Chapter 13
Local Replication

Replication is the process of creating an exact copy of data. Creating one or more replicas of the production data is one of the ways to provide Business Continuity (BC). These replicas can be used for recovery and restart operations in the event of data loss.

The primary purpose of replication is to enable users to have designated data at the right place, in a state appropriate to the recovery need. The replica should provide recoverability and restartability. Recoverability enables restoration of data from the replicas to the production volumes in the event of data loss or data corruption. It must provide minimal RPO and RTO for resuming business operations on the production volumes, while restartability must ensure consistency of data on the replica. This enables restarting business operations using the replicas.

KEY CONCEPTS

- Data Consistency
- Host-Based Local Replication
- Storage Array–Based Local Replication
- Copy on First Access (CoFA)
- Copy on First Write (CoFW)
- Restore and Restart

Replication can be classified into two major categories: local and remote. Local replication refers to replicating data within the same array or the same data center. This chapter provides details about various local replication technologies, along with key steps to plan and design an appropriate local replication solution. Remote replication is covered in Chapter 14.

13.1 Source and Target

A host accessing data from one or more LUNs on the storage array is called a *production host,* and these LUNs are known as source LUNs (devices/volumes), production LUNs, or simply the *source*.

A LUN (or LUNs) on which the data is replicated is called the target LUN or simply the *target* or replica. Targets can also be accessed by hosts other than production hosts to perform operations such as backup or testing. Target data can be updated by the hosts accessing it without modifying the source. However, the source and the target are not an identical copy of each other anymore. The target can be incrementally resynchronized (copying of data that has changed since the previous synchronization) with the source to make both source and target identical.

13.2 Uses of Local Replicas

One or more local replicas of the source data can be created for various purposes, including the following:

- **Alternate source for backup:** Under normal backup operations, data is read from the production volumes (LUNs) and written to the backup device. This places additional burden on the production infrastructure, as production LUNs are simultaneously involved in production work. As the local replica contains an exact point-in-time (PIT) copy of the source data, it can be used to perform backup operations. This alleviates the backup I/O workload on the production volumes. Another benefit of using local replicas for backup is that it reduces the *backup window* to zero.

The period during which a source is available to perform a data backup is called a *backup window*. Performing backup from the source sometimes requires the production operation to be suspended because the data being backed up is exclusively locked for the use of the backup process.

- **Fast recovery:** In the event of a partial failure of the source, or data corruption, a local replica can be used to recover lost data. In the event of a

complete failure of the source, the replica can be restored to a different set of source devices. In either case, this method provides faster recovery and minimal RTO, compared to traditional restores from tape backups. In many instances business operations can be started using the source device before the data is completely copied from the replica.

- **Decision-support activities such as reporting:** Running the reports using the data on the replicas greatly reduces the I/O burden placed on the production device.
- **Testing platform:** A local replica can be used for testing critical business data or applications. For example, when planning an application upgrade, it can be tested using the local replica. If the test is successful, it can be restored to the source volumes.
- **Data migration:** Local replication can also be used for data migration. Data migration may be performed for various reasons, such as migrating from a small LUN to a larger LUN.

13.3 Data Consistency

Most file systems and databases buffer data in the host before it is written to disk. A consistent replica ensures that data buffered in the host is properly captured on the disk when the replica is created. Ensuring consistency is the primary requirement for all the replication technologies.

13.3.1 Consistency of a Replicated File System

File systems buffer data in host memory to improve application response time. The buffered information is periodically written to disk. In UNIX operating systems, the *sync daemon* is the process that flushes the buffers to disk at set intervals. In some cases, the replica may be created in between the set intervals. Hence, the host memory buffers must be flushed to ensure data consistency on the replica, prior to its creation. Figure 13-1 illustrates flushing of the buffer to its source, which is then replicated. If the host memory buffers are not flushed, data on the replica will not contain the information that was buffered in the host. If the file system is unmounted prior to the creation of the replica, the buffers would be automatically flushed and data would be consistent on the replica.

If a mounted file system is replicated, some level of recovery such as *fsck* or *log replay* would be required on the replicated file system. When the file system replication process is completed, the replica file system can be mounted for operational use.

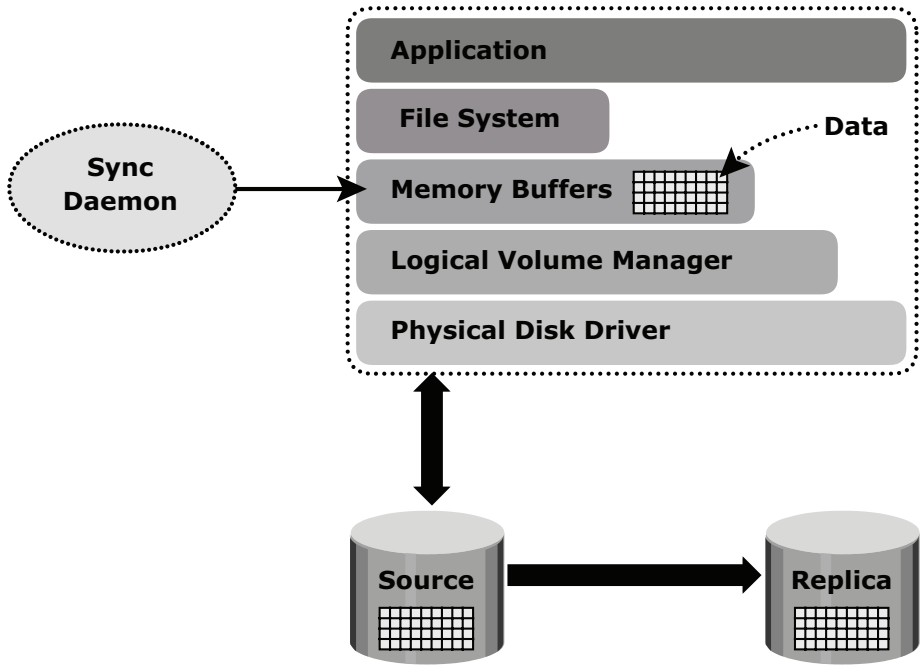


Figure 13-1: File system replication

13.3.2 Consistency of a Replicated Database

A database may be spread over numerous files, file systems, and devices. All of these must be replicated consistently to ensure that the replica is restorable and restartable. Replication can be performed with the database offline or online. If the database is offline, it is not available for I/O operations. Because no updates are occurring, the replica will be consistent.

If the database is online, it is available for I/O operations. Transactions to the database will be updating data continuously. When a database is backed up while it is online, changes made to the database at this time must be applied to the backup copy to make it consistent. Performing an online backup requires additional procedures during backup and restore. Often these procedures can be scripted to automate the process, alleviating administrative work and minimizing human error. Most databases support some form of online or hot backups. There will be increased logging activity during the time when the database is in the hot backup mode.

The sequence of operations in a hot backup mode is first to issue a database checkpoint to flush buffers to disk and place the database in hot backup mode. After taking a PIT copy, the database is taken out of hot backup mode. Logs collected are then applied to the replica to restore database consistently.

An alternate approach exploits the *dependent write I/O* principle inherent in any database management system (DBMS). According to this principle, a write I/O is not issued by an application until a prior related write I/O has completed. For example, a data write is dependent on the successful completion of the prior log write. Dependent write consistency is required for protection against power outages, loss of local channel connectivity, or storage devices. When the failure occurs a dependent write consistent image is created. A restart transforms the dependent write consistent image to a transactional consistent image — i.e., committed transactions are recovered, and in-flight transactions are discarded.

In order for a transaction to be deemed complete, databases require that a series of writes have to occur in a particular order. These writes would be recorded on the various devices/file systems. Figure 13-2, illustrates the process of flushing the buffer from host to source; I/Os 1 to 4 must complete, in order for the transaction to be considered complete. I/O 4 is dependent on I/O 3 and will occur only if I/O 3 is complete. I/O 3 is dependent on I/O 2, which in turn depends on I/O 1. Each I/O completes only after completion of the previous I/O(s).

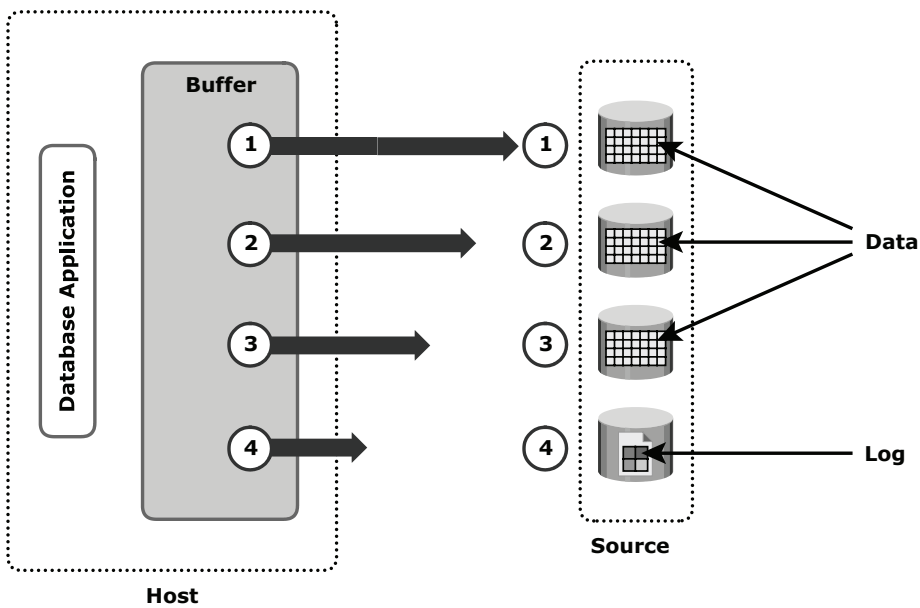


Figure 13-2: Dependent write consistency on sources

At the point in time when the replica is created, all the writes to the source devices must be captured on the replica devices to ensure data consistency. Figure 13-3 illustrates the process of replication from source to replica, I/O transactions 1 to 4 must be carried out in order for the data to be consistent on the replica.

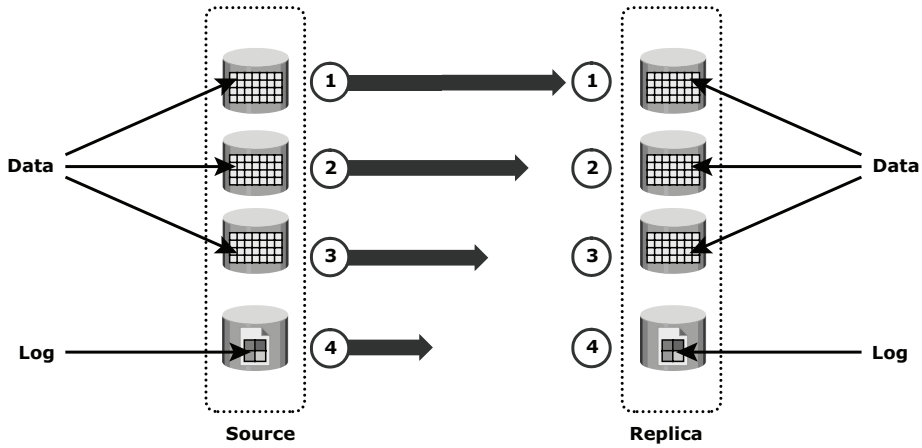


Figure 13-3: Dependent write consistency on replica

Creating a PIT copy for multiple devices happens quickly, but not instantaneously. It is possible that I/O transactions 3 and 4 were copied to the replica devices, but I/O transactions 1 and 2 were not copied. In this case, the data on the replica is inconsistent with the data on the source. If a restart were to be performed on the replica devices, I/O 4, which is available on the replica, might indicate that a particular transaction is complete, but all the data associated with the transaction will be unavailable on the replica, making the replica inconsistent.

Another way to ensure consistency is to make sure that write I/O to all source devices is held for the duration of creating the replica. This creates a consistent image on the replica. Note that databases and applications can time out if the I/O is held for too long.

13.4 Local Replication Technologies

Host-based and storage-based replications are the two major technologies adopted for local replication. File system replication and LVM-based replication are examples of host-based local replication technology. Storage array–based replication can be implemented with distinct solutions namely, full-volume mirroring, pointer-based full-volume replication, and pointer-based virtual replication.

13.4.1 Host-Based Local Replication

In host-based replication, logical volume managers (LVMs) or the file systems perform the local replication process. LVM-based replication and file system (FS) snapshot are examples of host-based local replication.

LVM-Based Replication

In LVM-based replication, logical volume manager is responsible for creating and controlling the host-level logical volume. An LVM has three components: physical volumes (physical disk), volume groups, and logical volumes. A *volume group* is created by grouping together one or more physical volumes. *Logical volumes* are created within a given volume group. A volume group can have multiple logical volumes.

In LVM-based replication, each *logical partition* in a logical volume is mapped to two physical partitions on two different physical volumes, as shown in Figure 13-4. An application write to a logical partition is written to the two physical partitions by the LVM device driver. This is also known as *LVM mirroring*. Mirrors can be split and the data contained therein can be independently accessed. LVM mirrors can be added or removed dynamically.

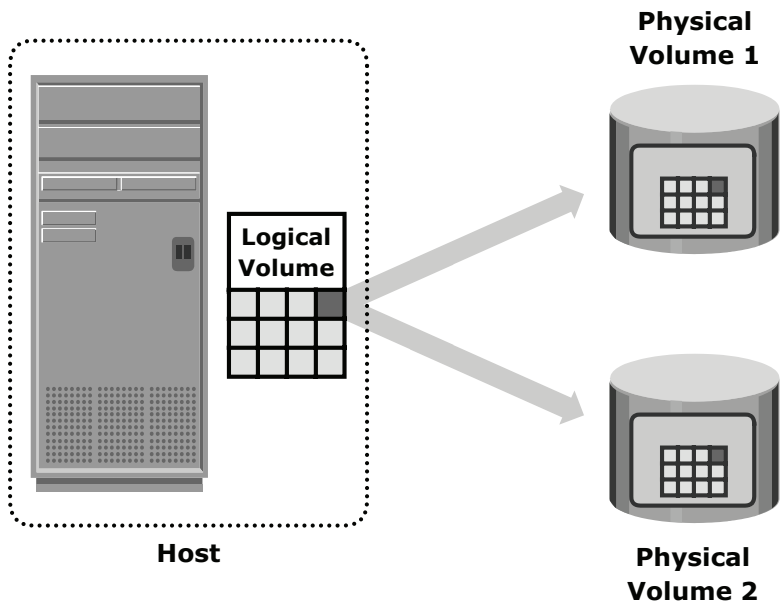


Figure 13-4: LVM-based mirroring

Advantages of LVM-Based Replication

The LVM-based replication technology is not dependent on a vendor-specific storage system. Typically, LVM is part of the operating system and no additional license is required to deploy LVM mirroring.

Limitations of LVM-Based Replication

As every write generated by an application translates into two writes on the disk, an additional burden is placed on the host CPU. This can degrade application performance. Presenting an LVM-based local replica to a second host is usually not possible because the replica will still be part of the volume group, which is usually accessed by one host at any given time.

Tracking changes to the mirrors and performing incremental synchronization operations is also a challenge as all LVMs do not support incremental resynchronization. If the devices are already protected by some level of RAID on the array, then the additional protection provided by mirroring is unnecessary. This solution does not scale to provide replicas of federated databases and applications. Both the replica and the source are stored within the same volume group. Therefore, the replica itself may become unavailable if there is an error in the volume group. If the server fails, both source and replica are unavailable until the server is brought back online.

File System Snapshot

File system (FS) snapshot is a pointer-based replica that requires a fraction of the space used by the original FS. This snapshot can be implemented by either FS itself or by LVM. It uses Copy on First Write (CoFW) principle. CoFW mechanism is discussed later in the chapter.

When the snapshot is created, a bitmap and a blockmap are created in the metadata of the Snap FS. The bitmap is used to keep track of blocks that are changed on the production FS after creation of the snap. The blockmap is used to indicate the exact address from which data is to be read when the data is accessed from the Snap FS. Immediately after creation of the snapshot all reads from the snapshot will actually be served by reading the production FS.

To read from the Snap FS, the bitmap is consulted. If the bit is 0, then the read is directed to the production FS. If the bit is 1, then the block address is obtained from the blockmap and data is read from that address. Reads from the production FS work as normal.

13.4.2 Storage Array–Based Replication

In *storage array-based local replication*, the array operating environment performs the local replication process. The host resources such as CPU and memory are not used in the replication process. Consequently, the host is not burdened by

the replication operations. The replica can be accessed by an alternate host for any business operations.

In this replication, the required number of replica devices should be selected on the same array and then data is replicated between source-replica pairs. A database could be laid out over multiple physical volumes and in that case all the devices must be replicated for a consistent PIT copy of the database.

Figure 13-5 shows storage array based local replication, where source and target are in the same array and accessed by different hosts.

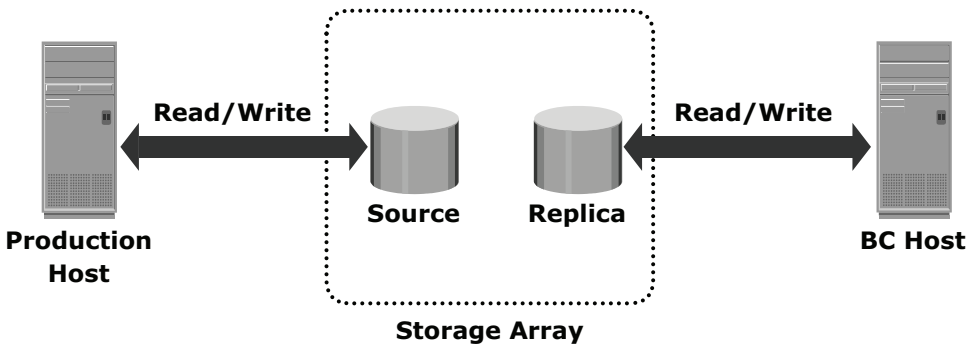


Figure 13-5: Storage array-based replication

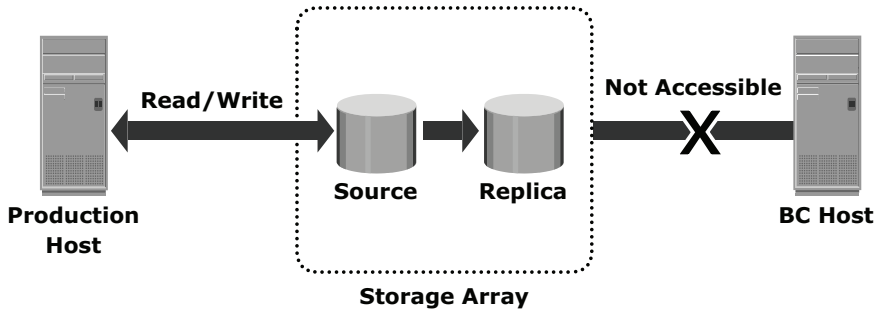
Storage array-based local replication can be further categorized as full-volume mirroring, pointer-based full-volume replication, and pointer-based virtual replication. Replica devices are also referred as target devices, accessible by business continuity host.

Full-Volume Mirroring

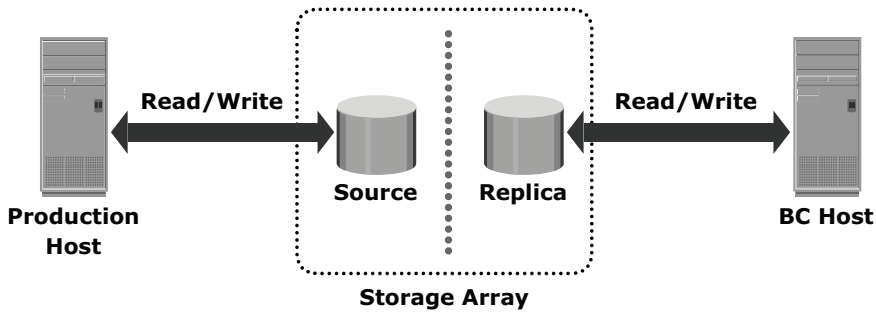
In *full-volume mirroring*, the target is attached to the source and established as a mirror of the source (Figure 13-6 [a]). Existing data on the source is copied to the target. New updates to the source are also updated on the target. After all the data is copied and both the source and the target contain identical data, the target can be considered a mirror of the source.

While the target is attached to the source and the synchronization is taking place, the target remains unavailable to any other host. However, the production host can access the source.

After synchronization is complete, the target can be detached from the source and is made available for BC operations. Figure 13-6 (b) shows full-volume mirroring when the target is detached from the source. Notice that both the source and the target can be accessed for read and write operations by the production hosts.



(a) Full volume mirroring with source attached to replica



(b) Full volume mirroring with source detached from replica

Figure 13-6: Full-volume mirroring

After the split from the source, the target becomes a PIT copy of the source. The point-in-time of a replica is determined by the time when the source is detached from the target. For example, if the time of detachment is 4:00 PM, the PIT for the target is 4:00 PM

After detachment, changes made to both source and replica can be tracked at some predefined granularity. This enables incremental resynchronization (source to target) or incremental restore (target to source). The granularity of the data change can range from 512 byte blocks to 64 KB blocks. Changes are typically tracked using bitmaps, with one bit assigned for each block. If any updates occur to a particular block, the whole block is marked as changed, regardless of the size of the actual update. However, for resynchronization (or restore), only the changed blocks have to be copied, eliminating the need for a full synchronization (or restore) operation. This method reduces the time required for these operations considerably.

In full-volume mirroring, the target is inaccessible for the duration of the synchronization process, until detachment from the source. For large databases, this can take a long time.

Pointer-Based, Full-Volume Replication

An alternative to full-volume mirroring is *pointer-based full-volume replication*. Like full-volume mirroring, this technology can provide full copies of the source data on the targets. Unlike full-volume mirroring, the target is made immediately available at the activation of the replication session. Hence, one need not wait for data synchronization to, and detachment of, the target in order to access it. The time of activation defines the PIT copy of source.

Pointer-based, full-volume replication can be activated in either Copy on First Access (CoFA) mode or Full Copy mode. In either case, at the time of activation, a protection bitmap is created for all data on the source devices. Pointers are initialized to map the (currently) empty data blocks on the target to the corresponding original data blocks on the source. The granularity can range from 512 byte blocks to 64 KB blocks or higher. Data is then copied from the source to the target, based on the mode of activation.

In CoFA, after the replication session is initiated, data is copied from the source to the target when the following occurs:

- A write operation is issued to a specific address on the source for the first time (see Figure 13-7).
- A read or write operation is issued to a specific address on the target for the first time (see Figure 13-8 and Figure 13-9) .

When a write is issued to the source for the first time after session activation, original data at that address is copied to the target. After this operation, the new data is updated on the source. This ensures that original data at the point-in-time of activation is preserved on the target. This is illustrated in Figure 13-7.

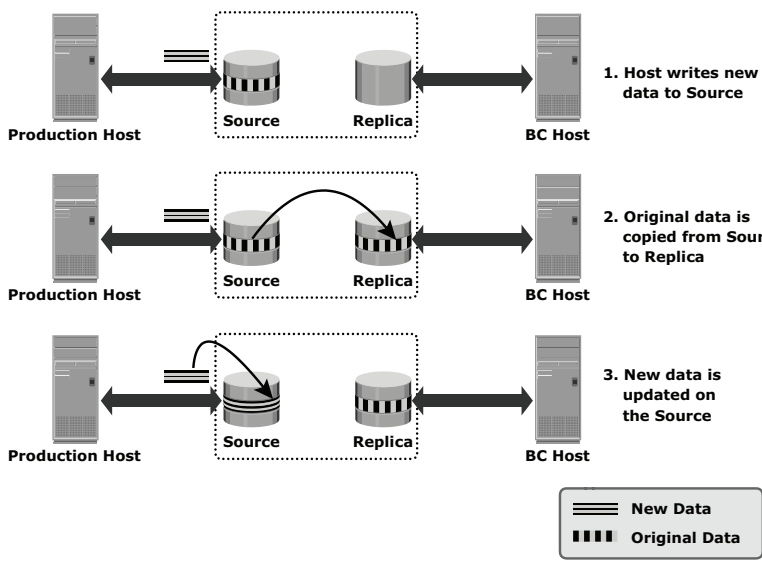


Figure 13-7: Copy on first access (CoFA) — write to source

When a read is issued to the target for the first time after session activation, the original data is copied from the source to the target and is made available to the host. This is illustrated in Figure 13-8.

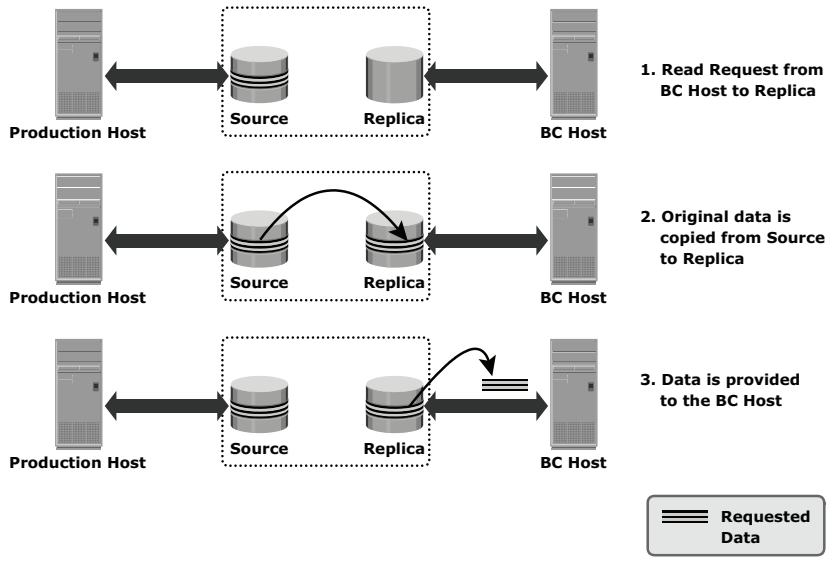


Figure 13-8: Copy on first access (CoFA) — read from target

When a write is issued to the target for the first time after session activation, the original data is copied from the source to the target. After this, the new data is updated on the target. This is illustrated in Figure 13-9.

In all cases, the protection bit for that block is reset to indicate that the original data has been copied over to the target. The pointer to the source data can now be discarded. Subsequent writes to the same data block on the source, and reads or writes to the same data blocks on the target, do not trigger a copy operation (and hence are termed Copy on First Access).

If the replication session is terminated, then the target device only has the data that was accessed until the termination, not the entire contents of the source at the point-in-time. In this case, the data on the target cannot be used for a restore, as it is not a full replica of the source.

In Full Copy mode, all data from the source is copied to the target in the background. Data is copied regardless of access. If access to a block that has not yet been copied is required, this block is preferentially copied to the target. In a complete cycle of the Full Copy mode, all data from the source is copied to the target. If the replication session is terminated now, the target will contain all the original data from the source at the point-in-time of activation. This makes the target a viable copy for recovery, restore, or other business continuity operations.

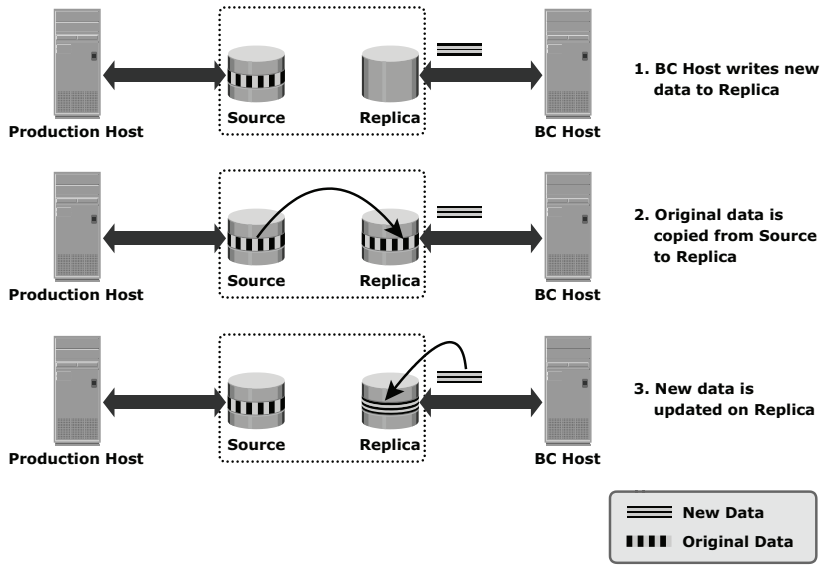


Figure 13-9: Copy on first access (CoFA) — write to target

The key difference between pointer-based, Full Copy mode and full-volume mirroring is that the target is immediately accessible on session activation in Full Copy mode. In contrast, one has to wait for synchronization and detachment to access the target in full-volume mirroring.

Both the full-volume mirroring and pointer-based full-volume replication technologies require the target devices to be at least as large as the source devices. In addition, full-volume mirroring and pointer-based full-volume replication in Full Copy mode can provide incremental resynchronization or restore capability.

Pointer-Based Virtual Replication

In *pointer-based virtual replication*, at the time of session activation, the target contains pointers to the location of data on the source. The target does not contain data, at any time. Hence, the target is known as a *virtual replica*. Similar to pointer-based full-volume replication, a protection bitmap is created for all data on the source device, and the target is immediately accessible. Granularity can range from 512 byte blocks to 64 KB blocks or greater.

When a write is issued to the source for the first time after session activation, original data at that address is copied to a predefined area in the array. This area is generally termed the *save location*. The pointer in the target is updated to point to this data address in the save location. After this, the new write is updated on the source. This process is illustrated in Figure 13-10.

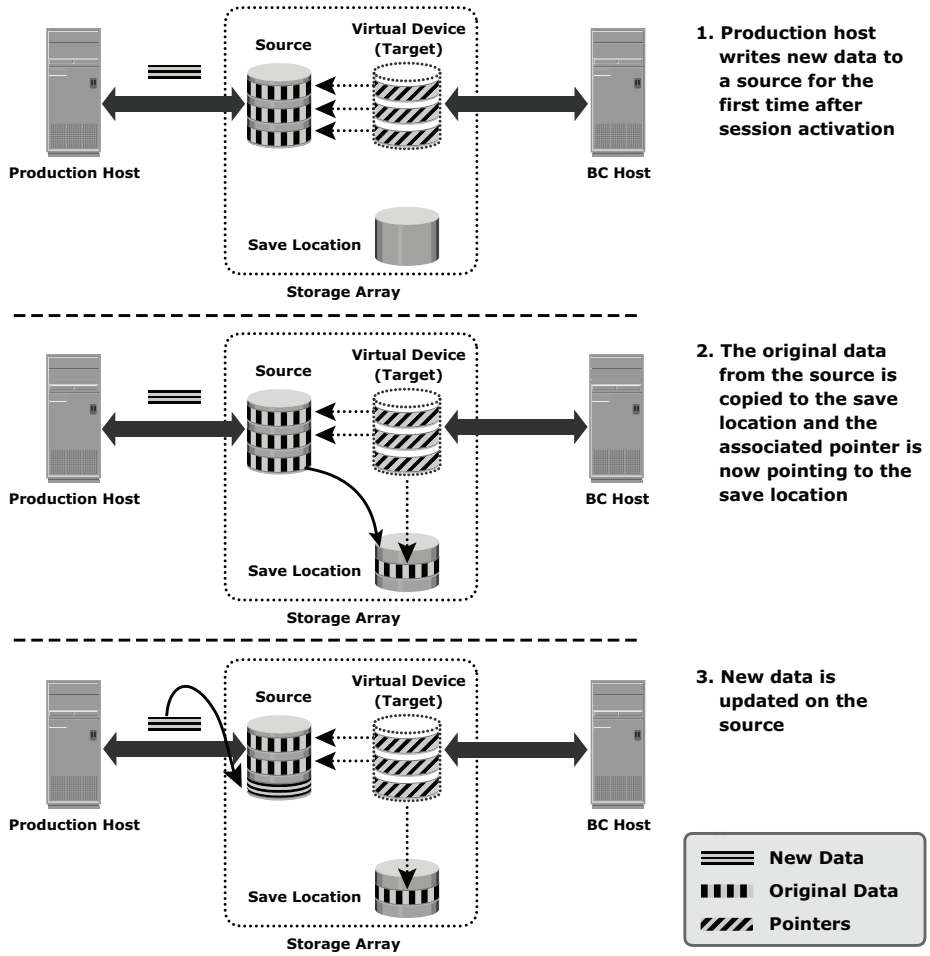


Figure 13-10: Pointer-based virtual replication — write to source

When a write is issued to the target for the first time after session activation, original data is copied from the source to the save location and similarly the pointer is updated to data in save location. Another copy of the original data is created in the save location before the new write is updated on the save location. This process is illustrated in Figure 13-11.

When reads are issued to the target, unchanged data blocks since session activation are read from the source. Original data blocks that have changed are read from the save location.

Pointer-based virtual replication uses CoFW technology. Subsequent writes to the same data block on the source or the target do not trigger a copy operation.

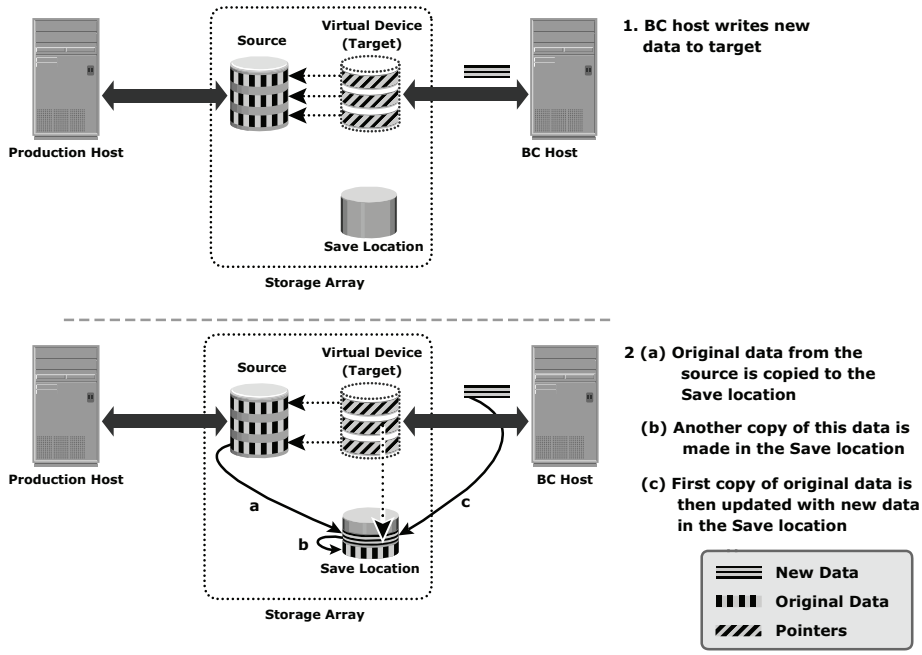


Figure 13-11: Pointer-based virtual replication — write to target

Data on the target is a combined view of unchanged data on the source and data on the save location. Unavailability of the source device invalidates the data on the target. As the target only contains pointers to data, the physical capacity required for the target is a fraction of the source device. The capacity required for the save location depends on the amount of expected data change.

13.5 Restore and Restart Considerations

Local replicas can be used to restore data to production devices. Alternatively, applications can be restarted using the consistent point-in-time copy of the data on the replicas.

A replica can be used to restore data to the production devices in the event of logical corruption of production devices — i.e., the devices are available but the data on them is invalid. Examples of logical corruption include accidental deletion of information (tables or entries in a database), incorrect data entry, and incorrect updating to existing information. Restore operations from a replica are incremental and provide a very small RTO. In some instances, applications can be resumed on the production devices prior to completion of the data copy. Prior to the restore operation, access to production and replica devices should be stopped.

Production devices may also become unavailable due to physical failures, such as production server or physical drive failure. In this case, applications

can be restarted using data on the latest replica. If the production server fails, once the issue has been resolved, the latest information from the replica devices can be restored back to the production devices. If the production device(s) fail, applications can continue to run on replica devices. A new PIT copy of the replica devices can be created or the latest information from the replica devices can be restored to a new set of production devices. Prior to restarting applications using the replica devices, access to the replica devices should be stopped. As a protection against further failures, a "Gold Copy" (another copy of replica device) of the replica device should be created to preserve a copy of data in the event of failure or corruption of the replica devices.

Full-volume replicas (both full-volume mirrors and pointer-based in Full Copy mode) can be restored to the original source devices or to a new set of source devices. Restores to the original source devices can be incremental, but restores to a new set of devices are a full-volume copy operation.

In pointer-based virtual and pointer-based full-volume replication in CoFA mode, access to data on the replica is dependent on the health and accessibility of the original source volumes. If the original source volume is inaccessible for any reason, these replicas cannot be used for a restore or a restart.

Table 13-1 presents a comparative analysis of the various storage array–based replication technologies.

Table 13-1: Comparison of Local Replication Technologies

FACTOR	FULL-VOLUME MIRRORING	POINTER-BASED, FULL-VOLUME REPLICATION	POINTER-BASED VIRTUAL REPLICATION
Performance impact on source	No impact	CoFA mode - some impact Full copy - no impact	High impact
Size of target	At least the same as the source	At least the same as the source	Small fraction of the source
Accessibility of source for restoration	Not required	CoFA mode - required Full copy - not required	Required
Accessibility to target	Only after synchronization and detachment from the source	Immediately accessible	Immediately accessible

13.5.1 Tracking Changes to Source and Target

Updates occur on the source device after the creation of point-in-time local replicas. If the primary purpose of local replication is to have a viable point-in-time copy for data recovery or restore operations, then the target devices

should not be modified. Changes can occur on the target device if it is used for non-BC operations. To enable incremental resynchronization or restore operations, changes to both the source and target devices after the point-in-time can be tracked. This is typically done using bitmaps, with one bit per block of data. The block sizes can range from 512 bytes to 64 KB or greater. For example, if the block size is 32 KB, then a 1 GB device would require 32,768 bits. The size of the bitmap would be 4 KB. If any or all of a 32 KB block is changed, the corresponding bit in the bitmap is flagged. If the block size is reduced for tracking purposes, then the bitmap size increases correspondingly.

The bits in the source and target bitmaps are all set to 0 (zero) when the replica is created. Any changes to the source or target are then flagged by setting the appropriate bits to 1 in the bitmap. When resynchronization or a restore is required, a *logical OR* operation between the source bitmap and the target bitmap is performed. The bitmap resulting from this operation (see Figure 13-12) references all blocks that have been modified in either the source or the target. This enables an optimized resynchronization or a restore operation, as it eliminates the need to copy all the blocks between the source and the target. The direction of data movement depends on whether a resynchronization or a restore operation is performed.

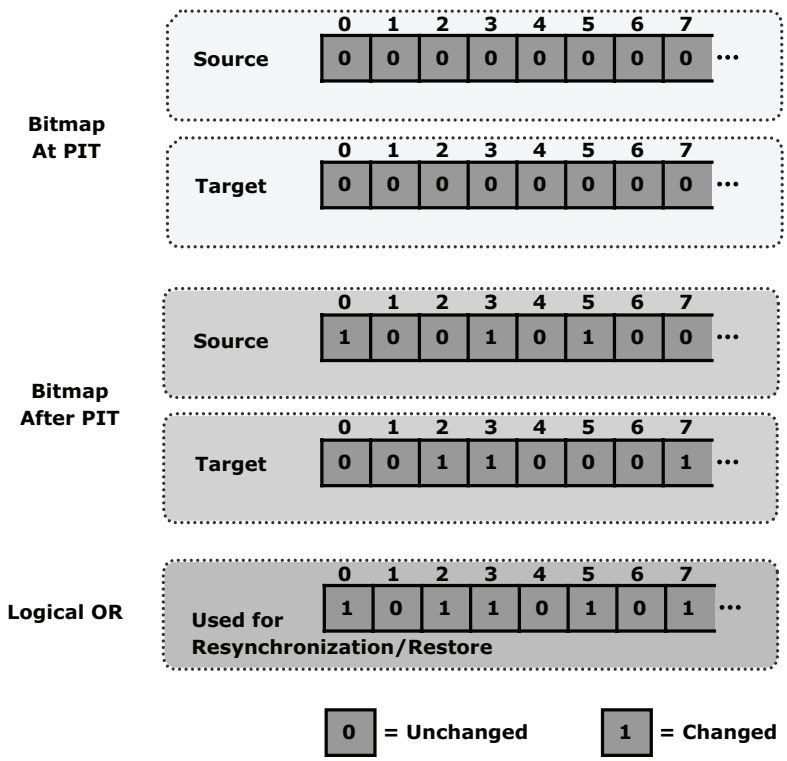


Figure 13-12: Tracking changes

If resynchronization is required, then changes to the target are overwritten with the corresponding blocks from the source. In this example, that would be blocks 3, 4, and 8 on the target (from the left).

If a restore is required, then changes to the source are overwritten with the corresponding blocks from the target. In this example, that would be blocks 1, 4, and 6 on the source. In either case, changes to both the source and the target cannot be simultaneously preserved.

13.6 Creating Multiple Replicas

Most storage array-based replication technologies enable source devices to maintain replication relationships with multiple targets. Changes made to the source and each of the targets can be tracked. This enables incremental resynchronization of the targets. Each PIT copy can be used for different BC activities and as a restore point.

Figure 13-13 shows an example in which a copy is created every six hours from the same source.

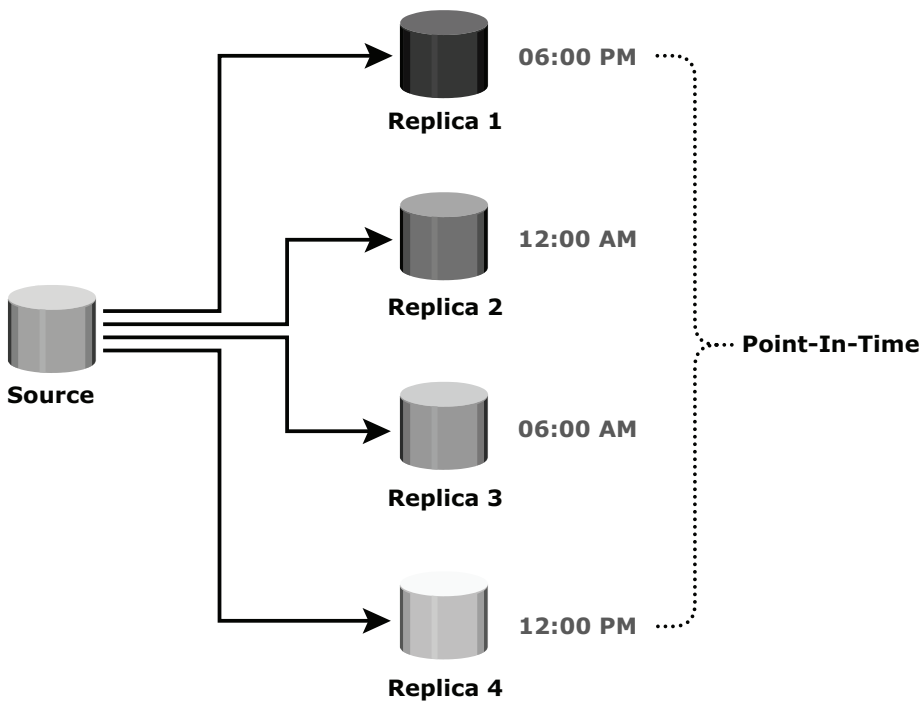


Figure 13-13: Multiple replicas created at different points in time

If the source is corrupted, the data can be restored from the latest PIT copy. The maximum RPO in the example shown in Figure 13-15 is six hours. More frequent replicas will reduce the RPO and the RTO.

Array local replication technologies also enable the creation of multiple *concurrent* PIT replicas. In this case, all replicas will contain identical data. One or more of the replicas can be set aside for restore or recovery operations. Decision support activities can be performed using the other replicas.

13.7 Management Interface

The replication management software residing on the storage array provides an interface for smooth and error-free replication. This management software interface provides options for synchronization, resynchronization, splitting, starting and stopping a replication session, and monitoring replication performance. In general, two types of interface are provided:

- **CLI:** An administrator can directly enter commands against a prompt in a CLI. A user can also perform some command operations based on assigned privileges. CLI scripts are developed to perform specific BC operations in a database or application environment.
- **GUI:** This type of interface includes the toolbar and menu bar options. A user can select an operation name and monitor performance in real time. Most of the tools have a browser-based GUI to transfer commands and are often integrated with other storage management suite products.

13.8 Concepts in Practice: EMC TimeFinder and EMC SnapView

EMC has several array-based specialized local replication softwares for different storage arrays. For Symmetrix array, the "EMC TimeFinder" family of products is used for full volume and pointer-based local replication. EMC SnapView and SnapSure are the solutions for CLARiiON and Celera storage arrays respectively.

TimeFinder family of product consists of two base solutions and four add-on solutions. The base solutions are TimeFinder/Clone and TimeFinder/Snap. The add-on solutions are TimeFinder/Mirror, TimeFinder/Consistency Groups,

TimeFinder/Exchange Integration Module, and TimeFinder/SQL Integration Module. Visit `http://education.EMC.com/ismbook` for the latest information.

TimeFinder is available for both open systems and mainframe. The base solutions support the different storage array–based local replication technologies discussed in this chapter. The add-on solutions are customizations of the replicas for specific application or database environments.

13.8.1 TimeFinder/Clone

TimeFinder/Clone is a point-in-time image of the full source volume in open systems that can be used for backups, decision support, data warehouse refreshes, or any other process that requires parallel access to production information. TimeFinder/Clone also includes a Mainframe SNAP facility that enables the creation of full-volume copies or dataset-level copies.

Clones can be protected using any type of supported protection scheme. Clones are available immediately for read and write access. TimeFinder/Clone also includes a "no copy" option, which enables performing a copy process only when the actual data is requested. As with all TimeFinder family of products, TimeFinder/Clone also supports the TimeFinder/Consistency Groups option to ensure data consistency between volumes and even across Symmetrix systems. TimeFinder/Clone is an ideal solution when high performance, RAID 5 protection, and highly functional point-in-time copies are required.

Clone Operation

A full-volume copy can be created by copying all data from the source as a background operation. When the copy is activated, data begins copying in the background. Track-by-track copying makes the target a clone of the source volume. While background copying, the state of the device pair is "CopyInProgress." When the operation completes, the state becomes "Copied." The copy session must be activated before the target host can access the data from the target volume. The Pre-Copy function starts copying tracks in the background, before the copy session is activated.

13.8.2 TimeFinder/Mirror

TimeFinder/Mirror performs local replication using full-volume mirroring. TimeFinder/Mirror uses special Symmetrix logical devices called *Business Continuance Volumes (BCVs)* for local replication. BCVs are dedicated devices for local replication and can be attached dynamically and nondisruptively with a *standard device*. All BCVs and their corresponding standard devices are of the same size and format.

TimeFinder/Mirror Operations

TimeFinder/Mirror performs four basic BCV operations — *Establish*, *Split*, *Restore*, and *Reestablish* — to preserve the mirror relationship between a standard device and a BCV:

- **Establishment of BCV pairs:** The TimeFinder establish operation is the first step in creating a TimeFinder/Mirror replica. The purpose of the establish operation is to synchronize the contents from the standard device to the BCV (see Figure 13-14[a]). The first time a BCV is established with a standard device, a full synchronization has to be performed. Any future resynchronization can be incremental in nature. The establish operation is a nondisruptive operation to the standard device. I/O to standard devices can proceed during an establish. However, all I/Os to the BCV device must be stopped before the establish operation is performed.

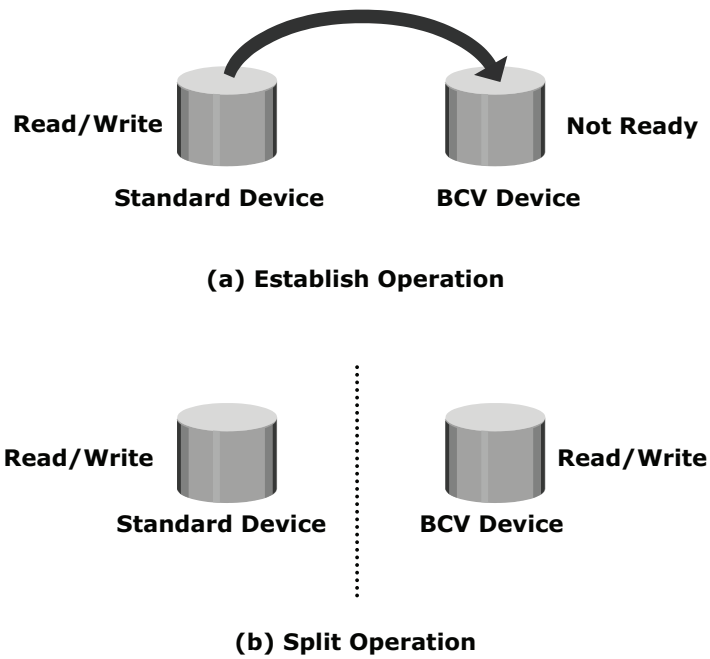


Figure 13-14: TimeFinder operation

- **Split of BCV pairs:** The replica is associated with the time when the split operation is executed. The split operation detaches the BCV from the standard device and makes the BCV device available for host access through its own device address (see Figure 13-14[b]). After the split operation, changes made to the standard or BCV devices are tracked by the

Symmetrix Enginuity Operating Environment. EMC TimeFinder/Mirror ensures consistency of data on the BCV devices via the *consistent split* option, described next.

- **Consistent split of BCV pairs:** The TimeFinder/Mirror consistent split option ensures that the data on the BCV is consistent with the data on the standard device. Consistent split holds I/O across a group of devices using a single consistent split command; thus all the BCVs in the group are consistent PIT copies. It can be used to create a consistent PIT copy of an entire system, a database, or any associated set of volumes.

  For ensuring consistency I/O can be held with either the EMC PowerPath multipathing software at the host level or with the Symmetrix Enginuity Consistency Assist at the array level. With a PowerPath-based consistent split executed by the host performing the I/O operation, I/O is held at the host before the split.

  An Enginuity Consistency Assist (ECA) based consistent split can be executed by the host doing the I/O or by a control host in an environment containing distributed and/or related databases. I/O is held at the Symmetrix until the split operation is completed; therefore, ECA can be used to perform consistent splits on BCV pairs across multiple, heterogeneous hosts.

- **Restoration of BCV pairs:** The BCV can be used to restore data to standard devices. Restore is a recovery operation, so all I/O activity to the standard device needs to be stopped, and the device must be taken offline, just before this operation. The restore sets the BCV to a "Not Ready" state. Therefore, all I/O activity to the BCVs must be stopped before issuing the restore command. Operations on standard devices can resume as soon as the restore operation is initiated, even while synchronization of the standards from the BCV is still in progress. The query command is used to obtain the current status of standard or BCV volume pairs. This operation also provides options for full or incremental restores.

Note the important difference between TimeFinder/Clone and TimeFinder/Mirror: With the former, the target is immediately available after the copy session is activated, whereas in the latter, it is only available after splitting from the source.

13.8.3 EMC SnapView

SnapView is an EMC CLARiiON array-based local replication software that creates point-in-time views or point-in-time copies of logical volumes using SnapView snapshots and clones, respectively. Snapshots are pointer-based snaps that require only a fraction of the source disk space. Clones are full-volume copies that require disk space equal to the source.

SnapView Snapshot

A SnapView snapshot is not a full copy of the production volume; it is a logical view of the original production volume based on the time the snapshot was created. Snapshots are created in seconds, and can be retired when no longer needed. Up to eight read/write snapshots per source LUN can be created to suit the needs of multiple business processes. A snapshot "roll back" feature provides instant restore to the source volume. The key terminologies of SnapView snapshot are as follows:

- **SnapView session:** The SnapView snapshot mechanism is activated when a session is started, and deactivated when a session is stopped. A snapshot appears "offline" until there is an active session. It is an exact copy of the source LUN when a session starts. Multiple snapshots can be included in a session. The source LUN can be involved in up to eight SnapView sessions at any time.
- **Reserved LUN pool:** This is a private area, also called a save area, used to contain Copy on First Write (CoFW) data. The "Reserved" part of the name refers to the fact that the LUNs are reserved and therefore cannot be assigned to a host.

To keep the number of pointers and the pointer map at a reasonable size, SnapView divides the source LUN into data chunks of 64 KB. Any changes to data inside a chunk cause that chunk to be written to the Reserved LUN pool, if it is being modified for the first time. The total number of reserved LUNs is limited and model dependent. To create snapshots on the source LUN, SnapView sessions need to be started on these LUNs and the Reserved LUN pool should be assigned to the source LUN. Once a session starts, the SnapView mechanism tracks changes to the source LUN.

13.8.4 EMC SnapSure

EMC SnapSure is an EMC Celerra Network Server software feature that enables the creation and management of checkpoints, which are point-in-time, logical images of a production file system (PFS). SnapSure uses a "copy on first modify" principle. A PFS consists of blocks. When a block within the PFS is modified, a copy containing the block's original contents is saved to a separate volume called the SavVol. Subsequent changes made to the same block in the PFS are not copied into the SavVol. The original blocks from the PFS in the SavVol and the unchanged PFS blocks remaining in the PFS are read by SnapSure according to a bitmap and blockmap data-tracking structure. These blocks combine to provide a complete point-in-time image called a checkpoint. A checkpoint reflects the state of a PFS at the point of time it is created.

TimeFinder/FS

TimeFinder/FS is an implementation of EMC TimeFinder technology. It is specifically tailored for the Celerra Network Server attached to a Symmetrix storage array. TimeFinder/FS enables the creation of independently addressable, physical, point-in-time copies (snapshots) of a PFS and provides "mirroring" existing snapshot to resynchronize it (and maintain its synchronization) with the PFS, so that an exact mirror of the PFS is always available. The process of mirroring on an existing snapshot is less time consuming than creating a new snapshot. A snapshot in mirror mode can be "mirror off" and an updated snapshot is available immediately for mounting and exporting.

Summary

There are an immense number of uses for a local replica in both data center production and BC operations. This technology has become an integral part of day-to-day operations. Replication eliminates the backup window and provides a quick resource to ensure protection against data corruption during major updates to the source data.

This chapter took a detailed look at the local replication process and described the uses of a local replica. Local replication can be accomplished using various technologies, such as host-based local replication and storage array-based local replication. This chapter also described the restore operations for storage array-based local replication, as well as the creation and use of multiple replicas. The planning and design considerations for establishing viable local replication processes, and implementation examples, were also provided in this chapter.

Though duplication of data with a local replica ensures high availability, dispersal of the duplicates to different sites is a way to ensure continuous operation for data centers in the event of a disaster that could incapacitate the entire site. Establishing the replicas at the remote site with replication has now emerged as a mature technology. Remote replication is covered in detail in the next chapter.

EXERCISES

1. What is the importance of recoverability and consistency in local replication?
2. Describe the uses of a local replica in various business operations.
3. What are the considerations for performing backup from a local replica?
4. What is the difference between a restore operation and a resynchronization operation with local replicas? Explain with examples.
5. A 300 GB database needs two local replicas for reporting and backup. There are constraints in provisioning full capacity for the replicas. It has been determined that the database has been configured on 15 disks, and the daily rate of change in the database is approximately 25 percent. You need to configure two pointer-based replicas for the database. Describe how much capacity you would allocate for these replicas and how many save volumes you would configure.
6. For the same database described in Question 5, discuss the advantages of configuring full-volume mirroring if there are no constraints on capacity.
7. An administrator configures six pointer based virtual replica of a LUN and creates eight full volume replica of the same LUN. The administrator then creates four pointer based virtual replica for each full volume replica that was created. How many usable replicas are now available?