the Availability Digest

Console Command Takes Down Active/Active System

December 2006

A Successful Active/Active System

A European company had been running a two-node NonStop active/active system for several years without an outage. In fact, the system had gone through many hardware, operating system, and application upgrades without ever having to take any planned downtime. This is what active/active is all about.

The capacity of each node was sufficient to carry the entire application load. Therefore, upgrades could be rolled across the two nodes one at a time without affecting any of the system's users. This was done by taking down one node, upgrading it, returning it to service, and repeating that procedure for the second node.

Zero Planned Downtime

Taking down a node for an upgrade was very straightforward. The upgrade started with Node A. Its users would first be switched over to Node B. The applications on Node A would then be shut down in an orderly manner, and the node would be stopped.

There was no big rush to upgrade the downed node. Since the system was configured to run in an active/active environment, and since each node could handle the entire application load, the upgrade could be accomplished in a very controlled and orderly manner. More importantly, there was plenty of time to test the node thoroughly before it was reintroduced into the network, to synchronize its database with the operational system, and to return its complement of users to it.

Following the return to service of Node A, Node B would be taken down and upgraded using an identical procedure.

The switchover and recovery procedures were properly documented. These procedures were used frequently enough so that the operational staff was satisfied that the upgrade plan was kept current and that the people involved were well practiced. Node upgrades had become almost second nature.

Until One Day …

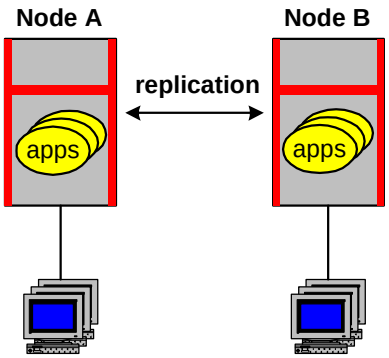
And therein lay the problem – complacency. Fingers could fly across the keyboard with little connection to the brain.

The upgrade in question started normally. The operations staff moved the users off of Node A in preparation for upgrading it. Once satisfied that all users were now being properly handled by Node B, the system manager brought up the RMI console (the Remote Maintenance Interface

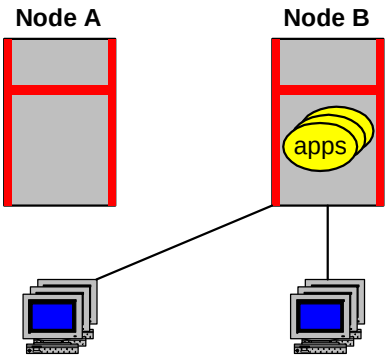
© 2006 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com

used to control a system) for Node A on his PC, and the command to stop the Node A processors was entered.

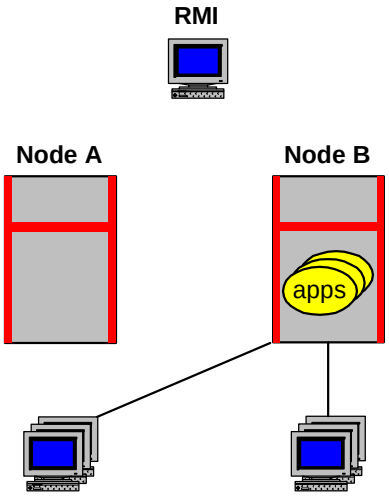
To the system manager's his horror, the entire system suddenly shut down. As it turned out, he had not brought up the RMI console for Node A. He had brought up the RMI console for Node B. Consequently, he had stopped the operational system.



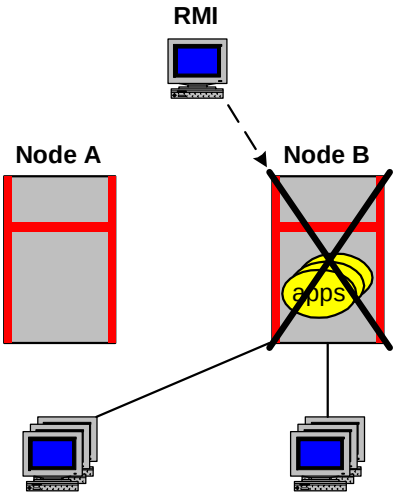
Normal Operation



Step 1: Isolate Node A



Step 2: Bring Up RMI Console



Step 3: Stop Node A
Oops!!

Now the brain suddenly became engaged. As we well know, when things go wrong, people get stupider. But the system manager managed to rise above this syndrome and had Node A returned to service within a very short time. (Fortunately, a database resynchronization was not required.) Afterwards Node B was brought back into service and the action restarted from the beginning and this time everything went well.

© 2006 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com

Lessons Learned

Subsequent analysis of this incident found that there was no password protection provided for the RMI/RCP consoles. Therefore, the use of the wrong console could easily go unnoticed. Distinguishing passwords were immediately added so that at least a second thought process was required before shutting down a node. And brains are now more fully engaged in what might otherwise appear to be rote actions.

A recent survey by The Standish Group found that 26% of all system faults were caused by operator error.[1] These faults represented 7% of all downtime (obviously, when the operator brings down a system, he is quick to bring it back up again since he knows exactly what happened).

This Never Again story exemplifies The Standish Group findings. It emphasizes the need for good documentation and checklists for common or critical tasks. Memorizing a checklist is not good enough; they must be read and followed during their pertinent procedure.

Fortunately, the perpetrator of this action did not receive a Darwin award.[2] Rather, he has gone on to become a very respected figure in the active/active community; and the system has since run for years without a single outage. After all, if you never make a mistake, you never learn.

[1] The Standish Group, The Other Side of Failure, paper MEA-22-U presented at the 2006 ITUG Summit; October, 2006..
[2] The Darwin Awards are awarded by popular vote to "salute the improvement of the human genome by honoring those who remove themselves from it." Our favorite Darwin award was given to the terrorist whose mail bomb was returned for insufficient postage. He was blown away when he opened it. See www.darwinawards.com.

© 2006 Sombers Associates, Inc., and W. H. Highleyman
www.availabilitydigest.com