"Achieving Century Uptimes"
An Informational Series on Enterprise Computing

As Seen in *The Connection*, A Connect 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 24: Is It Worth The Effort to Move to Active/Active?

September/October 2010

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

Active/active architectures eliminate all downtime, period. However, going active/active is a big step. Is it worth it?

It all depends. Moving to an active/active architecture is worth it if the savings to your company in downtime costs currently being experienced will provide an attractive return on the investment required to move to active/active.[1]

In this article, we look at what it takes to move a current application to active/active and what it might save you.

What is Active/Active?

As a brief review, an active/active system comprises two or more geographically-separated processing nodes cooperating in a common application. The database copies at each node are kept synchronized by data replication.

Should a processing node fail, all that needs to be done to recover is to route further transactions to surviving nodes. This can be accomplished in seconds. Furthermore, planned downtime for upgrade purposes can be eliminated by rolling upgrades through the application network one node at a time.

Consequently, active/active systems eliminate unplanned and planned downtime. If the cost of downtime is significant, the investment in active/active technology can bring significant returns.

The Causes of Downtime

Downtime can be caused by many factors, including failures in hardware, software, networks, power, and cooling. Operator error can cause downtime. An entire data center may be taken out of commission by a natural or manmade disaster.

One downtime cause that is not often considered is the downtime of your critical partners. If a partner outage can take you down, so far as your users are concerned, you are down.

[1] We will define the component costs of downtime later in the article.

Active/active architectures reduce the cost of downtime because they reduce recovery time from hours or days to seconds.

The Costs of Transitioning to Active/Active

An active/backup configuration cannot be converted to an active/active configuration simply by running the application on multiple nodes. There are several considerations that will probably add to the cost of your current system. Let us look at some of them.

Dual Data Centers

The first requirement is that there are at least two geographically-separated data centers, each containing one or more processing nodes in the application network. If you are already running an active/backup configuration, you probably are in good shape. If not, you have to plan a second data center with all of its attendant costs, including space, personnel, power and cooling, security, and so forth.

Redundant, Intelligent, and Fast Networks

The network you use to replicate changes between the database copies must be redundant, with load sharing or automatic failover for fast recovery. The loss of the replication network can be serious. If nothing is done, the separated processing nodes will continue to execute transactions; and their databases will diverge. Transactions may be processed differently depending upon to which node they are routed. This is called *split-brain mode*. Either split-brain mode must be acceptable in your application, or one node (or group of isolated nodes) must be shut down.

The user network should also be redundant and must support fast failover of users from a failed node to a surviving node if downtime is to be eliminated.[2]

Replication Engine

You will need to license a replication engine to keep your database copies in synchronism. If you are already running active/backup, you may currently be using data replication; but that replication engine may not be suitable for an active/active architecture. A suitable replication engine must support bidirectional data replication and must allow all databases to be open for read/write activity. Furthermore, the replication engine should be fast to minimize replication lag time.

There are two types of data replication – asynchronous and synchronous. Asynchronous replication[3] replicates data after the fact. With asynchronous replication, some data may be lost if a node fails. Furthermore, there is the possibility of data collisions, in which application

[2] Achieving Century Uptimes: Parts 22, 23 – Fast Failover with Active/Active Systems, *The Connection*; March/April and May/June, 2010.

[3] Chapter 3, Asynchronous Replication, *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, AuthorHouse; 2004.

instances in two nodes modify the same data object before each is aware that the data object has been modified by the other instance.

Synchronous replication[4] solves the problems of data loss and data collisions but imposes a performance penalty based on the distance by which the nodes are separated. This is because an application must wait for a transaction to be committed across the network before it can proceed.

A new replication technique to be commercially introduced in the near future is called *Coordinated Commits*. It solves these problems by combining asynchronous replication of data changes with transaction synchronization at commit time.[5]

Licenses

When configuring for active/active, you will probably find that your software license costs will increase. This is typically due to the fact that both nodes are now actively running the application rather than having one simply exist in a standby mode. In addition, there will be license costs for the replication engine(s).

Application Modifications

You must inspect your applications for structures that might prevent them from operating properly in an active/active environment and modify them to correct the problems.[6] For instance:

- There may be unique number generators (invoice numbers, for example) that will now be duplicated.

- Memory-resident context upon which the application depends, if any, must now be replicated.

- There may be transaction sequences that must be processed in absolute strict order. For instance, a transaction followed by a subsequent cancellation must be processed in order. Dependent transactions must typically be processed by the same node.

- The potential for data conflicts/collisions must be understood. This occurs if two application instances in different nodes attempt to lock or modify the same data item at the same time. With asynchronous replication, this results in data collisions. With synchronous replication, this results in distributed deadlocks. Regardless of the result, these must either be avoided or identified and resolved by the replication engine.

---

[4] Chapter 4, Synchronous Replication, *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, AuthorHouse; 2004.

[5] Achieving Century Uptimes Part 17: HP Unveils its Synchronous Replication API for TMF, *The Connection*; July/August 2009.

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

- Depending on the replication technology used, read locks may not be replicated. This can confuse an intelligent locking protocol that is unaware of a lock being held on a data item by another node.

- If the application opens files exclusively, the replication engine will not be able to open them and write remote changes to them.

- Periodic mini-batch runs scheduled by the application should run on only one node or be partitioned to support the active/active architecture.

This is certainly not a complete list. Some problems may only be apparent after testing the application in an active/active environment. If an application cannot run active/active, an option is to run it in an active/active infrastructure but only actively updating the database on one node. This is called a "Sizzling-Hot Standby." If the active node should fail, users can be switched to the standby node in seconds, preserving the downtime-elimination advantage of active/active systems.

Application and System Monitoring

Network tools for configuring, monitoring, and recovering distributed systems and applications must be acquired and system operators trained in their use.

Testing

Last but certainly not least is testing. Once you think that you have an active/active application ready to deploy, be sure to test it thoroughly. Operating procedures should also be well documented. Improper failover documentation can represent a single point of failure.[7]

The Cost of Downtime

The cost of downtime depends upon the application. Some applications can be down for days without serious impact to the company. Others can cause significant or catastrophic problems if they are down for just minutes or even seconds. It is this latter class of applications that should be considered for running in an active/active environment.

The cost of application downtime is one of the parameters that should be determined in the Risk Analysis phase of a properly executed Business Continuity Plan.[8] However, the "cost" of downtime is not measured only in dollars (or euros or yen or whatever) but can be a complex determination:

- Financial: The cost of downtime should include all real costs of an outage. These costs typically include lost revenues, employee overtime to recover lost work, SLA (Service Level Agreement) penalties, marketing expenses to recover lost customers, and compliance and regulatory fines.

[7] See Poor Documentation Snags Google, *Availability Digest*; April 2010.
[8] Business Continuity Planning: IT Examination Handbook, *Availability Digest*, October 2006.

- Customer Loyalty: Lost sales are only one impact imposed by customers. They may go elsewhere to satisfy their current need and may find the experience with their new supplier to be so good that they don't come back to you.

- Legal Liability: You may be subject to lawsuits if your outage has caused damage to another party.

- Publicity: You may suffer the embarrassing "CNN moment" when your outage is publicized by the press.

- Loss of Life or Property: At the far extreme is safety. If your system is one that protects life or property, it should be designed to never go down.

There have been several studies to determine the cost of downtime by industry. The results can be intimidating. For instance, The Standish Group has recently published a study of downtime. Some of its eye-opening findings include:[9]

Industry	**Hourly Downtime Cost**
Brokerage	$5,100,000
Home Location Register (cellular)	$2,100,000
Online Sales and Orders	$1,080,000
Enterprise Resource Planning (ERP)	$960,000
Customer Relations Management (CRM)	$600,000
Electronic Funds Transfer (EFT)	$450,000
E-Mail, Texting, IM	$300,000
ATM/POS	$240,000
Billing	$180,000
Social Web Applications	$6,000

Similar numbers have been published by the Meta Group.[10] The numbers are indeed attention-grabbing. But do they reflect your company's downtime cost? Do the demographics of the surveyed companies match yours? Probably not. For one, you may not be as big as the surveyed companies. These numbers do nothing more than to open your eyes to the fact that your cost of downtime may be much more than you think. If you haven't done so already, it is time to perform an in-depth Risk Analysis to determine what your downtime costs really are.

[9] The Standish Group, July 2010, Data Pinpoints.

[10] IT Performance Engineering and Measurement Strategies: Quantifying Performance and Loss, *Meta Group*; October 2000.

The Cost/Benefit Analysis

Let us look at some simple cost/benefit analyses as examples. We will focus on the financial cost – the other costs such as publicity and loss of customer loyalty must be separately considered.

To start, we note that the annual cost of downtime is easily calculated if we know the value of a transaction, the transaction rate, and the expected annual downtime. Cost of downtime is then simply

Annual Downtime Cost = Transaction Value × Transactions/Hour × Downtime Hours/Year

Case 1:

A small online retail store uses a Linux server that suffers ten hours of downtime per year. The store's average profit per transaction is $10, and it averages 100 transactions per hour. Thus, the store's hourly downtime cost is $10/transaction × 100 transactions/hour = $1,000 per hour. Its annual downtime cost is $1,000/hour × 10 hours downtime/year = $10,000. This business is probably not a candidate for active/active.

Case 2:

A large brokerage firm executes five transactions per second (18,000 transactions per hour). Its average trade is $50,000, and it charges an average commission of 2%. Thus, the firm's average transaction value is 2% × $50,000 = $1,000. It runs its trading application on a cluster, which is down an average of six minutes (0.1 hours) per year. Thus, its cost of downtime is $1,000/transaction × 18,000 transactions/hour × 0.1 hours downtime/year = $1,800,000 per year![11] This firm is definitely a candidate for active/active. (Note in this example that commission is revenue, not profit as in Case 1.)

Case 3:

A major North American bank reports that it handles about 10 million transactions a day over its ATM and POS (point-of-sale) network. If the average transaction is worth $2 to the bank in customer and merchant fees, then the bank's hourly downtime cost is 10,000,000 transactions/day × $2/transactions ÷ 24 hours/day = $833,000 per hour. No wonder so many ATM/POS networks are running active/active today.

[11] This particular downtime cost is pretty large (though it is consistent with The Standish Group's study). But is it realistic? On September 8, 2008, the U.S. government had just announced a massive bailout of troubled banks. It promised to be the busiest trading day of the year. At 9:15 AM, the London Stock Exchange's system crashed and didn't return to service until 4:00 PM that afternoon. It was estimated that the average brokerage firm lost about £700,000 in commissions, and that millions of pounds in commissions were lost in total. (See London Stock Exchange PC-Trading System Down for a Day, *Availability Digest*; October 2008.)

Case 4:

Telephone companies face a different problem. In some of their systems, the revenue loss due to a system failure may not justify an active/active system. However, of paramount importance to a telco is service availability.[12] Customers expect a dial tone to be absolutely reliable and a call to not get dropped. A large service outage can make the news headlines. Frequent outages can seriously tarnish their reputation, infuriate users, and lead to a loss of their customer base. This is why telcos turn frequently to active/active systems for their critical processing.

Summary

Moving to active/active can be a large undertaking. But its cost may be justified when compared to the downtime cost that it can save. Financial institutions, telecommunication companies, internet service providers, and health-care operations around the world have already made the move, many reporting zero downtime over years and in some cases over decades.

To determine whether active/active is right for your critical applications, you need to know two numbers:

- What will it cost me to deploy my applications in an active/active environment?
- How much will I save by eliminating downtime?

The first number comes from an evaluation of your current infrastructure and applications. The second comes from a detailed Risk Analysis of your critical applications. The effort of going through this exercise should well be worth it.

[12] Just think about the current ads for the major carriers claiming 100% reliability, such as the Verizon ad – "Can you hear me now? … Good!"