the Availability Digest

www.availabilitydigest.com
@availabilitydig

Should Hope Trump Failover?

February 2014

Critical IT services are typically protected by redundancy. Each production system is backed up by another system in the data center or by a system located in a geographically remote data center. Should the production system fail, the backup system is brought into service.

Maybe! What if the backup system won't come up? Now you have two systems down. This is the dreaded *failover fault*. Failover faults are a major concern for systems that must achieve high availability.

Failover Faults

Failover faults occur all too frequently.[1] A primary reason for this is that failover testing is risky and expensive. In general, to test failover, the production system must be taken down and the backup system brought into production. However, depending upon how the backup system is synchronized with the production system, this process – if it works – could normally take minutes to hours.

If the backup system is a cold standby and its database must be loaded before applications can be started, this can take hours. If the backup system is a warm standby with a replicated database that is up-to-date, all that is required is to bring the applications up. This can take minutes to hours. If it is a hot standby with a replicated database and with all applications running, failover might be able to be done in minutes. All that is required is the reconfiguration of the network connecting the users to the processing systems.

During the failover, the IT services provided by the applications being tested are down. If these are critical applications, exhaustive planning is required to do a failover test. An appropriate time window must be determined (usually late at night through early morning on a weekend). Users must be alerted in advance that the system will be down for maintenance for an estimated period of time. Fallback procedures must be in place to return to the production system when the test is complete, whether the test is a success or a failure. There must be coordination with partners who interface with the system to be tested. Most importantly, there must be key personnel available during the test who understand every aspect of the system. If the backup won't come up, it is these people who can trace the problem and fix it.

The result is that many companies do not do complete failover testing. They may test only certain aspects of failover. They do not really know whether their backup system will come up or not. If their production system fails, they depend upon faith and hope rather than thorough testing.

[1] JPMC Downed by Replicated Corruption, *Availability Digest*; November 2010.
http://www.availabilitydigest.com/public_articles/0511/jpmc.pdf
Poor Documentation Snags Google, *Availability Digest*; April 2010.
http://www.availabilitydigest.com/public_articles/0504/google_power_out.pdf
Triple Redundancy Failure on the Space Station, *Availability Digest*; November 2007.
http://www.availabilitydigest.com/public_articles/0211/iss_tmr_failure.pdf

Failover Delay

The result of inadequate failover testing is *failover delay*. Failover to the backup system is a serious decision with several major consequences, including an extended outage if there is a failover fault. Consequently, the failover decision is typically made by senior management. The first thing that they will want to know is how long it will take to restore the production system. If this time is expected to be shorter than or comparable to the time to failover, it is likely that management will chose to wait until the production system is returned to service.

If the production recovery time is misestimated and it cannot be brought back into service in a timely fashion, then the decision to failover will be made. This does nothing but extend the outage. If there is a subsequent failover fault, the help-desk telephones will definitely be ringing.

This leads to an interesting question, which is really a dilemma:

> How long should it take to make the failover decision?

The answer to this question lies in large part to the degree of failover testing that has been practiced. If there had been only cursory failover testing, then management will be reluctant indeed to authorize a failover. On the other hand, if failover testing has been rigorous and recent, failover may be authorized as soon as there is any question about returning the production system to service within the failover time.

We recently posted a question to our LinkedIn Continuous Availability forum, which asked:

> "How long should you wait until you give up on system restoration and attempt a failover? Does your organization have policies for this? How does your confidence in your failover testing procedures affect the time you are willing to wait before attempting a failover? How often have you experienced a failover fault?"

One interesting response from Chris Petre related a real experience:

> "I was involved in the post mortem of a major outage where there was a delay in declaring a failover. I don't think they had any hesitation to declare a failover. There was an unusual situation whereby transactions were still flowing very slowly so the focus was on diagnosis/correction. They eventually made the decision to go stand-in which worked, but by then the damage was done. If it was a clear outage, they would have been better off and cut over immediately."

Perhaps in this case, the organization would have been better served to do its diagnosis and correction after a successful failover. If the organization had been confident in its ability to failover, this would have an easy decision to make.

How Critical to Availability are Failover Faults?

Are failover faults really a serious availability issue? A little math answers that question (go to the end of this section if you are mathematically challenged).

Consider a dual system – one a production system and one a backup system. If the availability of each is *a*, the probability of a single system failure is (1-*a*). The probability of a dual failure (the production system fails and the backup is currently down as well) is (1-*a*)(1-a). The availability of the system is [1 - (1-*a*)^2]. The availability cannot get any better than this – this is what the system manufacturer has given us. We call this the *inherent availability* of the redundant system.

But a dual system failure is only one cause of system downtime. There are two other significant causes:

- The system is in the process of failover over (seconds to hours).

- The failover fails.

Let
mtr be the mean (average) time to repair a system.
mtfo be the mean time to failover.
d be the probability of a failover fault

It can be shown that the true availability of the system is[2]

$$1 - (1-a)[1-(a\text{-mtfo/mtr-}d)]$$

This is equivalent to the inherent availability of the system, but the system behaves as if it is one system with an availability of *a* and a second system with an availability of *a* reduced by failover time and the probability of a failover fault.

Ignoring failover time, if the system availability is three 9s (0.999) and the probability of a failover fault is .01 (1% - many would love to have 99 failover attempts out of 100 succeed), the second system has an effective availability of only two 9s. The overall availability of the dual system has been reduced from six 9s to five 9s – it is an order of magnitude less reliable.

Yes, failover faults are significant to high availability!

A Major Cause of Failover Faults

One of the main reasons for failover faults is *configuration drift*. Typically, the backup system must be configured exactly the same as the production system in order for it to come up properly. However, the production system is continually being upgraded software-wise. There are three classes of software objects that must be kept synchronized:

- *Audited Databases*: These are the application databases. They are typically transaction oriented with a high rate of activity.

- *Unaudited Files*: These are ancillary configuration files used by the applications. They are fairly static.

- *Configuration Changes*: Various utilities are used to modify and manage the configuration of the software systems.

All of these software objects must be kept synchronized between the production system and the backup system. Otherwise, the backup system may fail to properly process transactions. There are typically utilities to aid in the detection of configuration errors in the backup system and to automatically keep the backup system in synchronism with the production system.

For instance, in HP NonStop systems, there are many data replication products to replicate in real time changes made to the application databases, including those from Gravic (Shadowbase), Oracle (GoldenGate), Network Technologies (DRNet), and Attunity (Replicate). Unaudited files are kept synchronized by FileSync from TANDsoft or AutoSYNC from Carr Scott. Configuration changes can be replicated by TANDsoft's Command Stream Replicator.

Regardless of the systems used, it is mandatory that utilities such as these be used to maintain the backup system in synchronism with the production system. Otherwise, the organization will always have a high incidence of failover faults.

[2] Simplifying Failover Analysis – Part 1, *Availability Digest*; October 2010.
http://www.availabilitydigest.com/public_articles/0510/failover_analysis.pdf

Eliminating Failover Faults

There is a way to eliminate failover faults, and that is through the use of active/active systems.[3] An active/active system comprises two or more nodes, each of which is actively engaged in a common application. They each have their own copies of the application database, and the databases are kept synchronized via data replication.

A transaction can be sent to any node in the application network and be properly processed. Should one node fail, all that needs to be done is to reroute transactions from the failed node to other nodes. This can be done in seconds, leading to very fast failover. Moreover, it is known that the other nodes are working because they are processing transactions. Therefore, there are no failover faults.

Another advantage of active/active systems is that the nodes in the application network are loosely coupled. They do not have to have the same configuration. In fact, they can even be different systems. Wolfgang Breidbach of Bank-Verlag, perhaps the true pioneer in active/active systems, built what well may be the earliest active/active system in 1988. It was a two-node system. One was an IBM mainframe, and the other was an HP NonStop system.[4]

In some cases, an application cannot be run in a distributed environment such as this. It must run on its own system. In this case, the same result can be achieved with a "sizzling-hot" standby. A sizzling-hot standby configuration is a two-node active/active system in which all transactions are routed to only one node. However, the other node is completely active and ready to take over on an instant's notice. It can be periodically tested (like every second) by simply sending it a test transaction to process.

On the LinkedIn Forum in response to our question, Bank-Verlag's Wolfgang wrote:

> "We are trying to avoid that failover stuff. We are a central access point for ATM- and POS-authorization. So if our application is down, people will neither get money at the ATM nor will they be able to pay by card at a shop.
>
> As we are running active-active (meanwhile 4 active systems), there is no failover decision. If one system is down unexpectedly, we do everything necessary to get it online again but such an event is only serious and not critical. I remember a situation a few years ago. On late Saturday morning one of our systems went down because of a problem within the SNA product. Operation people called me, I came in, took a dump and we started the system again. This took about 3 hours, if I remember correctly. During that time we received only 2 calls requesting confirmation that the system was not available, nothing else. No customer was affected. And even more important: no manual intervention on our side was necessary.
>
> Of course this required of lot of regulations and organization in the past. But today we can be sure that the application will be available without manual intervention even if one datacenter goes completely down."

Interestingly, Chris replied:

> "The major outage for this large Canadian Bank was back in late 2001 and I was selected to be part of an external 12 person swat team of senior consultants and practice owners covering technologies in use at the Bank for a major one-month long assessment and improvement engagement. My focus covered anything Tandem (sorry ... HP NonStop) related. One of my recommendations for the final report delivered in 2002 was in fact to go active-active."

[3] What is Active/Active?, *Availability Digest*; October 2006.
http://www.availabilitydigest.com/public_articles/0101/what_is_active-active.pdf

[4] Bank-Verlag – The Active/Active Pioneer, *Availability Digest*; December 2006.
http://www.availabilitydigest.com/private/0103/bank_verlag.pdf

Many banks, telcos, and other major companies with critical applications that just cannot go down have gone to active/active systems. The Case Studies in the Availability Digest article archive[5] have several stories about such successes.

Summary

Failover faults are indeed a major factor in availability. They can easily reduce the availability of a redundant system by an order of magnitude.

To reduce failover faults, it is imperative that means be taken to ensure that the backup system is truly in synchronism with the production system at all times, and that failover be regularly and completely tested.

The ultimate defense against failover faults is active/active systems. In these systems, it is known that the backup system is always available because it is, in fact, currently processing transactions. An additional advantage of active/active systems is that no configuration synchronization is needed between systems.

[5] http://www.availabilitydigest.com/articles.htm