the Availability Digest

www.availabilitydigest.com

Is the Cost of Converting to Active/Active Worth It?

November 2009

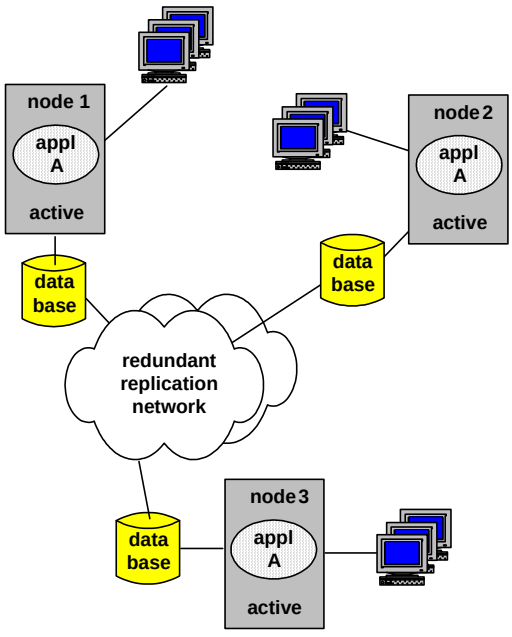
Active/active systems can bring continuous availability to applications that simply cannot be down. But converting your current active/backup system to active/active comes with a cost. Hardware may have to be upgraded, data replication engines licensed, and applications modified.

These costs are offset by the virtual elimination of both unplanned and planned downtime. But is the cost of conversion worth it? That all depends upon how much your current downtime is costing you.

In this article, we take a look at the costs of moving to active/active. We give two examples of determining whether these costs are justified based on the downtime savings afforded by an active/active architecture.

What is Active/Active?

For those of you to whom the concept of active/active is new, we first review the fundamentals of an active/active architecture.[1] An active/active system is a network of independent geographically-distributed processing nodes cooperating in a common application. Each node has access to a copy of the application database, and a transaction can be routed to any node for processing.



The database copies are kept in synchronism via data replication. Whenever one node makes a change to its copy of the database, this change is immediately replicated to all other database copies in the application network. In this way, all nodes have access to an up-to-date application database.

Should a node fail, all that needs to be done is to reroute users from the failed node to one or more of the surviving nodes. This rerouting can be accomplished in seconds or even subseconds. If this failover is fast enough, users will not even notice that there has been a failure and so, in effect, there has been no outage.

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

Furthermore, it is known that the node to which users are rerouted is up and running. After all, it is currently processing transactions. Thus, failover is fast and reliable, and continuous availability is achieved.

The Costs of Going Active/Active

There are several costs to be considered if you want to convert a critical application to take advantage of the continuous availability of an active/active configuration. Below is a brief look at many of these costs.

A Second Site

Additional Facilities

If your application is mission-critical, you probably already have a backup system. Hopefully, it is in a separate site that is far enough away from your primary system that no common disaster such as an earthquake, flood, hurricane, fire, or explosion will negatively impact both sites.

If this is not the case, a major cost of moving to active/active is the acquisition of a remote site and a second system. The second system need not be identical to the original system, but it must be large enough to handle the entire load should the original system fail. (We ignore here the costs of a multinode system of more than two nodes since, if you are moving from a single system, your first step is most likely to a dual-node active/active system.)

In addition to the IT costs of a second site and a second system, there are many ancillary costs for other facilities such as equipment cooling, fire suppression, and security.

Of course, you can implement an active/active system with two nodes collocated in a single site. Though this will save the cost of a second site, you have given up disaster tolerance – a bad move if your application is truly critical to your company.

Additional Staffing

With the second site comes the additional expense of additional technical staff to man that site. If you already have a second site, you probably have that staff already in place and trained. If you are setting up a second site, you can always consider a lights-out site with little or no staffing. Since you are running active/active with automatic failover, if your lights-out site fails, you have time to get technical support to that site to correct the failure since your staffed site is fully operational and handling the full load.

However, you don't want to wait too long to restore your failed site to service because, until you do, you are running with a single point of failure – your only operational site.

Additional Operating Costs

With a second site comes additional operating costs such as lighting, heating, equipment cooling, and the associated electric bills. In addition to technical staffing, you may have additional personnel costs such as for administration and security.

Additional Licenses

Even if your primary system vendor was giving you a break on licenses because you were using one system only as a backup, that break will now disappear because both systems will be in active use.

Replication Engine

In order to keep the two database copies synchronized, you will have to license a data replication engine. This should be a carefully considered choice. Data replication comes with certain problems.

If you are using asynchronous replication,[2] in which data changes are replicated to the target system after they have been made to the source system, *replication latency* is an important attribute. Replication latency is the time delay from when a change is made to the source system to the time that it is made to the target system. The longer the replication latency, the more data will be lost should the source node fail.

Also, data collisions are more likely. A data collision[3] occurs when both nodes try to update the same data item within the replication latency interval. Neither node will know of the other's change, and the database copies may begin to diverge. It is important that if data collisions are possible in the application, the replication engine has facilities for detecting and resolving these collisions.

An alternative to asynchronous replication is synchronous replication.[4] Synchronous replication engines ensure that each change is made to all database copies in the application network or that no copies are modified. Synchronous replication avoids the problems of data loss and data collisions associated with asynchronous replication. However, applications are slowed as they wait for changes to be applied across the application network.

Redundant Replication Networks

The copies of the application database are kept synchronized by replication changes across a communication network. This network must be very reliable, for if it fails, replication fails. If nothing is done, each node will continue to process transactions but will not be able to replicate its changes to the other system. The result is that the two database copies will begin to diverge. This is called *split-brain mode*. When the network is restored, the changes made to the database copies during the outage will have to be reconciled, and many data collisions are likely to have occurred.

The alternative is to shut down one of the nodes and run with only one node, facing the consequences of a failure of that node.

To avoid this problem, the replication network should be redundant. If one rail of the replication network should fail, replication can continue over the surviving rail.

Redundant User Networks

A key to the continuous availability provided by an active/active system is the ability to switch users or transactions from a downed node to an operating node. However, this could be thwarted by a communication failure in the network interconnecting the users with the processing nodes. It is important that this network be redundant and reliable so that users can always have access to an operating node in the presence of any single node or network fault.

---

[2] Asynchronous Replication Engines, *Availability Digest*; November 2006.
http://www.availabilitydigest.com/private/0102/asynchronous_replication.pdf

[3] Collision Detection and Resolution; *Availability Digest*; April 2007.
http://www.availabilitydigest.com/private/0204/collisions.pdf

[4] Synchronous Replication, *Availability Digest*; November 2006.
http://www.availabilitydigest.com/private/0102/asynchronous_replication.pdf

Additional Application Software Licenses

If you are running third party applications, you will probably have to pay additional license fees to run dual copies of the applications.

Application Modifications

Your applications were probably not designed to run in a distributed environment. Once you try to run multiple cooperating copies of an application, several problems could raise their ugly heads:[5]

- If data collisions can happen, strategies for resolving them must be determined and implemented.
- Unique number generators used, for instance, to create customer numbers or invoice numbers will no longer be unique when running multiple copies in different systems.
- Depending upon the replication engine used, read-only locks may not be replicated. This could make ineffective intelligent locking protocols designed to prevent deadlocks.
- Global locks such as mutexes held by one of the nodes may have to be implemented.
- Memory-resident context used by successive transactions may be insufficient if subsequent transactions can be routed to either node.
- It must be ensured that batch runs initiated by an application are not duplicated on the nodes.
- A means for distributing transactions between the nodes must be determined. This may have an impact on the application.

Distributed Management Tools

Managing a distributed system with distributed applications is a much more complex technical task than managing a single system. Good distributed system management and distributed application management tools must be licensed and your technical staff trained in their use.

Testing

Before deploying your new active/active solution, the system must be thoroughly tested. This could take many man-months.

Insurance

The additional site and equipment may increase your insurance premiums. On the other hand, the increased reliability and integrity of your applications may significantly reduce the amount of liability insurance that you must carry.

[5] Migrating Your Application to Active/Active, *Availability Digest*; March 2007.
http://www.availabilitydigest.com/private/0203/aa_ready.pdf
Appendix 4, A Consultant's Critique, *Breaking the Availability Barrier III: Active/Active Systems in Practice*, AuthorHouse; 2007.

Cost of Money

Finally, all of these costs must be financed. The cost of money should be a component of the overall conversion cost.

The Cost of Downtime

The cost of downtime is simple to calculate. It is the product of the value of a transaction, the transaction rate, and the expected downtime:

Cost of downtime per year = (value of tx) (tx/hour) (downtime hours/year)

Actually, this equation must be evaluated for every critical transaction handled by the application and these costs summed.

We say that this calculation is simple, and it is if you know the values to plug into the equation. Presumably, you know how much a transaction is worth to you and what your transaction rates are at different times during the day. One complexity is that a transaction's value may not only be measurable financially but may have other implications such as loss of customers, regulatory action, bad publicity, or the loss of life or property.

This leaves your downtime. Hopefully, you have kept a good record of your downtime over the years. This should have included the date, time, duration and cause of each outage. If you haven't done so, now is the time to start as you make an educated guess as to how much downtime you suffer in a year.

Cost/Benefit Analysis

Let us assume that you can get a handle on downtime parameters and the costs of going active/active. Is the move worth it? We consider two simple examples to demonstrate the cost/benefit approach.

Online Retail Store

The first example is that of a small online retail store. Its average sale is $100, and the store uses a standard markup of 20% for profit. Therefore, an average transaction is worth $20 to the store.

The store makes about 100 sales per hour and will consequently lose $2,000 per hour if its system is down. It is currently running an x86 Linux server that seems to go down twice a year and requires about five hours to bring back to life (sometimes a one-hour reboot, sometimes a one-day parts replacement).

Thus, it suffers about ten hours of downtime per year at a cost of $2,000 per hour. Therefore, its annual downtime cost is $20,000. This does not include ancillary costs such as the loss of customers, but the store feels that it is unique enough that its customers will remain loyal.

To go active/active, the store owner finds that he can lease time on a managed system in a local data center for $1,000/month. He is quite fortunate in that an expanded version of the third-party shopping cart that he is using is available to run in a distributed environment, but this will increase his application license fee by $800/month for the more powerful version running on two systems. He does not feel that he needs redundant networks, but a dedicated line required between the two systems for replication will cost another $400/month.

Thus, he can move to an active/active system for an estimated cost of $2,200 per month, or $26,400 per year. Since this will save him an estimated $20,000 per year, it is probably not worth

making the move. The one caveat is customer loss. If this is a real concern, the small increase in the cost of moving to active/active may be justified.

Brokerage System

Our next example is a major brokerage firm that uses a cluster of large Windows servers for its trading system. The system accepts online trades, forwards them to the appropriate exchanges for execution, and returns execution reports to the customers.

Its average trade is $50,000, on which it makes an average commission of 2%. Thus, each transaction is worth an average of $1,000.

The firm executes an average of five transactions per second, or 18,000 transactions per hour, during the trading day. Thus, its downtime cost is $18 million per hour!

That is why it runs its trading system on a high-availability cluster. Its experience over the last few years is that it is achieving almost five 9s of availability from its cluster, which is down for an average of 6 minutes (0.1 hours) per year. Thus, its annual downtime cost is $1,800,000.

The firm has determined that it will cost an initial $1,400,000 to move to an active/active system plus additional operating expenses of $250,000 per year. One interesting question to ask is to what extent would it have to reduce its downtime in order to have a one-year return-on-investment (ROI)?

Its cost of downtime for the first year, including facilities investment and operating expenses, is $1,650,000. Its annual cost of downtime, as calculated above, is $1,800,000. If it could reduce its annual downtime from six minutes to just 30 seconds, its annual downtime cost would be reduced to $150,000. This leads to a breakeven at the end of the first year. Thereafter, annual savings would accrue to more than a million dollars a year.

Our mythical firm implemented its active/active system and found that it is achieving less than ten seconds per year of downtime, which is a downtime cost of $50,000 per year. Coupled with the $250,000 increase in operating expenses, the firm is now spending $300,000 per year instead of $1,800,000 per year for downtime.

These are big numbers, but are they reasonable? September 8, 2008, just after the U.S. government announced its massive bailout of banks, promised to be one of the busiest trading days of the year. At 9:15 that morning, the London Stock Exchange crashed and didn't come back up until 4 that afternoon. Brokerage firms suffered an average of £700,000 each in lost commissions, totaling hundreds of millions of pounds over all of the London brokerage firms.[6] The LSE had recently switched from its NonStop systems to a network of PCs. If instead they had moved to an active/active system, millions of pounds would have been saved.

Summary

Determining whether or not to move to an active/active configuration to support your most critical applications requires some important input – what will the move cost you initially and in ongoing operational expenses, and how much does downtime cost you.

This information may require some effort to estimate accurately. But once you have these figures, the determination is quite straightforward. Knowing your initial and ongoing costs and your

[6] London Stock Exchange PC-Trading System Down for a Day, *Availability Digest*; October 2008. http://www.availabilitydigest.com/public_articles/0310/london_stock_exchange.pdf

estimated savings, you can determine your ROI and can then make the determination as to whether a move to active/active is right for your company.