159159

METADATA LOCK MANAGEMENT IN ADVANCED FUNCTION N-WAY SHARED STORAGE CONTROLLER SYSTEMS

Field of the Invention

The present invention relates to the field of computer storage controllers, and particularly to advanced function storage controllers connected in n-way storage configurations with shared storage.

Background of the Invention

In the field of advanced computer storage, there is increasing demand for what have come to be described as "advanced functions". Such functions go beyond the simple I/O functions of conventional storage systems. Advanced functions are well known in the art and depend on the control of the metadata used to retain state information about the real or "user" data stored in the system. The manipulations available using advanced functions enable various actions to be applied quickly to virtual images of data, while leaving the "real" data available for use by user applications. One example of such an advanced function is 'Flash Copy' and though the present invention is not limited to that function, a description of one embodiment thereof using Flash Copy as one example only is appropriate to explain the background of the invention.

At the highest level, Flash Copy is a function where a second image of 'some data' is made available. This function is sometimes known as Point-In-Time copy, or T0-copy. The second image's contents are initially identical to that of the first. The second image is made available 'instantly'. In practical terms this means that the second image is made available in much less time than would be required to create a true, separate, physical copy, and that this means that it can be established without unacceptable disruption to a using application's operation.

Once established, the second copy can be used for a number of purposes including performing backups, system trials and data mining. The first copy continues to be used for its original purpose by the original using application. Contrast this in the case of backup, where without Flash Copy, the application must be shutdown, and the backup taken, before the application can be restarted again. It is becoming increasingly difficult to find time windows where an application is sufficiently idle to be shut down. The cost of taking a backup is increasing. There is significant and increasing business value in the ability of Flash Copy to allow backups to be taken without stopping the business.

G00002

Flash Copy implementations achieve the illusion of the existence of a second image by redirecting read I/O addressed to the second image (henceforth referred to as "Target") to the original image (henceforth referred to as "Source"), unless that region has been subject to a write. Where a region has been the subject of a write (to either Source or Target), then to maintain the illusion that both Source and Target own their own copy of the data, a process is invoked which suspends the operation of the write command, and without it having taken effect, issues a read of the affected region from the Source, applies the read data to the Target with a write, then (and only if all steps were successful) releases the suspended write. Subsequent writes to the same region do not need to be suspended since the Target will already have its own copy of the data. This copy-on-write technique is well known and is used in many environments.

There are many variations in how Flash Copy is implemented. These variations show through in the various features of an implementation. For instance, particular implementations allow reads and writes to the Target, while others only allow reads. Some implementations allow only limited update to the Target. Some require the Target to be the same size as the Source, while some permit it to be smaller.

However, all implementations rely on some data structure which governs the above decisions, namely, the decision as to whether reads received at the Target are issued to the Source or the Target, and the decision as to whether a write must be suspended to allow the copy-on-write to take place. The data structure essentially tracks the regions which have been copied from source to target, as distinct from those that have not.

Maintenance of this data structure (hereinafter called metadata) is key to implementing the algorithm behind Flash Copy. Other 'advanced functions' such as Remote Copy (also known as PPRC or continuous copy or remote mirroring) or Virtualization rely on similar data structures, and thus fall within the scope of the present invention. The metadata for each of these functions differs, but in all cases, it is used to maintain state information, such as the location of data, the mappings of virtualized files to real storage objects, and so forth. The metadata is held in persistent storage.

A function such as Flash Copy is relatively straightforward to implement within a single CPU complex (possibly with SMP processors), as is often employed within modern storage controllers. With a little more effort, it is possible to implement fault tolerant Flash Copy, such that

(at least) two CPU complexes have access to a copy of the metadata. In the event of a failure of the first CPU complex, the second can be used to continue operation, without loss of access to the Target Image.

However, the I/O capability of a single CPU complex is limited. Though improving the capabilities of a single CPU complex measured in terms of either I/Os per second, or bandwidth (MB/s) has a finite limit, and will eventually impose a constraint on the performance of the using application(s). This limit arises in many implementations of Flash Copy, but a good example is in Storage Controllers. A typical storage controller has a single (or possibly redundant pair) of CPU complexes, which dictate a limit in the performance capability of that controller.

More storage controllers can be installed. But the separate storage controllers do not share access to the metadata, and therefore do not cooperate in managing a Flash Copy image. The storage space becomes fragmented, with functions such as Flash Copy being confined to the scope of a single controller system. Both Source and Target disks must be managed within the same storage controller. A single storage controller disk space might become full, while another has some spare space, but it is not possible to separate the Source and Target disks, placing the Target disk under the control of the new controller. (This is particularly unfortunate in the case of a new Flash Copy, where moving the Target is a cheap operation, as it has no physical data associated with it).

As well as constraining the total performance possible for a Source/Target pair, the constraint of single-controller storage functions adds complexity to the administration of the storage environment. The administrative cost is often cited as the biggest cost in the Total Cost of Ownership of storage. It would be significantly advantageous to reduce system cost and complexity by removing all possible arbitrary constraints.

Typically, storage control systems today do not attempt to solve this problem. They implement advanced functions that are confined to a single controller, and hence are constrained by the capability of that controller.

A simple way of allowing multiple controllers to participate in a shared Flash Copy relationship is to assign one controller as the Owner of the metadata, and have the other controllers forward all read and write requests to that controller. The owning controller processes the requests as if they came directly from its own attached host servers, using the algorithm described above, and completes each request back to the originating controller.

The main drawback of such a system, and the reason that it is not widely used, is that the burden of forwarding each I/O request is too great, possibly doubling the total system-wide cost, and hence approximately halving the system performance.

It is known in the field of distributed parallel database processing to have a distributed lock management facility that enables resources to be assembled into "lock clubs" and to assign lock club owners which in turn control all locking for their assigned regions and issue locking control messages to I/O-requesting clients. Such a system is implemented at the logical resource level, and does not allow for control of locks among storage controller systems, nor does it provide any form of lock management at a metadata level. It also introduces considerable overhead in the case of storage virtualization, when real data segments may be held in widely distributed physical media.

It has been suggested in the academic literature, for example in "Scalable Concurrency Control and Recovery for Shared Storage Arrays", by Amiri et al., that it is possible to use distributed lock management at the device level in storage controller networks. However, it is known that such lock management techniques are inhibited by the burden of messaging that must take place among the controllers, leading to long latency periods and increased potential for deadlocks and repeated retries.

There would therefore be considerable advantage in a low cost, high-performance, scalable scheme which allows for any of N storage controllers (an "n-way" system) to participate in advanced storage functions where the ownership and locking at a metadata level could be distributed among the devices.

Summary of the Invention

The present invention accordingly provides, in a first aspect, a storage control apparatus operable in a network of storage controller nodes comprising an I/O performing component; an ownership assignment component to assign ownership of metadata for data to an owner storage controller node; a lock management component at said owner storage controller node operable to control locking of said metadata during I/O activity to ensure coherency of said data with any copy of said data held from time to time; a messaging component at said owner storage controller node operable to pass one or more messages between storage controller nodes to request a metadata state, grant a lock, request release of a lock, and signal that a lock has been released; wherein said I/O performing component is operable to perform I/O on data whose metadata is owned by any owner storage

controller node, subject to said I/O performing component's compliance with metadata lock protocols controlled by said owner storage controller node; and wherein any said copy of said data held from time to time is maintained in a coherency relation with said data.

Preferably, said I/O performing component is operable to perform a Flash Copy, and wherein said metadata comprises grain copy state metadata.

Preferably, said I/O performing component is operable to perform a Remote Copy, and wherein said metadata comprises grain copy state metadata.

Preferably, said I/O performing component is operable to perform Storage Virtualization activity, and wherein said metadata comprises virtualization state metadata.

The present invention provides, in a second aspect, a method of operating storage control apparatus in a network of storage controller nodes comprising the steps of receiving an I/O request by an I/O performing component; assigning ownership of metadata for data to an owner storage controller node; controlling locking of said metadata during I/O activity to ensure coherency of said metadata with any copy of said metadata held from time to time; passing one or more messages between storage controller nodes to request metadata state, grant a lock, request release of a lock, and signal that a lock has been released; performing I/O on data whose metadata is owned by any owner storage controller node, subject to said I/O performing component's compliance with metadata lock protocols controlled by said owner storage controller node; and maintaining any said copy of said data held from time to time in a coherency relation with said data.

Preferably, said I/O is a Flash Copy, and wherein said metadata comprises grain copy state metadata.

Preferably, said I/O is a Remote Copy, and wherein said metadata comprises grain copy state metadata.

Preferably, said I/O is Storage Virtualization activity, and wherein said metadata comprises virtualization state metadata.

The present invention provides, in a third aspect, a computer program product tangibly embodied in a computer-readable storage medium, comprising computer program code means to, when loaded into a computer system and executed thereon, cause storage control apparatus in a network of storage controller nodes to perform the steps of receiving an I/O request by an I/O performing component; assigning ownership of metadata for data to an owner

storage controller node; controlling locking of said metadata during I/O activity to ensure coherency of said data with any copy of said data held from time to time; passing one or more messages between storage controller nodes to request metadata state, grant a lock, request release of a lock, and signal that a lock has been released; performing I/O on data whose metadata is owned by any owner storage controller node, subject to said I/O performing component's compliance with metadata lock protocols controlled by said owner storage controller node; and maintaining any said copy of said data held from time to time in a coherency relation with said data.

The presently preferred embodiment of the present invention thus employs messaging to coordinate locking activity at a metadata level among plural storage controllers (or nodes) in an n-way system. The messaging coordinates activity between the nodes in the system, but each node is still responsible for performing its own I/O.

Brief Description of the Drawings

A preferred embodiment of the present invention will now be described by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a flow diagram illustrating steps of a method in accordance with a preferred embodiment of the present invention.

Figure 2 is a block diagram illustrating an apparatus in accordance with a preferred embodiment of the present invention.

Detailed Description of the Preferred Embodiment

The preferred embodiment of the present invention employs messaging to coordinate activity between multiple storage controllers (or nodes) in an n-way storage system. The messaging coordinates activity between the nodes in the system, but each node is still responsible for performing its own I/O. Such messaging imposes a lower processing burden on the nodes in the system than the I/O forwarding scheme of the known storage prior art illustrated above. This is to a large degree because the protocol messages are smaller (in total number of bytes transferred) than the I/O they relate to, but also messages are amenable to coalescing or batching, to reduce system overhead further.

As an example, consider an n-way system implementing Flash Copy. Assume every node has access to the storage managed by the cooperating set of n nodes. One of the nodes is designated (102) as an owner for the

metadata controlling all I/O relationships of a region of storage. The other nodes are designated as clients. In a presently most preferred embodiment, one of the client nodes is further designated as a backup owner and maintains a copy of the metadata in order to provide continuous availability in the event of a failure of the owner node.

Consider a host I/O request arriving (104) at a particular client node ('C'). Suppose that the host I/O request is either a Read or Write of the Target disk, or possibly a Write of the Source disk. Client C begins processing by suspending (106) the I/O. C then sends (108) a message REQ to the Owner node O, asking if the grain has been copied.

On receipt of message REQ, O inspects its own metadata structures. If O finds that the region has already been copied, O replies (110) with a NACK message. If O finds that the region has not already been copied, it places a lock record against the appropriate metadata for the region within its own metadata structures, and replies (112) with a GNT message to indicate that the lock has been granted. The lock record is required to ensure compatibility between the request just received and granted, and further requests that might arrive affecting the same metadata while the processing at C continues. The details of how the lock record is maintained, and what the compatibility constraints are, are the same as if the I/O had been received locally by O, and are well known to those skilled in the art.

On receipt of a NACK message, C unpends (114) the original I/O request.

On receipt of the GNT message, C continues (116) by performing the data transfer or transfers required by the Flash Copy algorithm. In the case of a Target Read, this means performing the read to the source disk. Some time later, C will process completion (118) for the read request, and will issue (120) an UNL message to O to request that the lock be released, at the same time as completing the original I/O request to the host system that issued it.

O, on receipt of an UNL message, removes (122) the lock record from its metadata table, thus possibly releasing further I/O requests that were suspended because of that lock. In the presently most preferred embodiment, O then delivers (124) a UNLD message to C indicating that the lock has been released, and allowing C to reuse the resources associated with the original request. This is, however, not required by the Flash Copy algorithm itself.

In the case of a write (to either Target or Source) C must perform the copy-on-write (127). Having completed all steps of the copy-on-write, and with the original write I/O request still suspended, C issues (126) an UNLC request to O to request that the lock be released and the metadata be updated to record the copy-on-write.

O, on receipt of the UNLC message, marks (128) in the metadata the region affected as having been copied, removes (130) the lock record, informs (132) any waiting requests that the area has now been copied, and then issues (134) an UNLD message to C to indicate that the lock has been released.

C, on receipt of the UNLD message, releases (136) the suspended write operation, which will some time later complete, and then C completes (138) the write operation to the host.

Recovery paths are required to cater for the situation where a disk I/O fails, or the messaging system fails, or a node fails, but the requirements and implementations of these are well understood to persons of ordinary skill in the art, and need not be described here.

The above description has been cast in terms relating to a single I/O, and a single Client C. But it will be clear to one of ordinary skill in the art how the scheme continues to operate in the presence of multiple I/Os, from multiple client nodes, with O continuing to process all requests using the same algorithm.

Turning now to Figure 2, there is shown an apparatus in accordance with a presently preferred embodiment of the invention, the apparatus embodied in a storage controller network comprising an Owner (202), a Client (204) I/O performing component, a portion of metadata (206) relating to data (208) held under the control of the storage network, and a copy (209) of the data (208). The apparatus includes an ownership assignment component (210) to assign ownership of metadata to the Owner (202), and a lock management component (212) operable to control locking at said metadata (206) level during I/O activity to ensure coherency with any copy (209), which may be a Flash Copy image, a Remote Copy image, or a Virtualized Storage image. Included also is a messaging component (214) attached to the Owner (202) and to the Client (204), the messaging component (214) being operable to pass one or more messages between Client (204) and Owner (202) to request a response regarding a metadata state, grant a lock, request release of a lock, and signal that a lock has been released. The Client (204) is operable to perform I/O on data whose metadata is owned by any Owner (202), subject to the Client's (204)

compliance with the lock protocols at the metadata level controlled by said Owner (202).

The presently most preferred embodiment of the present invention allocates different nodes the role of being metadata owners for different regions of storage. This ensures that a single owner node does not itself become a bottleneck in the system. An efficient scheme for balancing region ownership (although at the database table level) is known from the distributed parallel database processing schemes of the known prior art.

It will be appreciated that the method described above will typically be carried out in software running on one or more processors (not shown), and that the software may be provided as a computer program element carried on any suitable data carrier (also not shown) such as a magnetic or optical computer disc. The channels for the transmission of data likewise may include storage media of all descriptions as well as signal carrying media, such as wired or wireless signal media.

The present invention may suitably be embodied as a computer program product for use with a computer system. Such an implementation may comprise a series of computer readable instructions either fixed on a tangible medium, such as a computer readable medium, for example, diskette, CD-ROM, ROM, or hard disk, or transmittable to a computer system, via a modem or other interface device, over either a tangible medium, including but not limited to optical or analogue communications lines, or intangibly using wireless techniques, including but not limited to microwave, infrared or other transmission techniques. The series of computer readable instructions embodies all or part of the functionality previously described herein.

Those skilled in the art will appreciate that such computer readable instructions can be written in a number of programming languages for use with many computer architectures or operating systems. Further, such instructions may be stored using any memory technology, present or future, including but not limited to, semiconductor, magnetic, or optical, or transmitted using any communications technology, present or future, including but not limited to optical, infrared, or microwave. It is contemplated that such a computer program product may be distributed as a removable medium with accompanying printed or electronic documentation, for example, shrink-wrapped software, pre-loaded with a computer system, for example, on a system ROM or fixed disk, or distributed from a server or electronic bulletin board over a network, for example, the Internet or World Wide Web.

It will be appreciated that various modifications to the embodiment described above will be apparent to a person of ordinary skill in the art.

CLAIMS

1. A storage control apparatus operable in a network of storage controller nodes comprising:

an I/O performing component;

an ownership assignment component to assign ownership of metadata for data to an owner storage controller node;

a lock management component at said owner storage controller node operable to control locking of said metadata during I/O activity;

a messaging component at said owner storage controller node operable to pass one or more messages between storage controller nodes to request a metadata state, grant a lock, request release of a lock, and signal that a lock has been released;

wherein said I/O performing component is operable to perform I/O on data whose metadata is owned by any owner storage controller node, subject to said I/O performing component's compliance with metadata lock protocols controlled by said owner storage controller node; and

wherein any copy of said data held from time to time is maintained in a coherency relationship with said data.

2. The apparatus as claimed in claim 1 , wherein said I/O performing component is operable to perform a Flash Copy, and wherein said metadata comprises grain copy state metadata.

3. The apparatus as claimed in claim 1 , wherein said I/O performing component is operable to perform a Remote Copy, and wherein said metadata comprises grain copy state metadata.

4. The apparatus as claimed in claim 1 , wherein said I/O performing component is operable to perform Storage Virtualization activity, and wherein said metadata comprises virtualization state metadata.

5. A method of operating storage control apparatus in a network of storage controller nodes comprising the steps of:

receiving an I/O request by an I/O performing component;

assigning ownership of metadata for data to an owner storage controller node;

controlling locking of said metadata during I/O activity;

passing one or more messages between storage controller nodes to request metadata state, grant a lock, request release of a lock, and signal that a lock has been released;

performing I/O on data whose metadata is owned by any owner storage controller node, subject to said I/O performing component's compliance with metadata lock protocols controlled by said owner storage controller node; and

maintaining any copy of said data held from time to time in a coherency relation with said data.

6. The method as claimed in claim 5, wherein said I/O is a Flash Copy, and wherein said metadata comprises grain copy state metadata.

7. The method as claimed in claim 5, wherein said I/O is a Remote Copy, and wherein said metadata comprises grain copy state metadata.

8. The method as claimed in claim 5, wherein said I/O is Storage Virtualization activity, and wherein said metadata comprises virtualization state metadata.

9. A computer program product tangibly embodied in a computer-readable storage medium, comprising computer program code means to, when loaded into a computer system and executed thereon, cause storage control apparatus in a network of storage controller nodes to perform the steps of:

receiving an I/O request by an I/O performing component;

assigning ownership of metadata for data to an owner storage controller node;

controlling locking of said metadata during I/O activity;

passing one or more messages between storage controller nodes to request metadata state, grant a lock, request release of a lock, and signal that a lock has been released;

performing I/O on data whose metadata is owned by any owner storage controller node, subject to said I/O performing component's compliance with metadata lock protocols controlled by said owner storage controller node; and

maintaining any copy of said data held from time to time in a coherency relation with said data.