Advanced functions for storage subsystems: Supporting continuous availability

by A. C. Azagury
M. E. Factor
W. F. Micka

Storage subsystems must support advanced copy functions. In particular, these functions are needed to enable the subsystem to support disaster recovery. Continuous remote copy functions ensure that all data written to a primary control unit are also written to a remote secondary control unit that, it is assumed, will not be impacted by a disaster. There are many variants of continuous remote copy, and it can be implemented in many ways and at various levels, including the application, the file system, and the disk storage subsystem. Although we mention other levels, our focus is on the storage subsystem. In this paper, we describe the variants of continuous remote copy and their implementation, emphasizing Peer-to-Peer Remote Copy (PPRC) supported by the IBM TotalStorage™ Enterprise Storage Server® (ESS).

Storage subsystems (or storage control units) were once expected only to store and retrieve randomly accessible data. That day, however, is long gone, and today storage subsystems—in particular high-end subsystems such as the IBM TotalStorage* Enterprise Storage Server* (ESS), EMC's Symmetrix**, or Hitachi's Lightning—are expected to play an integral role in supporting continuously available business operations.

To support continuously available operation, storage subsystems must support advanced copy functions. These include point-in-time copy and continuous remote copy functions. Point-in-time copy functions enable administrative operations to be performed on the data (e.g., backups, checkpoints, etc.), in almost zero time, in such a way that the applications using the data are not significantly impacted. Continuous remote copy functions support continuous availability by ensuring that all data written to a primary control unit are also written to a remote secondary control unit, which, it is assumed, will not be impacted by a disaster.

Point-in-time copy and continuous remote copy are building blocks for overall solutions, which enable continuously available business operation; they do not, themselves, provide these solutions. For instance, to allow backing up a large ERP (Enterprise Resource Planning) installation, a database hot backup mode is used to quiesce the application (that is, end it by allowing operations to complete normally) while a point-in-time copy is made of the underlying data, and it is this copy that is backed up.[1] Similarly, to ensure that a business can keep operating in the event of a disaster, some form of cluster management software is responsible for leveraging a continuous remote copy facility to "bring up" an application at a backup site in the event of a disaster. For an example, see Reference 2.

In this paper, we focus on *continuous remote copy*. We have described solutions for point-in-time copy

 0018-8670/03/$5.00 © 2003 IBM

in more detail in Reference 3 and provide a brief summary of this function here.

A disaster recovery solution for a data center must provide a copy of a corporation's data that is physically distant from the corporation's main data center, thereby allowing a business to resume operation in a reasonable amount of time after a disaster (e.g., a fire, flood, or power outage, that destroys or otherwise makes unusable the corporation's data center). A wide range of disaster recovery solutions is possible. The solutions differ in their implementation cost, their impact on normal business operations, the amount of data lost, and the length of time a disaster can cause a business's data to be unavailable. One of the main items that impacts the cost of a disaster recovery solution is the mechanism for ensuring the existence of a remote copy of the data.

Which solution is appropriate for a given situation depends upon the trade-offs made by the corporation. At one extreme are solutions where a nightly tape backup is placed on a truck and driven to a remote location; in the event of a disaster, new information technology equipment is obtained, and the data are restored from tape. Such solutions could make use of point-in-time copy facilities to minimize the impact of the backup on normal business activities. At the other extreme are the solutions in which an on-line copy of data is maintained at a site that is physically remote from the corporation's main data center. This on-line secondary copy can be kept continuously synchronized with the primary copy, making it a mirror of that copy, it can be continuously consistent with the primary copy but running behind, or it can be only periodically consistent. All of these on-line solutions build upon some type of continuous remote copy facility to approach continuous operation even in the face of a disaster.

A continuous remote copy facility differs from a point-in-time copy in two essential ways. First, as the name implies, the source and target of the remote copy (also referred to as primary and secondary) are located at some distance from each other. Second, and more significantly, a continuous remote copy facility is not aimed at capturing the state of the source at some point in time, but rather aims at reflecting all changes made to the source data at the target.

A remote copy facility can be viewed as having essentially two phases. An initial copy phase involves a bulk transfer of all of the data at the primary site to the secondary site. This phase involves transferring large amounts of data in large units. The second phase involves transferring modifications that occur at the primary site to the secondary site; this may include updates that occurred during the first phase. This continuous phase involves transferring smaller units of data. Given sufficient time without modifications to the primary site, any continuous remote copy solution should be able to ensure that the secondary copy of the data is identical to the primary copy.

Given this definition, it should be clear that a range of continuous copy solutions exists. These solutions can be implemented above the file system, by the file system,[4] at the volume manager or device driver level,[5,6] at the storage subsystem,[7–10] or via some combination of external facilities and support by the disk subsystem.[9] Although we mention solutions at other levels, our focus is on continuous remote copy facilities provided by storage control systems. As for point-in-time copy, the biggest benefit of providing this function at the level of the storage subsystem is performance—we do not needlessly add load to other components of the system. In addition, solutions can be uniform across applications. The biggest drawbacks are the lack of semantic knowledge (i.e., knowledge at the subsystem level concerning the content of storage blocks) and the requirement for software to integrate advanced copy functions with the applications in order to provide a total solution.

In the remainder of this paper, we provide a background description of point-in-time copy and a more detailed definition of continuous copy, describing the range of behaviors the latter function can display and the various generic ways of implementing this function in a block storage subsystem. We then show how these facilities are realized in practice by briefly surveying several existing implementations. In particular, we describe in detail the Extended Remote Copy (XRC)[9] of IBM; we also describe Network Appliance's SnapMirror**[4] and OceanStore.[11] We then describe the Peer-to-Peer Remote Copy (PPRC) facility of ESS,[9] including a discussion of the new Extended Distance option.[12] These last functions were developed in our labs.

Point-in-time copy

Point-in-time copies or "snapshots" of data can be made for a variety of reasons. They allow easy restoration of data in case of inadvertent corruption, backup of a consistent image of the data, easy rep-

lication of "production" data for test environments, mining of nearly current versions of data, and so on. The ability to create a point-in-time copy of the data efficiently, with minimal interruption and with minimal overhead, is critical in most production environments, where the expectation for continuous operation is commonplace. The storage capacity growth trend, with drive capacity doubling roughly every 12 months for the last few years, is expected to continue in the near future, and this trend has rendered obsolete point-in-time solutions based on an actual physical copy of the data.

Although conceptually simple, point-in-time copy functions come in a variety of flavors. In some cases, limitations are imposed on the copy, such as making the copy read-only, limiting its fault tolerance, limiting the number of "outstanding" copies, and so on. In addition, while some copy approaches allow almost immediate copying of the data (without advanced planning), some implementations require careful ahead-of-time planning. A more comprehensive review of point-in-time copy techniques is provided in Reference 3.

There are three major families of point-in-time copy implementation: split mirror, changed block, and concurrent. In a *split mirror* implementation, a copy of the data is created ahead of time, and the copy is then kept up-to-date synchronously with the source. When the "point-in-time" copy function is invoked, the source and copy of the data are decoupled, allowing them to be read or written independently. Although the decoupling of the source and the copy of the data can be performed very efficiently, split mirror implementations are space-inefficient, since they require an actual copy of the data to be prepared, and ahead-of-time planning must be done by the storage administrator. In addition, periodic resynchronizations are required in order to create new versions of the point-in-time copy. *Changed-block* implementations use copy-on-write techniques. The source and the copy of the data share the same actual backing store; that is, only one physical copy of the data is maintained, until either the source or the copy is overwritten. Changed block implementations can be space-efficient and do not require early planning, and the point-in-time copy can be very efficient, since only the meta-data (e.g., a bitmap) of the data need to be processed when the function is invoked. However, since the data are not necessarily copied, both source and target volumes are subject to data loss, should a single physical block be lost. Finally, in a *concurrent* implementation, the point-in-time copy is implemented similarly to changed block implementations, except that after the point-in-time function invocation completes, the data are copied in the background. This is particularly important in environments where an actual physical copy of the data is eventually required for reliability reasons.

Point-in-time copies can also be made at the file system level, where the additional semantic knowledge about the data (enabling the making of distinctions between data and-meta-data) can be leveraged for more efficient implementations. However, file system snapshots carry a toll on server CPU cycles and the storage network, since data and meta-data must be moved from the storage subsystem to the server and vice versa.

Continuous remote copy

As mentioned above, there are several different types of continuous remote copy solutions. Most commonly, a continuous remote copy is used as part of a disaster recovery solution, although there are other possible uses, for example, data migration or replica creation. Continuous remote copy facilities can be distinguished by several characteristics. Some of the more important characteristics include consistency, currency, latency impact, and bandwidth requirements.

Consistency—does the remote copy reflect a consistent view of the data as seen at the source site at some point in time? Consistency models are a field of significant research in computer science, and large numbers of these models have been defined. In the context of continuous remote copy, there are three types of consistency guarantees that are often considered. The weakest is no guarantee; in this case, we are ensured only that given sufficient time without modifications to the primary site, the continuous remote copy facility will eventually cause the secondary site to be equivalent to the primary.

A second type of consistency guarantee is *power-failure consistency*; this is the guarantee typically seen in continuous solutions with the most stringent requirements for currency (see below). In power-failure consistency, the image of the data seen at the secondary site is consistent with what would have been seen had there been a power failure and recovery at the primary site. This is a form of causal consistency in which if a write A completes at the primary site prior to a write B being initiated, then

the secondary image of the data will never contain the modifications due to write *B* without also containing the modifications due to write *A*. Note, that application recovery is typically required if the data image is in a power-failure-consistent state, for example, to apply a database log or reconstruct a file system's meta-data.

A final type of consistency guarantee is *application consistency*. In application consistency, the image of the data at the secondary site is consistent with what would have been seen at the primary site had the application been given a chance to shut down normally. An application-consistent copy implies greater data loss, that is, less currency (see below) than a power-failure-consistent copy, but it also implies that an application can be up and running based on the secondary copy much sooner, since application recovery is not required.

Currency—how out of date are the data at the remote site? The currency of the data at the remote site is a function of the lag between a write updating the primary site and this modification being sent to the secondary site: the greater the delay in transferring the data, the less the currency.

A continuous remote copy solution can be synchronous or asynchronous. A synchronous solution ensures that all modifications are transferred to the remote site prior to the acknowledgment of each write to the host. Synchronous solutions inherently guarantee power-failure consistency and also guarantee that the data at the secondary site are as up-to-date as possible. When used as part of a disaster recovery solution, synchronous continuous remote copy solutions minimize the amount of data lost in the event of a disaster.

Asynchronous solutions acknowledge a write and allow the application executing the write to proceed prior to the modifications being sent to the secondary site. There are three main reasons that asynchronous solutions are preferred in spite of the fact that they have inferior currency. First, by transferring the modifications to the remote site as part of the processing of a write, synchronous solutions can have a major impact on application performance (see below); asynchronous solutions can significantly reduce the impact of a continuous remote copy solution on the application. Second, by delaying transfer of the data, it is possible that a modified block will be overwritten before it is transferred, thus allowing the block to be transferred once rather than twice. Third, an asynchronous approach, by delaying the transfer of data, can use a lower-bandwidth connection between the primary and secondary copies.

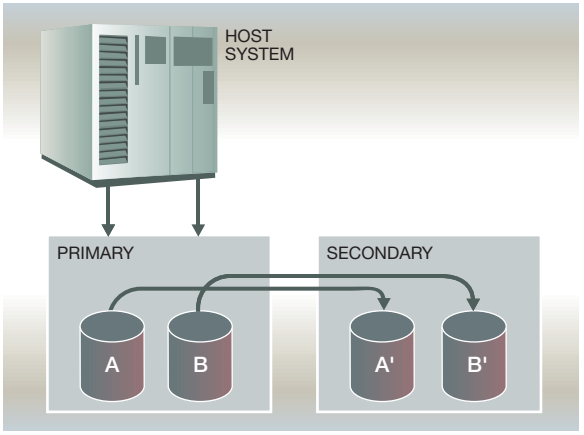
Latency impact—how much is application response impacted by the overhead of the continuous remote copy facility? The latency impact should be felt only on application writes. For synchronous solutions, the impact is a function of the distance between the primary and secondary sites; the overhead is at least 5 microseconds per kilometer, based on the speed of light in glass. This is probably a very optimistic lower bound, as there is additional overhead for the management of the transfer, switching delays, and so on, which can approach and even exceed a millisecond per write and which can also vary with distance. For asynchronous solutions, the latency impact involves only the overhead of keeping track of which data need to eventually be transferred to the secondary site. This can be as simple as setting a bit, or as complex as preparing a message to transfer the data; however, the overhead in all cases is local and is independent of the distance between the primary and secondary sites.

A related question is what reduction (if any) occurs in application throughput due to the overhead of the disaster recovery solution? Throughput impacts affect both write and read requests and are purely a function of the local overhead encountered in implementing the continuous remote copy. Throughput should not be impacted by the distance between the primary and the secondary sites.

Bandwidth requirements—what network bandwidth is required for the solution? For a synchronous solution, the bandwidth between the primary and secondary sites must be sufficient to carry the instantaneous peak write load, if the primary site expects to avoid even further performance impact on the application. For asynchronous solutions, the bandwidth required between the primary and secondary sites must be greater than the average write bandwidth to the primary, although to avoid too great an impact on currency due to peak load periods, greater bandwidth between the sites is desirable.

Actually, the calculation of required bandwidth is much more complicated for two reasons. First, synchronous solutions can transfer only the data blocks actually written. Some asynchronous solutions may track the data to be transferred at a coarser granularity and thus may need to transfer more data than were actually written. On the other hand, if a mod-

Figure 1 Basic architecture for continuous remote copy between two pairs of volumes



ified block were rewritten, an asynchronous solution may need to transfer less data than were written.

There are many different types of continuous copy solutions that can be supported at the level of disk control units. These solutions can be broadly characterized into four types:

1. *Synchronous, continuous copy* solutions, such as PPRC[9] or Symmetrix Remote Data Facility (SRDF**),[7] in which all modifications are transferred to a remote site prior to acknowledging the write to a host
2. *Asynchronous, consistent continuous copy* solutions, such as XRC[9] or Hitachi's NanoCopy**,[8] in which the secondary copy, used in the event of disaster, is an old but power-failure-consistent copy of the data that were at the primary site at the time of the disaster
3. *Asynchronous continuous copy* solutions, such as Extended Distance[12] (IBM) or Adaptive Copy,[7] (EMC) in which data are continuously copied from a primary to a secondary site, but consistency is not maintained. Such solutions are not directly useful for a disaster recovery solution (in general), but they can be components in a larger solution.
4. *Asynchronous, periodic copy* solutions, such as SnapMirror,[13] IBM's split mirror,[1] or solutions that can be built on top of asynchronous continuous copy solutions.

When implemented at the level of a disk control unit, the continuous copy solutions involve replicating some number of the logical units or volumes exported by the control unit. The volumes at the primary and secondary control units are organized into pairs as shown in Figure 1.

As mentioned previously, these continuous copy solutions have two phases. The first phase is a bulk copy phase in which there is an attempt to (basically) synchronize the primary copy and the secondary copy with each other. This phase is executed when the pairing is initially created between the primary and secondary site. It is also executed after a failure, for example, a line outage, preventing data transfer from the primary to the secondary site; in this resynchronization, it is desirable to send only those data that were modified while data transfer was not possible, and not the entire volume. In ESS the way only modified data are transferred during a resynchronization is by associating a persistent bit with each track and turning on this bit if data are modified at a time when the data cannot be transferred. A track is either 32K or 56K bytes, depending upon whether a volume is formatted for an open systems host (e.g., Microsoft Windows**, AIX*, Sun, etc.), or if it is formatted for a mainframe. When the primary site is ready to resynchronize with the secondary site, it transfers only those tracks whose associated bit has been set, turning off the bit as the track is transferred.

The second phase involves the continuous transfer of modified data. As a host modifies data, these data must be sent to the secondary site; the different solutions described above differ primarily in how they transfer data that were modified by the host.

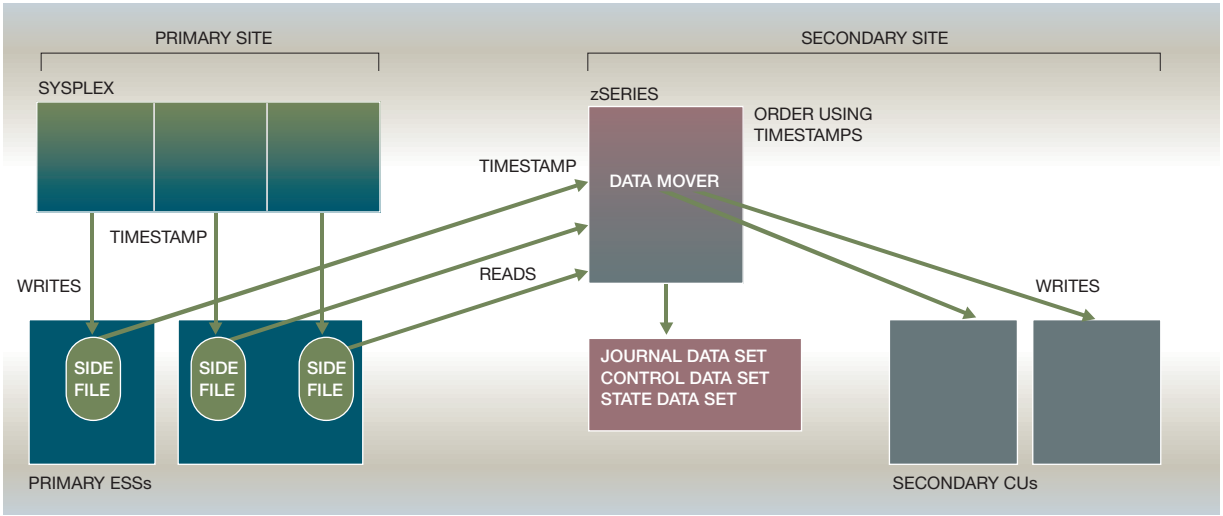
It is important to note that during the bulk transfer phase, host writes will be occurring and must be correctly transferred (according to the policy of the given solution) to the secondary site. In other words, we assume that the host continues using the disk control unit during this bulk transfer. If not done carefully, the bulk transfer will adversely impact host performance, in particular for synchronous solutions.

Continuous remote copy solutions today

The research literature is not rich in describing continuous remote copy solutions. In this section, we describe several industrial solutions for continuous remote copy.

Synchronous continuous copy. At a high-level architectural view, there is no significant difference between the synchronous continuous copy solutions

Figure 2 Schematic view of Extended Remote Copy (XRC)



available from different vendors today. PPRC,[9] SRDF,[7] and Remote Copy[8] from Hitachi all work in essentially the same way. We describe PPRC, which we developed, in more detail in the section "ESS Peer-to-Peer Remote Copy."

Asynchronous consistent continuous copy. XRC (Extended Remote Copy) is an asynchronous, continuous remote copy solution, which is supported by the disk subsystem and driven by software running on a zSeries* host. Figure 2 shows a schematic view of the XRC architecture.

In XRC, when a write is received from the host—which in the case of XRC is either a single zSeries or a zSeries Parallel Sysplex*—the control unit places information about the data that the host modified on a queue of updates (called a side file). This information includes a timestamp provided by the host and a pointer to the modified data, which are in the control unit's cache. After the information is queued and normal write processing completes, the control unit signals the successful completion of the write to the host. The zSeries Parallel Sysplex has a timer facility, which enables a single clock to be shared among multiple zSeries hosts comprising the sysplex. Thus, by associating a sysplex timestamp with each write, it is possible to reconstruct the order in which the write requests were executed.

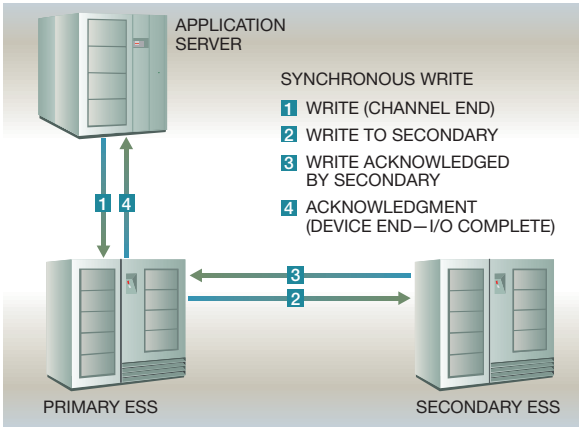
Independent of host writes, a data mover process running on a zSeries server, typically one located at the secondary site, issues commands to the primary control unit to read the host's modifications. The control unit uses the queue information to transfer the primary host's modification to the data mover; these modifications are transferred along with the timestamps, allowing the data mover process to ensure causal consistency among the writes by reordering them, if necessary. In Hitachi's Asynchronous Remote Copy, which implements the XRC architecture, the data mover facility executes in the control unit.

One subtlety to note is that if a write is received for data that are referenced from the queue before these data are transferred to the secondary site, the control unit cannot allow the data to be overwritten. Prior to allowing the write to proceed, the control unit makes a copy of the data and changes the queue to reference this copy. A new queue entry is then created for the data that are about to be modified.

Other approaches. In addition to these conventional, industrial-derived approaches at the level of the storage control unit, we consider three other approaches: host-driven replication, SnapMirror, and OceanStore.

Host-driven replication. Host-driven replication implementations intercept write operations at the host level and send the changes to a remote device or storage control unit. These implementations replace the device driver, stack a second device driver on top of

Figure 3 PPRC synchronous write



an existing one, or leverage some "logical volume manager" software already installed on the host.[5,6] This approach can therefore be made independent of the actual storage unit. However, this takes a toll on the host CPU cycles and communication bandwidth. Management of a replication solution becomes more cumbersome, since it needs to interact with all the hosts where the replication software is installed. Moreover, these implementations do not work in environments where volumes are shared, since this would require prohibitive coordination among hosts.

SnapMirror. Network Appliance's Filer implements a copy-on-write-based snapshot facility[13] that creates on-line, read-only copies of the entire file system. This facility is the basis of a remote copy solution.

The snapshot facility allows an administrator to create up to 20 snapshots of a file system. In order to support snapshots, the free block data structure is extended to mark the snapshots to which the block belongs. A block can be returned to the "free pool" only after its marker bit for this block in each snapshot is zero.

Network Appliance has integrated its snapshot facility with a SnapMirror/SnapRestore** capability. SnapMirror[4] allows automated, consistent replication of file systems to remote sites. It periodically creates a snapshot of the file system and then transfers the modified blocks to the remote site. After a baseline transfer is complete, SnapMirror uses the snapshot bitmaps to identify which blocks need to be transferred to the remote site. SnapRestore enables restoring a mirrored snapshot to the primary site.

OceanStore. OceanStore[11] is a research project that is developing a utility infrastructure for a system that is intended to span the globe and that will run on a very large set of distributed nodes. In such an environment, availability is a major concern, and OceanStore ensures availability of data by massive replication utilizing *erasure codes*. The process of erasure coding breaks a data object into a collection of e fragments, or extents, and creates ce independent subobjects, where c is the redundancy factor. Given any subset of size e of the ce subobjects, the original data object can be recreated.

In OceanStore, whenever a data object is archived, it undergoes the process of erasure coding. Each of the subobjects is then stored on a different node. By appropriately picking c and e, it is possible to ensure availability of any given data object with arbitrarily high probability.

ESS Peer-to-Peer Remote Copy

ESS Peer-to-Peer Remote Copy (PPRC) is a continuous disk data copy facility using a synchronous protocol. This copy function maintains an identical image of a logical volume on two geographically distributed volumes. As a host server application writes data to an attached primary volume, the ESS PPRC will ensure that the data written will be applied to a secondary volume before the application host is notified that the write has completed.

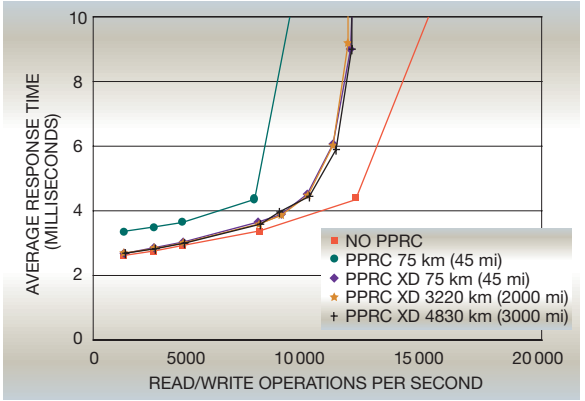
The synchronous write protocol is shown in Figure 3. The application server writes data blocks in step 1. The primary ESS stores the write blocks in a cache memory and sends them to the secondary ESS in step 2. When the secondary ESS has hardened the blocks in its cache structures, a signal is sent to the host-attached primary ESS that the operation with the host can be completed in step 3. The completion message is then sent from the primary ESS to the attached host in step 4, and the write operation is complete. As can be surmised, the PPRC synchronous operation elongates the normal write processing time by the time taken to send the modified blocks to the secondary ESS. This time consists of the following main periods: the additional processing internal to the primary ESS for the transfer of the blocks to the secondary ESS, the time for the secondary ESS to pro-

cess the transfer (which is the time for processing a fast write), and the additional propagation time of signals over communications lines or channels to access the remote secondary ESS.

Consistency of data at the remote location. Synchronous PPRC is ideal for database applications that must be restarted on the secondary volumes in the event of failures or disasters. PPRC is architected to include provisions to maintain a consistent image of the data at the remote or secondary location. The image at the secondary location can become inconsistent due to a particular behavior of synchronous copy operations called *volume suspension*. Suspension is a state that a volume may enter if there is a failure to update the secondary volume and the customer desires that the primary applications continue without interruption. When a PPRC volume pair becomes suspended, the changed tracks are recorded by means of a bitmap for future resynchronization, and the application continues with its normal operation. Due to the state of suspension, the secondary volume will not receive updates, although other secondary volumes that are not in a state of suspension may receive them. This creates a hole in the overall consistency at the secondary site and will result in a larger recovery effort if the secondary site must be used for the application program execution. To fix this hole, PPRC provides a message to the host processors and provides commands to freeze all of the secondary volumes upon detection of the first failing volume. The set of volumes that are frozen is called a consistency group. During the freeze process, application I/O is halted momentarily while automation scripts issue commands to freeze all volumes associated with the consistency group. These volumes may exist on any number of physical ESS machines. Once the freeze has been communicated to all volumes, application I/O is enabled again, and the applications are allowed to resume.

PPRC synchronous mode of operation. Synchronous PPRC operates in two modes depending on the phase of the copy—either full track asynchronous transfer mode or changed sector mode. Full track asynchronous transfer mode is always used when the initial establishment of a mirrored pair is performed and when resynchronizing changed tracks from the suspension bitmap. Full track mode ensures that the track format on both volumes is identical and that all the data for the tracks are updated. In full track mode, any updates to the volume will be recorded and sent as full tracks in an asynchronous fashion. Following the completion of the synchronization or

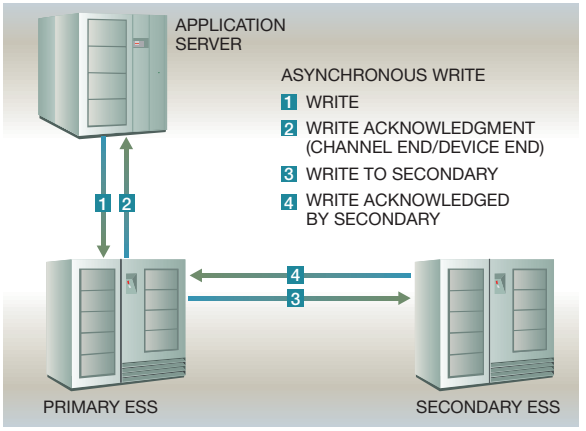
Figure 4 Response times for synchronous PPRC with cache standard workload



resynchronization phase, the volumes enter the synchronous transfer phase. A track of data stored on the ESS disks consists of blocks or sectors. When a write occurs from a host to the ESS, the data are placed into a track image in cache. The PPRC changed sector mode function will access the track image in the cache and send only the modified sectors to the remote secondary volume. This method reduces the amount of data transferred compared to full track mode, and saves connectivity expense.

Synchronous operations require confirmation of update completion at the secondary remote location before the primary host write can be acknowledged. This involves transit time for commands, data, and status to be communicated over the communications connections between the ESS machines. The time to complete the transaction is therefore impacted by the speed with which these communications can be performed. The impact that the customers' applications can tolerate can therefore limit the distance between the primary and secondary sites, in synchronous PPRC. In Figure 4, the orange plot shows the ESS response time for a range of I/O rates without any PPRC degradation. The green plot shows the same workload with PPRC established at a distance of 75 kilometers. The source for the figure is performance measurements from the test lab of the IBM Storage Systems Group on March 22, 2002. The performance penalty begins to be severe above 7000 I/O operations per second. This penalty dictates that some form of asynchronous operations should be used when the impact of distance on synchronous

Figure 5 PPRC asynchronous write



PPRC performance exceeds the acceptable value for a given customer.

PPRC asynchronous extended distance mode of operation. Asynchronous PPRC operations decouple the host-to-ESS transfer of data and status from the ESS-to-ESS transfer of data and status. Application writes are accepted in the primary ESS and queued for transfer to the remote ESS. The goal of an asynchronous transfer protocol is to eliminate the synchronous write penalty as seen in the green plot in Figure 4.

Figure 5 shows an illustration of the asynchronous sequence. At step 1, a host application write occurs at the primary ESS. Acknowledgment is made at step 2 as soon as the transfer from the host is complete and the data are stored in the cache. As part of completing the transfer from the host to the primary ESS, the modified data are enqueued for transfer to the remote ESS. Internal scheduling algorithms then select the write at step 3 for sending the write data to the remote ESS. Step 4 is the acknowledgment from the remote ESS of the successful transfer of the data. At this point the I/O can be released from the PPRC asynchronous queue at the primary ESS. When synchronous PPRC and asynchronous PPRC (on different volumes) are used concurrently, the scheduling algorithm gives higher priority to the synchronous PPRC writes, without causing starvation of the asynchronous ones.

In a remote copy application, a customer will typically mirror a large number of volumes that may exist on multiple ESS machines. In a simplified model of asynchronous mirroring, each write operation to a given volume that is received from a host and queued could be sent to the remote location in the exact sequence it was received, with respect to the writes to all other volumes. This would require coordination of the transfer of the queued writes among all ESS machines and all volumes. As can be imagined, this serialization of writes to the remote location would not necessarily provide the best performance and in fact, would eventually severely impact the customers' applications. To solve this problem, most asynchronous mirroring solutions will apply a large number of updates to the remote volumes in a batch and will not maintain the original update ordering within the batch. An algorithm determines the updates that are selected to be in a specific batch, such that consistency of data is maintained at the remote location when all of the updates within a batch are applied. This batch of updates can be called an asynchronous consistency group and differs from the synchronous consistency group in the way it is created.

Multiple asynchronous consistency groups may be created per second, in some mirroring solutions. Other mirroring solutions may create consistency groups minutes or hours apart. The highest frequency of consistency group creation results in the smallest data loss if there is a disaster and the remote volumes must be used for the customer applications. Data that are queued at the primary site waiting for transfer and any data in a partially applied batch of updates will be lost if a disaster breaks the links between the primary and secondary sites or if the primary site loses the ability to send the queued updates for any other reason. The point that is important to realize about any asynchronous solution is that the secondary volumes represent a state of the data at a point in time that is determined by the application of the last complete consistency group. To preserve the last consistent data point, an instant copy process such as FlashCopy* (which provides fast data duplication without requiring applications to pause for long periods) is usually applied to a tertiary set of volumes at the remote location.

The ESS feature that provides asynchronous PPRC for ESS is called PPRC Extended Distance (PPRC XD). PPRC XD is an asynchronous solution, because it decouples the application data write completion from the remote data propagation, and it does this with very little response time degradation. Figure 4 illustrates the effect of XD at three distances: 45 miles, 2000 miles, and 3000 miles. Shorter distances are not

shown, since most applications using this feature would be transferring data at a long distance. The three XD lines are indistinguishable and show very little deviation from base performance up to about 9000 I/Os per second. Since most customers do not average more than 9000 I/Os per second, XD will be almost transparent to the primary host applications.

Since asynchronous remote copy queues data to be sent to the remote location, a major issue to consider is the number of data blocks that reside at the primary ESS waiting to be sent. This number is directly related to the currency and consistency of the data at the remote location. This number is difficult to measure, because there are many variables to take into account. A few of these are: the number of PPRC ports, the number of host channels, the rate of update for host application writes, and the number and speed of the communications links between the primary and secondary sites. For the cache standard workload shown in Figure 4, a measurement was taken of the time to send all queued data following the termination of all new update activity. With the configuration used to generate Figure 4 and following the most active portion of the test case, the queues were empty within 20 seconds, at 2000 miles separation. Notice that, unlike XRC (see the section "Asynchronous consistent continuous copy"), in case of conflicts, PPRC XD allows data to be overwritten with no further action. PPRC XD today does not guarantee consistency of the remote volumes, because writes may be applied in a different order than that in which they occurred on the primary. The next subsection describes techniques that allow maintaining consistent copies of volumes with PPRC XD.

PPRC XD operation. PPRC XD uses the same commands and interfaces as those used for synchronous PPRC. Additional control bits are used to enable ESS to set up and perform the Extended Distance function. In operation, when the initial establishment of an XD relationship is created, the first phase of the function is to copy the entire primary volume to the secondary volume. When this is complete, the volume pair transitions to bit map mode, where change recording is performed. This mode is similar to the PPRC state of suspension, but differs in that the volume pair will remain in an active asynchronous mirroring state and the changed data will be sent automatically without any intervention external to the ESS. In bit map mode, host write commands are recorded by setting a bit to a "one" in an array in which each bit is related to a logical track on the volume. An internal scheduling algorithm determines when the changed data will be sent to the remote location. This algorithm manages multiple volumes and takes into consideration the bandwidth and number of PPRC links. Once all of the volumes considered to be related in a consistency group have been fully copied and are in change recording mode, a point-in-time consistent copy should be created at the remote site. This can be accomplished in two ways.

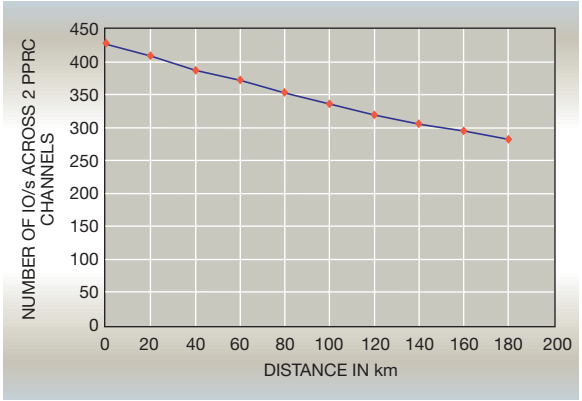
The first method is to issue a command to each volume pair to transition the pair from XD to synchronous PPRC. In synchronous mode, the volume pair transitions to full duplex state when all data are current at the remote location. When all pairs become full duplex, the application writes can be suspended, or a PPRC consistency group freeze operation can be performed. When either of these is complete, a FlashCopy can be made at the remote location. When the FlashCopy initialization is complete, usually within a few seconds, the freeze can be removed and XD reinstated. Note that the PPRC freeze will suspend all volume pairs, and host applications can be resumed with almost no performance impact.

The second method does not require the transition from PPRC XD mode to synchronous mode. Instead, the applications can be placed into "write inhibit" mode or quiesced, and the FlashCopy can be made as soon as all the queued asynchronous write data have been sent to the remote volumes. When the FlashCopy initialization is complete, application writes can resume.

PPRC performance comparison. Another important performance measurement is that of the throughput that can be achieved over distance with synchronous remote copy. Figure 6 shows the throughput using two ESCON* (Enterprise Systems Connection) ports attached to a DWDM (Dense Wave Division Multiplexer). The transfer length for each PPRC data block is 56K bytes. As shown, the rate of transfer drops more or less in a linear fashion, and this rate reduction can be attributed to the delay inherent in data and control handshakes over distance. When distances greater than 75 km are required, a channel extension product that provides compression and handshake reduction should be employed.

PPRC XD performance is highly dependent on the type of workload. For example, since the transfer operation is done asynchronously, the data might have been purged from the cache by the time they are scheduled to be transferred. A "random write" workload might pay the penalty of random writes and

Figure 6 Synchronous PPRC throughput versus distance using DWDM



reads from disk drives twice, with the corresponding impact on the host response time and throughput; sequential workloads might be less affected. On the other hand, PPRC XD performance might benefit from reduced bandwidth requirements for workloads where the same sectors are rewritten over and over (hot spots). Whereas synchronous PPRC would transfer these sectors for every write, PPRC XD may send these only once per multiple writes, with the corresponding network traffic savings.

Adding flexibility to the control of continuous remote copy

As described in the section "Continuous remote copy," there are trade-offs among the continuous remote copy variants regarding currency and consistency on the one hand, and cost (as measured by application impact and bandwidth requirements) on the other. Sophisticated combinations of these variants with point-in-time copy facilities can give greater flexibility to manage these trade-offs.

For instance, given a remote copy relationship between two volumes A and B, we can periodically switch between a synchronous-consistent and an asynchronous means of copying the data between the two sites. This can be done as follows: create the relation between the two volumes using an asynchronous mechanism that does not ensure consistency, such as PPRC Extended Distance, and periodically change the relationship to a synchronous one. After the two volumes are synchronized, create a point-in-time copy of the B volume before changing back to the asynchronous mechanism. If this point-in-time copy is executed at a time that is meaningful to the application, this mechanism can even be used to provide an application-consistent copy of the data at the remote site. By judiciously deciding when to change, an administrator can exercise a huge degree of flexibility in managing the cost/quality trade-offs.

Conclusions

We have described the current state of the art of advanced copy functions provided by high-end storage control units, focusing on the Peer-to-Peer Remote Copy facility of ESS. The different models for continuous remote copy support trade-offs between levels of performance, consistency, currency, and cost of the remote copy. Today's continuous remote copy facilities, and in particular PPRC, provide the building blocks for continuous operation even in the event of disasters.

Acknowledgments

PPRC would not exist today were it not for the diligent work of the IBM Haifa and Tucson development teams.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of EMC Corporation, Hitachi Ltd., Network Appliance, Inc., or Microsoft Corporation.

Cited references

1. S. Das, S. Schmidt, P. Pitterling, and B. Godavari, *Storage Management for SAP and DB2 UDB: Split Mirror Backup/Recovery with IBM's Enterprise Storage Server (ESS)*, IBM Corporation (2001). See http://www-1.ibm.com/servers/storage/solutions/sap/pdf/db2-sap-splitmirror.pdf.
2. *Geographically Dispersed Parallel Sysplex: The Ultimate e-business Availability Solution*, GF22-5114-03, IBM Corporation (March 2002).
3. A. Azagury, M. Factor, W. Micka, and J. Satran, "Point-in-time Copy: Yesterday, Today and Tomorrow," *Tenth Goddard Conference on Mass Storage Systems and Technologies/Nineteenth IEEE Symposium on Mass Storage Systems*, MD (April 2002).
4. H. Patterson, S. Manley, M. Federwisch, D. Hitz, S. Kleiman, and S. Owara, "SnapMirror: File System Based Asynchronous Mirroring for Disaster Recovery," *Proceedings of the FAST 2002 Conference on File and Storage Technologies*, Monterey, CA (January 2002).
5. *HAGEO for AIX and GeoRM for AIX on IBM e-server pSeries*, IBM Corporation (2002). See http://www.ibm.com/servers/aix/products/ibmsw/high_avail_network/hageo_georm.pdf.
6. *VERITAS Volume Replicator Successful Replication and Disaster Recovery*, Veritas Software Corporation (December 2001). See http://eval.veritas.com/downloads/pro/volume_replicator_whitepaper.pdf.

7. *Using EMC SnapView and MirrorView for Remote Backup*, Engineering White Paper, EMC Corporation (April 2002).
8. *Software Solutions Guide for Enterprise Storage*, Hitachi Data Systems Corporation (November 2000).
9. *Implementing ESS Copy Services on S/390*, SG24-5680-00, IBM Corporation (December 2000).
10. *Implementing ESS Copy Services on UNIX and Windows NT/2000*, SG24-5757-00, IBM Corporation (February 2001).
11. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage," *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems* (November 2000). See http://oceanstore.cs.berkeley.edu.
12. *IBM TotalStorage Enterprise Storage Server PPRC Extended Distance*, SG24-6568-00, IBM Corporation (May 2002).
13. K. Brown, J. Katcher, R. Walters, and A. Watson, *SnapManager for Microsoft Exchange 2000 Technical Overview*, Network Appliance, Inc. (2002). See http://www.netapp.com/tech_library/3198.html.

Accepted for publication February 6, 2003.

Alain C. Azagury *IBM Research Division, Haifa Research Laboratory, Haifa University, Mount Carmel, Haifa 31905, Israel (azagury@il.ibm.com).* Alain Azagury is the department general manager of the Storage and Systems department at IBM's Haifa Research Laboratory. He received B.S. and M.S. degrees in computer science in 1984 and 1987, respectively, from the Technion, Israel's Institute of Technology, where he currently teaches a course on Operating Systems Structure. The Storage and Systems department develops advanced systems and storage functions, including copy services for IBM's Enterprise Storage System, clustering and high-availability technologies, and Java™ runtime technologies. In addition, the department pioneered the development of the iSCSI standard and related technologies. The department is currently investigating novel storage technologies such as object storage and semantic access to file systems, as well as new architectures for services in grid computing. Mr. Azagury is a member of the Institute of Electrical and Electronics Engineers.

Michael E. Factor *IBM Research Division, Haifa Research Laboratory, Haifa University, Mount Carmel, Haifa 31905, Israel (factor@il.ibm.com).* Dr. Factor received the B.S. degree (Valedictorian) in computer science from Union College, Schenectady, NY in 1984, and the M.S., M. Phil., and Ph.D. degrees in computer science from Yale University in 1988, 1989, and 1990. Since graduating from Yale University, Dr. Factor has worked at the IBM Israel Haifa Research Lab (HRL). Currently, Dr. Factor's focus is on storage subsystems. He is one of the architects of advanced copy functions for the IBM Enterprise Storage Server. He is also the architect for the more advanced work being done in IBM on object stores. In addition, he takes a leading role in storage-related research in his lab, including topics such as distributed storage systems, SAN-NAS convergence, advanced storage networking technologies, storage management, and advanced file system organization techniques. In the past, Dr. Factor was the manager of the distributed and clustered systems department at HRL. Dr. Factor's areas of interest include storage subsystems, Java implementations, cluster computing, high availability, scalable servers, distributed systems, and file systems. In addition to ESS, projects that Dr. Factor has been involved with include the cluster VM for Java™, the XML file system, the IBM iSeries™ integrated file system, NAS for iSeries, and the Web server for the 1996 Atlanta Olympics.

William F. Micka *IBM Systems Groups,, 9000 S. Rita Road, Tucson, Arizona 85744 (micka@us.ibm.com).* Mr. Micka is a storage architect working on current and future storage controllers. His specialty is disaster recovery and data replication. He graduated from the University of Minnesota with a B.S. degree in electrical engineering and began his career with IBM in Rochester, Minnesota in 1968. He has worked in the field of optical character recognition, magnetic tape control unit design, fiber optic channel design, and magnetic disk controller advanced functions. He has 35 issued patents related to the various products that have been developed during his career.