the Availability Digest

Asymmetric Active/Active at Banco de Credito

November 2007

Banco de Credito e Inversiones (Bci) (www.bci.cl) is the third largest bank in Chile and serves 10% of Chile's population of 16,000,000 people. It offers various financial services, including personal, commercial, and corporate banking. Bci finances foreign trade and offers financial support for Latin American banks. It has 243 offices in Chile and has a presence in Peru, Mexico, Hong Kong, and the U.S. It also has several subsidiaries that provide services such as fixed income, financial trading, fund management, risk protection advice, international factoring, and recovery of bad debt.

Having introduced NonStop systems in 1993, Bci has been a technological pioneer in the Chilean banking industry.

The Challenge

Bci had chosen NonStop servers to run a critical proprietary checking account system developed in Cobol85 and imbedded SQL/MP. Bci had initially implemented its system with a pair of NonStop S76006s (each being a six-processor system). One system ran as the active system and the other as the standby system two kilometers away.

However, it became apparent that having a passive standby system was not in the best interests of the bank for several reasons:

- Maintaining an idle standby system did not maximize the value of the bank's IT investment.
- Over 300 batch processes had to be run every day. The batch processing requirements had grown as the bank had grown to the point that bank personnel had to work overtime. This violated a new human resources policy at the bank. If the standby system could be put to work, all processing could be completed during normal business hours.
- Moving to an active/active configuration would allow the bank to be compliant with new government regulations promulgated by the SBIF (loosely translated, the Superintendent of Banks and Financial Institutions, Chile's central bank).
- The separation of the standby site, only two kilometers away, was not considered a sufficient distance for man-made or natural disaster tolerances. A distance of at least 100

kilometers was needed. After all, the largest earthquake in recorded history, a massive 9.5 on the Richter scale, occurred in Chile.

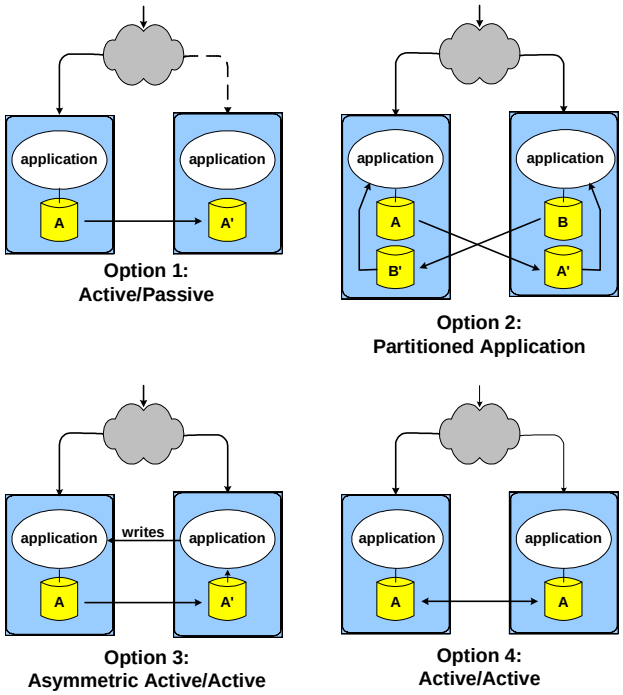
Therefore, it was decided to reconfigure the bank's two systems into an active/active architecture so that both could be actively processing online transactions during the business day.[1]

The Options

The bank considered four alternative architectures before making a choice.

Option 1: Active/Passive

This is the architecture with which the bank started.



Option 2: Partitioned Application

With this architecture, both systems have a complete copy of the database, and the application is running on both systems. However, the database is partitioned so that the two instances of the application, one running in each system, can be processing transactions simultaneously without data collisions. One of the application instances processes transactions against the data in one of the partitions, and the other processes transactions against the other partition.

As changes are made to one partition, they are replicated to the copy of that partition on the other system. Therefore, both systems have a complete and current copy of the database and consequently have access to all of the data in the database for query purposes. There can be no data collisions, as there is no way for both application instances to be processing the same data item due to the database partitioning.

Option 3: Asymmetric Active/Active

In an asymmetric active/active configuration, one system is the master, or principal, system; and the other system is the slave system. Both systems have a copy of the current application database.

All transactions that require database modification are routed to the master system, where they are processed and the database updated. All updates made by the master system are replicated to the database copy on the slave system so that both systems maintain a current copy. This allows either system to process any transaction that requires read-only access to the database. Since many applications are read-intensive (typically 80% read and 20% write in OLTP applications), this allows the processing load to be distributed evenly across the nodes.

[1] The material for this article was taken from a presentation given by Juan Pablo Nahuel Alvarado, Bci System Architect, entitled "An Asymmetrical Active/Active Implementation at Bci," given at the 2007 HPTF conference in Las Vegas.

Routing of write transactions to the master node can be done by intelligent routers, an intelligent front-end system, or by the applications themselves. Since only the master system can update a data item, there can be no data collisions.

Option 4: Active/Active

In a full active/active configuration, any system in the application network can process any transaction against its copy of the application database. Any changes that it makes to its database copy are replicated to the other database copies in the network.

In this configuration, all nodes are equal. If asynchronous replication is used, there may be data collisions caused by two systems trying to update the same row at the same time. If they can occur, data collisions must be detected and resolved.[2]

The Choice

Option 1, an active/standby configuration, was already implemented and was not a viable choice. The bank decided not to follow Option 2, application partitioning, since its applications were complex and the data highly interdependent. This prevented any consideration of partitioning the database and therefore precluded this choice.

Likewise, Option 4, a fully active/active configuration, was ruled out. It was felt that the complexities of bidirectional replication and data collisions would require special application management facilities in order to balance the accounts.

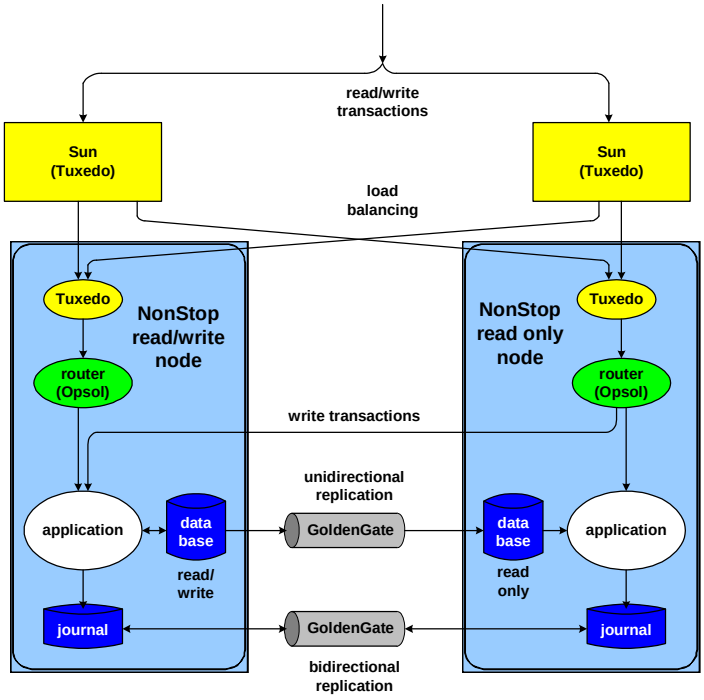
The bank's choice was Option 3, asymmetric active/active. This configuration allowed it to go active/active with a minimum of application changes. However, Bci sees this as an intermediate step toward achieving full active/active processing as it gains experience in this arena.

The System

System Architecture

In Bci's asymmetric active/active configuration, one of its NonStop systems acts as the read/write, or master, node and the other as the read-only, or slave, node. In the future, additional read-only slave nodes may be added to provide a scale-up of processing capacity. All write transactions are forwarded to the master node for processing, whereas a read-only transaction can be processed by either node.

The two NonStop systems are front-ended by a pair of Sun systems for transaction routing. The routing of transactions is done intelligently by the Sun systems so as to maintain a balanced load on both systems. However, the Sun systems cannot distinguish between write



[2] Collision Detection and Resolution, *Availability Digest*; April, 2007.

transactions and read transactions. Therefore, they distribute all transactions among the NonStop nodes evenly.

Tuxedo is used to move transactions from the Sun systems to the NonStop systems. There they are passed to the intelligent RTFS router, provided by Opsol Integrators (www.opsol.com), that can distinguish write transactions from read-only transactions. The Opsol router in the master node takes no action. It passes all transactions to its copy of the application for processing.

However, the router in the slave node will not pass write transactions to its application copy. Rather, it sends these transactions to the application copy on the master node. Read-only transactions are sent, however, to its local application copy.

The application copy in the master node processes all write transactions, thus updating its copy of the database. These updates are replicated to the slave's database copy to keep it in synchronism with the master database copy. Each node processes against its local database copy any read-only transactions that have been routed to it.

The application maintains an application journal that tracks all of its processing activities. Each application copy maintains its own journal. All entries in either journal are replicated via bidirectional replication to the journal on the other system. In this way, both nodes have a complete copy of the application journal. Since all updates to the journals are inserts only, there can be no collisions.

Bci uses GoldenGate's data replication engine (www.goldengate.com) for both the unidirectional replication of the application database and the bidirectional replication of the application journal.

Failover Procedures

There are several failure modes to consider in this asymmetric active/active configuration.

Slave Node Failure

Should a slave node or its attached database fail, the system simply continues unaffected (except for its available capacity). The Sun routers now route all transactions to the master node.

The master node, which had been processing write and read transactions, continues to do so with no impact on the users except perhaps for slower response times due to the increased load which it is carrying.

Prior to returning the slave node to service, its database must be resynchronized with the master's database either by draining the queue of updates that had accumulated during the outage or by copying the master database to the slave database.

Master Node Failure

Should the master node fail, the slave node must be promoted to master. It will now handle all transactions – both read and write.

When the failed system is returned to service, its database must first be synchronized with that of the new master as described above. At this point, operation in asymmetric active/active mode may continue uninterrupted with the old slave node now acting as master. Alternatively, the role of the master and slave may be reverted to the original configuration.

Communication Failure

A communication failure can occur in several ways. It can isolate one of the systems so that it is not receiving any transactions. In that case, the isolated system should be taken out of service until communication is restored. The surviving system carries on as master.

A communication failure could prevent the slave from sending write transactions to the master. In this case, the slave must be taken out of service and all transactions routed to the master.

A communication failure could also interrupt replication. In this case, the slave could be taken out of service; or in some applications it could perhaps continue as a query node, albeit with a database that will become more out-of-date with time.

In any case, the database copy of the system being restored must be resynchronized with the online database before that system can be returned to service.

The Implementation

Proof of Concept

Bci wisely divided this complex project into two distinct phases. One was a proof-of-concept phase, in which it tested the architecture and the various components that had to be developed to support the new system. Testing included:

- the router integration into the Bci transactional environment.
- router failure modes, including manual and automatic failover, and the change in the system's personality (from slave to master, or to master with no slave).
- the backup recovery process under all failure modes such as system, database, and communication channel failures.
- database synchronization using the GoldenGate data replication engines.
- automatic reactions of the system to all possibilities of simulated failures.

As it turned out, only minimal changes to the application were required.

From the start to the end, Bci accomplished the proof-of-concept in two months.

Project Implementation

With the proof-of-concept experience behind them, and with a closely knitted team of Bci employees and those from HP, Opsol, and GoldenGate, the system went live in just four months. This included all program modifications to accommodate the routing mechanisms and the database replication issues, the creation of automated scripts, and thorough testing of all failure conditions.

The system then underwent extensive testing and certification. This included tests of all failure conditions and the verification of all automatic scripts. Certification of the system took an additional two months before the system went operational.

Recommendations

As a result of its successful experience, Bci recommends the following to any organization wanting to follow in its footsteps:

- Isolate the working team from normal day-to-day operations. It should be 100% dedicated to the task of building the new system.

- Provide a common work space for the development team so that they can have continual face-to-face contact.

- Create and maintain a detailed project plan that focuses on project delivery dates.

- Perform intensive load and stress testing on the system before putting it into service.

- Bci developed the SQL queries using SYSKEY. However, for today's fast processors, SYSKEY values across the network might not be the same for a query or unique across queries (they are based on time calculations). This parameter must be managed inside the application to ensure its uniqueness and its consistency across the nodes.

Summary

This implementation by Bci shows the viability of migrating an existing complex application to an active/active environment. Such a move can provide improved utilization of systems, vastly improved application availability, and rapid (almost instant) failover.

As seen here, the first step is in clearly understanding the application. Bci considered several options for active/active configurations and was guided to an appropriate architecture by its knowledge of its own application. Also of immense importance was its detailed understanding of active/active architectures and issues.

Interestingly, we have written much on the use of asymmetric active/active configurations to offload extensive query processing to slave nodes.[3] This case study shows another powerful use of this architecture – migrating an application that would require significant modification to move to a full symmetric active/active configuration. In Bci's case, it avoided the need for significant application modification by moving to an asymmetric configuration.

[3] P. J. Holenstein, B. D. Holenstein, W. H. Highleyman, Chapter 14, Benefits of Multiple Nodes in Practice, *Breaking the Availability Barrier III: Active/Active Systems in Practice*, AuthorHouse; 2007.