High Availability – A Case Study

White Paper

Published: May 2000 (with revisions)

When “Great” Isn’t Good Enough: Design and Deployment of a Very High Availability System

Contents

Preface

It’s five-thirty in the morning and the telephone is ringing you awake. The system is down. This is more than an inconvenience, or an unpleasant way to start your day. The east coast, three time zones ahead of you, has been working for the last hour and a half and is trying to service customers, take orders, and promise critical shipments. No, this isn’t just a rushed breakfast – this is your company losing millions of dollars in lost orders and lost customers. How can you avoid having a really bad day? This is a case study of a high availability system that was designed and implemented for one of our customers.

Overview

More and more, companies are faced with the need to support users in several time zones, often spanning several countries. This may be in addition to the use of multiple work shifts. The ability to provide access to the production system(s) is paramount. This creates an increased burden on both the systems (hardware and application), and on the operations department. Downtime can be very expensive when factors such as: labor costs, lost sales, late delivery, and lost opportunity, are taken into consideration. This can be a very difficult and expensive problem to address.

In the summer of 1999 we were asked to help design a high-availability Ingres solution for a mission-critical OLTP and Reporting environment. This was the ideal project, since we were told to design the "perfect system, without consideration for cost". Once the generic specifications were created a RFP was sent to several major hardware vendors. Final selection was based on a combination of: performance results from benchmarking, overall price, and the hardware vendor's willingness to absolutely guarantee the success of this effort.

This site supports over 500 interactive users at any time, with over 750 connected DBMS sessions most of the time. Transaction volume can exceed a million transactions per day. The cost of downtime for this Client is several million dollars per day. This site supports near 24 x 7 operations for sites located in several countries. In the past, they experienced system outages that were now deemed unacceptable, due to the impact on the business.

Our objective was to design a system to accommodate their needs now, and for the next 2-3 years (which this system has done). The primary goals for the new system were to improve performance, which had steadily degraded as their business increased, and to provide capacity for future growth.

Another stipulation was that the new system could not have more than 2 unplanned outages per year (each no more than 2 hours long) – no matter what the root cause. This meant that we also needed to validate all software being used, patches (operating system and application software), and interfaces. A analysis of the computer room capacity (power and HVAC) was included as this effort in order to manage all identified risks. The effort had to be a complete success.

It should be noted that whenever there is concern about high availability, there is (or should be) a concern about Disaster Recovery and Business Continuation. For more information on Disaster Recovery, please see our white papers on the subject.

It should be noted that while this project involved the Ingres RDBMS product, it could have just as easily been another RDBMS like Oracle, or even an Application Server like WebLogic. The basic principles are the same. What differs is the specific implementation, and custom scripts required to make this project a success.

Our Design

For performance purposes, we decided to separate the OLTP activity from the reporting activity. Our primary production machine is designated as "OLTP", and handles all OLTP-related activity and any reporting that requiring current data. The backup machine is designated as "reporting", and handles all reporting activity that does not require current data. The reporting database is refreshed on a nightly basis. Any reports that require current information are run against the OLTP installation.

In the event of a failover condition, both OLTP and reporting activity occur on a single machine. High availability is maximized through the use of redundant hardware. With the sole exception of the system clock, every component in the machine is redundant and/or can automatically failover to another device.

Having a failover machine, identical in configuration to the production machine, provides failover protection. Each machine has a high performance fiber-channel disk array attached to it. Additionally, the arrays are mirrored to each other and cross-connected between the two machines. Through the use of floating IP addresses, we can quickly have the failover machine assume the identity of the primary production machine.

The system is designed with performance in mind. Multiple sort/work locations (4), and data locations (10) are used. All Ingres tables and indexes are striped across all locations (indexes and tables are striped in opposite directions - tables from location 1 through 10, and indexes from location 10 through 1). There is a very miniscule amount of overhead associated with this configuration (wasted INODES and additional file descriptors). This is because only 60% of the tables and/or indexes *need* to reside on all ten locations. Our opinion is that the benefits achieved from standardization, easily justify this small amount of overhead.

High availability is guaranteed by the minimization of Ingres downtime. We specifically targeted the following:

- Time required for database checkpoints (backups).
- Time required for ongoing maintenance (modification of tables and indexes, etc.).
- Proactive detection and resolution of potentially downtime causing problems.

Checkpoints are performed using customized checkpoint and rollforward scripts, permitting the 10 database locations to be written in parallel to 10 checkpoint locations on disk. The main reason behind this approach is to reduce the amount of time the database is unavailable to the users. Using this technique, check pointing a 30 GB database takes approximately minutes, and a full recovery typically takes approximately 45 minutes (including the application of journal files).

Maintenance time is kept to a minimum primarily through the speed of the production machine and its disk arrays. Most tables can be modified within 5-10 minutes. The largest of the production tables (~6 GB) can be modified in under 20 minutes. We are able to modify all tables, and recreate all indexes on all tables in the database, in 2 hours or less!

Better Support through Automation

Several production monitoring and support scripts were developed to support this environment. They are run in both "OLTP" and "Reporting" environments, in order to proactively detect and report problems. The use of these scripts relieves the support staff from performing manual monitoring of the system. <u>The scripts are smart enough to detect most common problems, and robust enough to be easily modified if new conditions or thresholds are detected.</u>

Our "ingmon" script is used to monitor the health of the Ingres installation. At two minute intervals, this script checks a number of Ingres installation parameters such as: log file utilization, number of Ingres processes running, and presence of critical errors in errlog.log. If problems are detected, a support staff member is paged with a description of the problem. A log of all problems is automatically maintained by this script.

Our "long run" script runs every 5 minutes and monitors queries within the DBMS servers. If a query is seen to run beyond a predefined threshold, email is sent to the DBA staff containing the text of the query, and the session information. The DBA staff can then take appropriate action to correct bad queries.

Our "PingProd" script is run every 5 minutes to check connectivity between the "OLTP", "reporting", and one of the key client nodes. This script uses the standard "ping" utility, to attempt to reach the other nodes on the network. If a node is unreachable, a page is sent to the operations group indicating a problem.

Our "ChkDistributedRpt" script runs every 5 minutes to verify the reporting database is available. It determines this by connecting to "reporting" from "OLTP", and attempting to select from the reporting database. If this fails, a page is sent to various members of the DBA staff.

Customized startup and shutdown scripts are used to enable and disable the various monitoring scripts. For example: the ingmon, PingProd, and ChkDistributedRpt scripts, all monitor the $II_SYSTEM/ingres directory for a flag file indicating the state of the Ingres installation. This flag is updated any time a normal startup or shutdown takes place. If the installation is in a normal down state, the monitoring scripts will not perform their installation checks (keeping the scripts from generating unnecessary pages). In addition to updating the flag file, the customized startup and shutdown scripts’ start-and-stop site specific daemons and interfaces. The startup script is also responsible for saving the Ingres log files to an archive directory, and clearing the contents of these files out, before restarting Ingres.

Finally, we created custom wrappers for the "rollforwarddb" and "destroydb" utilities. By default, these utilities make it too easy for a problem to occur due to carelessness. Since this is a mission critical system with high reliability requirements, we felt *everything* needed to be viewed as a potential cause of downtime, and therefore mitigated the risk as much as possible by "building fences" around the database.

Database Mirroring

The linchpin of the new system is the use of database mirroring. This needs to work properly in order to support the failover procedure, as well as to make possible the offloading of the reporting activity to the "reporting" machine.

The "OLTP" array holds two sets of the Ingres filesystems (data locations, sort locations, $II_SYSTEM, journal location, dump location), which are kept in sync through a hardware mirror. Additionally, it holds the operating system disks. No disks are internal to the servers - all Ingres and OS data are stored on the external arrays.

The "reporting" array holds two sets of disks - one is used for the reporting database, and the second is a software mirror of the OLTP database. This software mirror serves two purposes.

1. In the event of a catastrophic failure of the primary array, a current copy of the production database is already available to the failover machine.
2. The software mirror is used during the nightly backups, to synchronize the "reporting" database from the "OLTP" database.

In the current configuration there are several filesystems that are clustered (that is, shared between the two machines). These are primarily OS related filesystems, such as /usr/home. This is required by the particular hardware vendor. In general it is not a problem, but does add a little bit of complexity to the environment, with regard to things like common scripts and shell history files.

Reporting System Synchronization

This takes the "OLTP" machine offline for approximately 1 hour, and the "reporting" machine offline for 2 hours. Technically, we could do this with less than 5 minutes of production downtime through the use of online checkpoints (database backups), but we felt the extra margin of safety afforded by the offline checkpoints, was worth this window of downtime. This can easily be changed in the future if required.

During the synchronization process the following steps are taken:

1. The reporting subsystem is alerted to the fact that the "reporting" machine is going offline through an update to a configuration table. When this flag is set, all reports are run against the "OLTP" machine.

2. Ingres is brought down on the "OLTP" machine (to remove any active sessions), and then restarted in *maintenance mode*. This special mode bypasses the normal site-specific startup tasks such as: starting background daemon processes, resetting reports, and verifying database integrity. It also excludes user access to the production databases.
3. Offline checkpoints are taken of all databases on the "OLTP" machine.
4. The software mirror to the reporting array is fractured.
5. A flag file (manual semaphore) is created to inform the "reporting" machine that the disks are ready to be synchronized.
6. Ingres is brought down from maintenance mode, mirrors are fractured (a way of breaking the mirrors, providing for quicker re-synchronization), and then Ingres is brought back up in *normal* mode and production users are allowed back on the system.

<u>On the Reporting / Failover machine:</u>

1. Once the flag file is detected, Ingres is brought down.
2. A disk copy is performed to copy the data from the production mirror set to the reporting disks.
3. A post-synchronization script is run. This script resets files and variables, in order to allow the Ingres installation to start and run on this node.
4. Ingres is restarted.
5. A script is run to tell the OLTP installation that the "reporting" installation is available, and to start directing reports to it.
6. The checkpoint disks are copied to tape.

Was it Easy?

No! We encountered problems making the hardware / OS perform exactly in the manner we needed it to. By developing numerous scripts, we were able to finally get the system to behave consistently as expected.

We encountered architectural issues with the RDBMS, limiting performance and concurrency. Luckily, the Level 2 Support group was as interested in making this project a success as we were, and worked closely with us to isolate problems and create reproducible test scenarios. They made several architectural changes (several that were direct requests), providing benchmark results that exceeded our expectations. These changes are now part of the standard product.

So no, it certainly was not easy; but yes, it was well worth the effort. The system remained in production for five years and met the service levels established. Ongoing management and capacity planning was a large part of the success of this system for that period of time, especially since the business experienced significant levels of growth over that same period.

The Fruit of Our Labor!

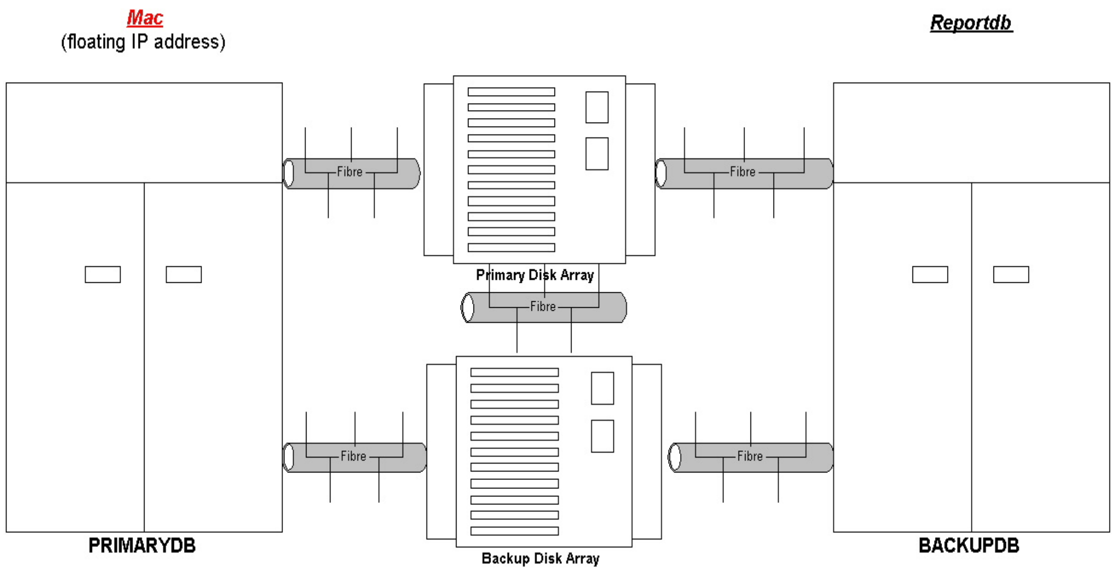
This configuration has been in production since January, 2000 and works extremely well. The full migration and production cutover took less than a weekend!

The users have been extremely happy with the changes - average response time for OLTP transactions has dropped, as well as the average report runtime. The operations staff is extremely happy, as the new setup has allowed them the flexibility to direct resource intensive ad hoc query requests to the reporting database, rather than risk impacting the OLTP environment. Management was pleased, as we were able to deliver on all aspects of their request.

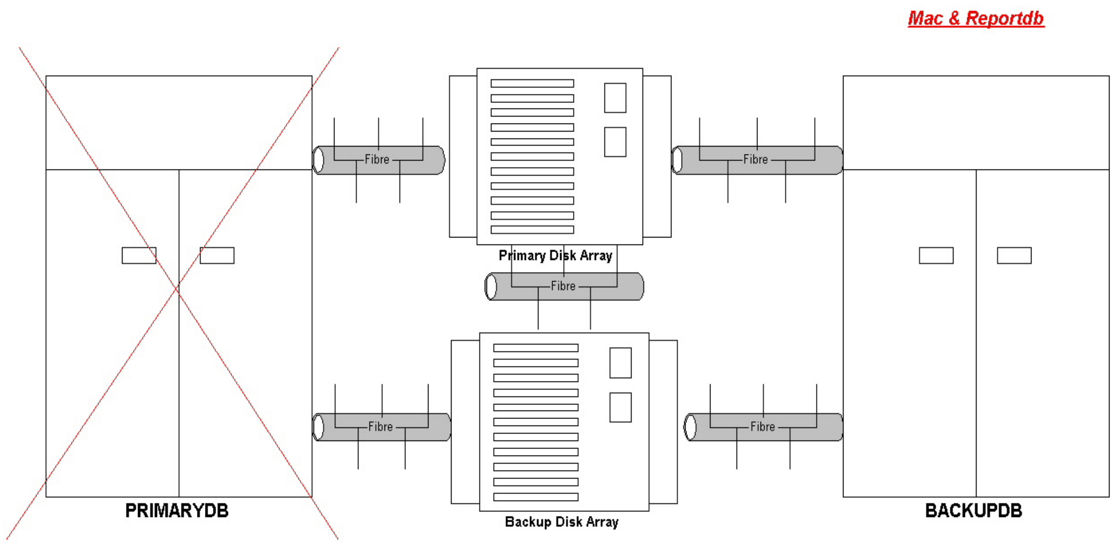
Please feel free to contact us at (info@Comp-Soln.com) if you have any questions or comments about this white paper.

<A diagram of the High Availability System follows on the next page>

Normal Ingres Cluster Operations



Failover Ingres Cluster Operations



About the Author

Chip Nickolett, MBA, PMP is the President of Comprehensive Solutions, and has over 22 years of experience in the Information Technology including programming, systems analysis & design, database design & administration, and project management & implementation. He has designed and implemented several high-performance and high-availability systems for many companies over the years. For more information please see http://www.Comp-Soln.com/chipn.html.

Let Us Help You Succeed!

Call today to discuss ways that Comprehensive Solutions can help your organization save money and achieve better results with your IT projects. We provide the ***confidence*** that you want and deliver the ***results*** that you need.

View a white paper on a different HA implementation

Back to White Papers

Back to Services

Comprehensive Solutions
4040 N. Calhoun Road
Suite 105
Brookfield, WI 53005
U.S.A.

Phone: (262) 544-9954
Fax: (262) 544-1236