"Achieving Century Uptimes"
An Informational Series on Enterprise Computing

As Seen in *The Connection*, An ITUG Publication
December 2006 – Present

About the Authors:

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein, have a combined experience of over 90 years in the implementation of fault-tolerant, highly available computing systems. This experience ranges from the early days of custom redundant systems to today's fault-tolerant offerings from HP (NonStop) and Stratus.

Gravic, Inc.
Shadowbase Products Group
17 General Warren Blvd.
Malvern, PA 19355
610-647-6250
www.ShadowbaseSoftware.com

Achieving Century Uptimes

Part 9: Where is My Database of Record?

March/April 2008

Dr. Bill Highleyman
Dr. Bruce Holenstein
Paul J. Holenstein

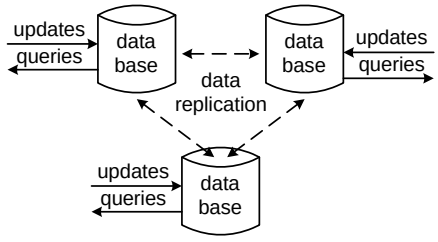
What is a Database of Record?

An important concept in today's complex corporate IT environments is the "single version of truth." Through corporate expansion, mergers, and acquisitions, companies often tend to end up with a hodgepodge of databases, many of them containing the same or similar data, such as names and addresses. The value of a data object may not necessarily be the same in the different databases. For instance, a customer may have moved; and the new address may have been put in one database but not in another. Which database is correct?

To solve this problem, companies often build a data warehouse or an operational data store (ODS) to hold all of their data. In this case, there is only one "version of the truth;" and this database is known as the database of record. Having a database of record is not only necessary for correct operations and good customer relations, but it is often mandated by regulation.

A common problem with the database of record is that it must be up and available to all applications in order for a company to provide the necessary IT services. Though the storage systems that hold the database of record are very reliable, they can and do fail. To protect against such failures, companies will often provide a backup storage system. However, in the event of a failure, there is a time interval during which data is not available (described by the RTO, or recovery time objective); and some data may be lost (described by the RPO, or recovery point objective).[1]

An active/active system largely solves these twin problems of data unavailability and data loss. However, in an active/active system, there are multiple copies of the database in the application network. This therefore begs the question as to which copy is the designated database of record. In this article, we provide practical answers to this question.



An Active/Active System

Active/Active Systems

An active/active system comprises multiple distributed processing nodes all cooperating in a common application.[2]

[1] Highleyman, W. H.; Holenstein, P. J.; Holenstein, B. D.; Chapter 6 – RPO and RTO, *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, AuthorHouse; 2004.

[2] Highleyman, W. H., What is Active/Active?, the *Availability Digest*, www.availabilitydigest.com; October, 2006.

Not only are there multiple processing nodes in the application network, but there also are two or more copies of the application database. Every user has access to two or more processing nodes via the communication network (this is necessary should one of the active nodes fail).

In many active/active systems, a user is assigned to a processing node; and that node processes all of his transactions. Should the user's node fail, his transactional activity is simply routed to another node in the network. In other systems, each transaction is routed to a node based on nodal loads or other parameters set by the network.

Thus, should a node fail, transactions are simply no longer routed to it. If the node had users assigned to it, those users are reassigned to surviving nodes. If the network is doing transaction routing, then transactions are no longer routed to the failed node.

The database copies in the application network must all contain the same data so that any application applied against any database will be correctly processed. To accomplish this, the databases are synchronized by data replication. Whenever a transaction makes changes to a database copy, those changes are immediately replicated to the other database copies in the application network. Therefore, all database copies will always be in the same state.

Asymmetric Configurations Using Asynchronous Replication

Asynchronous Replication

There are two primary types of data replication that can be used to keep application database copies synchronized – asynchronous replication and synchronous replication. Synchronous replication is discussed later.

With asynchronous replication,[3] changes made to a source database are captured by a replication engine and are sent over the network to the target system, where they are applied to the target database. Replication activity is totally transparent to the source system and to the application, and the target database is kept identical to the source database within a very small time interval.

The "very small time interval" is known as *replication latency*. It is the time from when a change is applied to the source system to the time that it is applied to the target system and is typically measured in the hundreds of milliseconds for efficient replication engines. Replication latency brings with it two significant problems:

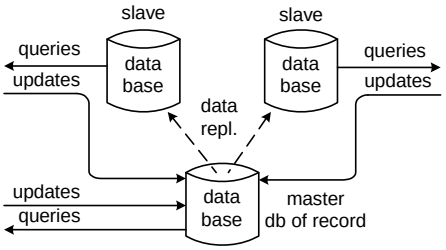
- *Loss of data* in the replication pipeline should the source system fail.
- *Data collisions* caused by changes made at separate database copies within the replication interval. These changes will be replicated and will overwrite each other unless otherwise accounted for.

[3] Highleyman, W. H.; Holenstein, P. J.; Holenstein, B. D.; Chapter 3 – Asynchronous Replication, *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, AuthorHouse; 2004.

There are in use several active/active configurations designed to eliminate or easily resolve data collisions when asynchronous replication is used. These configurations generally weight nodes differently so that there is a predominant node.[4] Therefore, these configurations are asymmetric. That is, the nodes are not all equal; and the database of record is that maintained by the predominant node.

Master/Slave

In one such configuration, one of the nodes in the application network is designated as the master node; and the other nodes are slave nodes. All updates are routed to the master node by the slave nodes. Changes made to the master database are then replicated to the slave databases. Since all updates are always made to only one database, the master database, collisions are avoided.

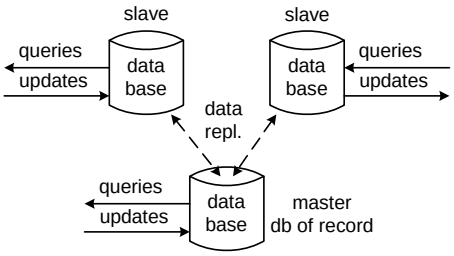


Master/Slave Configuration

In this configuration, the master database is chosen as the database of record. Should its node fail, another node is promoted to be the new master; and its database becomes the new database of record.

Designated Master

The designated master configuration is similar to the master/slave configuration in that one node is designated as the master. However, in this configuration, all updates are made to the database owned by whichever node receives the transaction, whether it is a slave node or the master node. If the node is a slave node, it replicates its database changes only to the master node. The master node will resolve any collisions according to business rules and will then replicate those changes to all of the slave nodes, including the one, if any, that initiated the transaction. Data collisions can occur between two slave nodes or between a slave node and a master node, but the master node is the final adjudicator.



Designated Master Configuration

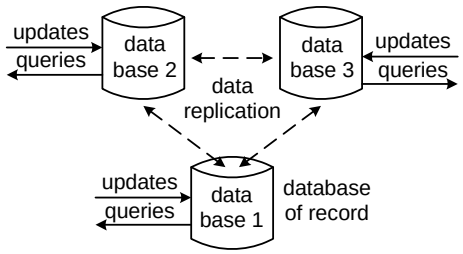
In this case, the database managed by the master node is declared the database of record. Again, should this node fail, one of the slave nodes is promoted to be the new master; and its database becomes the database of record.

Node Precedence

The node precedence configuration involves assigning a precedence (or priority) to each node. If a data collision should occur, it is resolved by accepting the competing change from the node with the highest precedence or priority.

[4] Highleyman, W. H.; Holenstein, P. J.; Holenstein, B. D.; Chapter 4 – Active/Active Topologies, *Breaking the Availability Barrier II: Achieving Century Uptimes with Active/Active Systems*, AuthorHouse; 2007.

For instance, consider the example in which there are three nodes in the application network with precedences 1, 2, and 3, and with precedence 1 being the highest. If a node receives a data collision between nodes 2 and 3, then the data change initiated by node 2 will be accepted and that from node 3 will be rejected.

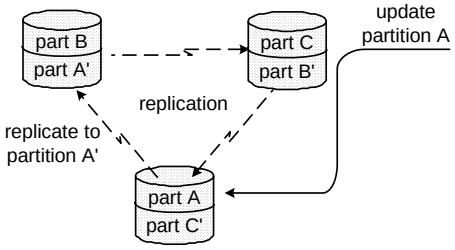


Node Precedence Configuration

In a node precedence configuration, the database managed by the node with the highest precedence is chosen to be the database of record. Should this node fail, then the database managed by the node with the next highest precedence will become the database of record.

Partitioned Database

In some active/active systems, the database is partitioned among the nodes. Typically, one node will own a partition; and that partition will be backed up on the database of another node.



Partitioned Database Configuration

For instance, in a three-node active/active system, the database might be partitioned into partitions A, B, and C. One node provides data storage for partition A and for the backup, C', for partition C. Another node provides storage for partition B and for the backup, A', for partition A. The third node houses partition C and the backup, B', for partition B.

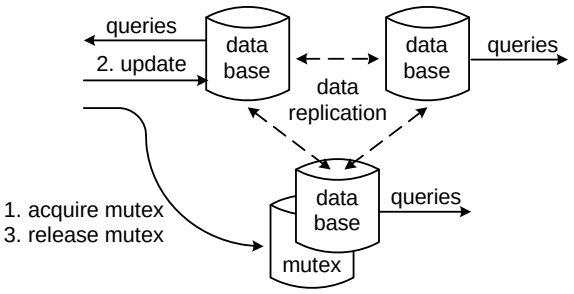
When a node receives an update for a partition, it routes that update to the node owning that partition. The owning node will make the update to its partition and will replicate that change to the other database copies in the application network. Since all changes to any given data object will only be made at one system, there will be no data collisions.

In this case, the database of record is also partitioned. It comprises the primary partitions that are distributed across the network.

Should a node fail, the backup partition for the primary partition on the failed node becomes the new primary partition and the new database of record for that partition.

Global Mutex

If all nodes in an active/active network are peers, data collisions can be avoided by using a global mutex. A *mutex* is a mutually exclusive lock (or other deterministic sequencing algorithm) on a data object. The lock must be acquired before that data object can be modified. A global mutex is a unique mutex somewhere in



Global Mutex Configuration

the network. The global mutex protects all of the copies of a data object across the network.

If a global mutex is used, an application that wants to update a data object must first acquire the global mutex for that object. It can then update the object, knowing that no other updates can be made while it is holding the mutex. Thus, data collisions are avoided. When the application has finished its update, it releases the global mutex.

The global mutexes are generally resident on one of the nodes or can be partitioned across nodes, residing either in memory or on disk storage units. The application database managed by the node containing the global mutexes or global mutex partition should be designated the database of record or the partition of the database of record.

Should the node containing the global mutexes fail, the global mutexes must be moved to another node. The database managed by that node should now become the database of record.

Symmetric Configurations Using Asynchronous Replication

A symmetric active/active system is one in which there is no distinction between the nodes. The nodes are equal peers, and there is no configuration reason to designate one as holding the database of record.

In this case, the determination of the database of record can be based on other attributes, such as:

- *Latency:* The database that is most central in the application network can be chosen to be the database of record. Its maximum replication latency to any other node will be the shortest in the network, and it therefore has the potential to lose less data should it fail.

- *Support:* The database copy that is the best supported from an administrative and maintenance viewpoint might be chosen as the database of record. This may be the database copy that is resident at corporate headquarters.

- *Capacity:* The database copy that is assigned to the node with the greatest capacity might be chosen as the database of record.

Synchronous Replication

An alternative to asynchronous replication to keep the database copies synchronized is the use of synchronous replication.[5] With synchronous replication, locks must be acquired on all copies of the data object to be modified before any change can be made. Synchronous replication guarantees that either all data object copies are changed or that none are. All data objects remain locked until their locks are simultaneously released by the updating application.

[5] Highleyman, W. H.; Holenstein, P. J.; Holenstein, B. D.; Chapter 4 – Synchronous Replication, *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, AuthorHouse; 2004.

As a result, no data collisions can occur; and no data will be lost following a failure. However, transaction response times will be slowed due to the necessity to coordinate updates across the network.

If synchronous replication is used, the considerations for the choice of a database of record are the same as those for symmetric active/active configurations using asynchronous replication, as discussed above.

Node Promotion Following a Failure

Should the node containing the database of record fail, another node must be chosen. The following considerations might be used as a basis for this determination:

- *Verification and Validation:* Often, a verification and validation facility is used to periodically compare the databases to the database of record. Should a discrepancy be found, it is corrected to bring the database being tested into synchronism with the database of record. If this is the case, the database copy that has been most recently validated can be chosen as the new database of record.

- *Minimize Data Loss:* The database copy that is geographically closest to the failed database of record can be chosen to be the new database of record since it is likely to have lost the least data following the failure of the node maintaining the database of record.

- *Support:* The surviving node with the best administrative and maintenance support can be chosen as the new database of record.

Summary

Because an active/active system contains multiple copies of the application database, there must be some method to determine which database copy is to be considered the database of record. In architectures in which one node has a predominant role, the database managed by the predominant node should be designated the database of record. This includes the master node in master/slave and designated master configurations, the node with the highest precedence in node precedence configurations, the node that owns a partition in partitioned databases, and the node that owns the global mutex if global mutexes are used to avoid data collisions.

If the system is totally symmetric and there is no dominant node, or if synchronous replication is used, the determination of the database of record can be made based on geographical location, administrative and maintenance support, or nodal capacity.