery task may be added to the recovery priority queue (S**330**). The recovery tasks in the queue may be serviced according to the order of the tasks in the queue (S**340**). It is noteworthy

that, depending on implementation, the recovery requests added to the recovery priority queue may be added in real time as information noted above about a damaged, or otherwise unavailable data object (i.e., a faulty data object) and its corresponding N replicas is obtained and analyzed.

[0033] In one example scenario, a request for recovery of a faulty data object may be added to the recovery priority queue as soon as possible and, optionally, prior to having determined the recovery priority for the faulty data object. For example, a first recovery request for a first data object and a second recovery request for a second data object may be added to the queue, with the first recovery request being scheduled ahead of the second recovery request. If the recovery of the second data object is determined to be of a higher priority over the recovery of the first data object (e.g., based on the results of the priority function), the second recovery request may be moved to a position in the queue that is ahead of the first recovery request.

[0034] In another example, recovery requests may be added to the queue after the priority is calculated. In this example, first priority for a first recovery request for a first data object may be calculated first and added to the queue and at some later time, a second priority which is higher than the first priority may be calculated for a second recovery request for a second data object such that the second request would be scheduled before the first request. In one implementation, the recovery for the first data object may not be interrupted, if the processing of the first recovery request has already started. As such, the recovery process may be optimized without slowing down the recovery requests that have already started processing, unless a certain predetermined set of conditions are met, for example.

[0035] In accordance with one embodiment, one or more factors may be considered in calculating a recovery plan for the purpose of introducing efficiency into the replica recovery process. As a simplified example, consider a scenario in which it may be determined that out of 10 faulty data objects, healthy replicas for 7 of the 10 objects may be recovered from the same set of remote clusters. A naive approach, in this example scenario, would be to submit 10 separate queries to determine the health or availability of replicas for each of the 10 faulty data objects from 10 separate remote clusters, followed by 10 independent retrieval requests to recover the local replica of the 10 faulty objects, possibly from 10 different remote clusters. It will be appreciated that such naive approach is associated with a large amount of overhead and is costly.

[0036] In one embodiment, health information stored in a (desirably local) recovery database about the replicas may be analyzed to determine the location of replicas available on a local or remote node or cluster. In the above example, 7 out of 10 replicas may be stored on a single local or remote node or cluster. It is noteworthy that at this point, while the location of the replicas is known, the actual health of the replicas may not be known. As such, health inquiries may need to be submitted for the 10 replicas to determine which of the 10 replicas may be suitable for the purpose of recovery. For efficiency, a health inquiry for the 7 replicas that reside on the same node may be submitted in a single inquiry (e.g., instead of submitting 7 separate inquiries).

[0037] In this example, at a health verification stage, responses may be received to the health inquiries submitted. The response would indicate the health of the 10 replicas. Desirably, in one implementation, a single response is received to the single inquiry for the 7 replicas to indicate the health level for one or more (and desirably all) the 7 replicas. Additional queries may be submitted to other remote clusters for health information of replicas on other nodes. For example, 3 separate health inquiries may have to be submitted to 3 separate nodes if the remaining 3 out of the 10 replicas are on 3 separate nodes. Note also that each faulty object may have multiple replicas and that inquiries needs to be sent to one or more clusters in which such replicas should exist to inquire the health of the object.

[0038] Once the health data for the 10 replicas is collected, the recovery priority for the faulty object may be calculated and a list of possible options for recovery of the object may be determined. In the above example, if 4 of the faulty objects being recovered have the same priority, and replicas for the 4 faulty objects may be retrieved from the same remote cluster, then a single recovery request may be submitted for those 4 faulty objects. That is, instead of issuing 4 independent retrieval requests, to the same remote cluster, one retrieval request is submitted. This implementation would save network and processing bandwidth and overhead costs associated with processing the recovery inquiries and requests.

[0039] The cluster recovering the 10 faulty objects therefore may design an efficient recovery scheme for the 10 faulty objects by grouping retrieval requests that share the same or similar priority and may be retrieved from the same source cluster using the information gathered during the health verification stage. Such grouped retrieval requests with their priority may be added to the priority queue once their priority is determined as described above. In one embodiment, the health verification stage and the recovery stage may be intertwined and use the same priority queue such that health verification stage tasks are assigned a priority defined by the apriority knowledge obtained from the database (e.g., based on the number of potential replicas that the faulty object may still have and the cost of losing such object). The recovery task priority may be calculated based on the responses provided at the health verification stage as provided above.

[0040] In accordance with one embodiment, if the replica recovery process from one or more identified nodes or clusters is unsuccessful, the replica recovery process may be repeated by returning to a health verification stage—i.e. by querying the health of the replicas by submitting the recovery inquiry and request to other nodes or clusters that may have replicas of the faulty data objects, even where said nodes or clusters do not provide the most optimal recovery process as calculated. Alternatively, other sources that were calculated at the previous health verification stage may be used to obtain the faulty object, while optionally adjusting the recovery request priority accordingly.

[0041] In summary, the following processes may be implemented for the purpose of recovery:

[0042] 1. Identify a faulty data object by determining that at least N replicas of a data object are lost.

[0043] 2. Queue Task 1 (Query health data & calculate priority):

[0044] 2.1 Query multiple nodes about the health of respective replicas for the faulty object,

[0045] 2.2 Calculate the priority of recovery task for the faulty object and decide the location from which to recover the replica, and

[0046] 2.3 Queue Task 2 to perform the recovery task based on the calculated priority for the faulty object.

[0047] 3. Task 2 (Recovery):

[0048] 3.1 Try to recover from a replica based on health data,

[0049] 3.2 If fail to recover: Queue Task 1 to query the health of replicas for the faulty object again. Depending on implementation, the priority of recovery for the faulty object may or may not be calculated again, according to one or more factors (e.g., if the priority calculation is independent of the health of the replicas).

[0050] As noted earlier, the recovery tasks may be prioritized based on the probability associated with the risk of losing one or more data replicas associated with a faulty data object. Further, a parallel recovery configuration may be implemented such that (a) some or optionally all nodes of a cluster may participate in the replica recovery process, (b) multiple processes may run in parallel to assist with the recovery from multiple available replicas, and (c) replicas may be grouped to speed up mass recovery processes.

[0051] Advantageously, the disclosed parallel replica recovery process from a focused group of nodes or clusters in a distributed data storage environment scales well in larger data storage systems and may be utilized to support an ongoing replica recovery process for recovering a relatively large number of replicas across a large-scale data storage system. In accordance with one embodiment, metadata including a replica's health information or availability status may be maintained in a database at the cluster level per replica (e.g., as opposed to the node level), such that the metadata may be accessed even if the replica data is lost. Such metadata may include the list of candidate clusters from which the replica may be recovered, optionally, the cost of losing the replicated data or the cost of replicating from each respective candidate cluster.

[0052] In one embodiment, the replica metadata may be replicated more times than the data object replica itself, allowing the replica metadata to survive more faults than the replica, for example. A cluster may contain one or more unprotected copies of the replica and one or more protected copies of an object's metadata, either via encoding (to achieve redundancy) or via replication. Depending on implementation, the priority of recovery for a data object or of obtaining the health of a faulty object during a health verification stage may be determined based upon, for example, at least one or more of the following:

[0053] the number of replicas lost for a data object,

[0054] the number of replicas remaining for a data object (e.g., whether de-facto replicas or in theory), and

[0055] the cost of losing one or more replicas of the data object.

[0056] If a candidate replica chosen or located for the purpose of restoration of a respective data object is deemed unavailable, the recovery priority for the data object may be determined as if the respective set of replica held by that candidate cluster is lost. In accordance with one embodiment, a designated cluster may be selected from one of a candidate set of clusters, if the designated cluster responds that the cluster contains a healthy replica. A candidate cluster may report its current state (e.g., the available resources), where a designated cluster may be determined based on the availability of its resources. If a cluster is provided with data about the cost of recovery from one or more candidate clusters, a designated cluster may be selected with a lower (or lowest) associated recovery cost.

[0057] In accordance with one embodiment, a cluster may have information about local replicas of a data object such that a designated cluster for the purpose of recovery may be determined based upon whether there are additional copies of the data object replicas in the cluster. The designated cluster may be determined from among the candidate clusters that responded with the highest number of healthy replicas from the set of replicas that share the same priority, where:

[0058] priority is calculated prior to selecting the designated cluster from which a faulty data object is to be recovered, and

[0059] when the designated cluster is selected, the same designated cluster is chosen for the largest possible subset of faulty data objects that share the same priority.

[0060] For example, in Table 1 below, Cluster 2 will be chosen for replicating faulty data objects 4, 5, 6, 7 and 8. And, Cluster 1 will be chosen for replicating lost replicas 1, 2 and 3.

replica	priority	cluster 1	cluster 2
1	5	ok	ok
2	5	ok	ok
3	5	ok	Missing
4	7	ok	ok
5	7	ok	ok
6	7	ok	ok
7	7	ok	ok
8	7	Missing	ok

[0061] In accordance with one embodiment, after a failure to recover a faulty data object from a designated cluster, a new priority and a new designated cluster may be determined based on the information previously queried from the candidate clusters or by making an additional query to the candidate clusters. A list of faulty data objects or lost replicas for a data object may be determined based on metadata stored in a recovery database. The list may be processed to find a common list of candidate clusters from which a faulty data object may be restored. List of lost replicas may be first divided based on the administrative domain controlling the data such that any further recovery processing is done based on privileges belonging to the said administrative domain.

[0062] References in this specification to "an embodiment", "one embodiment", "one or more embodiments" or the like, mean that the particular element, feature, structure or characteristic being described is included in at least one embodiment of the disclosed subject matter. Occurrences of such phrases in this specification should not be particularly construed as referring to the same embodiment, nor should such phrases be interpreted as referring to embodiments that are mutually exclusive with respect to the discussed features or elements.

[0063] In different embodiments, the claimed subject matter may be implemented as a combination of both hardware and software elements, or alternatively either entirely in the form of hardware or entirely in the form of software. Further, computing systems and program software disclosed herein may comprise a controlled computing environment that may be presented in terms of hardware components or logic code executed to perform methods and processes that achieve the results contemplated herein. Said methods and processes, when performed by a general purpose computing system or machine, convert the general purpose machine to a specific purpose machine.

[0064] Referring to FIGS. **4**A and **4**B, a computing system environment in accordance with an exemplary embodiment may be composed of a hardware environment **1110** and a software environment **1120**. The hardware environment **1110** may comprise logic units, circuits or other machinery and equipments that provide an execution environment for the components of software environment **1120**. In turn, the software environment **1120** may provide the execution instructions, including the underlying operational settings and configurations, for the various components of hardware environment **1110**.

[0065] Referring to FIG. **4**A, the application software and logic code disclosed herein may be implemented in the form of machine readable code executed over one or more computing systems represented by the exemplary hardware environment **1110**. As illustrated, hardware environment **110** may comprise a processor **1101** coupled to one or more storage elements by way of a system bus **1100**. The storage elements, for example, may comprise local memory **1102**, storage media **1106**, cache memory **1104** or other machine-usable or computer readable media. Within the context of this disclosure, a machine usable or computer readable storage medium may include any recordable article that may be utilized to contain, store, communicate, propagate or transport program code.

[0066] A computer readable storage medium may be an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor medium, system, apparatus or device. The computer readable storage medium may also be implemented in a propagation medium, without limitation, to the extent that such implementation is deemed statutory subject matter. Examples of a computer readable storage medium may include a semiconductor or solid-state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk, an optical disk, or a carrier wave, where appropriate. Current examples of optical disks include compact disk, read only memory (CD-ROM), compact disk read/write (CD-R/W), digital video disk (DVD), high definition video disk (HD-DVD) or Blue-ray™ disk.

[0067] In one embodiment, processor **1101** loads executable code from storage media **1106** to local memory **1102**. Cache memory **1104** optimizes processing time by providing temporary storage that helps reduce the number of times code is loaded for execution. One or more user interface devices **1105** (e.g., keyboard, pointing device, etc.) and a display screen **1107** may be coupled to the other elements in the hardware environment **1110** either directly or through an intervening I/O controller **1103**, for example. A communication interface unit **1108**, such as a network adapter, may be provided to enable the hardware environment **1110** to communicate with local or remotely located computing systems, printers and storage devices via intervening private or public networks (e.g., the Internet). Wired or wireless modems and Ethernet cards are a few of the exemplary types of network adapters.

[0068] It is noteworthy that hardware environment **1110**, in certain implementations, may not include some or all the above components, or may comprise additional components to provide supplemental functionality or utility. Depending on the contemplated use and configuration, hardware environment **1110** may be a machine such as a desktop or a laptop computer, or other computing device optionally embodied in an embedded system such as a set-top box, a personal digital assistant (PDA), a personal media player, a mobile communication unit (e.g., a wireless phone), or other similar hardware platforms that have information processing or data storage capabilities.

[0069] In some embodiments, communication interface **1108** acts as a data communication port to provide means of communication with one or more computing systems by sending and receiving digital, electrical, electromagnetic or optical signals that carry analog or digital data streams representing various types of information, including program code. The communication may be established by way of a local or a remote network, or alternatively by way of transmission over the air or other medium, including without limitation propagation over a carrier wave.

[0070] As provided here, the disclosed software elements that are executed on the illustrated hardware elements are defined according to logical or functional relationships that are exemplary in nature. It should be noted, however, that the respective methods that are implemented by way of said exemplary software elements may be also encoded in said hardware elements by way of configured and programmed processors, application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs) and digital signal processors (DSPs), for example.

[0071] Referring to FIG. **4**B, software environment **1120** may be generally divided into two classes comprising system software **1121** and application software **1122** as executed on one or more hardware environments **1110**. In one embodiment, the methods and processes disclosed here may be implemented as system software **1121**, application software **1122**, or a combination thereof. System software **1121** may comprise control programs, such as an operating system (OS) or an information management system, that instruct one or more processors **1101** (e.g., microcontrollers) in the hardware environment **1110** on how to function and process information. Application software **1122** may comprise but is not limited to program code, data structures, firmware, resident software, microcode or any other form of information or routine that may be read, analyzed or executed by a processor **1101**.

[0072] In other words, application software **1122** may be implemented as program code embedded in a computer program product in form of a machine-usable or computer readable storage medium that provides program code for use by, or in connection with, a machine, a computer or any instruction execution system. Moreover, application software **1122** may comprise one or more computer programs that are executed on top of system software **1121** after being loaded from storage media **1106** into local memory **1102**. In a client-server architecture, application software **1122** may comprise client software and server software. For example, in one embodiment, client software may be executed on a client computing system that is distinct and separable from a server computing system on which server software is executed.

[0073] Software environment **1120** may also comprise browser software **1126** for accessing data available over local or remote computing networks. Further, software environment **1120** may comprise a user interface **1124** (e.g., a graphical user interface (GUI)) for receiving user commands and data. It is worthy to repeat that the hardware and software architectures and environments described above are for purposes of example. As such, one or more embodiments may be implemented over any type of system architecture, functional or logical platform or processing environment.

[0074] It should also be understood that the logic code, programs, modules, processes, methods and the order in which the respective processes of each method are performed are purely exemplary. Depending on implementation, the processes or any underlying sub-processes and methods may be performed in any order or concurrently, unless indicated otherwise in the present disclosure. Further, unless stated otherwise with specificity, the definition of logic code within the context of this disclosure is not related or limited to any particular programming language, and may comprise one or more modules that may be executed on one or more processors in distributed, non-distributed, single or multiprocessing environments.

[0075] As will be appreciated by one skilled in the art, a software embodiment may include firmware, resident software, micro-code, etc. Certain components including software or hardware or combining software and hardware aspects may generally be referred to herein as a "circuit," "module" or "system." Furthermore, the subject matter disclosed may be implemented as a computer program product embodied in one or more computer readable storage medium (s) having computer readable program code embodied thereon. Any combination of one or more computer readable storage medium(s) may be utilized. The computer readable storage medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing.

[0076] In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device. A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

[0077] Program code embodied on a computer readable storage medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing. Computer program code for carrying out the disclosed operations may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages.

[0078] The program code may execute entirely on the user's computer, partly on the user's computer, as a standalone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

[0079] Certain embodiments are disclosed with reference to flowchart illustrations or block diagrams of methods, apparatus (systems) and computer program products according to embodiments. It will be understood that each block of the flowchart illustrations or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, a special purpose machinery, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions or acts specified in the flowchart or block diagram block or blocks.

[0080] These computer program instructions may also be stored in a computer readable storage medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable storage medium produce an article of manufacture including instructions which implement the function or act specified in the flowchart or block diagram block or blocks.

[0081] The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer or machine implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions or acts specified in the flowchart or block diagram block or blocks.

[0082] The flowchart and block diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical functions. It should also be noted that, in some alternative implementations, the functions noted in the block may occur in any order or out of the order noted in the figures.

[0083] For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams or flowchart illustration, and combinations of blocks in the block diagrams or flowchart illustration, may be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

[0084] The claimed subject matter has been provided here with reference to one or more features or embodiments. Those skilled in the art will recognize and appreciate that, despite of the detailed nature of the exemplary embodiments provided here, changes and modifications may be applied to said embodiments without limiting or departing from the generally intended scope. These and various other adaptations and combinations of the embodiments provided here are

within the scope of the disclosed subject matter as defined by the claims and their full set of equivalents.

1. A method for recovering data objects in a distributed data storage system, the method comprising:

storing one or more replicas of a first data object on one or more clusters in one or more data centers connected over a data communications network;

recording health information about said one or more replicas, wherein the health information comprises data about availability of a replica to participate in a restoration process;

in response to determining that the first data object is to be recovered, calculating a query-priority for the first data object;

querying, based on the calculated query-priority, the health information for the one or more replicas to determine which of the one or more replicas is available for restoration of the first data object;

calculating a restoration-priority for the first data object based on the health information for the one or more replicas; and

restoring the first data object from the one or more of the available replicas, based on the calculated restoration-priority, wherein the query-priority is calculated based on a priority function P(D)=Func(R(D),C(D),n), where:

D represents a data object with multiple replicas in multiple clusters;

$R(D)_i$, i=1 . . . n, where "i" and "n" are natural numbers, with a remote replica indexed i of D out of n remote replicas;

C(D) represents cost of losing N replicas of D;

P(D) represents priority given by the system for the query operation of D; and

Func( )represents some function.

2. (canceled)

3. The method of claim **1**, wherein the restoration-priority is calculated based on a second function with parameters that represent at least one of:

number of replicas available for restoring the first data object in the distributed data storage system,

projected costs associated with losing M replicas of the first data object in the distributed data storage system, or

risk of loss associated with delaying recovery of the first data object over a second data object, wherein the risk of loss is calculated based on the number of replicas or the costs of loss associated with the first and second data objects.

4. The method of claim **1**, wherein the one or more data centers are located remotely from the location in which the first data object is being recovered.

5. The method of claim **1**, wherein the health information comprises availability and location data about the one or more replicas.

6. The method of claim **1**, wherein querying the health information of a plurality of data objects from a location is optimized by aggregating queries for at least a subset of the plurality of the data objects in a single query.

7. The method of claim **6**, wherein data objects in the subset of the plurality of the data objects in the single query are associated with the same query-priority.

8. The method of claim **1**, wherein the restoration of a plurality of data objects is optimized by restoring at least a subset of the plurality of the data objects from a location where a maximum collective number of the plurality of data objects have replicas, using a single restoration request.

9. The method of claim **8**, wherein data objects in the subset of the plurality of the data objects restored using a single restoration request are associated with the same restoration-priority.

10. (canceled)

11. The method of claim **1**, wherein the restoration-priority is calculated based on a second priority function P(D)=Func (N(D),C(D),n), where:

D represents a data object with multiple replicas in multiple clusters;

$R(D)_i$, i=1 . . . n, where "i" and "n" are natural numbers, with a remote replica indexed i of D out of n remote replicas;

$H(D)_i$ represents health of $R(D)_i$ at the time a recovery query is run, wherein ad-hoc information provided by a remote cluster may indicate that a replica is "Available" or "Unavailable", for example;

N(D) represents number of remote replicas for which $H(D)_i$ is available;

C(D) represents cost of losing N replicas of D;

P(D) represents priority given by the system for the restoration operation of D; and

Func( )represents some function.

12. The method of claim **1**, wherein the health information for the one or more replicas is recorded by a first cluster in a database accessible by a recovery module of the first cluster.

13. The method of claim **1**, wherein a first priority queue is used in the one or more clusters to record and prioritize one or more tasks for restoring one or more data objects such that:

a first task associated with priority set to be the first data object query-priority is inserted into the first priority queue to query the health information for the one or more replicas of the first data object from one or more locations, in response to determining that the first data object is to be recovered, and

a second task with priority set to be the first data object restoration-priority is inserted into the first priority queue to restore the first data object from the one or more available replicas in the one or more locations, in response to determining the restoration-priority for the first data object;

wherein the one or more tasks are extracted from the first priority queue and executed, such that a task with higher priority is executed before a task with lower priority.

14. The method of claim **13**, wherein given a first plurality of data objects with health information that is to be queried from the same location or have the same query-priority, optimizing the querying such that a first task for querying health information for a subset of one or more of the first plurality of data objects is inserted as a single task in the first priority queue.

15. The method of claim **13**, wherein given a second plurality of data objects that are to be restored from the same location or have the same restoration-priority, optimizing the restoring such that a second task for restoring a subset of one or more of the second plurality of data objects is inserted as a single task in the first priority queue.

* * * * *