Hardware vs Software Data Replication for Business Continuity

Keith Evans >> Shadowbase Product Management >> Gravic Inc.

Introduction

Today, businesses with access to real-time online transactional data have a competitive advantage. To gain the greatest benefit from this data it must be current and available at any given time. The counter to this advantage is that the inability to access or update current data denies service to users and carries significant business costs, possibly measured in many thousands of dollars per second. These requirements necessitate an IT infrastructure that is continuously available.

Business continuity encompasses activities that an enterprise performs to maintain timeliness, consistency, and availability of its data, operations, and services. Application availability depends upon the ability of IT services to survive any fault, whether it is a server failure, a network fault, or a data center disaster. Data availability depends on the existence of up-to-date backup data copies. *Data replication* is an enabling technology for achieving high or continuous availability for application services and the timely backup of important data. There are two primary data replication technologies, hardware replication and software replication. Each of these technologies, and the differences between them, are discussed in this article.

Data Replication – The Fundamental Force Behind Business Continuity

Improving availability via data replication depends upon having at least two nodes (or disks), each capable of hosting data. As shown in Figure 1, the purpose of data replication is to keep target data synchronized in real-time with source data that is being updated by a source application.

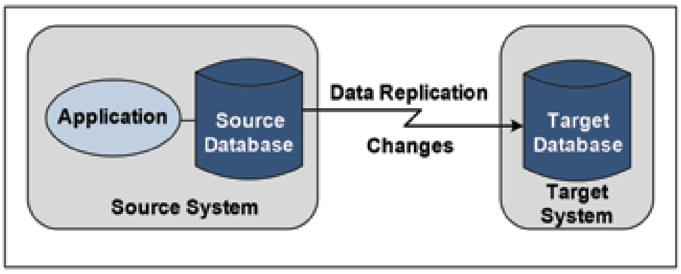

Figure 1 – Data Replication

The *source database* is hosted by the source node and the *target database* is hosted by the *target node*. The two nodes comprise the distributed data processing system. As an application makes changes (inserts, updates, and deletes) to its local source database, these changes are immediately sent by some means to the target system, where they are applied to the target database, which typically resides on another independent node. Because the target database is kept synchronized with the source database, if the source system becomes unavailable, processing can continue using the target system, maintaining service availability. The means by which the data is replicated between the disks on the source system and the disks on the target system falls into one of two categories, hardware replication or software replication.

Hardware Data Replication

Hardware replication is usually implemented in the storage system controller, which replicates disk blocks to a target disk as they are written to the source disk (Figure 2). If a failure occurs in the source system to which the source disk is attached, a backup system to which the target disk is attached can take over processing.

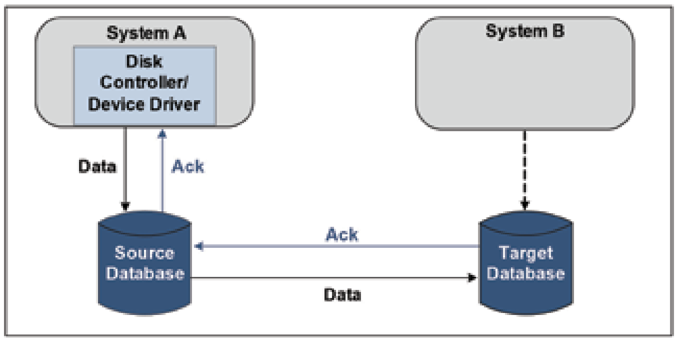

Figure 2 – Hardware Replication – On Disk Write

However, disk blocks are typically only written to disk when they are flushed from the disk's cache. There is no logical order to the disk-write sequence since other factors control cache flushing. The recent least used disk blocks are flushed to disk when cache space is needed for new blocks that must be read from disk. As a consequence: the target disk is not guaranteed to be consistent with the source disk; target disk blocks may be partially split; indices may exist without the rows or records to which they refer; and children may exist without parents. The data is consistent in cache, but the target disk image is generally useless. As a result, applications cannot use the target database for any application processing. Because of this inconsistency, if the source node fails, a lengthy recovery process is required to bring the target database into a useful, consistent state, which extends the period of service unavailability. Additionally, if synchronous replication is used, large amounts of data may be lost due to a source-system failure, as any data still in cache will not have been flushed nor replicated at the time of failure.

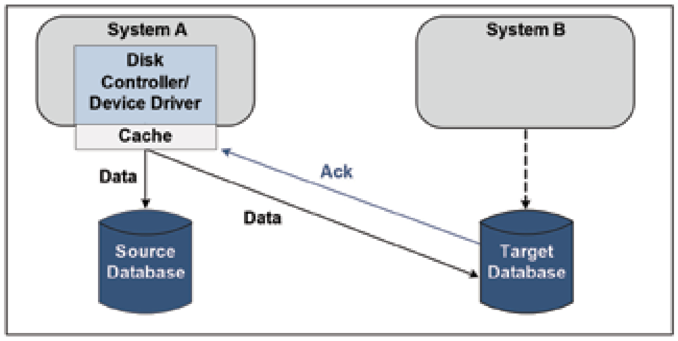

Figure 3 – Hardware Replication – On Cache Write

Some storage controllers replicate changes as they are made to a disk's cache regardless of whether or not they have been physically written to the source disk (Figure 3). The replication of cache updates ensures the logical consistency of the target database, since changes are replicated to the target system as soon as they are made at the source system. If synchronous replication is used, no data will be lost following a source system failure. However, due to other limitations (as described next), the target database may still not be useable by applications while replication is occurring.

Whether based on disk flushing or cache updates, hardware replication typically sends blocks of changes to the target. In some cases, the controller compresses data to only the changed bytes. In other cases, entire data blocks are sent, which requires high communication bandwidth and co-location of source and target disks. Both hardware replication techniques typically do not replicate current data locking protocols nor transaction end-state information (commits and aborts). Hence, the target database typically contains many "dirty records" and cannot be used by applications, even for read-only activities.

Hardware replication generally requires identical storage technology, including the version level, to be used at both the source and the target. This requirement means it cannot be used to integrate diverse systems and applications, to eliminate islands of information and implement new business services. Further, if any component is required to be upgraded (for example, to fix a fault), all components must be upgraded at the same time (or else the fault would bring down both components).

Hardware replication also typically does not allow the target database to be opened by applications at the same time that replication is taking place, thus preventing the use of hardware replication for active/active systems[1], or for enabling read-only activities to be performed on a backup system. Consequently, hardware replication is not an option in order to achieve recovery times measured in seconds or minutes, or for maximizing system utilization.

Another issue with hardware replication is that the maximum distance between the source and target disks is limited. Having the target disk insufficiently distant from the source disk increases the chance that a local area incident (e.g., a flood) will affect both the source and target disks and prevent a timely recovery.

Software Data Replication

Software replication may take place by: event, transaction (several events all treated as a single unit of work), request, or *log-shipping* (which is discussed further in the following section):

- *Event replication* replicates data-manipulation language (DML) events as they occur. DML events include insert, update, and delete operations. In some cases, event replication may also replicate data-definition language (DDL) operations that affect the database's data structure and schema.
- *Transaction replication* replicates entire transactions, either one operation at a time as they occur, or as a group of operations once the transaction has committed on the source. When replayed at the target, the transaction is either committed or, if the entire transaction is not received, is aborted.
- *Request replication* replicates the entire application request, which is reprocessed in its entirety by the application running on the target system.

In this article we are concerned about achieving the highest levels of replication performance, application service availability, data consistency, and minimizing data loss when a failure occurs. Of the various modes of software replication, *transaction replication* best meets these requirements. In transaction-level software replication, a data replication engine running on the source and target systems performs the replication task. The *data replication engine* is driven by a queue of database change events which are read on the source system and sent to the target system and applied

[1] Active/active systems spread the processing load over multiple environments, allowing for instantaneous takeover when one of the systems or environments fails, and lead to the highest Recover Time Objective (RTO) attainable. Active/active architectures are referred to as continuously available and disaster tolerant, and achieve the best RTO possible when implemented with transactional software data replication. See the Gravic white paper, *Choosing a Business Continuity Solution to Meet Your Availability Requirements* (http://shadowbasesoftware.com/white-papers/) for more information.

to its database. This form of replication is able to replicate at the transaction level because transaction control information is also replicated, and updates are applied to the target as transactions. That is, either all updates in a transaction are applied, or none are, thereby preserving source transaction consistency at the target database. In addition, the updates are applied to the target system in the same order as they were generated on the source system. As a result, the target database can always satisfy all of the requirements of referential integrity and database consistency, and consequently can be used by other applications for both read and write operations (the latter in an active/active architecture).

Unlike hardware replication, software replication works between heterogeneous systems and databases. This capability enables the integration of diverse applications and data, elimination of islands of information, and implementation of new business services, such as real-time business intelligence.

As with hardware replication, software replication may be asynchronous or synchronous. An *asynchronous data replication* engine is completely transparent to the applications running in the source node. As shown in Figure 4, it extracts changes made to the source database from a database change queue[2] and sends them after-the-fact to the target database.

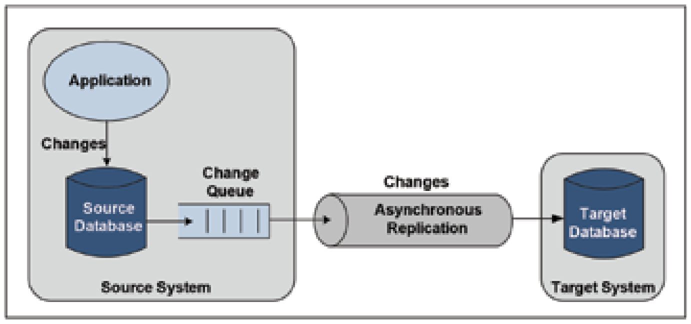

Figure 4 – Asynchronous Software Replication Engine

Unlike some forms of hardware replication (disk flushing), *synchronous software replication* guarantees that no data will be lost after a failure. Using a technique known as "coordinated commits"[3] (Figure 5), synchronous software replication makes no permanent changes to any database copy unless these changes can be applied to both source and target database copies. With coordinated commits, the replication engine participates in the source application's transaction, and at commit time, it votes "yes" or "no" dependent upon whether all the updates in the transaction have been replicated to the target system. If "no," then the source transaction aborts. It is guaranteed that all participating databases, source and target, received and/or applied the same updates, or none did; therefore no data will be lost in the event of a source system outage. Another major benefit of coordinated commit technology is that while it guarantees no committed data will be lost, impacts to application throughput are also minimized as synchronization only occurs at transaction commit time, and not on every database change event.

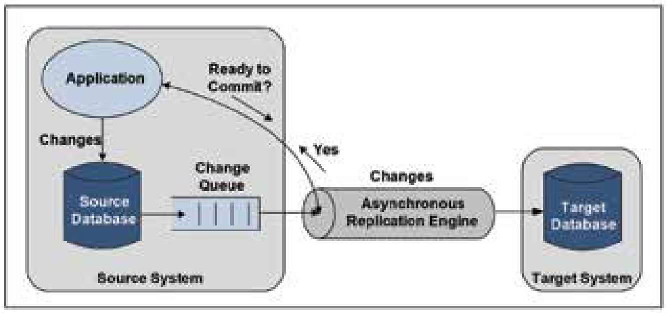

Figure 5 – Synchronous Software Replication using Coordinated Commits

Software Data Replication – Log-Shipping

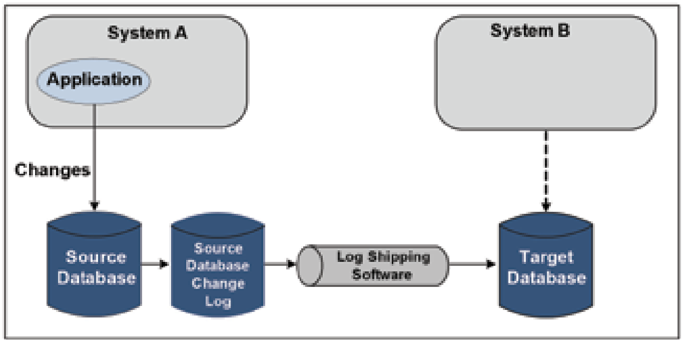

Figure 6 – Software Replication – Log-Shipping

Log-shipping (Figure 6) is a form of software data replication that operates more like hardware replication. Log-shipping sends the entire source database change log periodically to a target system. On the target system this change log is read and the data and index blocks are applied against the physical database structure, which is analogous to how hardware-based data replication works; it also has all of the same issues.

Compared to software transaction-based data replication, log-shipping has a higher amount of data loss on failure (a higher Recovery Point Objective, or RPO), because the target database is only current to the point of the most recent log-ship, which may only be when the source log file has been closed. Any subsequent changes on the source system are lost. Log-shipping data is usually applied to the target as index and data blocks of changes, which results in an inconsistent target database, rendering it practically useless while replication is taking place, as is the case with hardware replication. Even if the change events are extracted from the log and applied to the target, source transaction consistency is typically not maintained; events are applied without regard to source transaction bracketing. The result is an inconsistent and unusable target database.

Comparison of Hardware vs Software Data Replication

Although hardware replication (including software-based log-shipping) appears to offer a simple and cost-effective approach to maintaining data and service availability, it is often not a good business continuity solution for the reasons previously discussed. To summarize:

- Since there is no concept of transaction boundaries, database consistency or referential integrity, hardware replication cannot be used for active/active systems (i.e., it cannot provide continuous availability).

[2] A database change queue is a DBMS-maintained list of transactional insert, update, and delete operations that have been performed by applications against the database.
[3] For more information on synchronous replication and coordinated commits, see Dr. Bill Highleyman, Paul J. Holenstein, Dr. Bruce Holenstein, Chapter 4 – Synchronous Replication, Breaking the Availability Barrier: Survivable Systems for Enterprise Computing, AuthorHouse; 2004.

- The backup disk is highly inconsistent due to missing data not yet replicated, and is consequently not usable for query/ reporting or other functions.
- The primary and backup systems must generally use identical database hardware and software. All upgrades must be simultaneously applied to all hardware components, thereby increasing the risk that a fault in one component will affect all components.
- The source and target systems must be homogeneous, and integration of diverse systems, applications, and data is not possible.
- Recovery from a failure is a complex and lengthy task, requiring data "fix-up" on the backup disk, which leads to long recovery times and service unavailability.
- A significant amount of communication bandwidth is required since whole disk blocks rather than individual rows are typically replicated.
- Data corruption of the source database is replicated to the target, perhaps preventing it from being opened for recovery.
- Distance between disks is physically limited (typically about 100 km), which increases the likelihood that a local area incident could affect both disks and prevent recovery.
- If synchronous replication is used, data can still be lost when a failure occurs due to the cache flushing issue.

By contrast, none of these issues affect software replication:

- Primary and backup databases are consistent and maintain transactional and referential integrity. Software replication therefore enables active/active systems which deliver continuous service availability.
- Because software replication maintains database consistency, backup systems can be used for query/reporting and other online activities, while replication is taking place.
- Primary and backup systems can be completely different (heterogeneous). The platforms, operating systems, database software, and database structure can all be different. The data replication engine takes care of managing the necessary data transformations.
- Software replication can be used for the integration of different applications and data, enabling the implementation of new business services.
- Because the backup database is transactionally consistent and ready for use at any time, there is no need for a complex takeover process (database "fix-up" to bring it into a consistent and usable state), and recovery times as low as sub-seconds are possible.
- In an active/active architecture many users will not see an outage, and recovery is simply a matter of re-routing users from a failed node to an active system.
- Because only row change data is sent between systems, much less communications bandwidth is required for software replication.
- Software replication replicates changes described by a source transaction log. Such changes are executed completely independently on target systems, thereby avoiding the mirroring of corruption from the source database to the target database.
- Software replication has no physical distance limits between nodes, which can be positioned sufficiently far apart to ensure continued service, including an outage incident that affects a wide area.
- Synchronous software replication guarantees that all data associated with committed transactions is replicated, and hence no data will be lost in the event of failure.

Summary

On the surface, hardware data replication appears to offer a simple and cost-effective solution to the problem of maintaining data and service availability in the event of a system outage. Scratch the surface however, and it becomes clear that hardware data replication suffers from many significant issues which make it unsuitable for this task for mission-critical applications. Simply put, the likelihood of a timely recovery from an outage with minimal data loss is very low, and perhaps not possible at all. Furthermore, backup system capacity is wasted since the replicated data is inconsistent and unusable. It will only take one incident for it to become apparent that hardware replication is not cost-effective and does not enable the highest levels of service availability with minimal data loss. While suitable for some tasks, hardware data replication is inadequate for supporting business continuity of mission-critical applications.

Conversely, software-based transactional replication suffers from none of the issues which afflict hardware replication, including:

- distances are not limited;
- source and target databases are consistent;
- recovery is simple, fast, and repeatable;
- continuously available active/active architectures are supported;
- zero data loss is provided when running in synchronous mode;
- backup databases can be used for productive work;
- source and target systems can be completely heterogeneous.

For mission-critical applications, the highest levels of service availability and protection against data loss is required. A business continuity architecture built on software-based transactional data replication is the only viable solution to meet this requirement.

Shadowbase Software Data Replication

The Shadowbase product suite from Gravic, Inc. provides the full range of software data replication features to satisfy the most demanding IT business continuity and other replication requirements, including:

- active/active continuously available business continuity architectures to eliminate *unplanned downtime*;
- synchronous replication technology for zero data loss (ZDL) when disasters occur;
- zero downtime migration (ZDM) capability to eliminate *planned downtime* for upgrades and migrations;
- data and application integration between heterogeneous systems.

The Gravic white paper, *Choosing a Business Continuity Solution to Meet Your Availability Requirements* contains more information on the subject of hardware versus software data replication, and the requirements to consider in choosing the best solution to meet your business continuity needs.

Keith B. Evans works on Shadowbase business development and product management for Shadowbase synchronous replication products, a significant and unique differentiating technology. Asynchronous data replication suffers from certain limitations such as data loss when outages occur, and data collisions in an active/active architecture. Synchronous replication removes these limitations, resulting in zero data loss when outages occur, and no possibility of data collisions in an active/active environment. Shadowbase synchronous replication can therefore be used for the most demanding of mission-critical applications, where the costs associated with any amount of downtime or lost data cannot be tolerated. For more information and the availability of Shadowbase synchronous replication, please email sbproductmanagement@gravic.com.