the Availability Digest

Hardware Replication

January 2007

A fundamental premise of active/active systems is that the database copies in the application network must be kept in synchronism. In this article, we talk about database synchronization using hardware data replication. In our two previous articles, we talked about using software-based data replication to synchronize database copies. Before proceeding with a description of hardware replication, it is useful to briefly review those software replication techniques.

A Review of Software Replication

Creating a clone of a database in real-time is useful for many reasons, including

- creating a backup database for recovery purposes in the event of a system failure.
- providing query copies to support local access and to offload the primary system.
- providing active/active processing in which all database copies are actively being used.

Creating a database clone requires that updates made to a source database be *replicated* to one or more target databases. In previous articles, we have discussed software-based asynchronous replication[1] and synchronous replication[2] techniques to provide this function.

These replication techniques replicate specific changes made to a source database and maintain the sequence of these changes as they are applied to the target databases.[3] Transaction boundaries are maintained, as is the order of the events within the transactions.[4] Therefore, the target databases are always consistent and satisfy the requirements for referential integrity. As such, they are suitable for query and report processing and can be used in active/active architectures, in which all database copies are being actively updated by different instances of a common application.

Asynchronous replication is decoupled from the application and therefore imposes no performance penalty on an application. However, the inherent time from when an update is made to a source database to the time that the update is applied to a target database (the replication latency) opens up the possibility of data loss following a source system failure. Equally of concern is that data collisions can occur when the same data item is updated at roughly the same time in two different database copies.

[1] Asynchronous Replication Engines, the *Availability Digest*; Volume 1, Issue 2; November, 2006.
[2] Synchronous Replication, the *Availability Digest*; Volume 1, Issue 3; December, 2006.
[3] See Chapter 10, Referential Integrity, *Breaking the Availability Barrier; Survivable Systems for Enterprise Computing*, by Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein.
[4] Some replication engines even maintain the inter-threaded order of simultaneous source transaction events during the transaction replay at the target. Although this strict ordering is not required for target database consistency, it does improve the overall target system resource leveling to match the source's performance profile.

Synchronous replication solves these problems but introduces one of its own. Since an application must wait for a transaction to be committed across the network, its response time is increased (the application latency).

There are other techniques for data replication. One is replication at the hardware level – the topic of this article.[5]

Hardware Replication

Hardware replication is in common use today in many systems. It differs from software replication in that it replicates data at the physical block or sector level rather than at the logical transaction level. Though it may usually be implemented as software somewhere in the I/O stack, it is referred to as hardware replication because it happens not only beneath the application layer (where software replication happens) but even below the operating system layer.

Basically, whenever a disk block is written to the source disk, it is written immediately to the target disk. Therefore, the two disks are in near-real-time synchronism. By using synchronous hardware replication (discussed below), the disks can be kept in exact synchronism.

Hardware replication is generally much faster than software replication as no updates are made to the target database. Rather, only full block writes are made. In addition, if replication is done at the disk subsystem level (that is, at the physical layer – see the discussion below concerning the I/O stack), there is no processing load imposed for replication.

The Database Consistency Issue

A fundamental hardware replication characteristic that must be understood is that it is the disks that are synchronized, not the database copies. If updates are cached, the up-to-date copy of the database is reflected in the contents of the cache, not the contents of the disk.

Data blocks are only flushed from cache to disk when some algorithm so dictates. For instance, if a data block is to be read from disk, room must be made in the cache for the new block. To create this space, the oldest data block might be flushed to disk (if it has been modified since it was read or last flushed). This is the *least recently used (LRU)* algorithm.

Another case requiring flushing is a transaction commit. Typically, before a transaction can be committed, the transaction change log must be flushed to disk. However, data blocks affected by the transaction can remain in cache until they need to be flushed by an LRU algorithm.

Therefore, it is only the contents of cache that reflect the current state of the database. Since the contents of the source cache are not being replicated, the target database is not consistent. Index entries may exist with no associated data blocks. Child data items may not have a parent. Block splits may be incomplete.

As a result, there is no concept of referential integrity and database consistency in a database maintained by hardware replication. Therefore, while replication is active, the target database is not suitable for use for functions such as query or report processing. It is definitely not useful in

[5] The various forms of replication are analyzed in a great deal more detail in the forthcoming book, *Breaking the Availability Barrier: Achieving Century Uptimes with Active/Active Systems.*

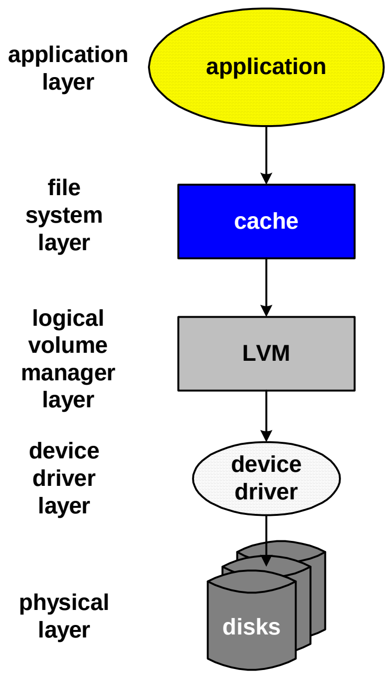
active/active architectures since every database copy must be consistent so that they can all actively participate in a common application.[6]

A target database can be made useful only by pausing the application, waiting until the target database has become synchronized, and then turning off replication. This technique can be used to create a consistent database copy that can then be moved to tape in an offline backup procedure.

However, hardware replication is quite suitable for creating a backup system that is almost ready to take over from the primary system in the event of a primary failure. “Almost ready” is meant to mean that the target database must first be recovered and made consistent. Because of certain characteristics of hardware replication, this can be a complex and lengthy process. We explain this later.

The I/O Stack

Hardware replication can occur at any one of several places in the I/O stack.[7]



Application Layer

This layer corresponds to the application stack running on a server. It includes not only the applications but also the database manager, if any. It is the database manager that provides transaction services. Replication products that run at this layer are the transaction-oriented software replication engines described in our previous articles.

File System Layer

The file system provides the interface between the logical I/O requests of the application and the block I/O addressing of the disk storage system. It typically caches data blocks until they are ready to be written to disk. There are several software products that offer data block replication at this layer.

Logical Volume Manager Layer

The LVM layer is a software layer that manages the disk devices and decides to which physical storage device a request should be routed. It is an optional layer and may not exist in a particular I/O stack. Various software products provide data block replication at this layer.

Device Driver Layer

Device drivers are software components utilized by the operating system to interact with the physical device. It is less common for this layer to provide hardware replication.

[6] Of course, if cache is turned off, every write will immediately be reflected in the target database; and the target database will be consistent. However, this practice is generally not employed for performance reasons.
[7] Selim Daoud, Hardware Replication Challenges, Sun BluePrints OnLine; November, 2003.

Physical Layer

The physical layer includes all of the storage hardware such as disk, RAID arrays, or SAN switches. Replication at this layer is handled by the storage device itself.

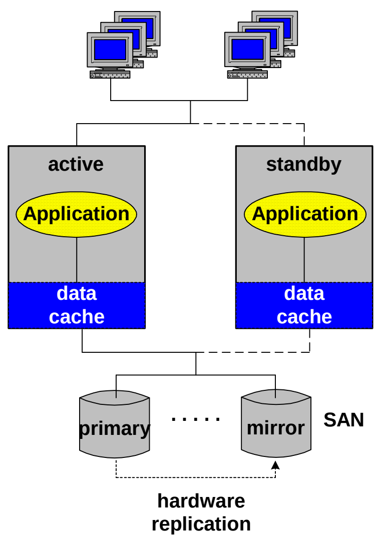
Some Examples

The particular level in the I/O stack at which hardware replication is implemented depends upon the system architecture as shown in the following two examples:

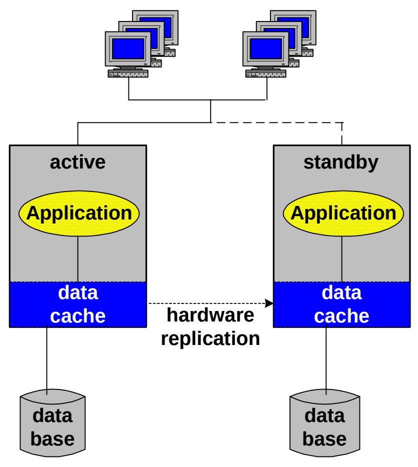
Storage Area Networks

A storage area network, or SAN, is isolated from the servers which use it by a communication network. The SAN is a freestanding system that handles storage requests from any server connected to it over the network.

Therefore, there is no full I/O stack running in a server that is associated with a SAN. Hardware replication is a function of the SAN and is typically implemented at the physical layer. A primary disk subsystem and a backup disk subsystem are provided. The primary disk arrays in the SAN replicate data blocks to the backup storage devices as data blocks are written to them.



Storage Area Network



Direct Attached Storage

Direct Attached Storage

On the other hand, if direct attached storage is used, there is an I/O stack associated with each set of storage devices. In this configuration, a primary server system and a backup server system are provided, each with its own data storage.

Though hardware replication could be done at the physical level in this case, it is more common to provide replication at the file system layer. It is this layer which maintains the storage cache. Whenever a data block is flushed from the cache of the primary system, it is replicated to the backup system's file layer, which writes that data block into its own storage system.

Asynchronous and Synchronous Replication

Hardware replication can either be asynchronous or synchronous. With asynchronous replication, as soon as a data block is flushed to disk on the source system, it is sent to the target system to be applied to the target disk. The source system immediately continues processing. There is no impact on the application.

With synchronous replication, as soon as a data block on the source system is flushed to disk, the data block is also sent to the target system to be written to disk. However, in this case, the source system's write is not complete until the target system has responded that it has successfully received and safe-stored the data block or has written it to disk.

Synchronous replication guarantees that the two disks are always exactly synchronized (though neither contains a consistent database, as explained previously). However, as with software-based synchronous replication engines, this is done at the expense of performance. If the application is awaiting a write completion, it will be delayed by network latency as the replication write propagates to the target system and as the target system's response is returned.

As a consequence, there is generally a distance limitation between the source and target disks if synchronous hardware replication is to be used.

There is also a compromise form of synchronous hardware replication. This can be thought of as semisynchronous replication. With this technique, when a write to disk occurs, the block is replicated asynchronously; but the next block's write will not complete until the replication of the previous block has completed.

Recovery

We indicated earlier that recovery from a system failure can be complex and lengthy when using hardware replication. This is because data blocks are not written to the disk in the same order that they are written to the database due to caching (unless caching is turned off).

Following a failure, there is no way to know how old the data on disk is. If there is a transaction log, it will be reasonably up-to-date since no data can be committed until the transaction log has been flushed to disk. However, just because the transaction log has been flushed to disk, there is no way to know which, if any, of the referenced data blocks were flushed. Much of this data is still in the cache of the source system and is lost.

When attempting to recover these transactions from the transaction log, there is no way to know how far back in the log one must go in order to recover all of the data that has yet to be flushed. The result is a very complex and potentially very lengthy recovery process. This is in sharp contrast to software replication, which guarantees that all changes have been applied to the target database when a transaction is committed.

Advantages of Hardware Replication

There are several advantages associated with hardware replication.

- It can replicate any kind of data, such as transaction data and logs, configuration files, control files, source files, executables, and other unstructured data.
- It imposes a minimal load on the target database since write activity is full block writes (which is generally very efficient) rather than updates.
- There is no processor overhead if replication is done at the physical layer.

- Hardware replication can be very useful for providing a backup database in an active/backup configuration.

Disadvantages of Hardware Replication

Inherent in hardware replication are several disadvantages.

- The target database is inconsistent and cannot be used for any processing functions such as query processing or reporting.
- Even the data that is stored on the target database can be quite stale.
- The source and target systems must be identical. There can be no conversions to different formats, database structures, or differing databases.
- Should there be corruption in a source data block, this is replicated to the target system. This could create the inability to use either set of disk data.
- Recovery from a primary system failure can be quite lengthy, as described above.
- A greater load is imposed on the communication channel since entire blocks rather than simply changes are replicated.
- When using synchronous hardware replication, the systems must generally be fairly local to each other and be connected by a high-speed communication network since the communications latency can dramatically impact the source application's write performance.

As a result of database inconsistency, hardware replication is not appropriate for active/active systems in which each database copy must be current so that it can be used by local applications to participate in a common application.

Summary

Hardware replication is a commonly available replication technique for maintaining a backup database to protect against a system failure. Hardware replication products are offered by many vendors, including EMS, Legato, Sun, Veritas, and EVA.

However, hardware replication suffers from the fact that the target database is not consistent and cannot be used for processing functions. This precludes its use to synchronize databases in an active/active application network. Software replication, including asynchronous replication, synchronous replication, and transaction replication (to be discussed in our next article), must be used to support an active/active configuration.