the Availability Digest

www.availabilitydigest.com

U.S. Bank Critiques Active/Active

May 2009

U.S. Bank manages its ATM network with a two-node, HP NonStop active/active system running ACI's BASE24. Having successfully gone through this implementation, the bank offers advice and encouragement for others looking for continuous availability in their mission-critical systems.

U.S. Bank

Minneapolis-based U.S. Bancorp (NYSE: USB), with $264 billion in assets, is the parent company of U.S. Bank, the 6th largest commercial bank in the United States. The company operates over 2,800 banking offices and over 5,000 ATMs. It provides a full line of banking, brokerage, insurance, investment, mortgage, trust and payment services products to consumers, businesses and institutions.

People around the world depend on ATMs for their banking needs anytime, anywhere. The availability of ATM networks is crucial to today's functioning society. In fact, as shown by recent incidents in this time of financial uncertainly, a brief failure in a bank's ATM network can trigger a run on the bank as depositors fear the bank's collapse.

Because of the importance of its ATMs, the availability of U.S. Bank's ATM network is crucial. A major focus within the bank's IT operations is to ensure continuous availability of this valuable service.

Continuous Availability for the Bank's ATM network

To meet this goal, the bank has migrated from its earlier active/passive architecture using a pair of HP NonStop servers to a full active/active configuration. Along the way, it learned many valuable lessons. We view the bank's migration path and the lessons it learned in this article.

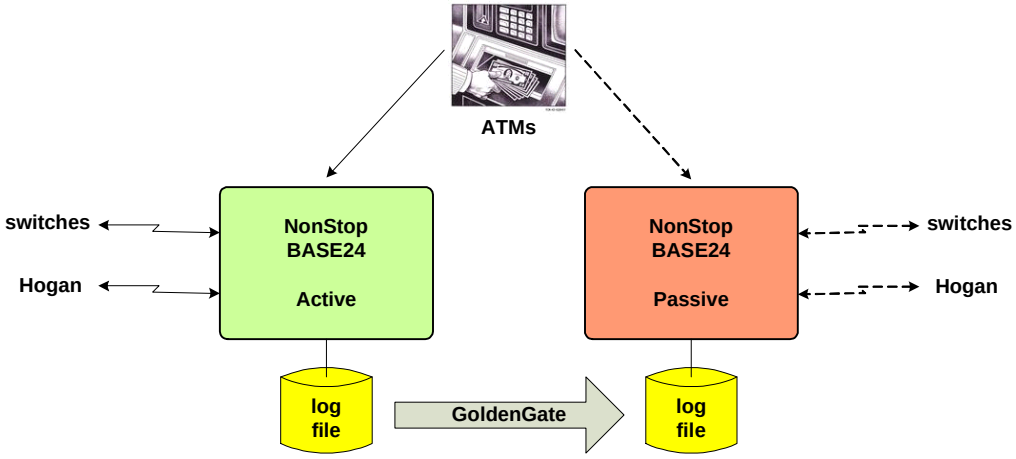
The Bank's Early High-Availability System

The bank's early ATM system comprised a pair of geographically separated HP NonStop servers configured as an active/passive pair. One server acted as the server actively processing ATM transactions; and the other server acted as a passive standby server, ready to take over should the active server fail.[1]

[1] Rich Rosales, The Road to Active/Active, Community-Connect Europe presentation; November, 2008. www.communityconnecteurope.org/education/Rich_Rosales_**USBank**.ppt

The ATMs were managed by BASE24 from ACI (www.aciworldwide.com). BASE24 provides a broad range of financial-services management functions, including managing ATM transaction routing and authorization. The BASE24 database on the passive node was kept synchronized with the active BASE24 database via log-based replication using the GoldenGate Transactional Data Management (TDM) replication engine (www.goldengate.com).

BASE24 managed the links to the various national authorization networks such as PLUS, CIRRUS, and VISA. It also connected to an IBM authorization switch running CSC's Hogan Core Banking System.



U.S. Bank's Early Active/Passive ATM System

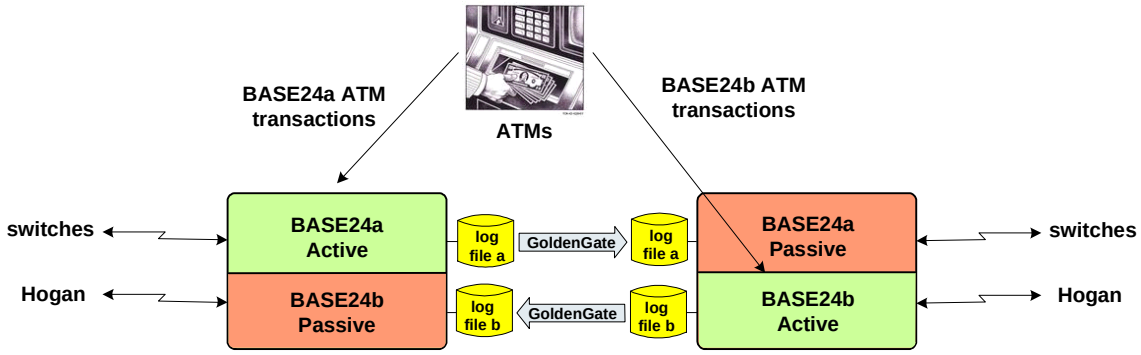
This configuration had its limitations. For one, the backup server's primary function was for disaster recovery in that it was prepared to take over transaction processing should the active server fail. To do so, it assumed control of the switch and ATM communication links and activated its copy of BASE24. It could also switch roles with the active server so that the active server could be upgraded with new hardware or software. Consequently, it was not available to share the transaction load. In effect, the bank had purchased two machines to do the work of one.

A major problem was that even though the backup server's database was synchronized with the active server's database, it still took anywhere from a half-hour to two hours to switch over. The active applications had to be taken down (if they had not failed), the communication links had to be moved to the backup server, the applications had to be restarted on the backup server, and the proper operation of the backup server had to be tested and verified before it could be put into service. If the verification process uncovered problems, they had to be corrected, increasing the switchover time even more.

Switchover to the backup server was not only time-consuming, during which the ATM network was down, but was risky as any unforeseen problem would only extend the outage. As a result, the bank was reluctant to make upgrades to the system. Perhaps even worse, the bank avoided testing disaster recovery because of the indeterminate ATM downtime that the system would impose.

Through an acquisition, the bank then acquired another ATM application that did not interact with its current network. The bank took advantage of this to put its passive server to work. The new ATM application was installed on what was the backup server, using what was the active server as its backup. Now both servers were processing transactions for independent applications, using the other server as a backup. Even better, if one server lost access to an authorization network, it

could regain that access by sending its authorization traffic over the NonStop Expand interconnect to the network connection on its backup server.



Expanding to Two Instances of BASE24

Still, if one server failed, it took the same half-hour to two hours or more to recover by switching the failed application to its backup. Competitive and regulatory pressures were growing to eliminate this downtime. New regulatory requirements dictated that disaster-recovery procedures be tested several times per year. Customers demanded 100% availability and enhanced functionality of the bank's ATM services. Continuous availability was needed.

Moving to Continuous Availability

How does a bank implement a sound disaster-tolerance strategy for mission-critical banking systems and achieve zero downtime – planned or unplanned? U.S. Bank decided to reconfigure its two NonStop servers as an active/active system to meet this need.

In an active/active system, all nodes in the application network are actively processing transactions for the same application. Each node has available to it a copy of the application database. The copies are kept synchronized via data replication. When one node makes a change to its database, that change is replicated to the other database copies in near real-time.

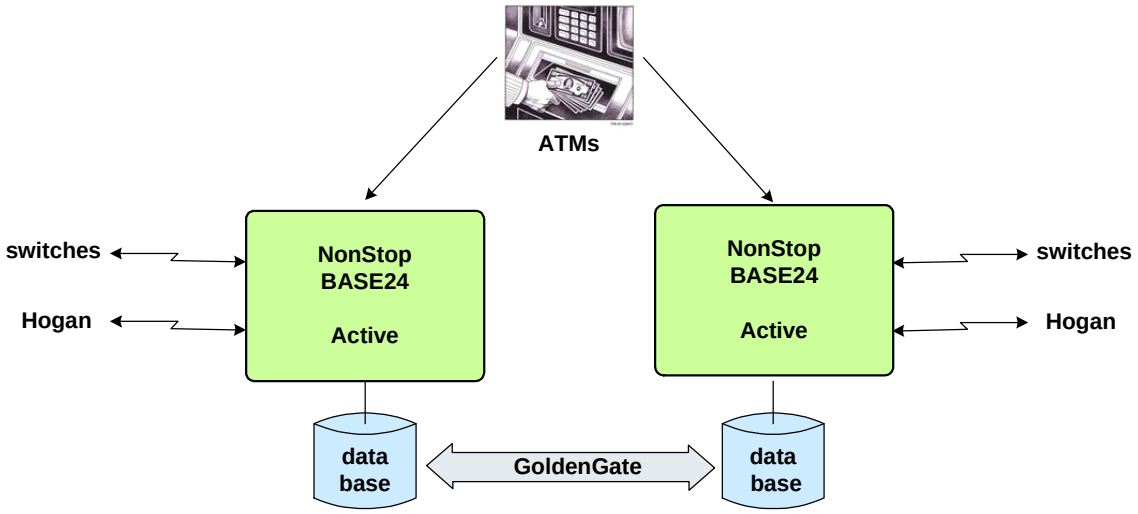
Unplanned failovers or planned switchovers for maintenance require only that users be rerouted to a surviving node, a process that generally can be accomplished in a few seconds. Therefore, failover or switchover can be accomplished quickly, resulting in almost imperceptible downtime to customers at the bank's ATMs.

Furthermore, failover is reliable since it is known that the other node is working – after all, it is currently processing transactions. Therefore, the risk of a failover or switchover fault is virtually eliminated. This allows the bank to feel comfortable making the upgrades necessary to keep up with customer services as well as to frequently test disaster-recovery failover.

Thus, by moving to an active/active configuration, the bank achieved its continuously available goals:

- It implemented a sound disaster-recovery capability with low downtime for testing.
- It was free to make frequent updates to meet competitive pressures.
- It ensured that the BASE24 applications stayed continuously available during any planned or unplanned downtime of either of the system's NonStop servers.

The resulting active/active system runs each BASE24 instance actively on both nodes. GoldenGate's bidirectional replication engine is used to keep the two databases synchronized. Therefore, an ATM transaction can be routed to either node for processing, thus balancing the load on the system. In fact, many of the national switches - VISA, for instance - round-robin transactions between the nodes for load balancing.



U.S. Bank's Active/Active ATM System

In addition to disaster tolerance and continuous availability during planned and unplanned downtime, an added advantage of the bank's active/active system is scalability. The bank currently processes one billion transactions annually, or about three million transactions per day. Should this load increase to the point that the system becomes too heavily loaded, additional nodes can be added to the system to scale it to the capacity needed.

The Biggest Issue – Data Collisions

The benefits of the bank's active/active approach to its ATM network management are impressive. But what are the challenges in this implementation?

Rich Rosales, BASE24 Development Manager for U.S. Bancorp, points out that data collisions are the most complex challenge. Unless the database can be partitioned so that only one node can update any given partition, data collisions will occur and must be dealt with. Such is the case in this application.

A data collision occurs when the two nodes update the same data item at roughly the same time. Each will update its local copy of the data item and will then replicate that change to the other node. The replicated changes will overwrite the original change made by each node. Now both nodes have a different value for that data item, and both are wrong.

U.S. Bank's approach to solving this problem was to categorize the many files in the system as dynamic or static (though its categorization was somewhat more complex than this). Dynamic files have to be maintained in real-time synchronization. An example of dynamic data is withdrawals so that up-to-date balances can be maintained. It is important that the application in each node know the current balance in an account so that it can properly authorize a withdrawal. These database updates must be replicated as quickly as possible (desirably in subseconds).

On the other hand, updates to card status (lost, stolen, inactive), PIN changes, and card limits can be replicated in a more relaxed way. The typical way to replicate these changes is to apply them as a batch run. Each node will process its own log files in a batch run, replicating its changes to the database of the other node.

The bank soon determined that a batch run and replication of the approximately three million static updates per day terribly slowed down replication during the batch processing, significantly increasing the chance of dynamic data collisions during the batch run. Furthermore, the batch latency increased the chance for data collisions in the static updates.

To solve this problem, the bank implemented a procedure that it calls *data targeting*. Rather than replicating individual changes or entire transactions, a transaction is broken down into its dynamic and static data. The dynamic data is deemed high priority, and the static data is deemed to be lower priority. A transaction's related dynamic data is replicated in real time as an event to the remote node. There it is applied to the remote database via user exits supported by the GoldenGate data-replication engine. The user exits implemented by the bank emulate the update business functions of the BASE24 applications.

The static data is batched and sent less frequently, for instance, every five minutes. In this way, the batch updates are not large enough to slow down replication significantly; and the lower batch latency significantly reduces the chance for data collisions in the replicated batch updates.

Date collisions in both the dynamic data and the batch data can still occur and must to be detected and resolved. Data-collision detection is performed by the GoldenGate replication engine. Collision resolution is resolved in some cases via standard methods provided by the replication engine or in other cases via specialized user exits provided by the bank. Depending upon the type of collision, one of several resolution methods are applied. For instance:

- The latest change is accepted. Examples of these collisions include card status and PIN changes. One requirement if time is to be used to resolve collisions is that both nodes must be synchronized with the same time. Otherwise, each can make a different decision; and the databases will be out of synchronization. That is, database corruption has occurred. To ensure proper time synchronization, one of the bank's NonStop servers has an atomic clock; and the NonStop time-synchronization facility is used to time-synchronize the other server.

- In some cases, GoldenGate's *deltas* capability is used. This capability allows an operation to be replicated rather than the end result. For instance, if one node wants to add 5 to a data item, and the other node wants to subtract 3, the +5 and -3 operations are replicated, resulting in a change of +2 at both nodes with no data collision.

- The resolution of many collisions is complex. To handle these situations, U.S. Bank developed their own in-house conflict resolution algorithms and embedded them as user exits into the data-replication engine.

In any event, all data collisions are logged for later manual review.

Database Verification

An important activity that is often overlooked is database verification. No matter how carefully the system is implemented and data collisions are resolved, database-update errors will occur. Though these errors may be quite infrequent, they lead to a contaminated database. The results of a transaction may then be different if executed on different nodes.

Update anomalies may occur because of administration errors, operating-system faults, subtle race conditions, or other faults. For instance, if data collisions are to be resolved according to time, it must be recognized that no matter how tightly the nodes in an active/active system are time-synchronized, synchronization is never perfect. Therefore, there is a small chance that two nodes will make a different decision as to which update to accept, resulting in database corruption.

Therefore, it is imperative that the databases be periodically verified by comparing them. Differences should be flagged and administrators notified so that they can take corrective action. This process is complicated by the fact that the databases are being actively updated during the verification process. There will be some differences that occur because of changes in flight and therefore do not represent database corruption.

In U.S. Bank's early active/passive system, there was no emphasis on database verification. In its active/active implementation, the bank uses GoldenGate's Veridata facility to periodically verify the databases.

Additional Advice

U.S. Bank makes the following observations for those contemplating a move to an active/active architecture:[2]

- Make sure that the application owners are involved early in the process. They can be a major help in identifying potential data collisions and crafting the strategy to resolve them.
- Learn the difference in active/active architectures.[3]
- Conduct a file categorization to determine the proper ways to replicate each file.
- Develop a strategy for out-of-sync detection, and test the resynchronization strategy.
- Review the network architecture with a goal of maximizing the effectiveness of that architecture.
- Determine the strategy to handle data collisions.
- Implement the project incrementally, one function at a time so far as possible.

To this we would add a detailed review of the application to find the "gotchas" that will cause the application to misbehave in a shared environment. Be prepared to modify the application where necessary.[4]

Summary

Rich Rosales summarizes his active/active journey nicely:

> "With this active/active system in place, there is no longer any need for us to take databases or software offline and deny service to our users during a system upgrade. Additionally, fast replication means fewer data collisions and less data loss, thereby allowing customers to experience the best service possible around the clock – which has always been our goal here at U.S. Bank. … Since the implementation, our customers have experienced no downtime, which helps us keep customer satisfaction and loyalty high."

He goes on to say:

---

[2] If I Knew Then What I Know Now: Implementing an Active-Active System in the No Downtime World of Banking, Rich Rosales, *The Connection*; January/February 2009.

[3] See the continuous availability seminars available from the Availability Digest: http://www.availabilitydigest.com/seminars.htm.

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

> "Active/active implementation can seem like a daunting task, but this should not discourage you from pursuing such a solution because the benefits are tremendous."

We here at the Availability Digest heartily endorse that observation.