the **Availability Digest**

# Active/Active Versus Clusters

May 2007

In our previous issues of the Availability Digest, we focused heavily on active/active architectures. But there is another, very important high-availability architecture, one, in fact, that is far more mature and predominant than active/active systems, That architecture is clusters. In this article, we describe the cluster architecture and compare it to active/active systems.

What is a Cluster?

A cluster is a set of interconnected servers that present a single-server interface to its users. The servers can back each other up, so in the event of a server failure, the functions of the failed server are transferred to a surviving server.

Similar recovery facilities are provided for the database and for the network. The user is unaware of any of this complexity. All a user sees is a single server image satisfying his needs.
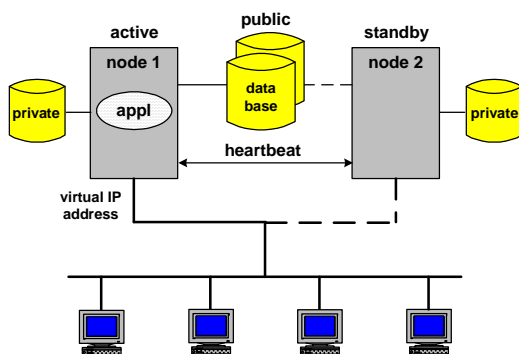
More specifically, a cluster comprises two or more servers, or nodes, that can connect to a common database. In general, only one node at a time, the *active node*, can operate on the database. Otherwise, database corruption can occur caused by two servers trying to update the same data item at the same time. It is important that both servers be the same. Clustering with heterogeneous servers can be terribly complex.

There are, in fact, two types of databases in a cluster. *Public* databases are accessible by multiple nodes in the cluster and hold the application databases. Each server also has a *private* database, which holds all of the software required for that server to operate when it is not the active server (i.e., it is a *standby* server).

One interesting question is where should the application executables reside? If they are resident on the public database, there is no version control problem. The entire cluster is always using the same version of the application. However, if upgrades are to be made to install a new application version, the entire cluster must be brought down.

If the executables are instead stored on the private databases, one node at a time can be brought down to upgrade its version of the application and then returned to service without having to take down the entire cluster. Of course, in this case, the cluster must be able to handle operations with different application versions; but the upgrade can be rolled through all of the nodes one by one.

Redundancy of the public database is as important as server redundancy in order to guarantee high availability. To protect against a database fault, either RAID disk arrays (RAID 5 typically) or mirrored disks (technically RAID 1) are used to hold the database. In either case, the database will survive one physical disk failure, though mirrored disks will have a higher availability since there are only two disks to fail rather than five or six as in a typical RAID 5 configuration.

In order to be truly highly available, the networks through which the nodes communicate with each other and with the users must also be redundant. Then, in the event of a network failure, all traffic can be routed over to the surviving network.

There is generally a failover management system that monitors the health of the cluster and that initiates failover actions.[1]

Cluster Resource Group

Actually, it is not a server that fails over. It is an application. In large systems, a server might be running many applications. A fault might affect only one application, and therefore it is necessary only to fail over that application to a backup server. The original server can continue to support its other applications.

Each application has its own IP addresses, which are used by the users to access application services. When an application is failed over, its IP addresses are remapped to the new server so that further requests are sent to it in its new home, transparently to the user.

An application comprises three components – its application code, its database, and its IP addresses. This set of components for an application is known as a *service group*.[2] When there is a failure, it is the service groups that are failed over, not the servers themselves.

When a service group fails, it may leave behind data corruption. This is because data has been left behind in cache memory and not written to disk. Therefore, the database must be recovered to eliminate corruption before it can be used (for instance, *scandisk* in Windows or *fsch* in Unix).

Following a service group failure, there may be an attempt to restart the application. Should that fail, recovery by failover may be initiated. The failover of a resource group generally takes a few minutes as the application is started (unless it is already up), the database is recovered, IP addresses are remapped, and the new active application connects to its database.

Heartbeats

The center of a cluster's high availability is the *heartbeat*. The servers (or the applications) exchange heartbeats periodically to inform each other of their health. Should a standby server that is backing up a currently active server or service group not receive a timely heartbeat from its active companion, it will declare that entity down and will initiate a failover.

It is extremely important that the heartbeat network be highly available. Should the heartbeat network be lost, both servers will declare the other server down and will attempt to assume control of the applications, thus operating in so-called *split-brain* mode. As a result, both will attempt to write to the database and as a result may corrupt the database.
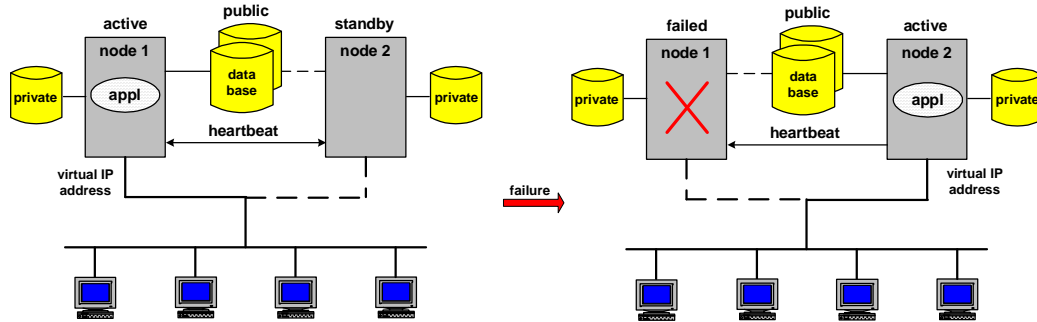
Therefore, the heartbeat network is generally a directly connected redundant network between the servers.

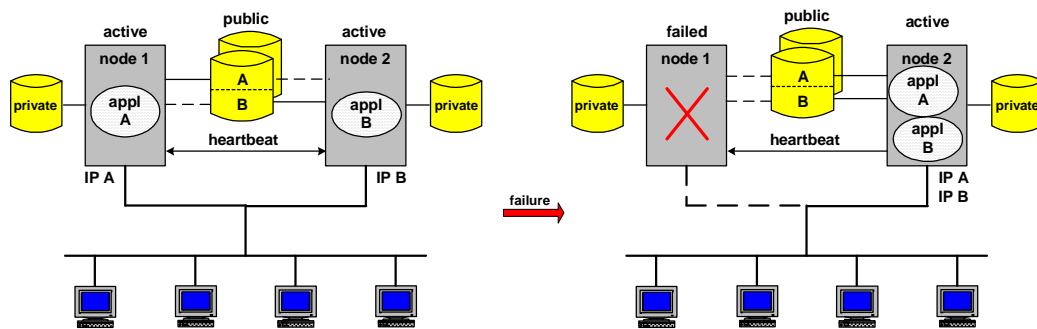[1] See our product review, *HP's ServiceGuard Clustering Facility*, in this issue of the *Availability Digest*.

[2] HP calls this a *package*.
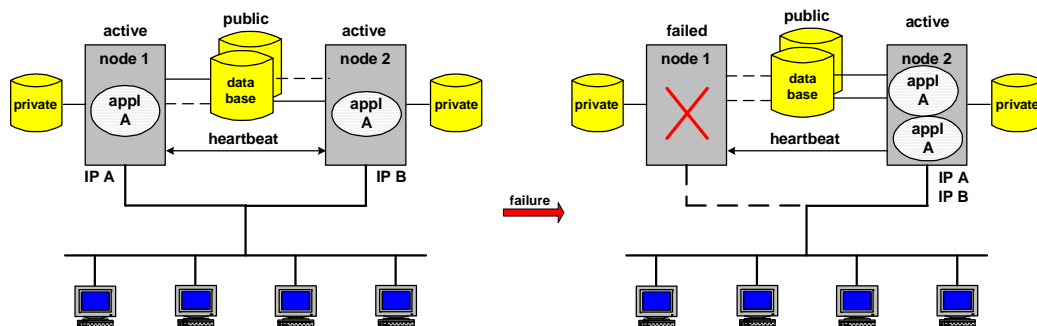
Cluster Configurations

There are three predominant configurations for clusters:

Active/Standby

Active/Active

Parallel Database

Active/Standby, in which one server backs up another. The standby server is either idle, or it can be used to run some noncritical applications which can be immediately terminated in the event of an active server failure. Active/standby configurations can include multiple standbys backing up multiple active servers.

Active/Active, in which each server in the cluster is running different critical applications and is also backing up one or more other servers.[3] Should one server fail, its service groups are failed over to its backup, which now must support its own applications and those of the failed server.

Parallel Database, in which multiple servers can be running the same application against a common database. This is a highly specialized configuration that requires a special database that can run in a distributed manner on multiple servers. Oracle's Real Application Cluster (RAC) database is such a database. It provides the facilities for distributed lock management and distributed cache management.

Disaster Recovery

Clusters can be arranged to be disaster-tolerant by creating a second cluster at a remote site. However, only one server can be updating the database at any one time. Therefore, an application can only be running on one server at a time. Its backup and the servers at the remote node are all idle so far as this application is concerned.
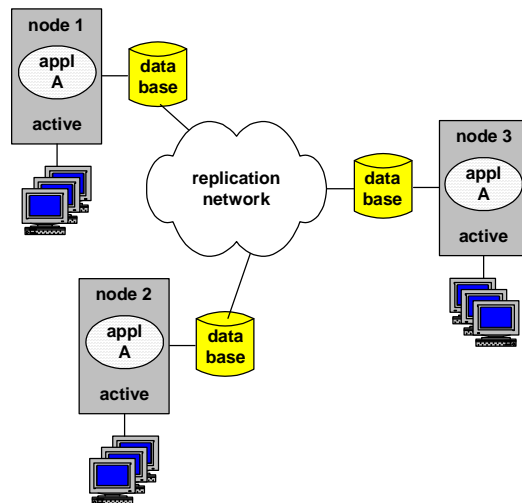
What is Active/Active?

We have described active/active architectures in many of our previous articles.[4] We briefly review this architecture so that we can compare it to clusters.

An active/active system is a network of independent processing nodes cooperating in a common application. Should a node fail, all that needs to be done is to switch over that node's users to a surviving node. Recovery is in subseconds to seconds.

Database Replication

Each processing node has access to a copy of the common application database. These database copies are kept synchronized via bidirectional data replication. Whenever a processing node makes a change to the database, it replicates that change to all of the other database copies in the application network.

Data replication is generally asynchronous, though there are techniques for synchronous data replication. A change queue of some sort, provided at each node, records all of the changes

---

[3] Note that this is not our definition of active/active, as described later. To resolve this confusion, Microsoft is pushing the term *multi-instance* for this cluster configuration.

[4] See *What is Active/Active?* In the October, 2006, issue of the Availability Digest.

made to the database at that node.[5] This may be a transaction log (audit trail) maintained by a transaction management facility, or it may be a log created by the application or by database triggers. The replication engine fetches changes from the change log and sends them to the target system, where they are applied to the database.

Replication Latency

There is a delay from the time that a change is applied to the source database and the time that it is applied to the target database. This time delay is known as *replication latency*. Replication latency brings with it two problems, data collisions and data loss following a failover.

> Data Collisions - Because there is a delay before the target system sees an update, it is possible that changes to the same data item can be made at two different nodes at about the same time. Both of these changes will be replicated to the other node, where they will overwrite the original change. Both databases are now different, and both are wrong. There are techniques for detecting and automatically resolving data collisions.

> Data Loss Following a Failover – Should one node fail, changes that are currently in the replication pipeline will be lost. Partial transactions will be rolled back at the surviving nodes, and these must be resubmitted.

Split-Brain Mode

As with clusters, if communication is lost between two nodes, they cannot replicate to each other. There are then two choices – take one node down by moving its users to the other node, or let both nodes continue to function in split-brain mode.

In the latter case, each node will continue to update its copy of the database and will queue its changes to be replicated to the other node when communications are restored. During this time, the databases will diverge; but their integrity will otherwise be maintained.

When communication is restored, each node will send its queued changes to the other node. There are bound to be data collisions, which must be resolved in the same way that the system normally resolves data collisions,

Note that whereas split-brain mode in a cluster results in data corruption, split-brain mode in an active/active system results only in database convergence.

Rolling Upgrades

Hardware, software, or applications can be upgraded at any node simply by moving its users to another node, upgrading the now idle node, and then moving its users back to their original node. An upgrade can be rolled through the entire application network one node at a time by using this process.

Disaster Recovery

Disaster recovery comes for free in an active/active system provided the nodes are geographically separated. Should one node be taken out of service for any reason, all that is required to restore full service to the users is to move those users who had been using the failed node to the surviving node, an operation that can be performed in seconds.

---

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

Comparing Clusters and Active/Active Systems

Both clusters and active/active systems provide high availability through redundancy. However, there are some striking differences.

Availability

Failover time for a cluster is typically measured in minutes as service groups are migrated, the database is recovered, and databases are opened. Failover time for active/active systems can be measured in subseconds or seconds as all that is required is to switch failed users to a surviving system.

Let us take the case of a cluster with a five-minute (300 second) failover time and compare it to an active/active system with a three-second failover time. Over a long period, the cluster will be down one hundred times as long as the active/active system, thus lopping off two 9s from its availability relative to the active/active system. If the servers in an active/active system are providing seven 9s availability (an average downtime of three seconds per year), those same servers in a cluster arrangement would provide five 9s of availability (an average downtime of five minutes per year).[6]

Furthermore, a cluster failure affects all users, whereas a failure of a node in an active/active system affects only the users being serviced by that system.

This leads to the primary active/active rule for extreme availabilities: *Let it fail, but fix it fast.*

Application Constraints

For both clusters and active/active systems, there are constraints on the applications in order to allow them to run in that mode. However the constraints are different.

Cluster Application Constraints – An application must:

- be easy to stop and restart.
- be easy to monitor.
- be able to use shared storage.
- be able to store its state on disk.
- be able to restart from its saved state.
- not corrupt data on a crash.

Active/Active Constraints – An application must:[7]

- be architected so that the database is loosely coupled to permit replication.
- be able to handle distributed deadlocks if synchronous replication is used.
- be able to generate unique numbers across the application network.
- not store local context in memory.
- be able to control batch processing in a distributed network.
- be able to manage, monitor, and configure distributed applications.

---

[6] See our companion article in this issue, *Calculating Availability - Cluster Availability*.

[7] See Appendix 4, A Consultant's Critique, *Breaking the Availability Barrier: Active/Active Systems in Practice*, AuthorHouse; 2007.

Other Comparisons

	Cluster	**Active/Active**
Split Brain	Data Corruption	Data Divergence
Data Sharing	None (unless RAC is used)	Unlimited
Application Scaling	No (unless RAC is used)	Unlimited
Disaster Recovery	Application active on one node	Application active on all nodes
Heterogeneity	None	Limited only by replication product
Rolling Upgrades	Yes, if executables are on private disk	Yes
Data Collisions	No	Yes (unless synchronous replication Is used)
Data Integrity following failover	Data corruption	Data loss (unless synchronous replication is used)
Maturity	Mature (VAX cluster introduced in 1984).	Developing

Summary

Cluster technology is very mature and is supported by a plethora of products.

Active/active technology is the new guy on the block. Its product support is growing, and there are a number of very successful implementations. Active/active technology can provide an order of magnitude or more improvement in availability.

Clusters are HA (high availability).

Active/Active is CA (continuous availability).