A group stamping style asynchronous replication service (100) utilizes a loosely-accurate global clock (320) in a data backup and recovery storage environment. The storage environment includes a set of source storage systems illustratively embodied as source replication nodes (300) that cooperate to maintain a consistency group that may span multiple geographical sites. The storage environment also includes one or more target storage systems illustratively embodied as target replication nodes (400) configured to service the consistency group. The group stamping style service utilizes the loosely-accurate global clock (320) to provide consistent replication of a storage space, e.g., a target storage space, of the consistency group.
GROUP STAMPING STYLE ASYNCHRONOUS REPLICATION UTILIZING A LOOSELY-ACCURATE GLOBAL CLOCK

FIELD OF THE INVENTION

The present invention relates to replication services and, more specifically, to a group stamping style asynchronous replication service utilizing a loosely-accurate global clock.

BACKGROUND OF THE INVENTION

Replication is typically employed as part of a data backup and recovery storage strategy and, as such, denotes the movement of data from a source storage space of a source domain to a target storage space of a target domain via a communications network (e.g., a computer network) in such a way that enables recovery of applications from the target storage space. As used herein, recovery denotes loading of the applications on possibly different hosts (e.g., computers) where they can access the target storage space, instead of the source storage space, resulting in the applications loaded to a valid state. Also, storage space denotes any storage medium having addresses that enable data to be accessed in a stable way and, as such, may apply to file system access, block access and any other storage access means.

The source domain contains at least the source storage space, but may also contain the hosts, a switching fabric and any source replication components situated outside of those components. In this context, a component may either be a physical entity (e.g., a special replication appliance) and/or software entity (e.g., a device driver). In remote disaster recovery, for example, the source domain includes an entire geographical site, but may likewise span multiple geographical sites. The target domain includes all of the remaining components relevant for replication services, including the target storage space. In addition, a replication facility includes components that may be located in both the source and target domains.

The replication facility typically has at least one component, i.e., a write interception component, which intercepts storage requests (e.g., write operations or
"writes") issued by a host to the source storage space, prior to sending the intercepted writes to the target storage space. The write interception component is typically embedded within a computing unit configured as a source replication node. When issuing a write, an application executing on the host specifies an address on the storage space, as well as the contents (i.e., write data) with which the storage space address is to be set. The write interception component may be implemented in various locations in the source domain depending on the actual replication service; such implementations may include, e.g., a device driver in the host, logic in the switching fabric, and a component within the source domain, e.g., a source storage system. The write interception component is typically located "in-band", e.g., between the host and the source storage system, although there are environments in which the component may be located "out-of-band", where a separate physical component, such as an appliance server, in the source domain receives duplicate writes by utilizing, e.g., an in-band splitter.

Synchronous replication is a replication service wherein a write is not acknowledged until the write data associated with the write is processed by the source storage space, propagated to the target domain and persistently stored on the target storage space. An advantage of synchronous replication is the currency of the target domain data; that is, at any point in time, the writes stored on the target domain are identical to the source domain. However a disadvantage of this replication service is the latency or propagation delay associated with communicating the writes to the target domain, which limits the synchronous replication service in terms of distance, performance and scalability.

An asynchronous replication service reduces such latency by requiring that the write only be processed by the source storage space without having to wait for persistent storage of the write on the target storage space. In other words, the write is acknowledged once its associated write data is processed by the source storage space; afterwards, the write (and write data) are propagated to the target domain. Thus, this replication service is not limited by distance, performance or scalability and, therefore, is often preferred over synchronous replication services. A disadvantage of the asynchronous replication service, though, is the possibility of incurring data loss should the
source storage space fail before the write data has been propagated and stored on the target storage space.

Prior asynchronous replication services may be classified into a plurality of techniques or styles, one of which is group stamping. According to this replication style, the write interception component intercepts all writes (e.g., synchronously before an acknowledgement is returned to the application) and buffers the intercepted writes. Instead of attempting to establish a relative order among all the writes, the group stamping style service establishes an interval, e.g., either in time or by trigger, and all writes that are intercepted by the write interception component within the interval are recorded to a current group of writes. Notably, the current group is defined by buffering writes during the established interval and associating metadata with the entire group without the need to associate the metadata with each write. The metadata may be an actual timestamp or, more likely, a timeless ordering mechanism (e.g., a sequence number).

Thereafter, according to a predetermined policy or other conditions, the write interception component declares the current group completed and records all subsequent writes to a newly established group. The current group of writes is propagated to the target domain and persistently buffered therein prior to being applied to the target storage space. The group stamping style is typically employed by asynchronous replication services because of its lack of concern with the actual order of writes within an interval; group stamping is generally only concerned with the fact that the writes belong to a same interval.

The replication services may be further adapted to planned recovery or unplanned recovery. Planned recovery is defined herein as an act of recovery where components, e.g., hardware and software, of the source domain are fully operational, whereas unplanned recovery is defined as recovery that takes place when the source components are fully and/or partially non-operational. As used herein, the source domain describes all of the components whose failure/unavailability should not impair the ability to do unplanned recovery.

For unplanned recovery services utilizing the group stamping style, an entire group of writes is propagated to the target domain for storage on the target storage
space in a manner that ensures consistency in light of an intervening disaster. For example, the writes are propagated to an intermediate staging area on the target domain to ensure that the target storage space can be "rolled back" to a consistent state if a disaster occurs. The replication services may utilize various intermediate staging areas (such as a persistent log or non-volatile memory) to buffer the writes in a safe and reliable manner on the target domain. In some cases, the intermediate staging area is the target storage space itself and consistent snapshots of, e.g., target volumes of the storage space are generated. In the event of a disaster, a snapshot of the target volume(s) is used rather than the "current" content of the target volume(s).

Assume a group stamping replication service utilizes one write interception component. A first interval is started and a first group of writes is intercepted and logged by the write interception component until the first interval completes. A second interval is then started and a second group of writes is intercepted and logged by the interception component. Meanwhile, the component propagates the first group of writes to a target storage system of a target domain. Where there are two or more writes directed to the same block (address) within the same interval, the write interception component may remove the duplication and send only the most up-to-date write to the target domain (in accordance with a data reduction replication technique). However, if a replication service is implemented that does not reduce such duplication, the write interception component propagates the writes to the target domain in the respective order using, for example, an in-order log or journal on the source domain.

A disadvantage of group stamping is that the achievable Recovery Point Objective (RPO) in the case of disaster may never approach zero because of the delay incurred by the writes at the interception component as a result of the interval. As used herein, RPO is defined as the difference (in time) between the time of a disaster and the time at which the source storage space contained a crash image established at the target storage space. For example, assume the smallest interval of a group stamping style replication service is 10 minutes. If a disaster occurs, the target domain is, on average, 5 minutes behind because the disaster does not necessarily occur exactly before the interval completes. Note that it may be impractical to develop a group stamping replication solution with very small intervals.
Often, a source domain configuration having multiple hosts and/or multiple source storage systems may include only one source replication node (i.e., one write interception component) configured to intercept all writes associated with a consistency group. As used herein, a consistency group comprises storage space that requires consistent replication at a target domain. Such a configuration introduces a scalability issue because there is a limit to the processing bandwidth that the interception component can sustain, thereby resulting in potentially substantial adverse impact on performance of the entire configuration. Thus, this configuration may obviate use of a single write interception component.

For example, assume that a large data center is configured with many source storage systems configured to serve many hosts, wherein the source storage systems cooperate to maintain a consistency group. If all write traffic is directed to the single write interception component, a substantial scalability issue arises because the interception component will not practically be able to sustain the entire traffic. Now assume that a consistency group is configured to span multiple geographical site locations such as, e.g., among several small data centers geographically dispersed throughout a country or a plurality of countries. Here, the main reason for not using a single write interception component is not necessarily the scalability issue as much as the substantial latency introduced by such a configuration. This may necessitate either use of smaller consistency groups, which facilitates reliable and consistent group recovery on the target domain, or acceptance of large latencies and performance impact, which is undesirable. Therefore, such configurations may dictate the use of multiple write interception components.

A prior solution provides consistent replication services using group stamping across multiple write interception components through coordination among all write interception components. Here, a coordinator is provided that sends a predetermined message (e.g., a freeze message) to all write interception components when it is desired to complete a previous interval N. Note that the components accumulate writes in a journal, and process (and acknowledge) those writes beginning at the start of the previous interval N. Upon receiving the freeze message, a write interception component "quiesces" all new write activity by, e.g., buffering any new incoming writes without
processing or acknowledging those writes. The coordinator then waits until all write interception components respond with freeze acknowledgements. Once the freeze acknowledgments are received from all the write interception components, the coordinator sends a thaw message to each component to thereby start a new interval N+1. In response, the new, buffered incoming writes are processed by the write interception components as part of the new interval.

The writes of interval N are then propagated from each write interception component to the target domain. Depending on the actual implementation, the writes of interval N may be differentiated among the components such as, e.g., interval N₁ from write interception component 1, interval N₂ from write interception component 2, etc. Only after all of the writes of interval N are propagated from all of the write interception components to the target domain is the target domain allowed to start applying them to the target storage space. In order to perform consistent group stamping, the write interception components are typically architected in "shared-nothing" relationships (i.e., between write interception components and storage) to obviate crossing of writes received at different write interception components.

A disadvantage of group stamping across multiple write interception components is that the quiescent penalty is substantial in terms of performance. In particular, the freeze-thaw protocol exchange between a coordinator and a plurality of write interception components is not scalable; i.e., the weakest/slowest point in the coordinator-component interchange sequence dominates. This is because the coordinator has to wait to receive every acknowledgement from every write interception component before the previous interval can conclude and a new interval can start, thereby imposing a potential global penalty for all writes issued by the hosts to the source storage systems.

This disadvantage may require placing of restrictions on the locations of the write interception components. For example, the group stamping style approach may be reasonable if the write interception components are placed inside of the source storage systems because (i) there are fewer of these systems then hosts, (ii) the source storage systems are typically not located far from each other and (iii) such an arrangement enables more control over the behavior of the components.
Yet another problem that limits scalability of group stamping across multiple write interception components is when a write interception component does not respond to the freeze-thaw protocol. In such a situation, the coordinator is stalled and cannot progress until all acknowledgments are received from all components. In addition, implementation of recovery procedures associated with such a situation (such as timeouts, etc) may be complex.

SUMMARY OF THE INVENTION

The present invention overcomes the disadvantages of the prior art by providing a group stamping style asynchronous replication service that utilizes a loosely-accurate global clock in a data backup and recovery storage environment. The storage environment includes a set of source storage systems illustratively embodied as source replication nodes that cooperate to maintain a consistency group that may span multiple geographical sites. The storage environment also includes one or more target storage systems illustratively embodied as target replication nodes configured to service the consistency group. The group stamping style service utilizes the loosely-accurate global clock to provide consistent replication on a storage space, e.g., a target storage space, of the consistency group.

In an illustrative embodiment, each source replication node comprises a write interception component adapted to intercept storage requests (e.g., write operations or "writes") and replicate those writes to the target replication nodes for storage on the consistency group. Each target replication node comprises a log, a consistency engine and an apply engine. The write interception component of each source replication node establishes an interval whereby all intercepted writes within the interval are recorded to a current group of writes. The current group is defined by buffering writes during the established interval and associating metadata with the entire group, wherein the metadata illustratively includes a Time Identifier (ID). The write interception component buffers the writes, their associated write data and the metadata of the group as an entry of a buffer and then propagates its buffered entry to the target replication node, which stores the entry as a buffered log entry in the log.
In accordance with the invention, the consistency engine is configured to establish, maintain and advance a consistency timeline. Only log entries that have an associated Time ID earlier than a current consistency timeline may be applied to the target storage space by the apply engine. The apply engine is thus configured to apply all log entries up to a given consistency timeline to thereby create a crash image on the target storage space, wherein the crash image is a storage image of the target storage space that is available at restart from an interruption, such as a power failure.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings in which like reference numerals indicate identical or functionally similar elements:

Fig. 1 is a schematic block diagram of an environment including a plurality of storage systems that may be advantageously used with the present invention;

Fig. 2 is a schematic block diagram of a storage system that may be advantageously used with the present invention;

Fig. 3 is a schematic block diagram of a source replication node that may be advantageously used with the present invention;

Fig. 4 is a schematic block diagram of a target replication node that may be advantageously used with the present invention;

Fig. 5 is a schematic block diagram of a Time ID structure that may be advantageously used with the present invention;

Fig. 6 is a chart depicting an illustrative embodiment of a consistency timeline that may be advantageously used with the present invention; and

Fig. 7 is a chart depicting a set of rules used to determine the validity of consistency timeline values in accordance with the present invention.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

Fig. 1 is a schematic block diagram of an environment 100, such as a data backup and recovery storage environment, which includes a plurality of storage systems 200, such as source storage systems 200s and target storage systems 200χ that
may be advantageously used with the present invention. Each storage system is a computer that provides storage service relating to the organization of information on storage devices, such as disks 130. The disks 130 of each storage system 200s, t cooperate to define a storage space 160s, t on each system. The source storage systems are connected to one or more hosts 110 over a switching fabric 140, which may comprise a conventional Ethernet or Fibre Channel (FC) switch.

In an illustrative embodiment described herein, the hosts 110, source storage systems 200s and source storage spaces 160s are organized as components of a source domain 170, while the target storage systems 200t and their target storage spaces 160t are organized as components of a target domain 180. The source domain components are coupled to the target domain components over a computer network 150, which may comprise a plurality of point-to-point connections or communication links of, e.g., a wide area network embodied as an Ethernet network or a FC network. The components of each domain may communicate over network 150 by exchanging discrete frames or packets of data according to pre-defined protocols, such as the Transmission Control Protocol/Internet Protocol (TCP/IP).

Each host 110 may be a general-purpose computer configured to execute applications 112. Moreover, each host 110 may interact with each storage system 200 in accordance with a client/server model of information delivery. That is, the client (host) may request the services of the server (storage system), and the storage system may return the results of the services requested by the host, by exchanging packets over, e.g., the switching fabric 140. The hosts may issue packets including file-based access protocols, such as the Common Internet File System (CIFS) protocol or Network File System (NFS) protocol, over TCP/IP when accessing information in the form of files and directories. Alternatively, the client may issue packets including block-based access protocols, such as the Small Computer Systems Interface (SCSI) protocol encapsulated over TCP (iSCSI) and SCSI encapsulated over FC (FCP), when accessing information in the form of blocks.

Fig. 2 is a schematic block diagram of a storage system 200 that may be advantageously used with one or more embodiments described herein as, e.g., a source storage system 200s or target storage system 200t. Each storage system 200 comprises a
processor 222, a memory 224, a network adapter 226 and a storage adapter 228 interconnected by a system bus 225. The network adapter 226 comprises the mechanical, electrical and signaling circuitry needed to connect the storage system 200 to the switching fabric 140 or computer network 150. The storage system 200 also includes a storage operating system 230 that preferably implements a virtualization system 240 to logically organize the information as a hierarchical structure of data containers, such as directories, files and logical units (luns). The virtualization system may be abstracted through the use of a database management system, a volume manager or, as described herein, a file system.

In an illustrative embodiment, the memory 224 comprises storage locations that are addressable by the processor and adapters for storing software programs and data structures associated with the present invention. The processor and adapters may, in turn, comprise processing elements and/or logic adapted to execute the software programs and manipulate the data structures. Storage operating system 230, portions of which is typically resident in memory and executed by the processing elements, functionally organizes the system 200 by, inter alia, invoking storage operations in support of processes, modules and/or services executing on the storage system. These processes, modules and/or services may include replication services such as, e.g., group stamping style asynchronous replication service 250. It will be apparent to those skilled in the art that other processing and memory means, including various computer readable media, may be used to store and execute program instructions pertaining to the inventive technique described herein.

The storage adapter 228 cooperates with the storage operating system 230 executing on the system 200 to access information requested by a host (or user). The information may be stored on any type of attached array of writable storage device media such as video tape, optical, DVD, magnetic tape, bubble memory, electronic random access memory, micro-electro mechanical and any other similar media adapted to store information, including data and parity information. However, as illustratively described herein, the information is preferably stored on the disks 130. The storage adapter includes input/output (I/O) interface circuitry that couples to the disks over an
I/O interconnect arrangement, such as a conventional high-performance, FC serial link topology.

Storage of information on each storage system 200s is preferably implemented as a collection of physical storage disks 130 cooperating to define an overall logical arrangement of storage space 160s on the system. To facilitate access to the disks 130, the storage operating system 230 implements a file system that cooperates with virtualization modules to provide virtualization system 240 that "virtualizes" the storage space 160 provided by disks 130. The file system logically organizes the information as a hierarchical structure of named directories and files on the disks. Each "on-disk" file may be implemented as a set of disk blocks configured to store information, such as data, whereas the directory may be implemented as a specially formatted file in which names and links to other files and directories are stored. The virtualization modules allow the file system to further logically organize information as a hierarchical structure of blocks on the disks that are exported as named luns.

In an illustrative embodiment, application 112 executes in host 110 to issue storage requests to access data containers exported by source storage systems 200s of the source domain 170. The virtualization system 240 of each source storage system transposes the storage requests into file system requests for local storage on the source storage space 160s, to thereby serve the storage requests issued by the host 110. The storage requests are illustratively write operations ("writes"). A write interception entity or component intercepts the writes and replicates those writes to one or more target storage systems 200 of the target domain 180, where the intercepted writes are stored for, e.g., data backup and/or disaster recovery purposes. The writes are thereafter consistently stored on the target storage space 160 such that, if a disaster arises, the storage on that storage space can be recovered in a manner that maintains the order of writes issued to the source storage systems by the host.

The write interception component may reside (i) in the host 110 (e.g., embodied as a device driver), (ii) within the switching fabric 140 (e.g., embodied as a software module in the network switch) or, as described herein, (iii) in the source storage system 200s, such that the component may intercept the writes in block-level or file-level semantics. A source replication node is the computing unit in which the write intercep-
tion component is embedded. For example, if the write interception component is implemented as a device driver inside host 110, the host is the source replication node, whereas if the component is located out-of-band, i.e., within an appliance server, the appliance server is the source replication node.

An advanced clock synchronization technique is adapted for use with replication service 250 in the data backup and recovery storage environment 100. The advanced clock synchronization technique establishes a software-only, loosely-accurate global clock based on a timestamp and an associated error boundary. The established clock is a loosely-accurate global clock because it is difficult to establish an extremely accurate, fined-grain global clock using general-purpose host systems and commercially-available storage systems, particularly if these systems are geographically dispersed. Notably, the timestamp and its error boundary are used as global clock information to enable synchronization (i.e., ordering of storage requests) among a plurality of source replication nodes, thereby ensuring consistent replication of the storage requests on the target storage systems. An example of a loosely-accurate global clock that may be advantageously used with the present invention is disclosed in commonly-assigned U.S. Patent Application Serial No. 112056-0405 titled, Advanced Clock Synchronization Technique, which application is hereby incorporated by reference.

Fig. 3 is a schematic block diagram of a source replication node 300 that may be advantageously used with the present invention. As noted, a source replication node (SRN) is the computing unit in which a write interception component is embedded. In an illustrative embodiment described herein, the write interception component 310 is located in a source storage system 200s and is embodied as a software module disposed over the virtualization system 240; as such, each source storage system 200s may illustratively be an SRN 300. Each SRN also contains a hardware clock (HClock) component 320 and a clock node (CNode) component 330.

Specifically, the HClock 320 is a conventional hardware clock circuit adapted to generate a clock or "timestamp" having a known drift as specified by a manufacturer, e.g., the timestamp drifts a predetermined number of seconds in an hour or in a day, maximum. Thus, there is an error boundary defined by the maximum drift, wherein the term "drift" denotes the time that the timestamp drifts from an actual, real time. The
CNode 330 is coupled in cooperating relation to (associated with) the HClock 320 and the write interception component 310. The CNode 330 is illustratively a software module configured to interact collectively with other CNodes 330 of other SRNs 300 to maintain a global clock and to provide an interface to the write interception component 310 to retrieve the current time.

In an illustrative embodiment, a set of CNodes that maintains a global clock comprises a clock group and the CNodes of a set of SRNs that maintains a consistency group belong to the same clock group. One CNode from among the clock group assumes a role of a master (hereinafter "master SRN") and the other CNodes assume the roles of slaves (hereinafter "slave SRNs"). Selection of the master and slave SRNs may be performed using conventional techniques known to those skilled in the art. In addition, determination of when a master SRN should be replaced may be performed using well-known techniques. Although the master SRN (CNode) is illustratively described as contained within an SRN, it will be understood to those of skill in the art that the master CNode could also be contained within a separate computing unit exclusive of an SRN having write interception functionality.

The present invention is directed to a group stamping style asynchronous replication service that utilizes a loosely-accurate global clock in a data backup and recovery storage environment. The storage environment includes a set of source storage systems illustratively embodied as SRNs that cooperate to maintain a consistency group that may span multiple geographical sites. The storage environment also includes one or more target storage systems illustratively embodied as target replication nodes configured to service the consistency group. A target replication node is a computing unit that resides in a target domain and that receives all writes for the consistency group issued from one or more SRNs. The group stamping style service utilizes the loosely-accurate global clock to provide consistent replication on a storage space, e.g., a target storage space, of the consistency group.

Although an illustrative embodiment described herein is directed to a single consistency group, it will be understood to those skilled in the art that the invention may apply equally to multiple consistency groups. In addition, although the illustrative embodiment description is directed to a single computing unit, e.g., a target replication
node, residing in the target domain 180 that receives all writes for the consistency
group maintained by one or more SRNs, the invention applies equally to multiple target replication nodes. Fig. 4 is a schematic block diagram of a target replication node (TRN) 400 that may be advantageously used with the present invention. In an illustrative embodiment described herein, each target storage system $200 \chi$ may illustratively be a TRN 400. As described herein, each TRN 400 comprises a log, e.g., TRN log 410 having a plurality of log entries 412, a consistency engine 430 and an apply engine 440.

The write interception component 310 of each SRN 300 establishes an interval whereby all intercepted writes within the interval are recorded to a current group of writes, i.e., Group N. The current Group N is defined by buffering writes during the established interval and associating a timestamp with the entire group. To that end, the write interception component 310 queries (requests) its associated CNode 330 for an indication of the current time so that it can time stamp the current Group N. In response, the CNode 330 generates (prepares) and returns a Time Identifier (ID) data structure configured to describe the current time. The write interception component 310 generates metadata for the recorded group of writes, wherein the metadata illustratively includes the contents of the Time ID structure, as well as addresses of the writes within the group, etc. The write interception component buffers the writes, their associated write data and the metadata of the Group N as an entry 312 of a temporary storage location organized as, e.g., a buffer 314. Note that various replication services, such as replication service 250, may implement the buffer in various ways, e.g., using volatile or non-volatile storage.

Fig. 5 is a schematic block diagram of a Time ID structure 500 that may be advantageously used with the present invention. The Time ID structure 500 illustratively contains a timestamp field 510, an error field 520 (the maximum error of the timestamp), a local sequence number (LSN) field 530 and a major sequence number (MSN) field 540. The timestamp field 510 contains a timestamp, preferably chronological, with sufficient resolution (e.g., microsecond). The error field 520 contains a dynamically calculated value specifying a maximum error boundary of the timestamp. For example, assume the timestamp is $t = 2007$, Jan 1 20:00:00:0000 and the error is $1000$ (microseconds). An illustrative interpretation of the maximum error boundary is that
the time may be lower or higher than the "correct" time in 500 microseconds, i.e.,
"real" time = [t-500..t+500]. It will be understood to those skilled in the art that there
may be other interpretations of the maximum error boundary such as, e.g., as "real" 
time = [t..t+1000], and those interpretations are covered by the present invention. The
LSN field 530 contains a counter value that is incremented every time the CNode 330
is queried to return the current time and the MSN field 540 contains a counter value
that increases in various low-frequency control events, such as master SRN replace-
ment, master SRN restart, etc.

As noted, a prior solution provides consistent replication services using group
stamping across multiple write interception components through the use of a conven-
tional freeze-thaw protocol to enable coordination among all of the write interception
components. In accordance with the invention, the problematic freeze-thaw protocol
may be replaced with a protocol based on the loosely-accurate global clock. In particu-
lar, all SRNs 300 agree (e.g., ahead of time) on specific times (e.g., every 10 minutes,
at "13:00:00", "13:10:00", "13:20:00") at which to switch from Group N to a next
group, i.e., Group N+1. The SRNs maintain the loosely-accurate global clock and
agree on the maximum error of the timestamp for that global clock that they consider as
"acceptable", e.g., ±100 milliseconds (msec). Each SRN then behaves (operates) inde-
pendently of the other SRNs, without the need for tight coordination that is inherent to
the freeze-thaw protocol. For example, at 100msec before 13:00:00 (according to its
"knowledge" of the current time) each SRN 300 starts a 100msec*2 (200 msec) "quiet"
period in which it suspends processing of all new writes. Those writes will be proc-
essed as part of the next Group N+1, once the quiet period completes. As long as the
global clock error of the SRNs is not greater than ±100 msec, this grants consistency.

If, during the quiet period, a particular SRN 300 detects a global clock error
greater than the maximum agreed one (e.g., ±100 msec), then that SRN does not com-
plete Group N and refrains from switching to Group N+1 because there is a danger for
consistency violation. In this case, the particular SRN 300 does not even need to intro-
duce a quiet period; the SRN need only continue the Group N until the next agreed time
(e.g., 13:10:00). Notably, the remaining SRNs need not know about the clock error of
the particular SRN 300. Upon receiving all the group information, the TRN 400 detects
this situation and, in response, "merges" the writes of Group N and Group N+1 received from the remaining SRNs.

Illustratively, all of the SRNs 300 transmit their buffered group entries 312 (e.g., Group N entries) to the TRN 400 via the network, wherein each group is identified by its timestamp. The TRN temporarily stores (buffers) each entry as a buffered log entry 412 in the TRN log 410 prior to applying it to the target storage space 160τ. That is, the entries 312 transmitted from all SRNs 300 are not immediately "hardened" to the target storage space 160τ, but instead are buffered on the TRN 400. The actual data propagation and TRN buffering implementation used the replication service 250 is independent of the present invention. However, the illustrative buffering implementation should (i) enable the entries 412 buffered on the TRN to be "logically" organized as a set of data structures, such as queues 420, e.g., one queue per SRN 300, wherein each queue 420 contains the entries 412 sorted by their order of interception at the SRN, (ii) provide the TRN 400 with the ability to conclude that up to time t, there are no other entries not yet propagated by the SRN, wherein the time t is updated frequently and denoted as T-info, and (iii) provide persistent TRN buffering for disaster recovery by, e.g., embodying the TRN log 410 as a persistent storage medium (such as a disk or non-volatile memory).

In accordance with the invention, the consistency engine 430 and apply engine 440 are illustratively software processes disposed over the virtualization system 240 (i.e., on top of the block storage) of the target storage system 200τ, although it will be apparent to those of skill in the art that the processes could also execute in the switching fabric 140 (e.g., embodied as a software module in the network switch) or in the host 110 (e.g., embodied as a device driver). The consistency engine 430 is illustratively configured to establish and maintain a value called a consistency timeline. As described herein, the consistency timeline is established at each group interval. That is, the consistency engine decides which SRNs send groups of writes during the appropriate time interval, i.e., which groups are stamped with the same timestamp. Only TRN log entries 412 that have an associated Time ID 500 earlier than a current consistency timeline may be applied to the target storage space 160τ by the apply engine 440.
Once the interval is determined, i.e., the consistency timeline is established, the apply engine 440 is invoked to apply the writes of the groups to the target storage space. The apply engine 440 is thus configured to apply all log entries 412 up to an established current consistency timeline (and no more than that), to thereby create a crash image on the target storage space. As used herein, a crash image is defined as follows: given a specific timed-sequence of writes to a storage space, e.g., the source storage space, and an initial storage image, the crash image is the storage image realized if, at time $t$ of the storage space, an interruption (e.g., power failure) occurs. The crash image is thus a storage image of the target storage space that is available at restart from an interruption, before any further writes take place.

Fig. 6 is a chart depicting an illustrative embodiment of a consistency timeline 600 that may be advantageously used with the present invention. This figure shows entries received at the TRN 400 from write interception components OfSRN 1-3. As noted, the entries are buffered (logged) upon receipt at the TRN and illustratively organized as queues, wherein each queue contains the entries 412 (i.e., groups) sorted by their order of interception at the SRNs, from the perspective of the TRN. Note the order of the groups within each queue may be guaranteed by the SRN or, alternately, the SRN may send the groups in any order and the TRN may reorder them. The specific ordering implementation is not limiting to the invention. The rectangular boxes on the timelines denote the timestamps provided by the CNodes of the SRNs for the actual entries 412 (groups).

**Consistency Engine**

In an illustrative embodiment, the consistency engine 430 is configured to establish, maintain and advance the consistency timeline 600, e.g., from time to time. The consistency engine 430 establishes (calculates) the timeline using parameters such as T-info 620 and Minimum T-info 630. As used herein, T-info 620 is defined as a guarantee (e.g., from a point of view of the TRN) that all entries with times earlier than T-info have been propagated by the specific SRNs associated with the specific T-info, while Minimum T-info 630 is the T-info with the lowest (earliest) value. The consistency engine 430 calculates the consistency timeline 600 in a manner that does not cause any uncertainty driven by the error boundaries of the groups.
Specifically, the goal of the consistency engine 430 is to advance the consistency timeline 600 to the most, up-to-date point possible so that the apply engine 440 can apply the groups of writes to the persistent target storage space 160 up to the calculated consistency timeline. To that end, the consistency engine 430 frequently (e.g., every 10 minutes or every time a group shipment to the TRN completes, but alternately according to policy) calculates a new consistency timeline, if possible. Typically, the desired timeline is the latest calculated timeline possible. Illustratively, groups of writes that occur earlier than the current consistency timeline (i.e., entries 412 located to the left of that consistency timeline) are allowed to be handled by the apply engine 440. The remaining entries 412 accumulate at the TRN log 410.

As noted, there may be a situation where a particular SRN 300 is not able to correctly form a current group, e.g., Group N, because of clock error and, accordingly, merges Group N with the next group, Group N + 1. The consistency engine 430 detects that it has received such a merged group, e.g., merged group 640, from the SRN and, therefore, skips establishment of a consistency timeline for Group N. Note that the remaining SRNs likely will not send merged groups because they are not aware of the error with the particular SRN. The consistency engine 430 does not actually merge the groups of writes sent by all SRNs at the TRN 400, but rather just skips establishment of the consistency timeline for Group N. That is, the consistency engine skips establishment of a consistency timeline for writes sent from all SRNs for Group N.

Fig. 7 is a chart depicting a set of rules used to determine the validity of consistency timeline values in accordance with the present invention. A current consistency timeline 700 is the timeline farthest to the left of the chart. The consistency engine 430 performs its function by advancing the consistency timeline from that position. Five examples of possible new consistency timelines 1-5 are provided, along with explanations as to their validity (V) or invalidity (X), the latter signifying that such timelines are not established (drawn). Note that each possible consistency timeline is investigated independently of the others; in a practical situation involving the five possible consistency timelines, the consistency engine 430 advances the consistency timeline to the most up-to-date point possible which, in this practical case, would illustratively be timeline 4.
In accordance with the invention, the consistency timeline is established at the "natural" boundary of entries 412 (groups); that is, the consistency timeline should not cross (be drawn) through the middle of any group. As can be seen, consistency timeline 1 crosses the middle of one or more entries (groups). Therefore, timeline 1 is an invalid consistency timeline and, thus, cannot be drawn by the consistency engine 430.

Consistency timelines 2, 4 do not cross any groups (merged or otherwise) and are smaller than or equal to the Minimum T-info 630. Therefore, these timelines are valid consistency timelines and can be drawn by the consistency engine 430. In contrast, consistency timeline 3 crosses a merged group is therefore invalid. Lastly, consistency timeline 5 is invalid because it is higher than the Minimum T-info 630.

Apply Engine

The apply engine 440 is configured to apply all groups of writes (entries 412) below (i.e., to the left) or equal to the current consistency timeline 700 and then delete (erase) those entries 412 from the TRN log 410. That is, the apply engine 440 retrieves entries 412 from the queues 420 up to the current consistency timeline 700 and flushes (stores) them to the target storage space 160_T. Concurrently, the consistently engine 430 advances the timeline to, e.g., consistency timeline 2, so that the apply engine can retrieve additional entries (up to the newly advanced current consistency timeline) and store them on the target storage space.

Assume the apply engine 440 is unable to complete storage of all entries 412 up to consistency timeline 2 because, e.g., the target storage space is slow responding to storage requests. Yet, the consistency engine 430 has advanced the current consistency timeline to, e.g., timeline 4. According to the invention, the apply engine 440 may continue processing (storing) entries 412 up to the consistency timeline 4. In other words, the progress of the apply engine (and/or target storage) does not prevent the consistency engine from continuing to calculate new consistency timelines. The apply engine 440 may apply the entries (write data of the groups) in virtually any order. That is, the apply engine does not necessarily have to apply the write data of the groups in the order that is illustrated in Fig. 6. The apply engine 440 may change the order of the groups of writes (or the order of writes within each group, if any such order exists in the specific
implementation) to optimize writing to the target storage space $160\chi$ as long as it does not reverse the order of two writes that address the same block.

Specifically, the apply engine 440 may apply the above writes to the target storage space $160\tau$ in any order it wishes, with the following exception: if two writes, $W_1$ and $W_2$, are associated with overlapped storage space addresses, and are both below the current consistency timeline 700, then those writes must be applied in the "correct" time order. The motivation for this exception is that, if the two writes are directed to the same address and if their true order is altered/switched, then the result will be an old ("stale") content, rather than a desired new ("valid") content, leading to a corrupted image that is no longer a crash image. The above exception is simple to maintain and realize, as long as the SRNs 300 have shared-nothing relations to the storage space (meaning that two different SRNs cannot write to the same storage space address).

While there have been shown and described illustrative embodiments for providing a group stamping style asynchronous replication service that utilizes a loosely-accurate global clock in a data backup and recovery storage environment, it is to be understood that various other adaptations and modifications may be made within the spirit and scope of the present invention. For example, assume that a particular SRN has a clock error greater than the maximum-agreed-upon error such that it merges the Group N interval with the Group N+1 interval. Assume also that the clock error persists at the SRN through interval N+1 and onto interval N+2. Note that if the system is configured correctly, then exceeding the maximum agreed error should be a temporary problem. However, if the temporary error persists, then in response to merging a number of group intervals at the TRN, user intervention is required. Other types of error scenarios are handled similarly; notably, a problem in one SRN does not affect any other SRN.

The foregoing description has been directed to specific embodiments of this invention. It will be apparent, however, that other variations and modifications may be made to the described embodiments, with the attainment of some or all of their advantages. For instance, it is expressly contemplated that the processes, modules and/or components described herein can be implemented as software, including a computer-readable medium having program instructions executing on a computer, hardware, firmware, or a combination thereof. Also, electromagnetic signals may be generated to
carry computer executable instructions that implement aspects of the present invention over, e.g., a wireless data link or a data network, such as the Internet. Accordingly this description is to be taken only by way of example and not to otherwise limit the scope of the invention. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

What is claimed is:
CLAIMS

L. A method for providing a group stamping style asynchronous replication service in a
data backup and recovery storage environment, the method comprising:

intercepting storage requests at a set of source replication nodes configured to
maintain a consistency group that spans multiple geographical sites;

establishing an interval at the source replication nodes whereby all intercepted
storage requests within the interval are recorded to a current group of storage requests;

maintaining a loosely-accurate global clock among the source replication nodes,
the loosely-accurate global clock based on a timestamp and a maximum error associ-
ated with the timestamp;

agreeing, among the source replication nodes, on specific times at which to
switch from the current group of storage requests to a next group of storage requests;

and

transmitting the current group of storage requests from the source replication
nodes to one or more target replication nodes configured to service the consistency
group to provide consistent replication of a target storage space of the consistency
group.

2. The method of Claim 1 further comprising:

identifying each current group of storage requests at each target replication node
by the timestamp of the loosely-accurate global clock.

3. The method of Claim 1 wherein the step of agreeing comprises agreeing, among the
source replication nodes, on an acceptable maximum error of the timestamps for the
loosely-accurate global clock.

4. The method of Claim 3 further comprising:

starting a quiet period at each source replication node using the acceptable
maximum error and one of the specific times; and

suspending processing of new storage requests during the quiet period.
5. The method of Claim 4 further comprising:
   processing the suspended storage requests as part of the next group of storage
   requests once the quiet period completes.

6. The method of Claim 5 further comprising:
   detecting, during the quiet period, a global clock error greater than the accept-
   able maximum error at a source replication node;
   refraining from switching to the next group of storage requests at the source
   replication node; and
   continuing to record all intercepted storage requests to the current group of stor-
   age requests at the source replication node.

7. The method of Claim 6 further comprising:
   merging the transmitted current group of storage requests with the next group of
   storage requests at the target replication node.

8. A system adapted to provide a group stamping style asynchronous replication ser-
   vice in a data backup and recovery storage environment, the system comprising:
   a set of source storage systems embodied as source replication nodes that coop-
   erate to maintain a consistency group that spans multiple geographical sites, the source
   replication nodes adapted to intercept storage requests and establish an interval
   whereby all intercepted storage requests within the interval are recorded to a current
   group of storage requests, the source replication nodes further adapted to agree on spe-
   cific times at which to switch from the current group of storage requests to a next group
   of storage requests;
   a loosely-accurate global clock maintained among the source replication nodes,
   the loosely-accurate global clock based on a timestamp and a maximum error associ-
   ated with the timestamp; and
   one or more target storage systems embodied as target replication nodes config-
   ured to service the consistency group, each target replication node adapted to receive
the current group of storage requests transmitted from the source replication nodes to provide consistent replication of a target storage space of the consistency group.

9. The system of Claim 8 wherein the source replication nodes are further adapted to stamp the current group of storage requests with the timestamp of the loosely-accurate global clock.

10. The system of Claim 9 wherein the target replication node is further adapted to identify each current group of storage requests by the timestamp of the loosely-accurate global clock.

11. The system of Claim 10 wherein the storage requests are write operations.

12. The system of Claim 8 wherein the source replication nodes are further adapted to agree on an acceptable maximum error of the timestamps for the loosely-accurate global clock.

13. The system of Claim 12 wherein the source replication nodes are further adapted to (i) start a quiet period using the acceptable maximum error and one of the specific times, and (ii) suspend processing of new storage requests during the quiet period.

14. The system of Claim 13 wherein the source replication nodes are further adapted to process the suspended storage requests as part of the next group of storage requests once the quiet period completes.

15. The system of Claim 14 wherein, in response to detecting a global clock error greater than the acceptable maximum error during the quiet period, a source replication node is further adapted to (i) refrain from switching to the next group of storage requests and (ii) continue to record all intercepted storage requests to the current group of storage requests.
16. The system of Claim 15 wherein each target replication node is further adapted to merge the transmitted current group of storage requests with the next group of storage requests.

17. The system of Claim 16 wherein each source replication node comprises a write interception component adapted to intercept the storage requests and stamp the current group of storage requests with the timestamp of the loosely-accurate global clock record, the source replication node configured to propagate the storage requests as entries to the target replication node for storage on the consistency group.

18. The system of Claim 17 wherein the target replication node comprises a log adapted to store each propagated entry as a log entry.

19. The system of Claim 18 wherein the target replication node further comprises a consistency engine configured to establish, maintain and advance a consistency timeline.

20. The system of Claim 19 wherein the target replication node further comprises an apply engine configured to apply log entries to the target storage space up to the consistency timeline to create a crash image on the target storage space.

21. A computer readable medium containing executable program instructions for providing a group stamping style asynchronous replication service in a data backup and recovery storage environment, the executable instructions comprising one or more program instructions for:

   - intercepting storage requests at a set of source replication nodes configured to maintain a consistency group that spans multiple geographical sites;
   - establishing an interval at the source replication nodes whereby all intercepted storage requests within the interval are recorded to a current group of storage requests;
maintaining a loosely-accurate global clock among the source replication nodes,
the loosely-accurate global clock based on a timestamp and a maximum error associ-
ated with the timestamp;
agreeing, among the source replication nodes, on specific times at which to
switch from the current group of storage requests to a next group of storage requests;
and
transmitting the current group of storage requests from the source replication
nodes to one or more target replication nodes configured to service the consistency
group to provide consistent replication of a target storage space of the consistency
group.
FIG. 1

SWITCHING FABRIC 140

HOST 110

APPLICATION 112

SOURCE STORAGE SYSTEM 200S

TARGET STORAGE SYSTEM 100T

SOURCE STORAGE SPACE 160S

TARGET STORAGE SPACE 160T

COMPUTER NETWORK 150

SOURCE DOMAIN 170

TARGET DOMAIN 180
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME STAMP</td>
<td>510</td>
</tr>
<tr>
<td>ERROR</td>
<td>520</td>
</tr>
<tr>
<td>LSN</td>
<td>530</td>
</tr>
<tr>
<td>MSN</td>
<td>540</td>
</tr>
</tbody>
</table>

FIG. 5
## INTERNATIONAL SEARCH REPORT

### A. CLASSIFICATION OF SUBJECT MATTER

<table>
<thead>
<tr>
<th>INV.</th>
<th>G06F11/20</th>
<th>G06F11/14</th>
<th>G06F17/30</th>
</tr>
</thead>
</table>

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

### B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, INSPEC, COMPENDEX

### C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>US 5 682 513 A (Candelaria Susan Kay [US] ET AL) 28 October 1997 (1997-10-28) column 8, line 10 - column 10, line 9 figure 3</td>
<td>1-21</td>
</tr>
<tr>
<td>Y</td>
<td>US 6 144 999 A (Khalidi Yousef A [US] ET AL) 7 November 2000 (2000-11-07) column 10, line 64 - column 11, line 30; figures 3,8</td>
<td>1-21</td>
</tr>
</tbody>
</table>

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

Special categories of cited documents:

1° document defining the general state of the art which is not considered to be of particular relevance

E° earlier document but published on or after the international filing date

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

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

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

1° 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

A° document member of the same patent family

Date of the actual completion of the international search
13 August 2008

Date of mailing of the International search report
26/08/2008

Name and mailing address of the ISA/
European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 Hl Rijswijk
Tel: (+31-70) 340-2040, Tx: 31 651 epo nl.
Fax: (+31-70) 340-3016

Authorized officer
Bowler, Alyssa
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, wild indication where appropriate, of the relevant passages</th>
<th>Relevant to claim No</th>
</tr>
</thead>
</table>
### INTERNATIONAL SEARCH REPORT
Information on patent family members

<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 5682513 A</td>
<td>28-10-1997</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>US 2004260736 A1</td>
<td>23-12-2004</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GB 2341958 A</td>
<td>29-03-2000</td>
</tr>
<tr>
<td>US 6502205 B1</td>
<td>31-12-2002</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>US 2004250029 A1</td>
<td>09-12-2004</td>
<td>US 2004250031 A1</td>
<td>09-12-2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2004250032 A1</td>
<td>09-12-2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2007517290 T</td>
<td>28-06-2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2005064469 A1</td>
<td>14-07-2005</td>
</tr>
<tr>
<td>US 5592618 A</td>
<td>07-01-1997</td>
<td>NONE</td>
<td></td>
</tr>
</tbody>
</table>