

High Availability - A Case Study

White Paper

Published: September 2006

High Availability on a Budget

Contents

Preface	1
Project Background	2
Project Planning / Management	3
Applications	4
Database	4
Interfaces	4
Cluster Operations	5
Hardware	7
Cluster Management	7
Documentation	8
The End Result	9
About the Author	9
Let Us Help You Succeed!	9
Supplemental Hardware Overview	11

Preface

In early 2004, we were approached by one of our clients to build a high availability (HA) cluster to manage their warehouse operations. There were several constraints on this system - the hardware for this system was (with some notable exceptions) leftover and reassigned hardware from defunct or end of life projects for this company. We later learned that an acquisition was in the works, so the combination of improved performance and cost reduction were very critical.

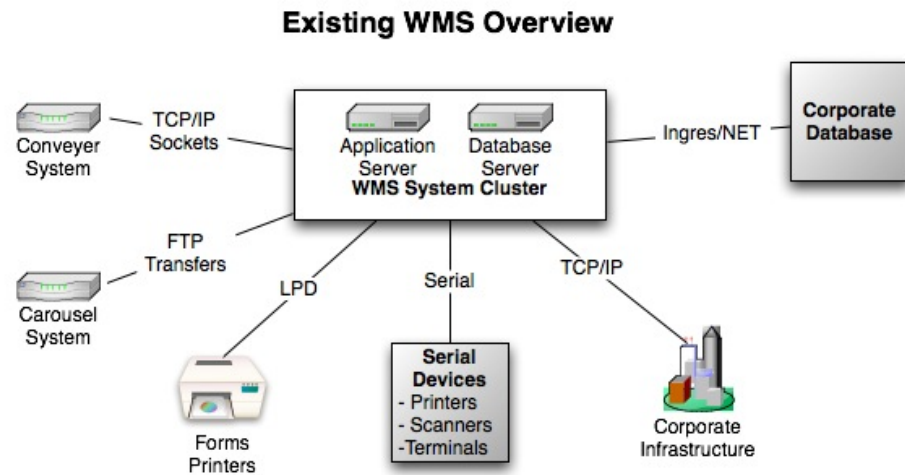
The budget for software and additional hardware was low to non-existent. We had to make do with the unused equipment we could find in the corporate data center. The overriding requirement was that the system needed to be constructed so it could be managed by the on-site IT resources that had minimal experience with Unix, Ingres (the RDBMS in use), and the custom applications and interfaces. The dependency on the corporate IT infrastructure, including staffing, was to be minimized.

Project Background

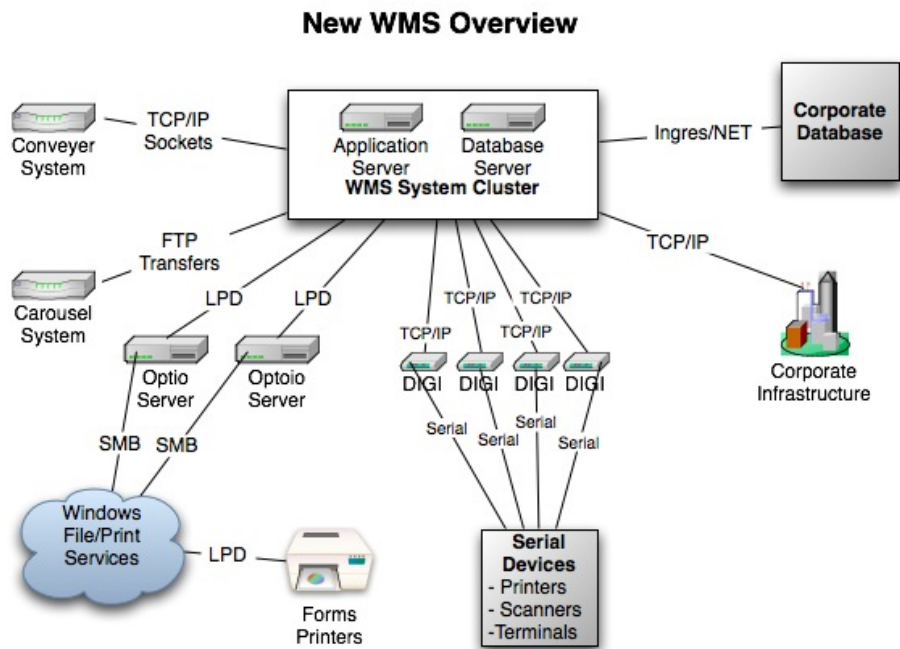
The project was executed over a two month period in 2004. One complication was that other than the Comprehensive Solutions team, the rest of the project team were primarily tasked to other efforts. This resulted in competition for time and effort between this project and other projects and tasks, especially with regard to scheduling. Another complication was that the resources used for the project reported to five separate managers, all of whom had varying degrees of interest in and involvement with the project. Luckily, we were also managing this and other projects and had some control over resource utilization and availability.

The existing WMS (Warehouse Management System) was a custom package written entirely in Ingres ABF (application by forms, a character based 4GL development environment) and ESQCLC (SQL embedded in C code). This was back ended by an Ingres database running Ingres II 2.0. The hardware configuration used was a two-node Data General AViiON cluster which communicated to a shared disk array through a SCSI hub. The nodes were designated as “application server” and “database server” and had been reconfigured by Comprehensive Solutions a few years earlier to support single-node operations in the event of a failure.

The warehouse floor was interfaced into the system via serial devices – terminals, scales, bar code readers – as well as through FTP and socket communications for the warehouse conveyer system and carousel system control.



The new cluster would need to run the existing WMS application, communicate with the carousel and warehouse conveyer systems, and interface with the existing warehouse floor hardware. Additionally, the existing Ingres database would need to be migrated to the new platform that was not binary compatible. As support on Ingres II 2.0 was in sunset, the migration would also include an Ingres upgrade to the most current supported version (at the time this was Ingres 2.6/SP1). A final requirement was that the cutover to the new system could take no longer than one weekend, including verification/validation of the existing data and testing. This meant that our migration, test, and failover plans would require even more optimization and coordination than usual to ensure that we achieved our goal.



Project Planning / Management

The overall project was divided into five major areas of responsibility:

1. Applications
2. Database
3. Interfaces
4. Cluster Operations
5. Hardware

The applications area included the WMS - the ABF application, ESQCLC supporting apps, all ancillary shell scripts, configuration files, directory layout, and complete end-to-end testing. Database systems included the Ingres installations on both nodes (client and server), as well as the configuration, backup and recovery testing, performance tuning, and data migration from the existing system. Interfaces included the serial device management (which would be migrated to use DIGI Serial/IP adapters as opposed to the multi-port serial adapters used on the existing system), as well as printer forms management, and interfaces to the corporate headquarters located in another state.

The most complex area of the project was Cluster Operations - this involved all the necessary scripts and programs required to manage the new cluster and all auxiliary components such as the DIGI adapters and the printer failover management. System handover requirements defined rigorous acceptance tests which required simulation of all possible failures and full validation of the recovery techniques. Full documentation of the system also fell under this area of responsibility. The system needed to be documented so any reasonably technically competent person in the warehouse could use the documentation to bring the system back online in the event of a problem or a failure. This turned out to be a bigger effort than we originally estimated, as we identified several potential new failure conditions during testing of known failure scenarios.

Applications

The application migration effort was the smoothest part of the migration. Source code and database objects were migrated from the Ingres 2.0 development environment into an Ingres 2.6/SP1 development environment and recompiled and re-imaged. Only minor changes to the source were required, mainly due to reserved words and some non-standard C functions used in the Data General optimized code.

Once the source code was recompiled, an end-to-end test was conducted using the development lab, which contained identical hardware to the warehouse. Conveyor and carousel operations were successfully tested using test programs developed for this purpose. Minor performance issues were identified and addressed at this time.

Database

Similarly to the applications effort, very few issues were encountered with the database component of the project. In order to prepare for the actual migration, a number of unload and reload tests were conducted. This included tests in both the development environment between a DG development server and the Sun development server, and using the live database during scheduled downtime for the warehouse.

Metrics were collected during these tests in order to assist in scheduling of the actual cutover - the amount of time to unload the database, transfer the datafiles across the network, and reload the database on the new platform were tracked for each test run. Additionally, a proprietary Comprehensive Solutions tool (“ChecksumDB”) was used to validate the data integrity between the “old” and “new” databases. This cross-platform tool works in a manner similar to the “checksum” Unix utility, in that it computes a checksum for a database table (either all columns or a specified subset of columns) based on data contained in the table.

The ChecksumDB utility identified a few differences in floating point values between the “live” database and the migration test database. An investigation into these discrepancies revealed that this was due to differing floating point implementations between DG/UX and Solaris. The differences expressed themselves in terms of precision - the data in the new database was actually more precise (out to 8 decimal places) than the data in the old database (which was rounded to 7 decimal places). This was communicated back to the client, and it was determined that there was no significant impact to the migration due to this issue.

Interfaces

The major interface change – the incorporation of the DIGI terminals – was extensively tested in the development environment using the actual DIGI devices that were scheduled to be deployed in the warehouse. The “realport” software provided with the devices works by the emulation of Unix device files, which are then used in the same manner a serial device file would be used.

The method by which the WMS accessed the serial devices differed by type. Data was pushed directly to the bar code printers by the application, terminals were monitored by port monitors managed via the “sacadm” Solaris command, and scales and bar code readers were managed by custom daemon processes on the application server that would poll the devices for activity. Each was critical, and each had minor configuration issues that needed to be addressed to make everything work as needed.

Since multiple DIGI devices were required to cover all the components in the warehouse, discussions were held with the client staff in order to determine how best to manage the configuration. The solution agreed upon was to have each “station” or “pod” be associated to a specific DIGI, with each overall area (such as picking, manifesting, shipping) spread across multiple DIGIs. This configuration approach ensured that the failure of one DIGI device would not idle all warehouse functions. It would be possible for work to continue on a single DIGI, albeit in a degraded mode.

The other major interface issues included the compilation and testing of the conveyor and carousel systems. These systems had customized handlers written in ANSI C and Ingres ESQCL, and communicated with the WMS application and the external systems via FTP and Unix Sockets. Some minor problems were encountered with the Solaris socket implementation, as it differed from the DG/UX implementation, but these were quickly addressed and resolved.

Another issue was printer form generation. The existing system utilized a legacy and de-supported software package called “Reform” to generate and process print jobs. However, the new corporate standard was “Optio” running on Windows 2000, so a conversion was required. The Optio software functions by presenting a virtual printer queue to the network. Print jobs, which included special Optio commands, were “printed” to this virtual device via the LPD spooler on Solaris. The Optio software would then process the print job by applying the appropriate form template and then spool it to the output printer defined in the original print job. Two Windows 2000 servers were acquired to run Optio in order to provide failover for the warehouse; however, our request to purchase a content switch to handle load balancing was denied.

In order to address the lack of a content switch, a suite of OS level scripts were written to manage printer failover and load balancing. The WMS code was modified to call these scripts with any Optio print jobs. These scripts would handle load balancing by checking a configuration file listing available Optio servers, and then pseudo-randomly picking a server based on an algorithm which used a time value (the current time in seconds) to assign a job to a particular server. Although this was not a true load-balance in the strictest sense of the term, production tests proved that it worked well enough to provide a nearly 50/50 division of the print jobs between the two servers. To support failover in the event of an Optio server crash (or if one needed to be taken offline for maintenance), a separate daemon process would run in the background and check for server availability (using a combination of ping and smbclient) and update the configuration file as necessary. By marking a server as “DOWN”, we could quickly and easily exclude it from the load balancing process. Another aspect of this approach is that additional servers could be added, or existing servers renamed by simply updating the configuration file – no script changes required.

Cluster Operations

The majority of the constraints encountered on this project concerned the implementation of the clustering solution. There were three main components to the cluster operations – the operating system (OS) level, the cluster software level, and the application level. All of the areas - application, Ingres database, and interfaces - needed to be “cluster-aware” and be capable of running on either node in the event of a failover, either planned or unplanned. Additionally, versions and patches needed to be coordinated in a way that supported all components of the environment.

The corporate standard for clustering software at the time of this implementation was Veritas Cluster Services (VCS). As was done with the previous system, the functionality would be split between the two nodes – application server and database server. The clustering software would be used to run the environment on only one node, with either node being able to assume the roles running on the other node within 5 (best case) to 15 minutes (worst case).

A rather severe challenge was encountered concerning the ability of the two nodes to simultaneously share a filesystem. In the original DG cluster, the major application filesystems were shared between both nodes as a “clustered, shared” filesystem. This arrangement allowed both machines read/write access to the filesystem and handled all file locking and consistency issues. Veritas Clustered Filesystem support was requested for the new system to handle this requirement. However, we were informed that there was no money in the budget for this option. As a result, we were forced to develop a solution that would support the environment, work in a failover situation, and only involve utilities and functionality that were part of the core Solaris OS and our licensed Veritas products.

The solution to this problem was to use NFS to export the filesystem from the application server node and mount it on the database server node. In the event of a failure where the database server needed to assume all cluster roles, the filesystem would be failed over using VCS. There were many issues encountered during the implementation of this requirement, as well as some performance related issues that needed to be addressed through application and configuration changes. Although the final solution arrived at handled the requirements, it should be noted that in the project close-out review it was determined that the amount of time spent addressing these problems easily cost 2 to 3 times the purchase price of the clustered filesystem support.

The design decided upon for the new cluster was similar to that already in use for the DG system. Each node would have the ability to run in “normal” mode (either application server or database server) or in “failover” mode (where both the application and database server roles would be run on the same node). Failover mode would assume that the other node was already down or unreachable. The nodes would not be permitted to “swap” roles - that is, the database server would never run the application server functionality in anything other than a failover scenario. This provided the maximum level of flexibility for assuring uptime.

An unlikely series of problems was encountered when configuring and testing the single-node scenarios in cases where the second node was unavailable at startup. VCS would often hang, waiting for the second node to join the cluster in some situations. As a result, code was built into the Failover script to manually “seed” the cluster and handle these scenarios.

Other problems were encountered with Veritas default failover behavior. For example, when it encounters certain errors, Veritas would offline resource groups. This caused problems with Ingres, as the filesystems in use by Ingres would be dismounted when the resource group managing these filesystems was offlined. This was done without regard for Ingres state at the time, which would cause the database to crash and require recovery when the filesystems were remounted. To solve this problem logic was built into all of the startup scripts to freeze the Veritas resource groups once they were set properly. This prevented Veritas from automatically changing the state of the resource group. Although this had the effect of negating some of the Veritas functionality, we were willing (and really required) to manage this through our scripts in order to ensure the integrity of Ingres and, by extension, the database.

Hardware

The main hardware for the project included two older Sun T3 Disk Arrays, a pair of fibre switches, and two Sunfire v480 servers. Ancillary and supporting hardware included three Sun Netras - two for DNS/NIS services, and one to run Veritas NetBackup and manage a tape library. Basic setup of these infrastructure components – backup server configuration, NIS authentication server setup, DNS, network configuration, etc., was handled in accordance with the client’s normal implementation and our recommended best practices guidelines and procedures.

The one issue encountered with the hardware concerned the disk arrays for the system. Our initial specification on the T3 arrays was to build multiple RAID-1 (mirrored) sets in each T3 and then mirror these sets between the two T3’s. However, because of RAID configuration limitations in the T3, we were unable to do this. Our fallback position was to create two RAID 0+1 volumes in each T3 and mirror these between the two arrays using Veritas Volume Manager. This provided the ability to lose any one of the following components – T3 Array, Fibre Switch, or a whole disk volume – without impacting the system uptime.

Cluster Management

All tasks related to the management of the cluster were reduced to the following core scripts:

- StartupDBServer.ksh
- StartupApplServer.ksh
- StartupSingleNode.ksh
- ShutdownServices.ksh

Each script contained a mixture of Solaris, Veritas, Ingres, Interface, and WMS application commands. Each script was responsible for not only starting/stopping components, but also for validating that the resources being manipulated were in the correct state. Also, the scripts would perform dependency checks between tasks to make sure that all components (such as filesystems, configuration files, network devices, etc) were in place and ready before allowing the next task to be processed.

Additional scripts were created which would provide additional functionality, such as the current status of the cluster, the availability of key components on the network (such as NIS servers, DIGI devices, etc), the ability to check on interface daemons, and to view an activity log for recent administrator actions on the server. Also, wrappers were provided for rebooting and halting the server. This allowed these actions to be tracked and logged.

For example, the “StartupDBServer.ksh” script would validate that all required Ingres filesystems were in place and that the configuration files were available before running the function to start the Ingres database system.

The scripts were designed with two levels of logging - a terse mode informed the operator of tasks being run and notified on success or failure. The verbose mode would provide all applicable output from the command being executed. In both modes the full command detail was saved off to a log file that was rotated on a weekly basis, with archived copies backed off to tape in accordance with the client’s backup and data retention policy.

In order to provide the simplest method for managing the cluster, a menu script was created that provided the administrative users with a simple, clean interface for managing the system:

```
Cluster Operations
  Version: 1.5 2004/09/03 13:15:11

Current Server: morpheus
  Logged in as: jschmidt
Current Date: Thu Sep 8 16:40:42 EDT 2005

1) TestConnectivity    5) StartupDBServer    9) ViewActivityLog
2) StatusSummary      6) StartupSingleNode 10) CheckDaemons
3) ShutdownServices   7) ShutdownNode      11) Quit
4) StartupAppServer   8) RebootNode

Select an option and press Enter:
```

This script, called “SysMenu”, runs on both nodes and performs user validation when run – users that are not part of the administrative group receive an error message and are prohibited from running the menu. Commands not applicable to a node would return an error if selected. For example, an attempt to run “StartDBServer.ksh” on the application server would return an error, as the command was only valid on the database server. In addition to the management commands on the “SysMenu”, a number of status and test commands were provided. These commands provide a “snapshot” of the current state of the system, including testing connectivity to the other core components of the overall system infrastructure (such as DIGI devices, NIS servers, Optio servers, and other devices).

Documentation

The key to this entire project was the documentation set that was created for the client. The goal was to provide several levels of detail for the system, usable by both the on-site technical resources and the Corporate IT group. This documentation was presented to the client in both printed form, as well as electronic copy stored in their collaborative software repository.

At the highest level, a “Cluster Operations Guide” was produced. This guide explained the process of starting and stopping the cluster at a very low level of detail - each server in the environment (including the NIS servers, backup servers, and Optio Print servers) was identified by function, impact, and location in the server room. Detailed instructions were provided for server power-on and power-off, as well as management of the services provided by the server. Instructions were provided to allow any IT resource to log on to the cluster nodes, start the “SysMenu” detailed out above, and perform any management tasks required. Common errors were listed out in the appendix, along with possible cause, and suggested corrective action.

At a more detailed level, a “Cluster Procedures/Configuration Guide” was created which described the cluster, the assumptions made in its setup and configuration, and the location and description of all key scripts and configuration files. The repository for these scripts and process for modification, testing, and deployment of changes was also detailed.

At the lowest level, a document was created for each script and configuration file used. Calling syntax and data flow was documented, as were any dependencies and side effects. A detailed breakdown of each function in the script was included in the documentation in order to aid troubleshooting or future modification of the scripts.

The End Result

The migration to the new system was completed over a weekend, and was devoid of any major issues. User response was very positive - the new system provided a faster, more stable platform for warehouse operations. Corporate and warehouse management were very happy with the new system, and its reduction in IT-related downtime. However, the greatest praise was provided by the onsite IT staff months later when they told us that the new system was great because “it just worked.”

About the Author

Jason Schmidt has been instrumental in the management and recovery of several “at risk” projects for our clients ranging from re-hosting and migration projects through Disaster Recovery and Business Continuity efforts. The success of the project described in this case study are the result of the experience gained by Jason and the entire Comprehensive Solutions Team over many years working in a wide variety of environments with a wide range of technologies.

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 another HA white paper](#)

[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

Copyright © 2006-2008 Comprehensive Consulting Solutions, Inc.
All Rights Reserved

No part of this document may be copied without the express written permission of Comprehensive Consulting Solutions, Inc., 4040 N. Calhoun Rd., Suite 105, Brookfield, WI 53005.

This document is provided for informational purposes only, and the information herein is subject to change without notice. Please report any errors herein to Comprehensive Consulting Solutions. Comprehensive Consulting Solutions, Inc. does not provide any warranties covering and specifically disclaims any liability in connection with this document.

All product names referenced herein are trademarks of their respective companies. Use of these names should not be regarded as affecting the validity of any trademark or service mark.

Supplemental Hardware Overview

WMS Ingres Clustering Project *Equipment and Services Overview*

