Automating IBM DB2 UDB HADR with HACMP

1. ***Overview***
2. ***System Configuration***
3. ***HACMP Setup***
4. ***Configure the HADR DB2 with HACMP***
5. ***Testing the HADR DB2 with HACMP***
6. ***Conclusion***

1. Overview

In this paper, we describe how to automate IBM® DB2® Universal Database ™ High Availability Disaster Recovery Option takeover in an High-Availability Cluster Multi-Processing (HACMP™) environment. We also describe the failover test procedure for HADR controlled by the HACMP facility.

1.1 DB2 HADR Overview

DB2 HADR is a data replication feature that provides a high availability solution for both partial and complete site failures. DB2 HADR is designed for quick failover, easy setup, and manageability. HADR protects against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby. It is an example of database log shipping technology; transactions that occur on the primary database are also applied to a designated standby database, so that if the primary database becomes inaccessible, clients can continue using the standby database. The logic to perform the state change can be embedded within a high availability cluster manager to provide unattended HADR failover.

1.2 HACMP Overview

HACMP for AIX® provides a highly available computing environment. This facilitates the automatic switching of users, applications, and data from one system to another in the cluster after a hardware or software failure. The primary reason to create HACMP clusters is to provide a highly available environment for mission-critical applications. For example, an HACMP cluster could run a database server program that services client applications. The clients send queries to the server program that responds to their requests by accessing a database that is stored on a shared external disk.

In an HACMP cluster, to ensure the availability of these applications, the applications are put under HACMP control. HACMP takes measures to ensure that the applications remain available to client processes even if a component

in a cluster fails. To ensure availability, in case of a component failure, the HACMP software moves the application, along with resources that ensure access to the application, to another node in the cluster.

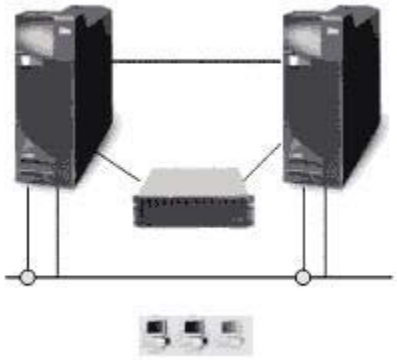
2. System Configuration

The following is a description of the hardware and software environment that was used as a basis for this white paper.

2.1 Hardware Configuration

2 x IBM7038-6M2 connected to the SSA160.

- Machine type: IBM, 7038-6M2
- Memory: 262144 x 4KB
- Network Card: 10/100M Ethernet adapter
- Storage: SSA160.



2.2 Software Configuration

- AIX 5.2 ML05
- HACMP 5.1 PTF Level 4
- DB2 UDB Version 8.2 (FixPack 10 or later required)
- HADR

3. Setting up HACMP

Setting up HACMP software usually requires following steps:

1. Define the cluster and add nodes to the cluster. The cluster nodes in our test scenario are named db2hadevp1 and db2hadevp2.

2. Configure the HACMP network and decide the method for IP address takeover. In our test case, we use IP aliasing.

3. Configure the communication interface and devices. A heart-beating disk is used as the heart beat device in our test case. To configure the heart-beating disk, the shared disk should be in an enhanced concurrent volume group.

4. Configure the high availability service IP.

5. Configure the application server.
It is often useful to place a simple test script for the corresponding start and stop script entry to test HACMP and, after verifying that the HACMP environment is functioning properly, modify the start and stop script to work with HADR.

6. Configure the high availability resource group.
This will associate the HA service IP, the application server and all the resources required by the application into one resource group. In case a node failure happens, the HACMP will move the resource group running on the failed node to the surviving node in the cluster.

After configuring HACMP, you must verify and synchronize the topology and resource group using the facility provided by HACMP (for example via SMIT command panels). Every time the HACMP configuration is changed, you must resynchronize the cluster from the node where the change has been made.

After the verification and synchronization is successful, you may start the HACMP service.

4. Configuring the HADR DB2 application to use HACMP

To configure the HADR DB2 application:

1. Make a directory (with the exact same directory path) on a local file system on each node in the cluster, copy the application start, and stop scripts to it. You can use the following commands on each node in the cluster:

```
# mkdir /home/hadr_app
# cp /home/db2inst1/sqllib/samples/hacmp/rc.hadr.* /home/hadr_app
```

2. For each start and stop script, edit the following lines, replacing *db2instp* with the instance name at this local node, *db2insts* with the instance name at the other node, and *hadrdb* with the actual name of the HADR database.

```
DB2HADRINSTANCE1=${1:-db2instp}
DB2HADRINSTANCE2=${2:-db2insts}
DB2HADRDBNAME=${3:-hadrdb}
```

For our tests, we are using instance name *db2inst1* on both nodes and HADR database name *hadrdb*, so the lines in question would be edited to look like the following:

```
DB2HADRINSTANCE1=${1:-db2inst1}
DB2HADRINSTANCE2=${2:-db2inst1}
DB2HADRDBNAME=${3:-hadrdb}
```

3. Modify the application server and supply the location of the HADR DB2 start and stop script file (rc.hadr.start and rc.hadr.stop respectively) to the corresponding entry, then resynchronize the cluster and start the cluster service.

5. Testing DB2 HADR with HACMP

We implement the test for three scenarios. In the first scenario, we use the facility provided by the HACMP application to test the failover between two nodes. For the second scenario, we power off the primary system and verify that the failover properly occurs in a real system outage environment.
In the third test scenario, we focus on the application failure rather than the system failure. We will test the circumstance in which the application or process failure happens. To carry out the test scenarios, complete the following steps:

1. Start HADR in Peer state prior to starting the HACMP service.

2. Make sure that the primary HADR database will initially be running on the cluster node that is supposed to keep the resource group when HACMP is started. For example, for a rotating resource group, if you bring up the primary HADR database on db2hadevp1, remember to start the HACMP service first on db2hadevp1 as well, so that the resource group will be brought on line on that node.

3. To start the standby HADR DB, issue the following command:

   ```
   % db2 start hadr on db hadrdb as standby
   ```

4. To start the primary HADR DB, issue the following command:

   ```
   % db2 start hadr on db hadrdb as primary
   ```

   Remember the standby HADR DB should be started prior to starting the primary HADR DB.

To view the current state of the HADR database, issue the following command:

```
% db2 get snapshot for all on hadrdb \
| grep -A14 "HADR Status"
```

You should see output similar to the following on node db2hadevp1:

```
HADR Status
  Role                    = Primary
  State                   = Peer
  Synchronization mode    = Nearsync
  Connection status       = Connected, 07/29/2005 10:06:54.151206
  Heartbeats missed       = 0
  Local host              = db2hadevp1
  Local service           = 24592
  Remote host             = db2hadevp2
  Remote service          = 24592
  Remote instance         = db2inst1
  timeout(seconds)        = 30
  Primary log position(file, page, LSN) = S0000002.LOG, 20,
                                          000000000139C1EA
  Standby log position(file, page, LSN) = S0000002.LOG, 20,
                                          000000000139C1EA
  Log gap running average(bytes) = 11773
```

We will create an alias for the above command and call it *hdrsnap* hereafter.

5. To start the HACMP service, using smitty clstart→ select the node to start the service → press enter.

6. To stop the HACMP service with takeover, using smitty clstop → shutdown mode (take over)→ press enter.

The relationship of the participating nodes in the cluster is set to rotating; Rotating resources have no site priority, when a site fails, the resource will be acquired by the next available standby site. When a node at the failed site rejoins, the resource will remain with its new owner and failover only when the current node fails.

Test Scenario 1

In this test scenario, we use the test facility provided by HACMP to implement the failover test. The primary HADR DB will be running on db2hadevp1 first.

1. On db2hadevp1, check the HADR status by issuing the following command:

   ```
   % hdrsnap
   ```

   You should see output similar to the following:

```
HADR Status
  Role                   = Primary
  State                  = Peer
  Synchronization mode   = Nearsync
  Connection status      = Connected, 03/24/2005
20:45:44.481552
  Heartbeats missed      = 0
  Local host             = db2hadevp2
```

2. On db2hadevp2, check the HADR status by issuing the following command:

   ```
   % hdrsnap
   ```

   You should see output similar to the following:

   ```
   HADR Status
     Role                   = Standby
     State                  = Peer
     Synchronization mode   = Nearsync
     Connection status      = Connected, 03/24/2005
   20:45:44.312617
     Heartbeats missed      = 0
     Local host             = db2hadevp1
   ```

3. First, we failover the primary HADR database to the node db2hadevp2. Check the status of HACMP before doing the failover test. Use smitty hacmp→ problem determination tool→view current state. At the bottom of the display, you will see the resource group state as follows:

   ```
   Cluster Name: "db2_cluster"
   Cluster State: UP
   Cluster Substate: STABLE
   Node Name: "db2hadevp1"           State: UP
   ……

   Node Name: "db2hadevp2"           State: UP
   ------------------------------------------------------------
   Group Name     Type       State      Location
   ------------------------------------------------------------
   hadr_rg        rotating   ONLINE     db2hadevp1
                             OFFLINE    db2hadevp2
   ```

4. At the node db2hadevp1, issue the following command:

   ```
   smitty clstop → shudown mode (takeover)→ press enter
   ```

5. Check the log file /tmp/hacmp.out on db2hadevp2 and make sure the resource group has been successfully taken over. The output of the log should be similar to the following if the failover is successful:

```
% tail -f /tmp/hacmp.out
hadr_rg:rc.hadr.start[61] rc=20
hadr_rg:rc.hadr.start[62] [ 20 -eq 10 ]
hadr_rg:rc.hadr.start[65] [ 20 -eq 20 ]
hadr_rg:rc.hadr.start[66] rc=0
hadr_rg:rc.hadr.start[69] return 0
hadr_rg:rc.hadr.start[3] logger -i -p info -t
/home/db2inst1/hadr/rc.hadr.start returning 0
hadr_rg:rc.hadr.start[4] return 0
hadr_rg:rc.hadr.start[183] echo 0
0
hadr_rg:rc.hadr.start[184] exit 0
```

6. Check the current HACMP status on db2hadevp1:

   smitty hacmp→ problem determination tools→ view current state

   You should see the following output:

```
-------------------------------------------------------------

Node Name: "db2hadevp1"           State: UP
……
Node Name: "db2hadevp2"           State: DOWN
……
Group Name     Type       State      Location
-------------------------------------------------------------
hadr_rg        rotating   OFFLINE    db2hadevp1
                          ONLINE     db2hadevp2
```

Check the status of the HADR database on the node db2hadevp2:

```
% hdrsnap
HADR Status
  Role                   = Primary
  State                  = Disconnected
  Synchronization mode   = Nearsync
  Connection status      = Disconnected, 03/28/2005
                           14:32:35.951693
  Heartbeats missed      = 0
  Local host             = db2hadevp2
  Local service          = 24592
  Remote host            = db2hadevp1
  Remote service         = 24592
  Remote instance        = db2inst1
  timeout(seconds)       = 30
```

You can see that the HADR application has been successfully failed over to the node db2hadevp2 from db2hadevp1.

7. Check the HADR database status on node db2hadevp1 by issuing the following command:

   ```
   % hdrsnap
   ```

   (No output)

8. Next, we will restart the HADR database on db2hadevp1 in the standby role. Before starting the HADR database, you have to start the DB2 instance by issuing the following command:

   ```
   % db2star*t*
   ```

   You should see the following output:
   ```
   SQL1063N  DB2START processing was successful.
   ```

9. Issue the following command to restart the HADR database in the standby role:

   ```
   % db2 start hadr on db hadrdb as standby
   ```

10. To view the HADR status on db2hadevp1, issue the following command:
    ```
    % hdrsnap
    ```

    You should see the output similar to the following:

    ```
    HADR Status
      Role                   = Standby
      State                  = Peer
      Synchronization mode   = Nearsync
      Connection status      = Connected, 07/29/2005
    14:56:16.015878
      Heartbeats missed      = 0
      Local host             = db2hadevp1
      Local service          = 24592
      Remote host            = db2hadevp2
      Remote service         = 24592
    ```

    Note that once the standby database has been started, the HADR state will become “peer” on both nodes.

Next, you can check if the resource group can fail back to the original node using the same procedure mentioned above. Keep in mind that you have to start the HADR database first and then to start the HACMP services on the same node.

Test Scenario 2

In test scenario 2, we test the failover in a node crash environment. Note that we cannot do the failover test by issuing the shutdown command on the AIX operating system, because when AIX is shut down (with the shutdown command) while HACMP is up, the processing in the HACMP shutdown scripts (for example, /usr/es/sbin/cluster/etc/rc.shutdown) performs a forced shutdown; the resource group failover action will not happen. In order to implement this failover test, you could power off the node directly. However, there is an alternative way to simulate the system crash, by issuing the following command:

```
# cat /etc/hosts > /dev/kmem
```

In this test scenario, we implement the failover test using this second method, that is, by writing data to kernel memory and crashing the node.

In this test scenario, the primary HADR database is running on db2hadevp1, and when the system crash happens, the HADR database will become Primary on the standby node, which is db2hadevp2.

1. First, check the HACMP and HADR status of the two nodes and make sure the HACMP service is started on both nodes:

   smitty hacmp→ problem determination tools → view current states

   You should see the following information:

   ```
   Node Name: "db2hadevp1"           State: UP
   Node Name: "db2hadevp2"           State: UP
   ------------------------------------------------------------
   Group Name     Type       State      Location
   ------------------------------------------------------------
   hadr_rg        rotating   OFFLINE    db2hadevp2
                             ONLINE     db2hadevp1
   ```

2. Issue the following command on node db2hadevp1:

   ```
   % hdrsnap
   ```

   You should see the following output:

   ```
   HADR Status
     Role                   = Primary
     State                  = Peer
     Synchronization mode   = Nearsync
     Connection status      = Connected, 07/29/2005
   14:56:16.907320
     Heartbeats missed      = 0
   ```

```
  Local host             = db2hadevp1
  Local service          = 24592
  Remote host            = db2hadevp2
  Remote service         = 24592
```

Then, on node db2hadevp2

```
% hdrsnap

HADR Status
  Role                   = Standby
  State                  = Peer
  Synchronization mode   = Nearsync
  Connection status      = Connected, 07/29/2005
14:56:16.015878
  Heartbeats missed      = 0
  Local host             = db2hadevp2
  Local service          = 24592
```

3. On db2hadevp1, issue the following command as root:

```
# cat /etc/hosts > /dev/kmem
```

The db2hadevp1 node will terminate processing immediately. Check the log /tmp/hacmp.out on db2hadevp2 and see if the resource group has been successfully taken over by the node db2hadevp2. You should see output similar to the following:

```
# tail -f /tmp/hacmp.out

hadr_rg:rc.hadr.start[61] rc=20
hadr_rg:rc.hadr.start[62] [ 20 -eq 10 ]
hadr_rg:rc.hadr.start[65] [ 20 -eq 20 ]
hadr_rg:rc.hadr.start[66] rc=0
hadr_rg:rc.hadr.start[69] return 0
hadr_rg:rc.hadr.start[3] logger -i -p info -t
/home/db2inst1/hadr/rc.hadr.start returning 0
hadr_rg:rc.hadr.start[4] return 0
hadr_rg:rc.hadr.start[183] echo 0
0
hadr_rg:rc.hadr.start[184] exit 0
```

4. Check the HACMP and HADR status of on node db2hadevp2:

smitty hacmp→ problem determination tools→ view current state:

You should see the following information:

```
Node Name: "db2hadevp1"           State: DOWN
```

```
Node Name: "db2hadevp2"           State: UP

-------------------------------------------------------------
Group Name     Type       State      Location
-------------------------------------------------------------
hadr_rg        rotating   ONLINE     db2hadevp2
                          OFFLINE    db2hadevp1
```

5. Issue the following command

   ```
   % hdrsnap
   ```

   You should see the following output:

   ```
   HADR Status
     Role                   = Primary
     State                  = Disconnected
     Synchronization mode   = Nearsync
     Connection status      = Disconnected, 07/29/2005
        15:45:15.110238
     Heartbeats missed      = 0
     Local host             = db2hadevp2
     Local service          = 24592
     Remote host            = db2hadevp1
   ```

6. The HADR database is now Primary on db2hadevp2; it is successfully failed over to the node db2hadevp2. Next, we will fail over the resource group back to the original node db2hadevp1. To do this, first start the standby HADR database on db2hadevp1 (once this node comes back online) by issuing the following command:

   ```
   % db2start
   ```

   You should see the following output:

   ```
   SQL1063N  DB2START processing was successful.
   ```

   Now issue the following command, again on node db2hadevp1:

   ```
   % db2 start hadr on db hadrdb as standby
   ```

   Note that to successfully take over HADR state in this scenario, you have to issue a forced HADR TAKEOVER. After this operation, the re-integration step given above might fail; in that case, you have to re-image the database standby from a backup of the current primary. This may occur in cases where the HADR pair was not in Peer State at the time of the Primary node failure.

7. Next, issue the hdrsnap command (again on node db2hadevp1) to check the status of HADR database to see if the status is “peer”.

   ```
   % hdrsnap
   ```

   You should see the following output:

   ```
   HADR Status
     Role                   = Standby
     State                  = Peer
     Synchronization mode   = Nearsync
     Connection status      = Connected, 07/29/2005
   14:56:16.907320
     Heartbeats missed      = 0
     Local host             = db2hadevp1
     Local service          = 24592
     Remote host            = db2hadevp2
   ```

8. Next, start the HACMP service on db2hadevp1:

```
smitty clstart → select node db2hadevp1→ press enter
```

9. Make sure the HACMP service startup has completed by checking the /tmp/hacmp.out file.

10. We will then simulate the system crash on db2hadevp2 and test if the resource group can be failed over back to db2hadevp1. To do this, issue the command on db2hadevp2 as root:

    ```
    # cat /etc/hosts > /dev/kmem
    ```

    This command will crash the db2hadevp2 and cause it to be rebooted.

11. Check the log file /tmp/hacmp.out on db2hadevp1 and make sure the failover process has been successfully completed.

12. Check the HADR status on db2hadevp1 by issuing the hdrsnap command:

```
% hdrsnap

HADR Status
  Role                   = Primary
  State                  = Disconnected
  Synchronization mode   = Nearsync
  Connection status      = Disconnected, 07/29/2005
                           16:10:41.696273
  Heartbeats missed      = 0
  Local host             = db2hadevp1
  Local service          = 24592
```

```
Remote host             = db2hadevp2
Remote service          = 24592
```

Test Scenario 3

In this test scenario, we test the instance failure. To do this test, the HACMP customer application monitors must be configured. Select the application server to monitor by using the fast path

smitty clappserv_to_custom_monitor→select

You should configure the following entries:

1. Monitor Method
2. Monitor Interval
3. Stabilization Interval
4. Restart Count
5. Restart Interval
6. Action on Application Failure
7. Cleanup Method
8. Restart Method

The monitor method is the full path name of a shell script that examines if the monitored application is running or not. The shell script should exit with 0 if the application is running; otherwise, the return code of the shell script would be different from 0. Also, you cannot provide any argument to the script, as HACMP cannot pass arguments to the script.

The application monitor works as follows. At the beginning, the Failure Count is set to 0. The monitor will periodically (at each monitor interval) check if the application is running. If the application has failed, the Failure Count is set to 1. HACMP will then try to reactivate the application by calling the cleanup and restart scripts. After that, it waits for the stabilization interval to expire. Once the stabilization interval expires, the monitoring process will resume and check whether the application is still active. If the application is running and further, it reaches the Restart Interval with the application still alive, the Failure Count will be reset to 0; otherwise, the HACMP will try to restart the process repeatedly with the Failure Count being incremented accordingly until the Failure Count is equal to the Restart Count, at which time it will report a “server_down” event in the log file (if action is set to notify) and monitoring process will not resume.

The application monitor parameters in the test scenario are set as follows:

```
Application Server Name                    db2_app
* Monitor Method                  [/home/db2inst1/hadr/db2mon.ksh]
```

```
  Monitor Interval                        [6]                 #
  Hung Monitor Signal                     [9]                 #
* Stabilization Interval                  [16]                #
  Restart Count                           [3]                 #
  Restart Interval                        [72]                #
* Action on Application Failure           [notify]            +
  Notify Method                           []
  Cleanup Method                  [/home/db2inst1/hadr/db2kill.ksh]
  Restart Method                  [/home/db2inst1/hadr/db2restart.ksh]
```

Here are examples of what the scripts should look like (these are not shipped and must be created by the user).

The db2mon.ksh should look like this:

```
% cat db2mon.ksh

#!/bin/ksh
su - db2inst1 -c "db2gcf -s"
```

The db2kill.ksh should look like this:

```
% cat db2kill.ksh

#!/bin/ksh
su - db2inst1 -c "db2gcf -k"
```

The db2restart.ksh should look like this:

```
% cat db2restart.ksh

#!/bin/ksh
su - db2inst1 -c "db2gcf -u"
```

In the test scenario, the primary server and resource group are originally on db2hadevp2. Log on to the node db2hadevp2 as instance owner and issue the following command (this will simulate an instance failure):

```
% db2_kill
```

Immediately check the HADR status on node db2hadevp2 by issuing the following command:

```
% hdrsnap
```

There should be no output.

Within a short period (monitor interval), the instance failure should be detected and HACMP will attempt to restart the instance. Check the

/tmp/hacmp.out log file; the following information should be present once the instance has been restarted successfully:

```
                        HACMP Event Summary
Event: server_restart_complete db2hadevp2 13
Start time: Wed Mar 30 17:23:47 2005

End time: Wed Mar 30 17:23:48 2005

Action:         Resource:                       Script Name:
------------------------------------------------------------
No resources changed as a result of this event
------------------------------------------------------------

Instance  : db2inst1
DB2 Start : Success
```

You can also check if the instance is successfully started by issuing following command:

```
% ps –ef |grep db2sysc |grep –v grep
```

The output should be similar to the following:

```
% ps -ef |grep db2sysc |grep -v grep
 db2inst1 774288 807070   0 17:23:49      -  0:00 db2sysc
```

Then connect to the database hadrdb by issuing the following command on the same node; this will activate the hadrdb database.

```
% db2 connect to hadrdb
```

The output should be similar to the following:

```
  Database Connection Information

 Database server        = DB2/6000 8.2.3
 SQL authorization ID   = DB2INST1
 Local database alias   = HADRDB
```

Check the HADRDB status using the hdrsnap command; the output should be similar to the following:

```
% hdrsnap

HADR Status
  Role                  = Primary
  State                 = peer
  Synchronization mode  = Nearsync
  Connection status     = Connected, 03/30/2005
                          18:50:12.863173
```

```
  Heartbeats missed      = 0
  Local host             = db2hadevp2.torolab.ibm.com
  Local service          = 24592
  Remote host            = db2hadevp1.torolab.ibm.com
  Remote service         = 24592
  Remote instance        = db2inst1
  timeout(seconds)       = 30
  Primary log position(file, page, LSN) = S0000003.LOG, 0,
                           0000000001770BE7
  Standby log position(file, page, LSN) = S0000003.LOG, 0,
                           0000000001770BE7
  Log gap running average(bytes) = 0
```

The Application Monitor must be defined inside an application server, and this application server must be present in a resource group, so the application monitor process will run on the node where the resource group is brought online, that is, it will only run on the primary HADR node.

Note that you can check if the application monitoring process is running by issuing the following command:

```
% ps –ef |grep appmon |grep –v grep
```

The output should be similar to the following:

```
    root 729286 794832   0 17:24:04      -  0:06
/usr/es/sbin/cluster/clappmond db2_app
    root 794832 364786   0 17:24:04      -  0:00 run_clappmond
-sport 1000 -result_node db2hadevp2 -script_id 0 -command_id
13 -command db2_app -
environment ?CLUSTER_VERSION=6??GS_NODEID=4??APPLICATION_SERVE
R=db2_app??MISC_DATA=??RESOURCE_GROUP=hadr_rg??RESTART_METHOD=
/home/db2inst1/hadr/db2restart.ksh??CLEANUP_METHOD=/home/db2in
st1/hadr/db2kill.ksh??NOTIFY_METHOD=??MONITOR_METHOD=/home/db2
inst1/hadr/db2mon.ksh??FAILURE_ACTION=notify??RESTART_INTERVAL
=72??HUNG_MONITOR_SIGNAL=9??RESTART_COUNT=3??STABILIZATION_INT
ERVAL=16??MONITOR_INTERVAL=6??INSTANCE_COUNT=0??PROCESS_OWNER=
??PROCESSES=??MONITOR_TYPE=user??HACMP_VERSION=__PE__??PATH=/u
sr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin??ODMDIR=/etc
/es/objrepos??LC_FASTMSG=true??PING_IP_ADDRESS= ??LOCALNODEID=
db2hadevp2??LOCALNODENAME=db2hadevp2??CM_CLUSTER_NAME=db2_clus
ter??CM_CLUSTER_ID=1110668033?
```

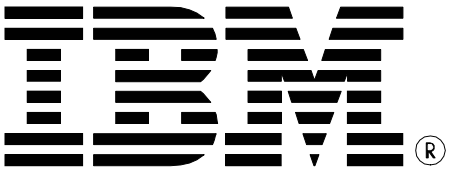
6. Conclusion

From the tests presented previously, we can see that an HADR database can work seamlessly with the HACMP environment with little human intervention. The combination provides high availability in two aspects: The HADR transaction log shipping will ensure the redundancy and

disaster recovery of the database, while HACMP will provide the almost uninterrupted high availability service in a mission critical environment.

Authors

Steven Raspudic, IBM Toronto Lab
Jing Yang, IBM Toronto Lab



© Copyright IBM Corporation, 2004
IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

All Rights Reserved.

IBM, DB2, AIX, HACMP, DB2 Universal Database, and Tivoli are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this white paper is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of 07/30/2003, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.