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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 September 2013 (19.09.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/137917 A1

(51) International Patent Classification:
G06F 9/06 (2006.01) *G06F 12/16* (2006.01)

(21) International Application Number:
PCT/US2012/034794

(22) International Filing Date:
24 April 2012 (24.04.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/611,046 15 March 2012 (15.03.2012) US

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

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **CAMBLE, Peter Thomas** [GB/GB]; Longdown Avenue, Stoke Gifford Bristol BS34 8QZ (GB). **TODD, Andrew** [GB/GB]; Longdown Avenue, Stoke Gifford Bristol BS34 8QZ (GB). **CHANDRASEKHARAN, Kaushik** [IN/GB]; Longdown Avenue, Stoke Gifford Bristol BS34 8QZ (GB).

(74) Agents: **ORTEGA, Arthur** et al.; HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P., Intellectual Property Administration, Mail Stop 35 3404 E. Harmony Road, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

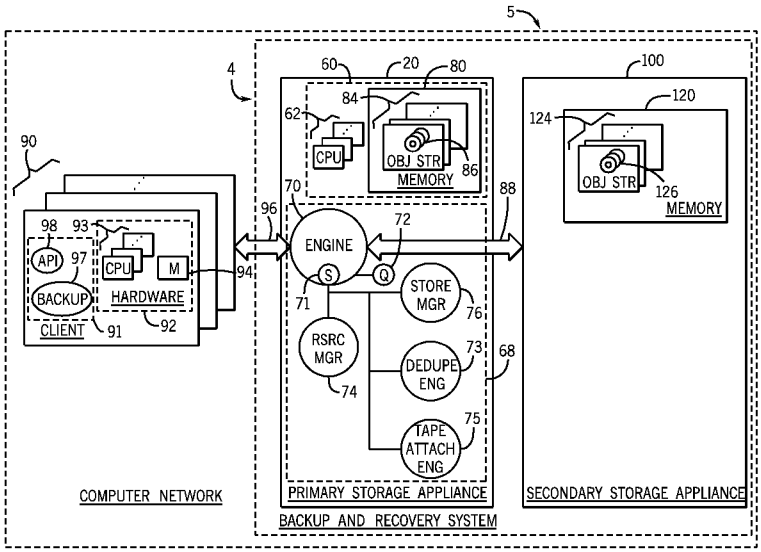
— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

[Continued on next page]

(54) **Title**: DETERMINING A SCHEDULE FOR A JOB TO REPLICATE AN OBJECT STORED ON A STORAGE APPLIANCE



(57) **Abstract**: A technique includes queuing jobs to replicate object data stored on a storage appliance. The technique includes, for at least one of the jobs, selectively regulating when the job appears in the schedule based at least in part on a number of failed attempts to complete the job.

WO 2013/137917 A1

Published:

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

DETERMINING A SCHEDULE FOR A JOB TO REPLICATE AN OBJECT STORED ON A STORAGE APPLIANCE

Background

[0001] A typical computer network may have a backup and recovery system for purposes of restoring data (data contained in one or multiple files, for example) on the network to a prior state should the data become corrupted, be overwritten, subject to a viral attack, etc. The backup and recovery system typically includes mass storage devices, such as magnetic tape drives and/or hard drives; and the system may include physical and/or virtual removable storage devices.

[0002] For example, the backup and recovery system may store backup data on magnetic tapes, and after a transfer of backup data to a given magnetic tape, the tape may be removed from its tape drive and stored in a secure location, such as in a fireproof safe. The backup and recovery system may alternatively be a virtual tape library-based system that emulates and replaces the physical magnetic tape drive system. In this manner, with a virtual tape library-based system, virtual cartridges, instead of magnetic tapes, store the backup data.

Brief Description Of The Drawings

[0003] Fig. 1 is a schematic diagram of a computer network that includes a backup and recovery system according to an example implementation.

[0004] Fig. 2 is an illustration of an object store used by the backup and recovery system of Fig. 1 according to an example implementation.

[0005] Fig. 3 is an illustration of objects in an object store created during a backup session according to an example implementation.

[0006] Fig. 4 is a flow diagram depicting a technique to replicate backup data according to an example implementation.

[0007] Fig. 5 is a flow diagram depicting a technique to access object-based backup data stored on the backup and recovery system of Fig. 1 and control at least one aspect of an operation to replicate the backup data according to an example implementation.

[0008] Fig. 6 is a flow diagram depicting a technique used by a backup application of Fig. 1 to regulate replication of data by the backup and recovery system according to an example implementation.

[0009] Fig. 7 is a flow diagram depicting a technique used by the backup application of Fig. 1 to search and/or group data objects stored on the backup and recovery system according to an example implementation.

[0010] Fig. 8 is a flow diagram depicting a technique to schedule replication jobs according to an example implementation.

[0011] Fig. 9 is a flow chart depicting a technique to set a rate at which replication jobs are attempted according to an example implementation.

[0012] Fig. 10 is a flow chart depicting a technique to anticipatorily tag jobs as failing according to an example implementation.

[0013] Fig. 11 is a flow diagram depicting a technique to regulate a timing of status request inquiries according to an example implementation.

[0014] Fig. 12 is a flow chart depicting a technique to regulate a time for a client to resubmit a status request inquiry according to an example implementation.

Detailed Description

[0015] Fig. 1 depicts an example computer network 5 that includes a backup and recovery system 4 and one or multiple clients 90 of the system 4, which generate backup data (during backup sessions) stored on the system 4. The backup data may include numerous types of data, such as application-derived data, system state information, applications, files, configuration data and so forth. In general, a given client 90 may access the backup and recovery system 4 during a recovery session to restore selected data and possibly restore the client to a particular prior state. As a non-limiting example, client(s) 90 may, in general, be servers of networks that are not illustrated in Fig. 1.

[0016] In accordance with example implementations, the backup and recovery system 4 includes a primary storage appliance 20 that stores backup data for the client(s) 90 and a secondary storage appliance 100 that stores copies of this backup data. In this manner, for such purposes of adding an additional layer of backup security, the primary storage appliance 20 may occasionally replicate backup data stored on the primary storage appliance 20 to produce corresponding replicated backup data stored by the secondary storage appliance 100.

[0017] Depending on the particular implementation, the primary storage appliance 20 and the secondary storage appliance 100 may be located at the same facility and share a local connection (a local area network (LAN) connection, for example) or may be disposed at different locations and be remotely connected (via a wide area network (WAN) connection, for example). In the example that is depicted in Fig. 1, the primary storage appliance 20 communicates with the secondary storage appliance 100 using a communication link 88. The communication link 88 represents one or multiple types of network fabric (i.e., WAN connections, LAN connections wireless connections, Internet connections, and so forth).

[0018] The client(s) 90 communicate with the primary storage appliance 20 using a communication link 96, such as one or multiple buses or other fast interconnects. The communication link 96 represents one or multiple types of network fabric (i.e., WAN connections, LAN connections wireless connections, Internet connections, and so forth). In general, the client(s) 90 may communicate with the primary storage

appliance 20 using one or multiple protocols, such as a serial attach Small Computer System Interface (SCSI) bus protocol, a parallel SCSI protocol, a Universal Serial Bus (USB) protocol, a Fibre Channel protocol, an Ethernet protocol, and so forth.

[0019] Depending on the particular implementation, the communication link 96 may be associated with a relatively high bandwidth (a LAN connection, for example), a relatively low bandwidth (a WAN connection, for example) or an intermediate bandwidth. Moreover, a given client 90 may be located at the same facility of the primary storage appliance 20 or may be located at a different location than the primary storage appliance 20, depending on the particular implementation. One client 90 may be local relative to the primary storage appliance 20, another client 90 may be remotely located with respect to the primary storage appliance, and so forth. Thus, many variations are contemplated, which are within the scope of the appended claims.

[0020] In accordance with some implementations, the primary storage appliance 20, the secondary storage appliance 100 and the client(s) 90 are "physical machines," or actual machines that are made up of machine executable instructions (i.e., "software") and hardware. Although each of the primary storage appliance 20, the secondary storage appliance 100 and the client(s) 90 is depicted in Fig. 1 as being contained within a box, a particular physical machine may be a distributed machine, which has multiple nodes that provide a distributed and parallel processing system.

[0021] In accordance with some implementations, the physical machine may be located within one cabinet (or rack); or alternatively, the physical machine may be located in multiple cabinets (or racks).

[0022] A given client 90 may include such hardware 92 as one or more central processing units (CPUs) 93 and a memory 94 that stores machine executable instructions 93, application data, configuration data and so forth. In general, the memory 94 is a non-transitory memory, which may include semiconductor storage devices, magnetic storage devices, optical storage devices, and so forth. The client 90 may include various other hardware components, such as one or more of the following: mass storage drives; a network interface card to communicate with the communication link 96; a display; input devices, such as a mouse and a keyboard;

and so forth.

[0023] A given client 90 may include machine executable instructions 91 that when executed by the CPU(s) 93 of the client 90 form a backup application 97. In general, the backup application 97 performs various functions pertaining to the backing up and restoring of data for the client 90. As a non-exhaustive list of examples, the functions that are performed by the backup application 97 may include one or more of the following: generating backup data; communicating backup data to the primary storage appliance 20; accessing the backup data on the primary storage appliance 20; searching and organizing the storage of backup data on the primary storage appliance 20; reading, writing and modifying attributes of the backup data; monitoring and controlling one or multiple aspects of replication operations that are performed at least in part by the primary storage appliance 20 to replicate backup data onto the secondary storage appliance 100; performing one or more functions of a given replication operation; restoring data or system states on the client 20 during a recovery session; and so forth.

[0024] The client 90 may include, in accordance with exemplary implementations that are disclosed herein, a set of machine executable instructions that when executed by the CPU(s) 93 of the client 90 form an application programming interface (API) 98 for accessing the backup and recovery system 4. In general, the API 98 is used by the backup application 97 to communicate with the primary storage appliance 20 for purposes of performing one of the above-recited functions of the application 97.

[0025] In accordance with implementations, the client 90 may include a set of machine executable instructions that form an adapter for the backup application 97, which translates commands and requests issued by the backup application 97 into corresponding API commands/requests, and vice versa.

[0026] A given client 90 may include other various other sets of machine executable instructions that when executed by the CPU(s) 93 of the client 90 perform other functions. As examples, a given client 90 may contain machine executable instructions for purposes of forming an operating system; a virtual machine hypervisor; a graphical user interface (GUI) to control backup/restore operations;

device drivers; and so forth. Thus, many variations are contemplated, which are within the scope of the appended claims.

[0027] Being a physical machine, the primary storage appliance 20 also contains hardware 60 and machine executable instructions 68. For example, the hardware 60 of the primary storage appliance 20 may include one or more CPUs 62; a non-transitory memory 80 (a memory formed from semiconductor storage devices, magnetic storage devices, optical storage devices, and so forth) that stores machines executable instructions, application data, configuration data, backup-related data, and so forth; and one or multiple random access drives 63 (optical drives, solid state drives, magnetic storage drives, etc.) that store, back-up related data, application data, configuration data, etc.; one or multiple sequential access mass storage devices (tape drives, for example); network interface cards; and so forth.

[0028] As also depicted in Fig. 1, the machine executable instructions 68, when executed by one or more of the CPUs 62 of the primary storage appliance 20 form various software entities for the appliance 20 such as one or more of the following, which are described herein: an engine 70, a resource manager 74, a store manager 76, a deduplication engine 73 and a tape attach engine 75.

[0029] Similar to the primary storage appliance 20, the secondary storage appliance 100 is also a physical machine that contains hardware, such as memory 120; one or more CPU(s); mass storage drives; network interface cards; and so forth. Moreover, the secondary storage appliance 100 also contains machine executable instructions to form various applications, device drivers, operating systems, components to control replication operations, and so forth.

[0030] In accordance with implementations that are disclosed herein, the backup and recovery system 4 manages the backup data as "objects" (as compared to managing the backup data as files pursuant to a file based system, for example). As can be appreciated by the skilled artisan, an "object" is an entity that is characterized by such properties as an identity, a state and a behavior; and in general, the object may be manipulated by the execution of machine executable instructions. In particular,

the properties of the objects disclosed herein may be created, modified, retrieved and generally accessed by the backup application 97. In accordance with some implementations, the object may have an operating system-defined maximum size.

[0031] The objects that are stored in the backup and recovery system 4 may be organized in data containers, or "object stores." In general, in accordance with exemplary implementations, an object store has a non-hierarchical, or "flat," address space, such that the objects that are stored in a given object store are not arranged in a directory-type organization.

[0032] For the example that is depicted in Fig. 1, the primary storage appliance 20 stores backup data in the form of one or multiple objects 86, which are organized, or arranged, into one or multiple object stores 84. Moreover, for the example that is depicted in Fig. 1, the objects 86 and object stores 84 are depicted as being stored in the memory 80, although the underlying data may be stored in one or multiple mass storage drives of the primary storage appliance 20.

[0033] The secondary storage appliance 100 stores the replicated backup data in the form of one or multiple replicated objects 126, which are organized, or arranged, in one or multiple object stores 124. In other words, the replicated objects 126 are derived from the objects 86 that are stored on the primary storage appliance 20. Moreover, for the example that is depicted in Fig. 1, the objects 126 and object stores 124 are depicted as being stored in the memory 120, although the underlying data may be stored in one or multiple mass storage drives of the secondary storage appliance 100.

[0034] During a given backup session, the backup application 97 of a given client 90 accesses the primary storage appliance 20 over the communication link 96 to create, modify (append to, for example) or overwrite one or more of the backup objects 86 for purposes of storing or updating backup data on the primary storage appliance 20. Likewise, during a given restoration session, the backup application 97 of a given client 90 may access the primary storage appliance 20 to retrieve one or more of the backup objects 86. In accordance with some implementations, an object 86 on the primary storage appliance 20 may be restored from a corresponding replicated object 126 stored on the secondary storage appliance 100.

[0035] For purposes of reading from or writing to a given object 86, the backup application 97 opens the object 86 and then seeks to a given location of the opened object 86 to read/write a collection of bytes. Moreover, because the data stored in the object 86 may be compressed (as further disclosed herein), the read/writing of data may include reading/writing without first decompressing, or rehydrating, the data; or the reading/writing may alternatively involve first rehydrating the data.

[0036] The API 98, in general, provides a presentation of the object stores 84 and objects 86 to the backup application 97, which allows the backup application 97 to search for objects 86, modify objects 86, create objects 86, delete objects 86, retrieve information about certain objects 86, update information about certain objects 86, and so forth. Referring to Fig. 2 in conjunction with Fig. 1, as a more specific example, the API 98 may present the backup application 97 with a given object store 84, which contains N objects 86 (objects 86-1. . . 86-N, being depicted as examples). In general, the objects 86 may contain data generated during one or more backup sessions, such as backup data, an image of a particular client state, header data, and so forth. The API 98 further presents object metadata 150 to the backup application 97, which the backup application 97 may access and/or modify. In general, the metadata 150 is stored with the objects 86 and describes various properties of associated objects 86, as well as stores value-added information relating to the object 86.

[0037] As examples, the metadata 150 may indicate one or more of the following for a given associated object 86: an object type; a time/date stamp; state information relating to a job history and the relation of the object 86 to the job history; an identifier for the associated object 86; a related object store for the associated object 86; information pertaining to equivalents to legacy-tape cartridge memory contents; keys; etc. As examples, the object type may refer to whether incremental or full backups are employed for the object 86; identify the backup application 97 that created the object 86; identify the client 90 associated with the object 86; a data type (header data, raw backup data, image data, as examples); and so forth.

[0038] Access and control of the objects 86 occurs via interaction with the primary storage appliance's engine 70, the resource manager 74, the store manager 76, the deduplication engine 73 and the tape attach engine 75. In accordance with some

exemplary implementations, the engine 70 serves as an external service end point for the communication links 88 and 96 for data path and control. More specifically, in accordance with some implementations, the commands and requests that are issued by the client 90 are processed by the engine 70, and vice versa. As non-limiting examples, the commands that are processed by the engine 70 include commands to open objects, close objects, write to data to objects, overwrite objects, read objects, read object data, delete objects, modify/write metadata-related information about objects, read metadata-information about objects, set preferences and configuration parameters, and so forth. The requests may include, for example, status inquiry requests, such as a request, for example, concerning the status of a particular replication job. The engine 70 further controls whether the backup and recovery system 4 operates in a low bandwidth mode of operation (described below) or in a high bandwidth mode of operation (described below) and in general, controls, replication operations to create/modify the replicated objects 126 on the secondary storage appliance 100.

[0039] The resource manager 74 manages the locking of the objects 86 (i.e., preventing modification by more than one entity at a time), taking into account resource constraints (the physical memory available, for example). In general, the resource manager 74 preserves coherency pertaining to object access and modification, as access to a given object 86 may be concurrently requested by more than one entity.

[0040] The store manager 76 of the primary storage appliance 20 is responsible for retrieving given object stores 84, controlling entities that may create and delete object stores 84, controlling the access to the object stores, controlling how the object stores 84 are managed, and so forth.

[0041] The deduplication engine 73 of the primary storage appliance 20 controls hashing and chunking operations (described below) for the primary storage appliance 20 for the primary storage appliance's high bandwidth mode of operation (also described below). The deduplication engine 73 also checks whether a chunk has already been stored, and hence, decides whether to store the data or reference existing data. The deduplication engine 73 performs this checking for both low and high bandwidth modes, in accordance with exemplary implementations.

[0042] The tape attach engine 75 may be accessed by the client 90 for purposes of storing a replicated physical copy of one or more objects 86 onto a physical tape that is inserted into a physical tape drive (not shown in Fig. 1) that is coupled to the tape attach engine 75.

[0043] Referring to Fig. 3 in conjunction with Fig. 1, in accordance with exemplary implementations, the backup application 97 may create and/or modify a given set of objects 86 during an exemplary backup session. For this example, the objects are created in an exemplary object store 84-1 on the primary storage appliance 20. The creation/modification of the objects 86, in general, involves interaction with the engine 70, the resource manager 74 and the store manager 76.

[0044] The objects 86 for this example include a header object 86-1, which contains the header information for the particular backup session. As a non-limiting example, the header object 86-1 may contain information that identifies the other objects 86 used in the backup session, identifies the backup session, indicates whether compression is employed, identifies a particular order for data objects, and so forth. The objects 86 for this example further include various data objects (data objects 86-2. . .86-P, being depicted in Fig. 3), which correspond to sequentially-ordered data fragments of the backup session and which may or may not be compressed. For this example, the objects 86 include an image object 86-P+1, which may be used as a recovery image, for purposes of restoring a client 90 to a given state.

[0045] It is noted that the backup application 97 may randomly access the objects 86. Therefore, unlike backup data stored on a physical or virtual sequential access device (such as a physical tape drive or a virtual tape drive), the backup application 97 may selectively delete data objects 86 associated with a given backup session as the objects 86 expire. Moreover, the backup application 97 may modify a given object 86 or append data to an object 86, regardless of the status of the other data objects 86 that were created/modified in the same backup session.

[0046] For purposes of generating the replicated objects 126 that are stored on the secondary storage appliance 100, the backup and recovery system 4 uses data replication operations, called "deduplication operations." The deduplication operations, in general, reduce the amount of data otherwise communicated across

the communication link 88 between the primary storage appliance 20 and the secondary storage appliance 100. Such a reduction may be particularly beneficial when the communication link 88 is associated with a relatively low bandwidth (such as a WAN connection, for example).

[0047] Fig. 4 generally depicts an example replication operation 200, in accordance with some implementations, for purposes of replicating the objects 86 stored on the primary storage appliance 20 to produce corresponding replicated objects 126, which are stored in corresponding object stores 124 on the secondary storage appliance 100. Referring to Fig. 4 in conjunction with Fig. 1, in accordance with exemplary implementations, the replication operation 200 includes partitioning (block 204) the source data (i.e., the data of the source object 86) into blocks of data, called "chunks." In this manner, the partitioning produced an ordered sequence of chunks to be stored on the secondary storage appliance 100 as part of the destination, replication object 126.

[0048] For purposes of reducing the amount of data communicated over the communication link 88, the chunk is not communicated across the communication link 88 if the same chunk (i.e., a chunk having a matching or identical byte pattern) is already stored on the secondary storage appliance 100. Instead, a reference to the previously stored chunk is stored in its place in the destination object, thereby resulting in data compression.

[0049] For purposes of determining whether a given chunk has already been stored on the secondary storage appliance 100, a signature of the chunk is first communicated to the secondary storage appliance 100. More specifically, in accordance with exemplary implementations, a cryptographic function may be applied to a given candidate chunk for purposes of determining (block 208 of Fig. 4) a corresponding unique hash for the data. The hash is then communicated to the secondary storage appliance 100, pursuant to block 212. The secondary storage appliance 100 compares the received hash to hashes for its stored chunks to determine whether a copy of the candidate chunk is stored on the appliance 100 and informs the primary storage appliance 20 of the determination.

[0050] If a match occurs (decision block 216), the primary storage appliance 20 does

not transmit the candidate chunk to the secondary storage appliance 100. Instead, the primary storage appliance 20 transmits a corresponding reference to the already stored chunk to be used in its place in the destination object, pursuant to block 220. Otherwise, if a match does not occur (pursuant to decision block 216), the primary storage appliance 20 transmits the candidate chunk across the communication link 88 to the secondary storage appliance 100, pursuant to block 224. The secondary storage appliance 100 therefore stores either a chunk or a reference to the chunk in the corresponding object 126.

[0051] If there is another chunk to process (decision block 228), control returns to block 208. The chunks are therefore processed in the above-described manner until the source data has been replicated in its compressed form onto the secondary storage appliance 100. The data reduction due to the above-described data deduplication operation 200 may be characterized by a data compression, or "deduplication," ratio.

[0052] Referring back to Fig. 1, in accordance with exemplary implementations, the above-described replication of the objects 86 may be performed in one of two modes of operation for the backup and recovery system 4: a low bandwidth mode of operation; or a high bandwidth mode of operation. For the low bandwidth mode of operation, the client 90 performs the above-referenced chunking and hashing functions of the replication operation. In other words, the client 90 partitions the source data into chunks; applies a cryptographic function to the chunks to generate corresponding hashes; transmits the hashes; and subsequently transmits the chunks or references to the chunks, depending on whether a match occurs. The low bandwidth mode of operation may be particularly advantageous if the client 90 has a relatively high degree of processing power; the communication link 96 is a relatively low bandwidth link (a WAN connection, for example); the deduplication ratio is relatively high; or a combination of one or more of these factors favor the chunking and hashing to be performed by the client 90.

[0053] In the high bandwidth mode of operation, the chunking and hashing functions are performed by the primary storage appliance 20. The high bandwidth mode of operation may be particularly advantageous if the primary storage appliance 20 has a relatively high degree of processing power, the communication link 96 has a

relatively high bandwidth (a LAN connection, for example); the deduplication ratio is relatively low; or a combination of one or more of these factors favor the chunking and hashing to be performed by the primary storage appliance 100.

[0054] In accordance with some implementations, the backup application 97 may specify a preference regarding whether the low bandwidth or the high bandwidth mode of operation is to be employed. As an example, the preference may be communicated via a command that is communicated between the client 90 and the engine 70. Based on this preference, the engine 70 either relies on the client 90 (for the low bandwidth mode of operation) or on the deduplication engine 73 (for the high bandwidth mode of operation) to perform the chunking and hashing functions.

[0055] Referring to Fig. 5 in conjunction with Fig. 1, to summarize, in accordance with exemplary implementations, the API 98 permits the backup application 97 to perform a technique 250. Pursuant to the technique 250, the API 98 provides an interface to the client of a storage appliance, which allows the client to access an object (the "source object") that is stored on the storage appliance, pursuant to block 254. The client may communicate (block 258) with the storage appliance to control at least one aspect of an operation to replicate at least part of the source object to produce a destination object. Thus, as set forth above, as an example, pursuant to a technique 260 (see Fig. 6), the backup application 97 may access (block 262) an object 86 that is stored on a primary storage appliance 20 and cause metadata (block 266) for the object 86 to indicate a preference regarding whether the client 90 or the primary storage appliance 20 performs compression (chunking and hashing) for deduplication of the object 86.

[0056] It is noted that replication may occur between differ object stores on the same storage appliance, or even data between two objects within a given object store. Although the entire object may be replicated, a given replication operation may involve replicating part of a given object, rather than the entire object. Moreover, a destination object may be constructed from one or multiple replicated regions from one or multiple source objects; and the destination object may be interspersed with one or multiple regions of data backed up from the client directly to the destination object. Thus, many variations are contemplated, which are within the scope of the appended claims.

[0057] The use of objects by the backup and recovery system 4 allows a relatively richer searching and grouping of backup data, as compared to, for example, a virtual tape drive-based system in which the backup data is arranged in files that are stored according to a tape drive format. More specifically, referring to Fig. 7 in conjunction with Fig. 1, pursuant to a technique 270, the backup application 97 may access (block 274) objects that are stored on the primary storage appliance and search and/or group the objects based on the associated metadata, pursuant to block 278.

[0058] In accordance with an example implementation, the replication engine 70 includes a scheduler 71 for scheduling replication jobs to replicate the objects 86 to produce the corresponding replicated objects 126 that are stored on the secondary storage appliance 100. In this manner, the scheduler 71 stores, or queues, identifiers for pending replication jobs in a queue 72 for purposes of copying part or all of the data in a given object 86 to a defined location of a target object 126 in a destination object store 124. It is noted that a given replication operation may involve a complete or partial overwrite of an object.

[0059] In accordance with implementations disclosed herein, the scheduler 71 manages when jobs in the queue 72 are run based upon a number of potential criteria. As non-limiting examples, these criteria may include the number/extent of free resources; blackout windows (imposed by the customer); network connectivity; and when source and target appliances are online and available.

[0060] In general, the scheduler 71 pauses the running of the jobs due to such events as a given appliance going offline or another pausible condition occurring (network link unavailability, for example); and the scheduler 71 resumes the jobs when such an event terminates. The scheduler 71 further cancels a given job when an unrecoverable error occurs, such as, as non-limiting examples, a destination appliance exhausting its disk space, a license not being present, an account not being permitted; or the customer canceling the job.

[0061] In general, the scheduler 71 uses the techniques disclosed herein for purposes of running the jobs relatively efficiently without incurring a significant amount of time scanning for possible runnable jobs. In this manner, the number of jobs stored in the queue 72 may be on the scale of millions of possible jobs, in

accordance with some implementations. Therefore, the techniques, which are disclosed herein for scheduling the jobs are directed to imparting a relatively low overhead and latency for the scheduler 71, in accordance with example implementations.

[0062] As a non-limiting example, the scheduler 71 determines a schedule for performing the jobs, i.e., times for each of the jobs to be run or re-run. The scheduler 71, in accordance with exemplary implementations, determines how long to wait before trying to run a replication job that failed, based on previous run attempts.

[0063] Using Eqs. 1 and 2, the scheduler 71 may schedule the jobs, pursuant to a technique 300, which is generally depicted in Fig. 8. Pursuant to the technique 300, the scheduler 71 queues (block 304) the jobs to replicate objects stored on a first storage appliance onto a second storage appliance and determines (block 308) times for performing the jobs. For at least one of the jobs, the scheduler 71 selectively regulates when the job appears in the schedule based at least in part on a number of failed attempts to complete the job, pursuant to block 312.

[0064] As a more specific example, the scheduler 71 may regulate how often a job is attempted (i.e., regulate an "attempt rate" for a given job) based at least in part on the number of one or more failed attempts in completing the job. For example, Fig. 9 depicts an example technique 320, which may be employed by the scheduler 71 in accordance with some implementations. According to the technique 320, the scheduler 71 progressively sets a slower attempt rate for running a given job, depending on the number of failed attempts. For this example, the constants N_1 (decision block 322), N_2 (decision block 326)and N_P (decision block 330) are monotonically increasing from N_1 to N_P, such that $N_1<N_2<N_P$. Initially, the attempt rate for a given job may be relatively high (i.e., may occur at a relatively high frequency). However, as the number of attempts for a given job increase, the corresponding attempt rate decreases. In this manner, Fig. 9 discloses exemplary attempt rates R_1 (block 324), R_2 (block 328) and R_P (block 332), such that $R_1>R_2>R_P$. The attempt rates R_1, R_2 and R_P correspond to failed attempt constants N_1, N_2 and N_P, respectively. In this manner, if the number of failed attempts is less than N_1 (decision block 322), the scheduler 71 sets (block 324) the corresponding attempt

rate at R_1, which is a relatively higher attempt rate. However, if the failed attempts increase such that the attempts are greater than N_1 and still less than N_2, the scheduler 71 then (pursuant to decision block 326) sets (block 328) the attempt rate at a lower attempt rate R_2. The progressive backing off of the time intervals between attempts continues in that when the failed attempts surpass N_P (decision block 330), the scheduler 71 sets (block 334) the attempt rate at the lowest attempt rate R_{P+1}.n accordance with example implementations, the scheduler 71 does not run a given job when the scheduler 71 detects that the failure of a previous job has failed for a reason that would be common to this job.

[0065] More specifically, the scheduler 71 periodically scans the queue 72 for replication jobs, which are ready to run, based on the schedule that is determined above, pursuant to the technique 320. In this regard, replication jobs may, in accordance with exemplary implementations, target a relatively small number of storage appliances (i.e. more than one job per target storage appliance). If during a particular scan, a replication job to a particular appliance is attempted but fails to run due to a reason (a disk space full error, a link error, a blackout window, as non-limiting examples) which would also affect all of the other jobs that may begin running to that storage appliance in this scan, then the other replication jobs are not attempted. Instead, the scheduler 71 anticipatorily presumes that these other jobs would fail as well to the community shared problem and correspondingly tags these jobs as failing as well. This approach avoids the overhead in attempting to run jobs, which are not able to run (at least for the current scan).

[0066] Thus, in accordance with example implementations, the scheduler 71 may perform a technique 334 that is depicted in Fig. 10. Pursuant to the technique 334, the scheduler 71 determines (decision block 336) whether a given replication job has failed and if so, determines (decision block 338) whether the same problem that precipitated the failure applies to one or multiple other replication jobs in the queue 72. If so, the scheduler 71 tags (block 340) the other replication job(s) as failing (e.g., makes one or multiple corresponding entries in status fields stored by the queue 72.

[0067] Although a blackout window is common to multiple jobs, the difference between a failure due to a blackout window and other failed reasons is that the

blackout window is configured at the primary storage appliance 20, in accordance with some implementations. Therefore, the primary storage appliance 20 knows when the blackout window no longer applies. In accordance with an example implementation, the queue 72 stores the next run time as well as an identifier indicating the reason why the job did not run. On the next scan, if a given status identifier for a given job indicates that the last job was not run due to a blackout window, the scheduler 71 resets the associated next run time to "immediately" and resets the number of failed attempts, so that if the job fails to run in the future for a different reason, the job is starting from a clean slate.

[0068] Referring back to Fig. 1, the clients 90, in general, submit status inquires to the primary storage appliance 20 for purposes of acquiring the statuses relating to corresponding replication jobs. For purposes of managing these status inquiries for such purposes as reducing network traffic and reducing the overhead on the scheduler 71, the scheduler 71 serves as a job manager that replies to a given status request inquiry from a requesting client 90 with a corresponding time for the requesting client 90 to wait before re-checking the status.

[0069] In general, the scheduler 71 performs a technique 350 that is depicted in Fig. 11, in accordance with an example implementation. Pursuant to the technique 350, the scheduler 71 queues (block 354) jobs to replicate object data stored on one or multiple storage appliances. The scheduler 71 receives (block 358) a status request inquiry from a client 90 and replies (block 362) to the status request inquiry, and the reply indicates a time (i.e., a minimum wait time) for the client 90 to provide another status request inquiry.

[0070] In determining status request inquiry times, the scheduler 71 may determine a percentage of completion for a given job (called "PercentageComplete"), as described below:

$$\text{PercentageComplete} = \text{Origin Object Extent Size}/(\text{Bytes Copies so far}), \quad \text{Eq. 1}$$

where "Origin Object Extent Size" represents the size of the object 86, and "Bytes Copied so far" represents the number of bytes that have been copied to the secondary storage appliance 100. The scheduler 71 may also estimate a completion

time (called "EstimatedCompletionTime"), as set forth below:

EstimatedCompletionTime=timeNow + (Job RunTimeSeconds * (100 Job PercentageComplete))/(Job PercentageComplete)), Eq. 2

where "Job RunTimeSeconds" represents the current time that the job has been running and "100 Job PercentageComplete" represents a constant, such as "100."

[0071] In this regard, in accordance with an example implementation, the scheduler 71, in response to a given status request inquiry, responds or replies with a time for the client 90 to wait before resubmitting a status inquiry. It is noted that depending on the particular implementation, the time may be an absolute time or may be a relative wait time interval from the time at which the client 90 has submitted the previous inquiry or has received the response from the scheduler 71.

[0072] As an example, Fig. 12 depicts a technique 400 that may be employed by the scheduler 71 for purposes of determining one or multiple inquiry times (as further described below) for a received status request inquiry about a particular, replication job. Pursuant to the technique 400, the scheduler 71 determines (block 404) a percentage of completion for the job (using Eq. 1, for example) and estimates (block 408) a completion time for the replication job (using Eq. 2, for example). In general, if the scheduler 71 determines (decision block 412) that the replication job is paused or pending (the job is in the queue 72 waiting to be run again), the scheduler 71 holds off any more status inquiries pertaining to the replication job until the time that is estimated pursuant to the technique 300. In this manner, for a paused or pending job, the scheduler 71 sets (block 416) the next status inquiry time to the next run attempt time.

[0073] If the scheduler 71 determines (decision block 412) that the replication job is not paused or pending, then the scheduler 71 determines (decision block 420) whether the job is currently running. If so, the scheduler 71 holds off any more status inquiries until the job progress status has measurably changed. More specifically, the scheduler 71 may, in accordance with example implementations, set (block 424) the status inquiry time to the estimated time for measured progress to occur. For example, depending on the particular implementation, the scheduler 71

may deem the job progress to have measurably changed based on, as examples, a given granularity of change (a one % change for example) set forth by the PercentageComplete determination of Eq. 1, a fixed number of bytes (1 gigabyte (GB), for example) being transferred, or the maximum of either of these criteria.

[0074] Thus, in accordance with some implementations, the scheduler 71 regulates the status inquiries by a given client 90 such that the client 90 queries just often enough to receive an indicated change in status from the scheduler 71. If the scheduler 71 determines (decision block 420) that the job is not currently running, then the scheduler 71 determines (block 428) whether the job is cancelled or completed. If not, the status request inquiry targets a non-identified job; and the scheduler 71 takes the appropriate corrective action. Otherwise, if the job is cancelled or completed, the scheduler 71 sets (block 432) the inquiry time to a time that is based on a fixed time interval. For example, the scheduler 71 may set the next query time to a maximum value (five minutes, as an example), as the cancellation is the terminal state for that job.

[0075] A given client status inquiry many inquire about the status of multiple replication jobs. For these requests, the scheduler 71 determines a suggested next query time for each job in the returned status reply and then sets the next overall query time to coincide with the shortest interval of the determined query times. Therefore, the client 90 has up-to-date information for the most rapidly changing job status via the reply. Thus, in accordance with example implementations, the scheduler 71 determines (decision block 436) whether the status request inquiry is associated with multiple jobs. If not, the scheduler 71 replies (block 440) with the next inquiry time for the single replication job. Otherwise, in accordance with example implementations, the scheduler 71 replies (block 437) with an inquiry time for each job and further replies with the next overall inquiry time (the minimum of the individual inquiry times, for example).

[0076] In accordance with example implementations, the scheduler 71 may bound, or constrain, the next query time within a range defined by a minimum value (thirty seconds, for example) and a maximum value (five minutes, for example).

[0077] While a limited number of examples have been disclosed herein, those

skilled in the art, having the benefit of this disclosure, will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations.

What Is claimed Is:

1. A method comprising:

queuing jobs to replicate object data stored on a storage appliance;

determining a schedule for performing the jobs; and

for at least one of the jobs, selectively regulating when the job appears in the schedule based at least in part on a number of failed attempts to complete the job.

2. The method of claim 1, wherein selectively regulating comprises varying a wait interval for performing the job based on the number of failed attempts such that a longer wait interval corresponds to a larger number of failed attempts.

3. The method of claim 1, wherein selectively regulating comprises comparing the number of failed attempts to a second schedule of failed attempts and adjusting a wait interval for performing the job based at least in part on the comparison.

4. The method of claim 1, further comprising receiving the jobs into a queue in response to at least one backup session generated by a backup application executing on a client coupled to the first storage appliance.

5. The method of claim 1, further comprising further basing the schedule on whether the at least one job failed due to a user imposed replication blackout interval.

6. The method of claim 1, further comprising:

determining whether a given job of the jobs has failed and is subject to a failure problem associated with at least one of the other jobs; and

selectively tagging the at least one of the other jobs as failing based at least in part on the determination.

7. An apparatus comprising:

a queue to identify jobs to replicate object data stored on a storage appliance; and

a processor-based job manager to:

receive a status request inquiry from a client to the storage appliance for a status of at least one of the jobs; and

in response to the status request inquiry, indicate a time for the client to provide another status request.

8. The apparatus of claim 7, wherein the job manager is adapted to indicate the time based at least in part on a time at which the job is expected to be completed.

9. The apparatus of claim 7, wherein the job manager is adapted to set the time based at least in part on a next run attempt data for the job.

10. The apparatus of claim 7, wherein the job manager is adapted to base the time on a fixed time interval and on a determination of whether the job has been cancelled or completed.

11. The apparatus of claim 7, wherein the status request is associated with a plurality of jobs, and the job manager is adapted to indicate a time for each of the jobs and an overall time for the client to provide another status request.

12. An article comprising a computer readable storage medium to store instructions that when executed by at least one processor cause the at least one processor to:

queue jobs to replicate object data stored on a storage appliance;

determine a schedule for performing the jobs; and

for at least one of the jobs, selectively regulate when the job appears in the schedule based at least in part on a number of failed attempts to complete the job.

13. The article of claim 12, the storage medium to store instructions that

when executed by the at least one processor cause the at least one processor to vary a wait interval for performing the job based on the number of failed attempts such that a longer wait interval corresponds to a larger number of failed attempts.

14. The article of claim 12, the storage medium to store instructions that when executed by the at least one processor cause the at least one processor to compare the number of failed attempts to a second schedule of failed attempts and adjust a wait interval for performing the job based at least in part on the comparison.

15. The article of claim 12, the storage medium to store instructions that when executed by the at least one processor cause the at least one processor to further base the schedule on whether the at least one job failed due to a user imposed replication blackout interval.

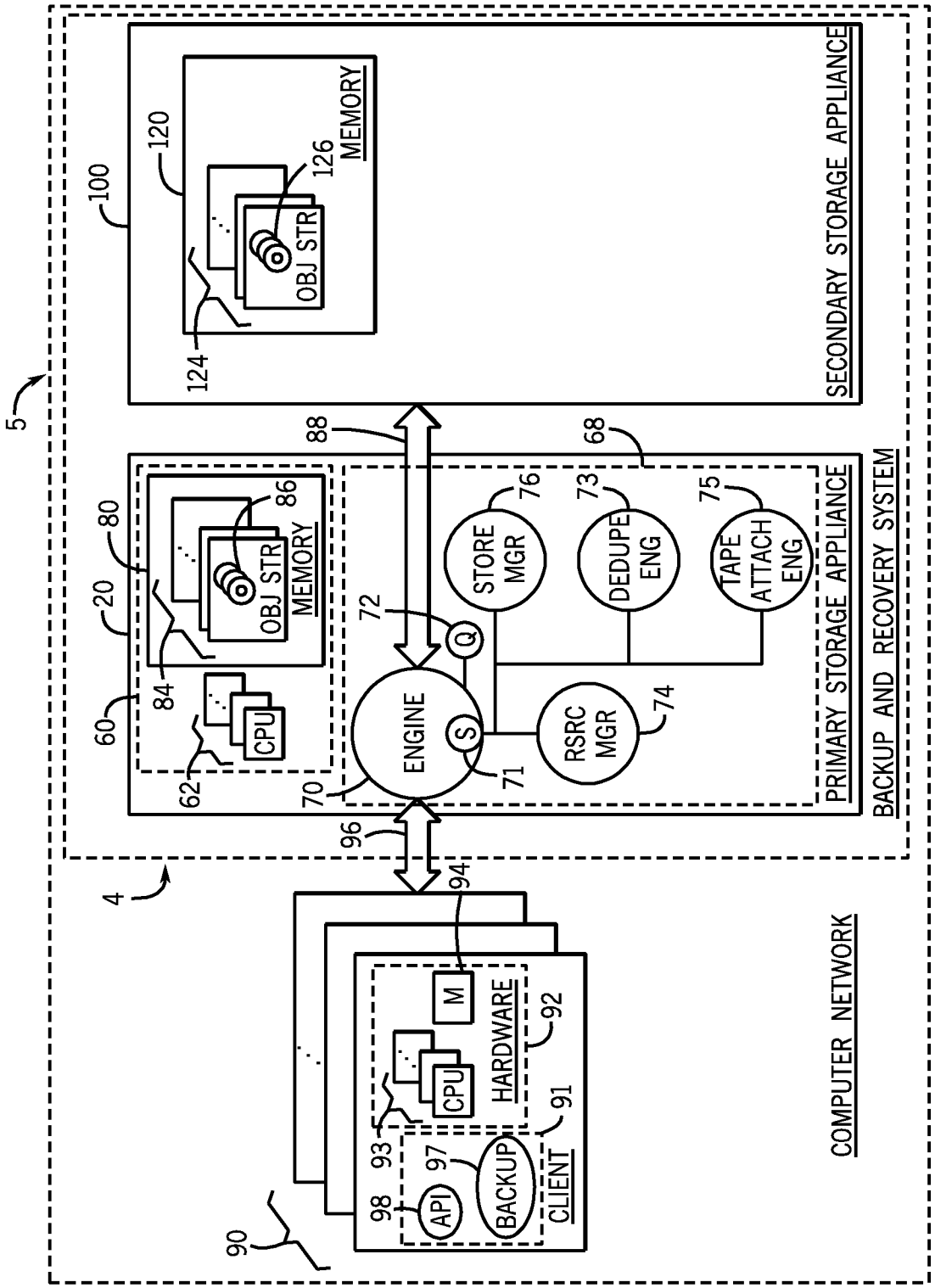

5
4
100
120
126
MEMORY
OBJ STR
124
SECONDARY STORAGE APPLIANCE
88
68
80
86
OBJ STR
MEMORY
20
84
60
CPU
62
72
Q
ENGINE
S
70
71
STORE MGR
76
DEDUPE ENG
73
TAPE ATTACH ENG
75
RSRC MGR
74
PRIMARY STORAGE APPLIANCE
BACKUP AND RECOVERY SYSTEM
96
94
M
CPU
HARDWARE
92
91
93
97
BACKUP
CLIENT
98
API
90
COMPUTER NETWORK

FIG. 1

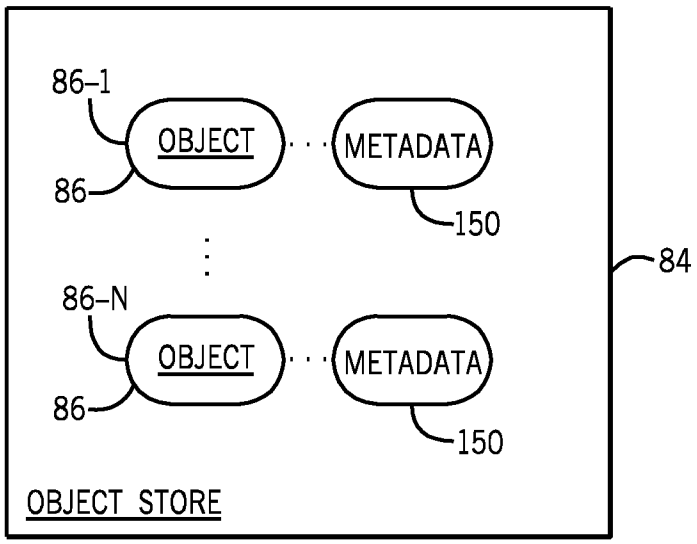

86–1
OBJECT
METADATA
86
150
86–N
OBJECT
METADATA
86
150
84
OBJECT STORE

FIG. 2

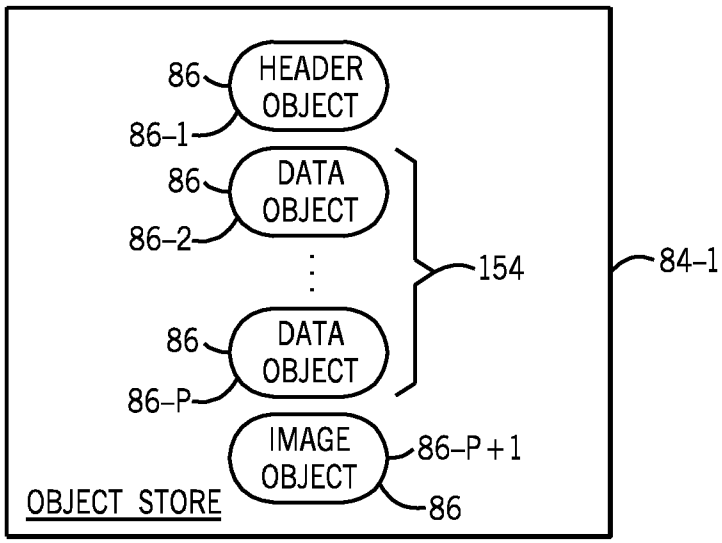

86
HEADER OBJECT
86–1
86
DATA OBJECT
86–2
154
86
DATA OBJECT
86–P
IMAGE OBJECT
86–P+1
86
84–1
OBJECT STORE

FIG. 3

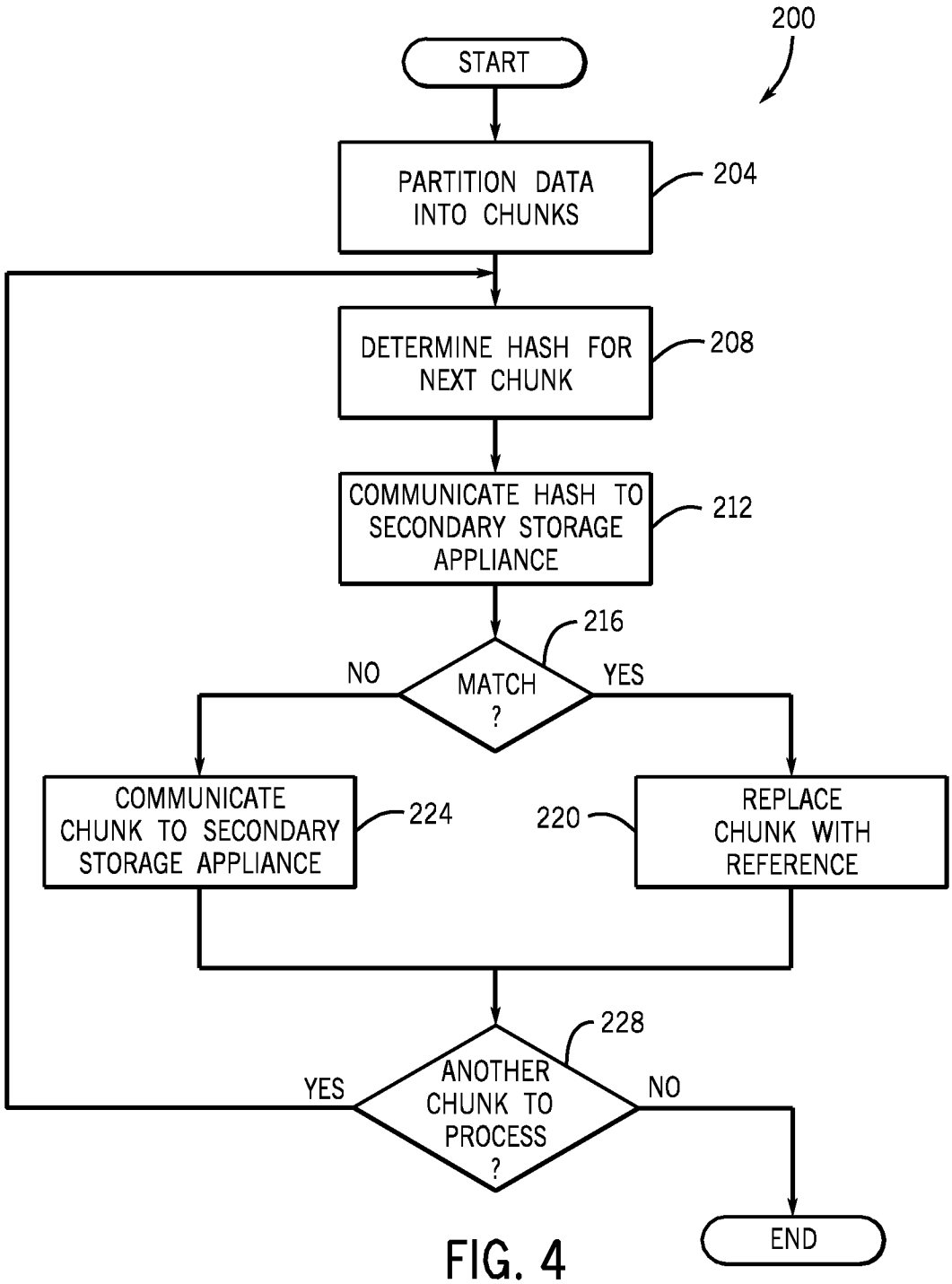

200
START
PARTITION DATA INTO CHUNKS
204
DETERMINE HASH FOR NEXT CHUNK
208
COMMUNICATE HASH TO SECONDARY STORAGE APPLIANCE
212
216
MATCH ?
NO
YES
COMMUNICATE CHUNK TO SECONDARY STORAGE APPLIANCE
224
220
REPLACE CHUNK WITH REFERENCE
228
ANOTHER CHUNK TO PROCESS ?
YES
NO
END

FIG. 4

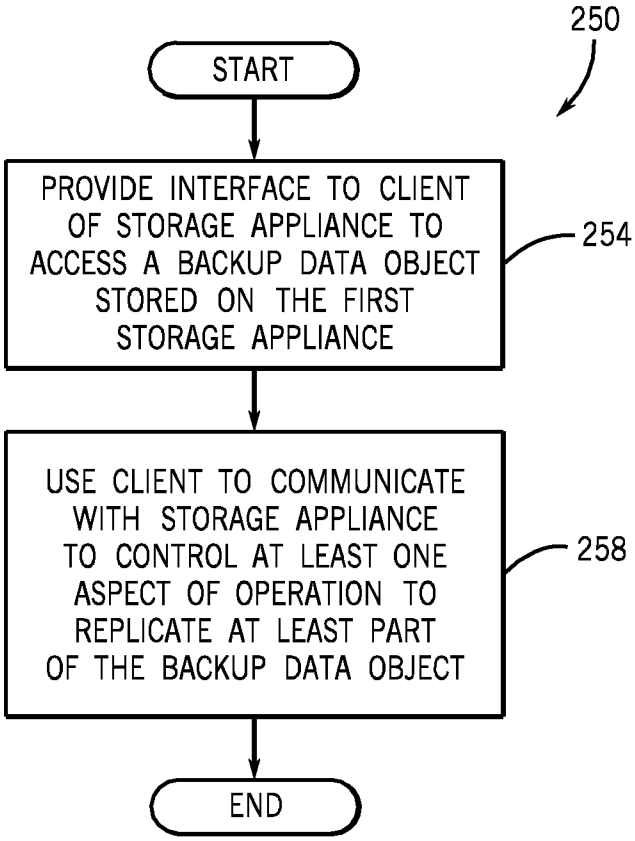

250
START
PROVIDE INTERFACE TO CLIENT OF STORAGE APPLIANCE TO ACCESS A BACKUP DATA OBJECT STORED ON THE FIRST STORAGE APPLIANCE
254
USE CLIENT TO COMMUNICATE WITH STORAGE APPLIANCE TO CONTROL AT LEAST ONE ASPECT OF OPERATION TO REPLICATE AT LEAST PART OF THE BACKUP DATA OBJECT
258
END

FIG. 5

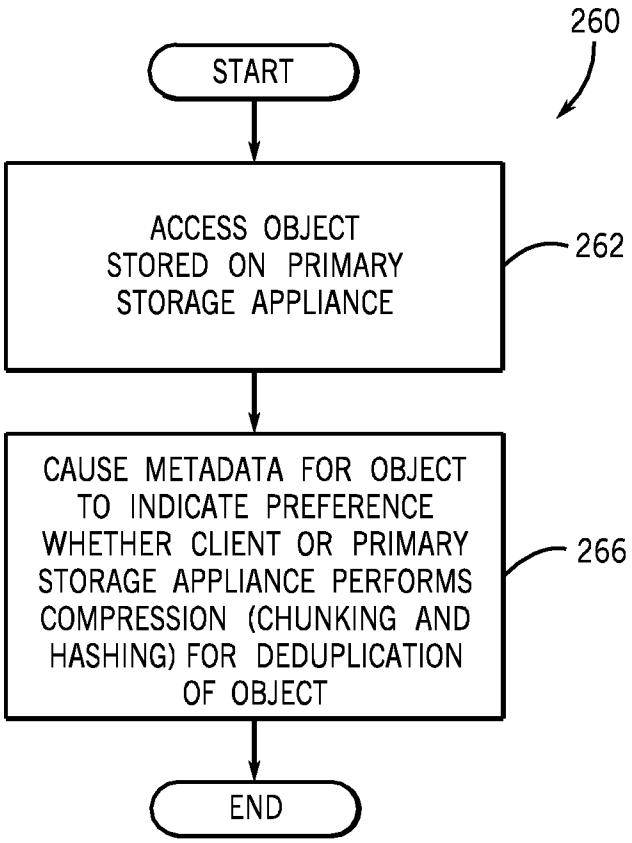

260
START
ACCESS OBJECT STORED ON PRIMARY STORAGE APPLIANCE
262
CAUSE METADATA FOR OBJECT TO INDICATE PREFERENCE WHETHER CLIENT OR PRIMARY STORAGE APPLIANCE PERFORMS COMPRESSION (CHUNKING AND HASHING) FOR DEDUPLICATION OF OBJECT
266
END

FIG. 6

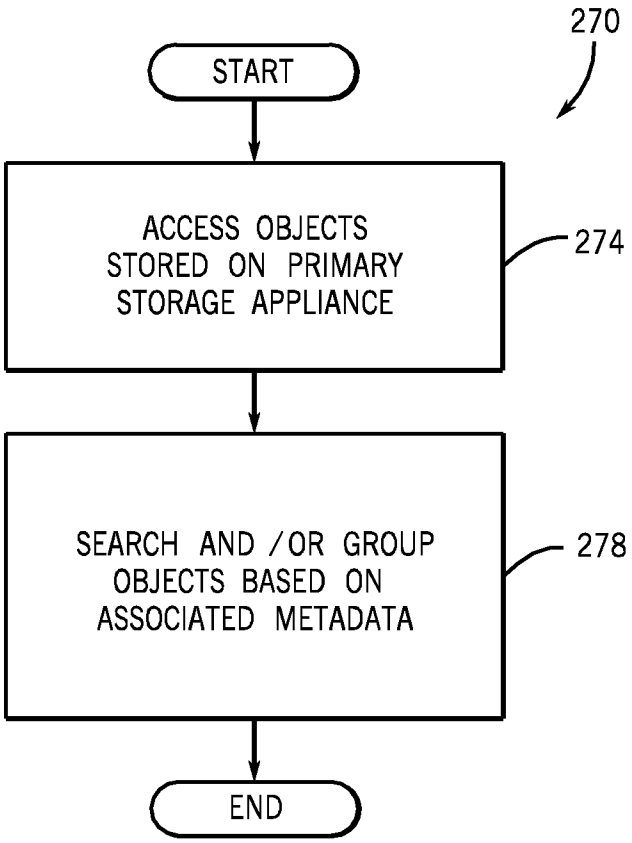

270
START
ACCESS OBJECTS STORED ON PRIMARY STORAGE APPLIANCE
274
SEARCH AND /OR GROUP OBJECTS BASED ON ASSOCIATED METADATA
278
END

FIG. 7

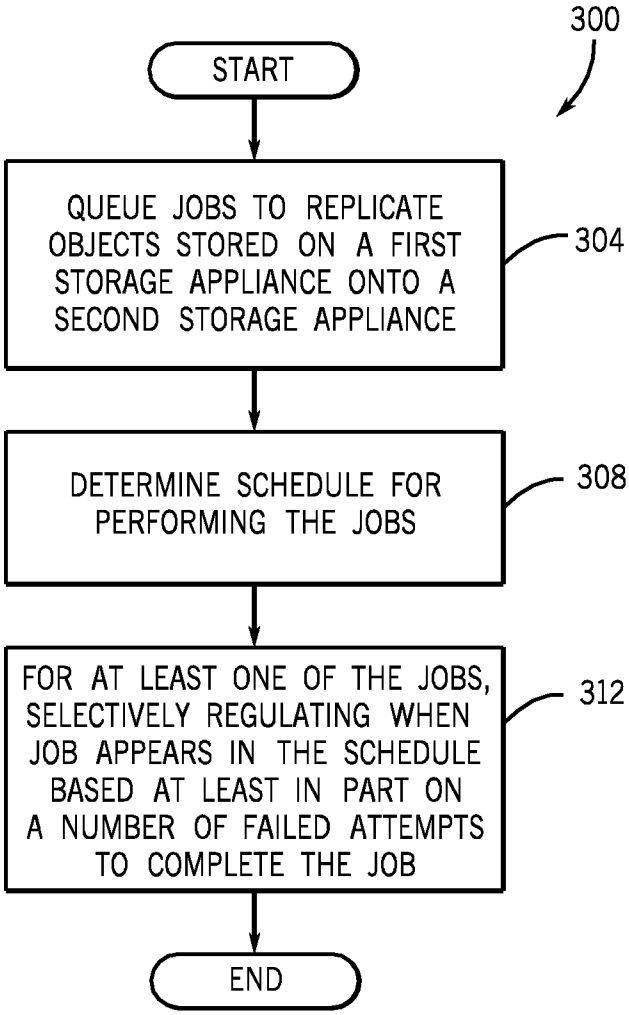

300
START
QUEUE JOBS TO REPLICATE OBJECTS STORED ON A FIRST STORAGE APPLIANCE ONTO A SECOND STORAGE APPLIANCE
304
DETERMINE SCHEDULE FOR PERFORMING THE JOBS
308
FOR AT LEAST ONE OF THE JOBS, SELECTIVELY REGULATING WHEN JOB APPEARS IN THE SCHEDULE BASED AT LEAST IN PART ON A NUMBER OF FAILED ATTEMPTS TO COMPLETE THE JOB
312
END

FIG. 8

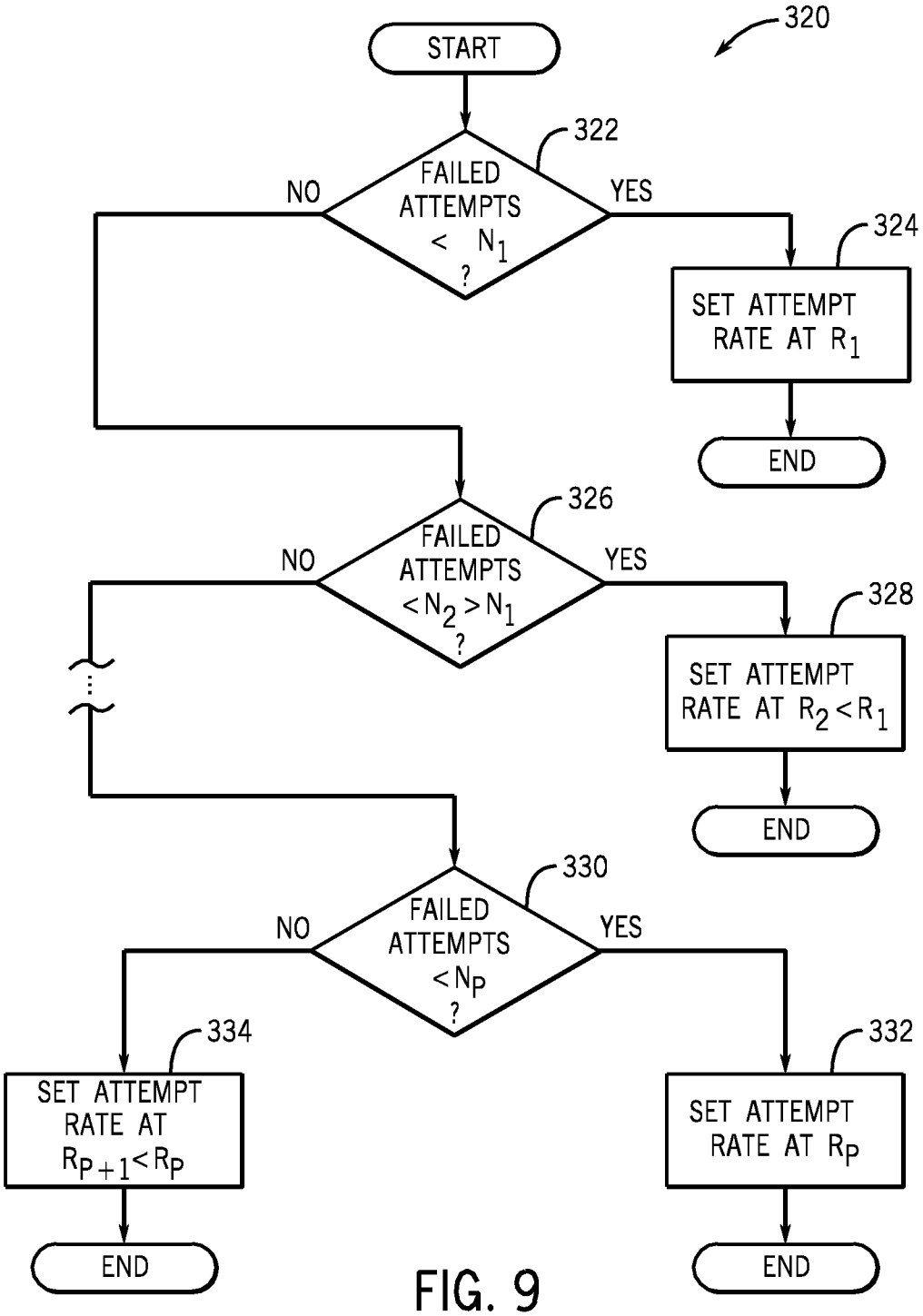

320
START
322
FAILED ATTEMPTS < N_1 ?
NO
YES
324
SET ATTEMPT RATE AT R_1
END
326
FAILED ATTEMPTS < N_2 > N_1 ?
NO
YES
328
SET ATTEMPT RATE AT $R_2 < R_1$
END
330
FAILED ATTEMPTS < N_P ?
NO
YES
334
SET ATTEMPT RATE AT $R_{P+1} < R_P$
END
332
SET ATTEMPT RATE AT R_P
END

FIG. 9

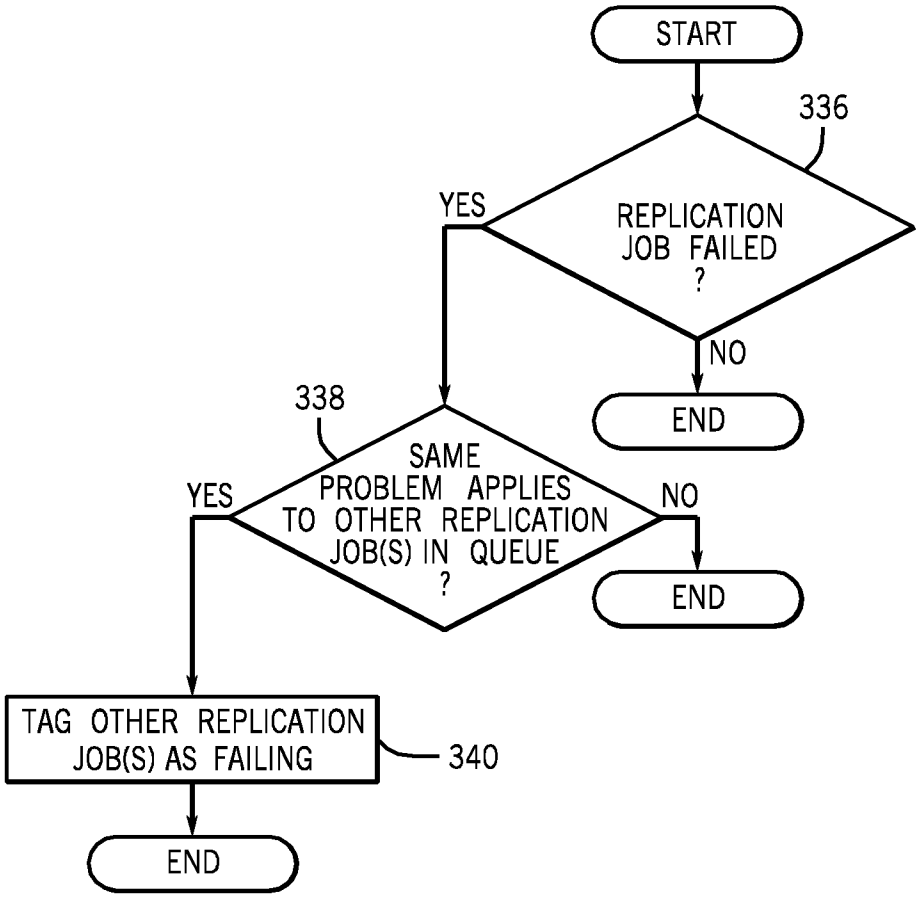

START
336
REPLICATION JOB FAILED ?
YES
NO
END
338
SAME PROBLEM APPLIES TO OTHER REPLICATION JOB(S) IN QUEUE ?
YES
NO
END
TAG OTHER REPLICATION JOB(S) AS FAILING
340
END

FIG. 10

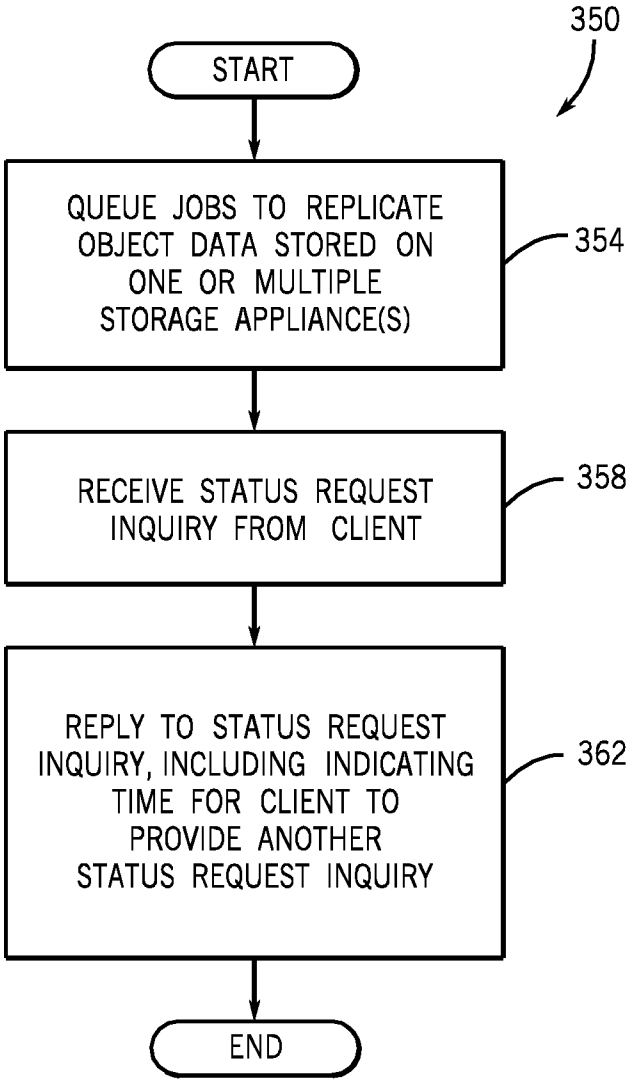

350
START
QUEUE JOBS TO REPLICATE OBJECT DATA STORED ON ONE OR MULTIPLE STORAGE APPLIANCE(S)
354
RECEIVE STATUS REQUEST INQUIRY FROM CLIENT
358
REPLY TO STATUS REQUEST INQUIRY, INCLUDING INDICATING TIME FOR CLIENT TO PROVIDE ANOTHER STATUS REQUEST INQUIRY
362
END

FIG. 11

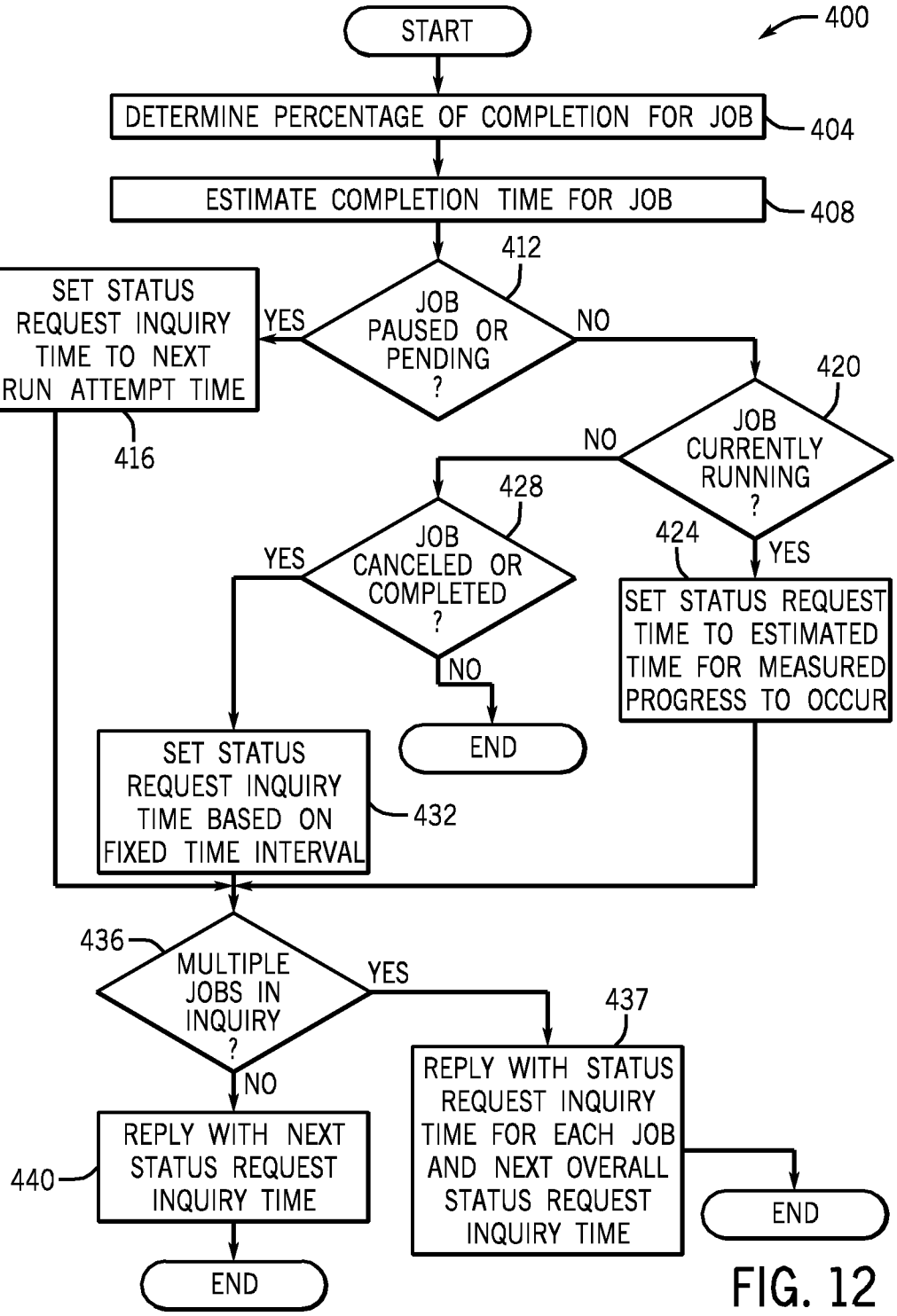

START
400
DETERMINE PERCENTAGE OF COMPLETION FOR JOB
404
ESTIMATE COMPLETION TIME FOR JOB
408
412
JOB PAUSED OR PENDING ?
YES
NO
SET STATUS REQUEST INQUIRY TIME TO NEXT RUN ATTEMPT TIME
416
420
JOB CURRENTLY RUNNING ?
NO
YES
424
SET STATUS REQUEST TIME TO ESTIMATED TIME FOR MEASURED PROGRESS TO OCCUR
428
JOB CANCELED OR COMPLETED ?
YES
NO
END
SET STATUS REQUEST INQUIRY TIME BASED ON FIXED TIME INTERVAL
432
436
MULTIPLE JOBS IN INQUIRY ?
YES
NO
437
REPLY WITH STATUS REQUEST INQUIRY TIME FOR EACH JOB AND NEXT OVERALL STATUS REQUEST INQUIRY TIME
END
440
REPLY WITH NEXT STATUS REQUEST INQUIRY TIME
END

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2012/034794

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/06(2006.01)i, G06F 12/16(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 9/06; G06F 15/16; G06F 17/00; G06F 17/30; G06F 12/00; G06F 11/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: fail over interval number of, queing job, replicate object data, schedule for performing the job;

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	US 6738923 B1 (NORBERT M. BLAM et al.) 18 May 2004 See abstract, column 6, lines 15-50, claim 1, and figure 5.	1,12 2-11,13-15
X A	US 7734591 B1 (MERCIER CHRISTINA WOODY et al.) 08 June 2010 See abstract, and column 6, lines 19-30.	7 1-6,8-15
A	US 2007-0061385 A1 (RICHARD J. CLARK et al.) 15 March 2007 See abstract, and paragraph [0126].	1-15
A	US 7765187 B2 (BERGANT MILENA et al.) 27 July 2010 See abstract, and column 11, lines 37- 50.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 FEBRUARY 2013 (19.02.2013)	**21 FEBRUARY 2013 (21.02.2013)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

BOK, Jin Yo

Telephone No. 82-42-481-5113

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2012/034794

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6738923 B1	18.05.2004	US 2004-0122935 A1	24.06.2004
		US 7401247 B2	15.07.2008
US 7734591 B1	08.06.2010	None	
US 2007-0061385 A1	15.03.2007	CA 2524794 A1	25.11.2004
		CA 2524794 C	30.03.2010
		EP 1620778 A2	01.02.2006
		US 2007-0198789 A1	23.08.2007
		US 7558927 B2	07.07.2009
		US 7657509 B2	02.02.2010
		WO 2004-099993 A1	18.11.2004
		WO 2004-102325 A2	25.11.2004
		WO 2004-102325 A3	25.11.2004
US 7765187 B2	27.07.2010	US 2007-0136389 A1	14.06.2007