"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 16: Zero-Downtime Migrations for Active/Backup Configurations

May/June 2009

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

System or application upgrades are stressful! Even if you have a multi-hour or weekend maintenance window to make and test the upgrade, you have a *big-bang* leap-of-faith that the upgraded system will indeed come up. Thankfully, it does…and then it crashes! You must now fall back to the original configuration. But will the fallback work? Will you lose any data during this process?

With a technique known as *Zero Downtime Migration*, or ZDM, these risks can be avoided. With ZDM, a new environment (operating system, platform, database, and/or application version) is configured[1] and brought up-to-date with the production environment that continues to remain online and available to users! The new environment then can be tested for as long as needed in parallel with the active production environment. Finally, the upgraded environment's database is synchronized with the operational database while the operational system continues providing services. With proper testing, this means there will be no leap of faith that everything will actually work when the cutover occurs.

In our previous article,[2] we showed how ZDM was a natural capability of active/active systems that comprise multiple nodes actively cooperating in a common application. But what if you are not running active/active? Can you still enjoy the advantages of ZDM to eliminate planned downtime for upgrades? The answer is "yes," and in this article we show you how.

The Problem

Many systems today must be up 24 hours a day, 7 days a week. Continuous availability is needed for applications such as online shopping, online banking, ATM, POS, patient monitoring, nuclear system monitoring, telecommunications, global trading, and the list goes on. Even email applications must be always up for global enterprises. Not only is continuous availability a customer service issue, but in many cases it is also a regulatory requirement.

However, market forces dictate that these systems undergo major upgrades periodically to meet changing user needs and to take full advantage of advancing technology for performance improvement and cost reduction. If a periodic maintenance window is available, the upgrade may be done in that window; but what happens if the upgrade fails? If no maintenance window is available, and if the application simply cannot be taken out of service, what then?

---

[1] Note that it is possible in certain cases to configure the new environment on the production system, in which case a redundant system is not needed.

[2] Part 15: Zero Downtime Migrations for Active/Active Systems, *Connection*; March/April, 2009.

With ZDM, you can sleep better at night since ZDM eliminates the need for planned downtime and substantially mitigates the risk of cutover.

The Old Way

We call the current method of system upgrading the "Big-Bang" approach. The system is taken down during the maintenance window, typically at night or during a weekend. If the upgrade will impact the database – for instance, if the database schema is being modified – the first thing that must be done is to make a full backup copy of the database (or an incremental copy of the current database, saving all of the changes since the last full copy). The new environment is then set up, and the database is reloaded if necessary into the new environment. The new system is tested to the extent that it can be within the maintenance window.

When time is up, the new system must be put into service. If all goes well, the users are happy. But all too often, things do not go well.

So what are the problems with the "Old Way"? First of all, the Big-Bang approach requires that the system be taken down. Therefore, it cannot be used if continuous operation is a requirement. Given that a maintenance window is available for the upgrade, what if the testing performed isn't sufficient to test all system aspects? What if the upgraded system has problems and cannot be put into service? One must be able to fall back to the original system. But is this possible? If it is possible, will the fallback fail?

It is the limited test time and the complex and difficult fallback requirement that makes the Big-Bang approach so risky.

What is Needed?

In our previous article, we pointed out that three things are required to ensure smooth upgrades with no planned downtime:

- Redundancy so that the new environment can be brought up without affecting the current production environment.
- Fast failover so that system services can be switched from the current production system to the upgraded system without impacting the users and can be switched back quickly to the original production system if need be. In addition, it is usually very important not to lose the new data added to the upgraded environment when a failback occurs.
- Reliable failover so that it is assured that the switchover to the new system and, if needed, the fallback to the original system will work.

The New Way – ZDM

Using the technique of Zero Downtime Migration, applications can be available during the entire upgrade process, including the switchover to the upgraded environment. Except for new features, users will substantially be unaware of the switch to the new environment. Furthermore, should the new environment experience problems either immediately after the switchover or at some

later time, users can be returned to the original production system with little if any interruption in services and with no loss of data.

Active/Active ZDM – A Review

In our previous article, referenced above, we explained how active/active systems inherently achieved the elimination of unplanned downtime with ZDM. An active/active system comprises two or more nodes cooperating in a common application.

Users can be moved off of a node to be upgraded to one or more other nodes within seconds or subseconds, thus satisfying the need for fast failover. Furthermore, it is known that the other nodes are operational since they are already processing transactions, thus satisfying the requirement for reliable failover.

By the same token, users can be quickly and reliably switched back to the upgraded node or failed back if there are problems with the upgraded node.

Active/Backup ZDM

ZDM is not limited to active/active systems. Most highly-available systems today are not run in an active/active mode. Rather, a backup system stands by to take over operations should the primary system fail. The existence of a backup system satisfies the first requirement of redundancy. However, what is needed is fast and reliable failover in order to eliminate planned downtime. This is the role of ZDM.

Given that a redundant backup system does exist, the ZDM process proceeds as follows.

Step 1: Configure Environment to be Upgraded – Setup and configure the new environment on a redundant system (usually the backup system, though in some cases the new environment can be configured on the production system). The new environment will include the appropriate hardware, operating system version, database management system version, database schema, and application versions. It is a complete system ready for production.

Step 2: Load Test Database – Load a test database onto the system being upgraded. This could be a special test database designed for system verification, a snapshot of the current production database, the current production database or subset thereof acquired via an online copy, or any other database that will allow the new system to be thoroughly tested.

Step 3: Test, Test, Test – Test the system as long as needed. This could take days, weeks, or even longer. Thoroughly test not only the application logic but all interactions with ancillary systems (which usually support verification transactions to ensure operability). Proper testing must include testing at the full anticipated loads and beyond. During this extended test time, the users continue to be serviced by the original production system.

Step 4: Synchronize the Database – When testing is complete, and when the upgraded environment has been certified for use, the preparation for switchover is begun. The first step is to synchronize the upgraded environment's database with that of the production system.

This can be done with an online copy that will replicate the production database to the upgraded system while the production system continues in operation. There are several available products for accomplishing this. Alternatively, if the test database is a full copy of the production database, the production system can queue changes made during the test process and can drain these changes to the upgraded environment via data replication.

Step 5: Maintain Database Synchronization – When the current production database has been loaded onto the upgraded environment, it is now important to keep it synchronized. This is done by configuring a data-replication engine that will replicate all new updates made to the production system to the upgraded environment so that the upgraded environment's database is always an up-to-date copy of the production database.

Step 6: Verify the Database – As an option, it is wise to use one of the available Verification and Validation utilities to ensure that the upgraded database is indeed a viable copy of the production database, especially if there has been a change in the database manager or database schema.

Step 7: Configure Reverse Replication – To ensure that users can be failed back to the production system if necessary, configure reverse replication so that changes made to the database of the upgraded environment after users were cut over are replicated back to the original production system.

Step 8: Switch Over Users – At this point, the upgraded environment is ready to be put into production. Users can be switched over to the upgraded environment either en mass (admittedly a big bang, but onto a thoroughly tested system) or, if the database and user groups are logically partitioned, piecemeal by switching over one group of users at a time. If the data-replication engine is synchronous, or if it is asynchronous and can handle data collisions, users can be slowly moved over a few at a time to ensure proper operation. Regardless, switching users over in a controlled fashion allows you to check scaling as the load increases, hopefully avoiding any latent loading issues. When all users have been switched over, the upgraded system is fully in production. However, it may be wise to keep the old production system running with its database synchronized via the reverse replication channel so that users can be switched back to it if necessary.

Step 9: Fall Back If Necessary – If problems should appear in the upgraded system either during the cutover process or afterwards, users can be switched back to the original production system while the problems in the upgraded environment are corrected. Just as with cutover, fallback can be very fast and is reliable, with no data loss.

Step 10: Upgrade Original Production System - Once the upgraded system has been in production long enough to inspire confidence, the original system can be decommissioned and upgraded via the same ZDM process.

Thus, in terms of the requirements stated earlier, the switchover of users is fast because they are being transferred from one operational system to another. Both failover and fallback are reliable because the users are being switched back to a system that is known to be operating correctly.

A Case Study

A major Internet Service Provider (ISP) serves millions of users worldwide. At any one time, several million of these users may be logged on. Continuity of service is mandatory.[3]

Login requests were being handled by a sixteen-server Linux/Sybase Login Request Complex backed up by an additional sixteen servers. This complex had reached the limits of its capacity. Expanding it would not only be very costly, but the sixteen independent databases created a system-management nightmare. The ISP decided to move its Login Request Complex to a four-processor, NonStop 16200 active/active system. It had to do this with minimal impact to its customer base.

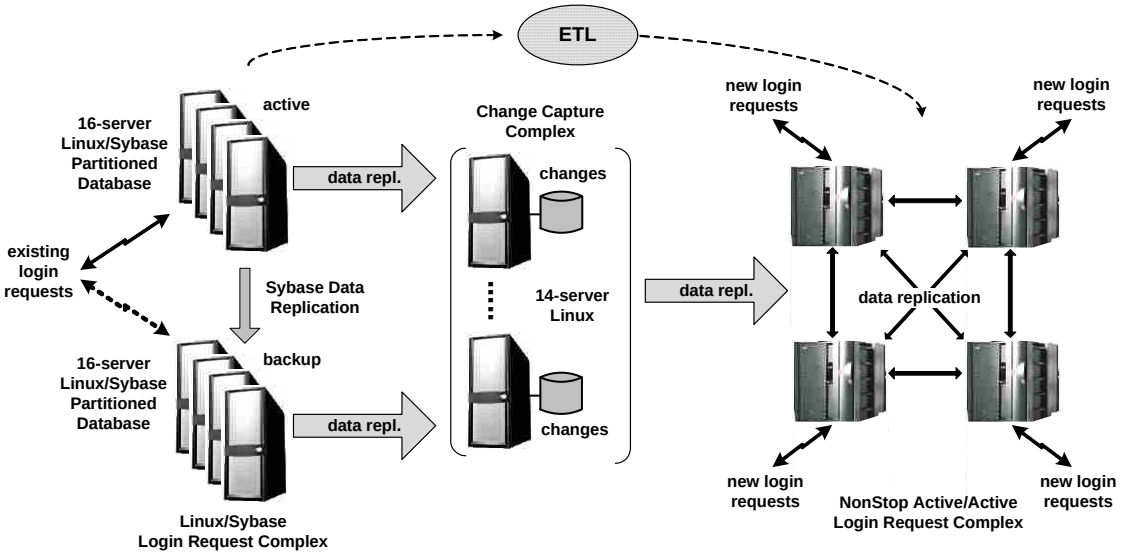

Figure 1: ISP Login Request Complex Migration

The first step was to provide for the capture of Sybase database changes by setting up a fourteen-processor Change Capture Complex using smaller Linux servers, as shown in Figure 1. As this complex received and stored new login updates, the Login Request Complex database was copied to the NonStop system via an ETL (extract, transform, and load) utility. Following the initial load, the changes that had accumulated during the load as well as new changes were replicated from the Change Capture Complex to the NonStop database, thus maintaining the NonStop active/active system's database in synchronism with the production login database. A Verification and Validation utility was used to compare the production and active/active databases to correct conversion errors.

At this point, the NonStop system was ready to be put into production. At first, only read requests were routed to the NonStop system. Update requests were still routed to the Linux/Sybase systems, with the changes being replicated to the NonStop system. After a period of satisfactory performance, new users were assigned to the NonStop system, which handled

[3] Major ISP Migrates from Sybase to NonStop with No Downtime, *Availability Digest*; November, 2008.

both read requests and update requests for these users. Finally, all user logon requests, both read and update, were routed to the NonStop system.

The migration proceeded cautiously over a period of months. Several hundred million user accounts were migrated to the NonStop system with no impact on user service. The new NonStop active/active system not only provides a unified database of all user accounts, but it is also easily expandable by adding nodes to the active/active system.

Summary

System upgrades without any planned downtime and with reduced business risk are being performed today via ZDM. By using data replication to keep the production and upgraded databases in synchronism, users can be switched between the two systems rapidly and reliably, eliminating the significant risk in the Big-Bang approach to system upgrades.