the Availability Digest

Blueprints for High Availability

May 2007

Evan Marcus and Hal Stern are well known in the cluster community. Their book, *Blueprints for High Availability: Designing Resilient Distributed Systems*,[1] is one of the most referenced books on this topic. With backgrounds from Sun Microsystems and Veritas, the authors bring to the table a long history of high-availability experience, which they share in this book.

Easy to read and thorough in its content, it is an excellent reference for anyone wanting to learn about clustering technology. Also included are many insights into what is required outside of the cluster to achieve high availability.

Levels of Availability

The authors define several levels of availability:

- Basic nonredundant systems (the lowest level)
- Redundant data (the use of RAID-5 or mirrored disks)
- System failover (active/backup pairs)
- Disaster recovery (geographic distribution of active and backup systems)
- Fault tolerance (which is outside the scope of the book)

They also define several availability terms, such as resiliency, availability, and downtime.[2]

There are several ways in which a system can fail, thus affecting its availability. These include hardware, software, environment, network, database, and Web server faults.

20 Key System Design Principles

The authors suggest twenty design principles that should guide any migration to a high-availability system. In order of increasing significance, they are:

20. Spend money … but not blindly.
19. Assume nothing.
18. Remove single points of failure.
17. Maintain tight security
16. Consolidate your servers.
15. Automate common tasks.
14. Document everything.
13. Establish service level agreements

[1] Marcus, E., Stern, H., *Blueprints for High Availability: Designing Resilient Distributed Systems*, John Wiley and Sons, Inc.; 2000.

[2] Interestingly, they say that "Given today's technology, [six 9s] is unachievable for all practical purposes and an unrealistic goal." Clearly, they were not thinking active/active, though they do make a reference to similar technology later in the book.

12. Plan ahead.
11. Test everything.
10. Maintain separate environments
9. Invest in failure isolation.
8. Examine the history of the system.
7. Build for growth.
6. Choose mature software.
5. Select reliable, serviceable hardware.
4. Reuse configurations.
3. Exploit external resources.
2. One problem, one solution.
1. KISS: Keep it simple.

Data Management

Disks are the most likely components to fail and must be redundant, either by using RAID arrays or by using mirrored pairs.

The authors discuss several disk connection and storage methodologies, including SCSI and Fibrechannel connections, multihosting and multipathing, disk arrays, JBODs (just a bunch of disks), Storage Area Networks (SANs), and the various levels of RAID.

Logical Volume Management (LVM) is an important part of any storage facility. A good LVM facility will provide online reconfiguration, will remove operating system limitations, will interface with system management tools, and will support software RAID if desired.

A high-availability disk subsystem requires more than just redundant storage media. It also must include redundant data paths and controllers, redundant cabinets and racks so that each disk of a mirrored pair can be housed separately, redundant cables, and redundant power supplies and management.

File system recovery is a particularly critical problem. In Windows, Unix, and Linux systems, data in the disk cache is not flushed to disk immediately. If a crash occurs, that data will be lost; and the file system will be corrupted. A file recovery utility must be run to repair the file system (*scandisk* for Windows, *fsch* for Unix), and this can take a long time. If the file is journaled, i.e., it provides a change log such as that maintained by transaction processing systems, file corruption is not typically a problem.

Clusters

Failover should be transparent and quick, with minimum data loss, minimum manual intervention, and guaranteed access to data (that is, the application has access to the same copy of the data following failover).

Though we think of failover as being from one server to another, it is really the application that is failing over. In large servers, the application may be the unit of failover rather than the entire computing system.

To this end, the authors define a service group as all those elements which make up an application. This includes the application executables, one or more IP addresses by which external users connect with the application, and the application database. Note that the computer itself is not part of the service group. The service group can run on any computer system.

Though a cluster can be more complex, it is most often thought of as a pair of identical and independent processors – the same hardware, the same operating system, the same database management system, and so on. If the cluster is not homogeneous, failover becomes incredibly complex.

There should be no single point of failure in the cluster.

The cluster also includes physical network connections, disks, and applications.

Network Connections

There are three types of network connections needed by a cluster:

- A pair of independent *heartbeat connections* over which the applications send “I’m alive” messages to inform each other of their health. Preferably, these are direct connections and are not routed through any external network that might fail.

- A *public network* which the users use to connect to the cluster.

- An *administrative network* which can guarantee the system administrator an access path to each server.

Disks

There are two classes of disks in a cluster:

- Each processor in the cluster owns a *private (or internal) disk*. This disk includes the operating system and all of the other software needed by the processor to function when it is not the active processor.

- The *public (or shared) disk* holds all of the application data. The public disk migrates back and forth between the processors depending upon which is active. Only one processor – the active processor – can access the public disk at any one time.

The public disk must be redundant. Either RAID arrays or mirrored pairs may be used.

Applications

An important application consideration is licensing. Software vendors may charge additional partial or full licenses for their software running on a backup system even though only one copy is being used at any one time.

An interesting question is where should the application executables go? Putting them on the public disk means that there is no versioning problem. There is only one version of an application on the system at any one time.

On the other hand, if the application executables are on the public disk, there is no opportunity to do a rolling upgrade of an application. The system must be taken down to do an application upgrade, thus destroying its availability unless there is dead time that can be scheduled for this purpose.

Therefore, for 24x7 systems, the application executables must be resident on the private disks.

The Heartbeat Network

The heartbeat network is arguably the most important network in the cluster. Without it, there is no way for the backup system to know about the active system’s health and therefore no way for it to initiate a failover should the active system fail.

There are several reasons that one system may cease receiving heartbeats from the other system:

- The other server is really down.
- There is a heartbeat network failure.
- The process that generates the heartbeat in the other server has failed.
- The other server is too heavily loaded and is running too slowly.

If the heartbeat fault is other than a server failure, manual intervention may be required. One severe problem with heartbeat failure is operation in split-brain mode. In this mode, both servers think that they are the active server since they think that the other server is down. This can cause both servers to try to write to shared files, thus corrupting them. Manual intervention may be required to correct this situation, though there are techniques for ensuring that split-brain mode will not happen.

Public Network

The public network is used by the users of the system to access the application. It should be a redundant network so that no single network failure will deny users access.

Each application has its own set of virtual IP addresses that are used by the users to access that application. Should the application fail over to another server, its IP addresses go with it; and the failover is transparent to the users.

There are several ways in which a virtual IP address may be migrated to another server. A gratuitous ARP may be issued;[3] the ARP will redefine the mapping between the virtual IP address and the new server's MAC address. The MAC address can be moved. Alternatively, the client can be responsible for detecting that the connection is down and reconnecting to the alternate IP address.

Consequently, a cluster provides a set of IP addresses. Each server will have its own IP address to which the virtual IP addresses of the applications are mapped. In addition, each server will have an administrative IP address to which system administrators can connect over the independent administration network.

Failover Management

Failover is generally managed by an independent software facility, the Failover Manager. The Failover Manager may either be a commercial product or it may be a home-grown facility. However, the authors note that the failover process is very complex; and home-grown facilities do not have the maturity, depth of testing, and support that commercial facilities have.

The Failover Manager is responsible for monitoring the health of its system – the processor, disks, networks, and software components. Should a problem be detected, a typical Failover Manager provides several options for the action to take:

- It can ignore the fault.
- It can attempt to restart the failed component.
- It can initiate a failover.
- It can send a notification to appropriate system technicians.

[3] A gratuitous ARP (address resolution control) is a request sent by a server for its own MAC address. All listening routers will map the IP address of the server to its MAC address on the LAN. In effect, the server has seized this IP address.

Notification implies manual action. Though this may be required in some cases, the requirement for manual action can lead to long recovery times as the appropriate personnel are located and notified, following which they must get to the data center (or find a terminal and log onto the cluster), analyze the problem, and determine the action to be taken.

Failover Configurations

Two-node cluster configurations are the most common cluster architecture. They may be used in asymmetric or symmetric arrangements.

An asymmetric arrangement is one in which one node is active and the other is standing by, monitoring the active node and ready to take over should the active node fail. One question addressed by Marcus and Stern is whether the standby can be used for any useful purpose prior to having to take on the active role. The authors mention several possibilities:

- Bad – use it for code development, but bugs in new code might take down the backup.
- Acceptable – use it for database development, which is not so risky.
- Good – use it for some critical, mature application which can be terminated if necessary.
- Best – do nothing with it, and let it simply stand by.

In a symmetric arrangement, both servers are hosting different applications and are backing up their companion. Should one server fail, all processing is moved to the surviving server.

In larger systems, a server may be hosting multiple service groups. In these cases, failover is by service group rather than by server. It is quite possible that a fault has affected only one service group, and that group can be restored to service by failing it over to the other server.

There are many other failover configurations. Examples are N to 1, where one standby server backs up several primary servers; N to M, in which multiple standby servers back up multiple primary servers; and ring configurations, in which each server is backed up by the next server in the ring.

Failover Faults

A faulty failover can result in quite undesirable system operation. One such situation occurs when each server determines that the other is down. This can cause two potentially severe problems:

- *Split-Brain Mode*, in which both servers think that they are the active server and both try to write to the public database, thus overwriting each other's work and corrupting the database.

- *Ping-Ponging*, or tug-of-war, in which one server seizes control only to have the other server seize back control.

Proper system monitoring and a good failover utility are musts for proper failover. But the best protection against failover faults is constant testing. Nobody likes to do it, but there is no other way.

Redundant Network Services

Network faults are very difficult to locate. They can be transient and can come and go. The delayed delivery of messages on a congested network may appear to be a network fault. Denial-of-service attacks are a fact of life today. Faults may be caused by hardware or by IP problems (configuration or routing errors). Networks can be so complex that once a problem is detected, it may be very difficult to locate.

The consequence is that it is very important to build redundancy into the network so that communication can still proceed even in the event of the failure of a portion of the network.

Data Service Reliability

Data must be always available to the applications that need it, or the applications will be down. The commonly used Network File System (NFS) can provide many of the needed availability functions. Database servers and Web servers are also configurable in high-availability architectures.

One common question is whether "deep" or "wide" is better. For high availability, is it better to have a stack of low-end disks; or is it better to use a high-end database server. This is a cost/availability tradeoff that has to be carefully investigated for each installation.

Replication Techniques

Data replication technologies are important when switchover times measured in minutes are not enough. A separate independent backup system with a mounted database copy that is kept synchronized with the active system can provide rapid and deterministic failover times. However, this is primarily the realm of mainframe systems. The authors warn that Windows and Unix users should tread lightly here.

There are many other uses for multiple independent hosts, each with a database copy kept synchronized by data replication:

- Load balancing among many replicated servers (primarily used for Web servers).
- Resolving WAN latencies by providing a database copy locally to a group of users.
- Failover and recovery in the event that one of the database copies becomes corrupted.
- Disaster recovery following a total server failure.

Many replication techniques exist today:
:

- File replication (for example, via *ftp* or *tar*).
- Block-level replication by device drivers or disk controllers.
- Transaction-level replication.
- Process state replication.

Each of these techniques has its own uses. File replication can only be done while the file is quiescent. Block-level replication does not guarantee the integrity of the database, which must be recovered before use. Transaction-level replication depends upon a change log, but maintains a database that is always consistent and which can be used while replication proceeds. Process state replication is used for high speed applications in which the backup process needs to remain synchronized with its active process.

Application Recovery

Applications can fail for several reasons, such as memory exhaustion or software bugs. When an application module fails, there are several techniques that can be considered to return it to service:

- Restart the application and try again.
- Do a controlled shutdown of the application.
- Crash the application (the application bug may have done this anyway) and restart.
- Restart from the last checkpoint if available.

Backup and Restore

Database redundancy, even with full mirrors, does not replace proper backup procedures. The most common use of backup tapes isn't for disaster recovery. It is for the restoration of corrupted or deleted files.

Like all other procedures, backup procedures must be frequently tested. Magnetic tape units must be properly maintained, and tapes should not be used beyond their useful life. Two backup copies should be made of all critical files since a tape error will render a backup tape useless.

There are many commercial backup utilities. Marcus and Stern provide many useful features to use in an evaluation of a commercial product. In addition, other considerations include the duration of the backup window, backup scheduling (both complete and incremental backups), and the handling and storage of backup tapes.

The only purpose of backing up data is so that it can be restored. Just as with the backup procedures, the restoration of the data must be thoroughly tested.

System Operations

System administration plays a large part in achieving high availability. Maintenance plans and processes must be in place. Change is a constant. Changes must be documented and well tested. One must control the change rate to prevent confusion and consequent failure. The distribution of files must be automated. There are good change management tools to aid in this endeavor.

Spare parts policies must be established. They define what spares, if any, will be kept on site. RAID disk replacements are a good candidate since RAID disks are inexpensive, critical, and failure-prone. Good preventive maintenance procedures will extend the life of components and may detect failing components early.

Environmental and physical issues must be thought out. These include the planning for adequate space with the proper power and cooling, security, water, fire protection, and the need for occasional new construction. Vendors must be managed.

Disaster Recovery

Disasters that can take out a single site include the loss of building power or power in a wider area, flood, fire, earthquake, tornado, terrorism, and many others. Recovery from such disasters requires a geographically separate disaster recovery site.

This site must be properly populated with the required processors, disks, telecommunication facilities, and licensed software as well as with desks, telephones, and other facilities needed by the personnel at that site.

Finally, once in place, the failover procedures to the backup site must be thoroughly and frequently tested.

The Bottom Line

High availability is insurance for your business. The amount that you are willing to spend on it depends on what downtime will cost you. If downtime costs your business $100,000 per hour, it is certainly justifiable to spend $25,000 on shortening each outage by thirty minutes. But if downtime costs your business $1,000 per hour, this expenditure may not be worth it.

The bottom line is that the requirements for high availability and its worth are different for every business. *Blueprints for High Availability* provides the background required to effectively evaluate the availability/cost tradeoff for your business.