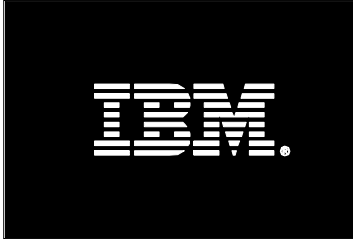
September 2004

DB2 Information Management Software



DB2 HADR and Sun Cluster 3.1

Author:
Steve Raspudic, IBM Toronto Lab

Contents

1. OVERVIEW

1.1 Objective

In this paper, we describe the implementation of an automated IBM® DB2® Universal Database™ (DB2 UDB) failover solution based on the high availability disaster recovery (HADR) feature used in conjunction with Sun Cluster 3.1 (SC3.1).

1.2 Overview: DB2 UDB

DB2 Universal Database (DB2 UDB) is the industry's first multimedia, Web-ready relational database management system, powerful enough to meet the demands of large corporations and flexible enough to serve medium-sized and small e-businesses. DB2 UDB combines integrated power for business intelligence, content management, and on-demand business with industry-leading performance and reliability. For more information, visit *www.software.ibm.com/data*.

1.2.1 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, where transactions that occur on the primary database are applied to a designation 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 an HA cluster manager, to provide unattended HADR failover.

1.3.2 Overview: Client Reroute Overview

The automatic client reroute (ACR) feature in DB2 UDB allows client applications to recover from a loss of communication with the server so that they can continue to work with minimal interruption. If client to server communication fails, then the client is automatically rerouted to an alternate server. The alternate server is specified through a command line processor (CLP) command by invoking an application programming interface (API), or when adding a database using the Control Center or the advanced view of the Configuration Assistant. Our examples focus on the use of CLP for configuration of the ACR feature in DB2 UDB.

ACR can be used with HADR to make client applications connect to the new primary database after a takeover operation. If ACR is not enabled,

client applications will receive error message SQL30081, and no further attempts will be made to establish a connection with the server.

1.3.3 Client Reroute Overview vs. Floating IP Address

ACR is a feature that allows transparent client failover to one of two possible target machines. Sun Cluster 3.1 provides a method to allow floating IP addresses in the cluster. It is possible to use either ACR or the cluster manager's ability to move IP addresses between physical nodes in the cluster to 'redirect' the clients to the physical location of the new server. The main differences between the two approaches are as follows:

- ACR client must have connected at least once to the primary database prior to failure
- With ACR, the connection is preserved through failover
- With floating IP, connection must be re-established by the client

Generally, ACR is the preferred option because of its greater degree of transparency to the clients, and thus, in this paper all examples will be performed with the use of ACR.

1.3.4 Overview: Sun Cluster 3.1

Sun Cluster 3.1 provides high availability (HA) by enabling application failover. The state of each individual node is periodically monitored. The cluster software automatically relocates a cluster-aware application from a failed primary node to a designated secondary node.

By allowing other nodes in a cluster to host workloads when the primary node fails, Sun Cluster 3.1 can significantly reduce downtime and increase productivity, providing HA service to all users.

Sun Cluster has the following key components:

Hardware

- Cluster nodes with local disks (unshared)
- Multihost storage (disks shared between nodes)
- Cluster interconnect for private communication
- Public network interfaces

Software

- Sun Solaris operating system
- Sun Cluster software

- Volume management (Solaris Volume Manager, Solstice DiskSuite or VERITAS Volume Manager)
- Data service application

Note that DB2 UDB does not pose any additional hardware or software requirement to implement in Sun Cluster. For the current software and hardware requirement for Sun Cluster, refer to Sun Cluster product Web site (http://www.sun.com/software/cluster) or contact your Sun account team.

- **Multihost Storage**

Sun Cluster 3.1 requires multihost disk storage—disks that can be connected to more than one node at a time. In the Sun Cluster 3.1 environment, multihost storage allows disk devices to become highly available. Disk devices resident on the multihost storage can tolerate single-node failures since there still exists a physical path to the data through the alternate server node.

Multihost disks can be accessed globally through a primary node; this node is said to master the disk. If client requests are accessing the data through one node and that node fails, the requests are switched over to another node that has a direct connection to the same disks.

A volume manager provides for mirrored or RAID5 configurations for data redundancy of the multihost disks. Currently, Sun Cluster 3.1 supports Solstice DiskSuite and VERITAS Volume Manager as volume managers. When running with Solaris 9 (or greater), Sun Cluster 3.1 supports Solaris Volume Manager. Combining multihost disks with disk mirroring and striping protects against both node failure and individual disk failure.

- **Global Devices**

Sun Cluster 3.1 uses global devices to provide cluster-wide, highly available access to any device in a cluster, from any node, without regard to the device's physical location. All disks are included in the global namespace with an assigned device ID (DID) and are configured as global devices. Therefore, the disks themselves are visible from all cluster nodes.

- **Cluster File Systems/Global File Systems**

A cluster file system, also referred to as a global file system, is a proxy between the kernel (on one node) and the underlying file system volume manager (on a node that has a physical connection to the disks). Cluster file systems are dependent on global devices with physical connections to one or more nodes. They are independent of the underlying file system and volume manager. The data only becomes available to all

nodes if the file systems on the disks are mounted globally as a cluster file system.

- **Device Group**

In Sun Cluster 3.1, all multihost storage must be controlled by the Sun Cluster framework. Disk groups are first created on the multihost disk. Then, they are registered as Sun Cluster disk device groups. A disk device group is a type of global device.

Multihost device groups are highly available. Disks are accessible through an alternate path if the node currently mastering the device group fails. The failure of the node mastering the device group does not affect access to the device group except for the time required to perform the recovery and consistency checks. During this time only, all requests are blocked (transparently to the application) until the system makes the device group available.

- **Resource Group Manager (RGM)**

The cluster facility known as the Resource Group Manager (RGM), provides the mechanism for HA. The RGM runs as a daemon on each cluster node. It automatically starts and stops resources on selected nodes according to preconfigured policies. The RGM allows a resource to be highly available in the event of a node failure or a reboot by stopping the resource on the affected node and starting it on another. The RGM also automatically starts and stops resource-specific monitors that can detect resource failures and relocate failing resources onto another node. It can also monitor other aspects of resource performance.

- **Data Services**

The term data service is used to describe a third-party application that has been configured to run on a cluster rather than on a single server. A data service includes the application software and Sun Cluster 3.1 software that starts, stops, and monitors the application.

Sun Cluster 3.1 supplies data service methods that are used to control and monitor the application within the cluster. These methods run under the control of the RGM, which uses them to start, stop, and monitor the application on the cluster nodes. These methods, along with the cluster framework software and multihost disks, enable applications to become highly available data services. As such, they can prevent significant application interruptions after any single failure within the cluster, regardless of whether the failure is to a node, to an interface component, or to the application itself.

The RGM also manages resources in the cluster, including network resources (logical hostnames and shared addresses) and instances of an application.

- **Resource Type, Resource, and Resource Group**

A resource type consists of three things:

• A software application to be run on the cluster
• Control programs used as callback methods by the RGM to manage the application as a cluster resource
• A set of properties that form part of the static configuration of a cluster

The RGM uses resource-type properties to manage resources of a particular type.

A resource inherits the properties and values of its resource type. It is an instance of the underlying application running on the cluster. Each instance requires a unique name within the cluster.

Each resource must be configured in a resource group. The RGM brings all resources in a group online and offline together on the same node. When the RGM brings a resource group online or offline, it invokes callback methods on the individual resources in the group.

The nodes on which a resource group is currently online are called its primaries, or its primary nodes. A resource group is mastered by each of its primaries. Each resource group has an associated Nodelist property, set by the cluster administrator, to identify all potential primaries or masters of the resource group.

- **HAStorage and HAStoragePlus**

HAStorage/HAStoragePlus is a type of Sun Cluster resource to ensure the affinity of the underlying storage devices with the HA resource. The HAStorage was introduced with the initial release of Sun Cluster 3.0. The HAStoragePlus was released with Sun Cluster 3.0 Update 3. The HAStorage resource supports raw device and cluster file system (CFS) only. The HAStoragePlus resource supports raw device, cluster file system (CFS), and failover file system (FFS).

- **CFS (Cluster File System) and FFS (Failover File System)**

CFS (Cluster File System) is a highly available, distributed, cache-coherent file system that allows ufs, hsfs, and vxfs file systems to be concurrently accessed on multiple cluster nodes.

FFS (Failover File System) refers to a locally mounted file system in a cluster environment. The FFS is only accessible to the cluster node

where it is mounted. In order to access this file system after a failover, this file system has to be failed over to the backup node.

1.4 Full System Overview

The HADR database is the database that is being replicated with the DB2 HADR technology. We refer to this database as *hadrdb* for the purposes of this document. Note that once the initial HADR pair is established, the HADR role of the instance can change; in response both to administrator commands and to external system state changes. Thus, it is possible that for any particular point in time, the instance db2instp can host the HADR standby database, it can host the HADR primary database, or it might not be part of a valid HADR pair at all. Regardless, the DB2 instance that initially hosts the HADR primary database is referred to as *db2instp,* and the DB2 instance that initially hosts the HADR standby database is referred to as *db2insts*, consistently. The physical machine that initially hosts the db2instp instance is referred to as *hostp* and the physical machine that initially hosts the db2insts instance is referred to as *hosts*.

Each instance is registered with the SC3.1. In contrast to a typical active/passive scenario, each instance is constrained to restart only on the physical host at which it is initially registered. Thus, instance db2instp will only restart on host hostp, while instance db2insts will only restart on host hosts.

Once the instances are made highly available, the HADR pair is created. The HADR pair can be formed using any of the methods described in the DB2 Version 8.2 documentation.

Once the pair is established, the cluster manager will control the state of the HADR pair. That is, it will control which of the two DB2 instances will host the database hadrdb as an HADR primary and which will host the database hadrdb as an HADR standby.

Sun Cluster 3.1 DB2-HA Agent

The DB2 agent is a software product consisting of the methods required to start, stop, monitor, and administer DB2 UDB in a Sun Cluster 3.1 environment. This agent integrates the robust, highly available DB2 database technology with Sun's clustering technology. The combined system architecture provides reliable, high-performance data services in environments from a workgroup up to the entire enterprise.

Most of these methods are implemented as shell scripts, which allows for maximum flexibility in the hands of a skilled user. For example, additional functionality can be added to alert the administrator by e-mail that a failure has occurred, or to start some other resources or processes in conjunction with a DB2 resource failover.

For more information, see the white paper called IBM® DB2® Universal Database™ and High Availability on Sun Cluster 3.1 at http://www.ibm.com/software/data/pubs/papers/.

2. CLUSTER SETUP

The sample two-node cluster, which we will reference later, is illustrated below and includes:

? Storage shared between nodes
? Two-node cluster nodes named sun-ha1, sun-ha2
? Two DB2 instances running DB2 UDB Version 8.2 (or greater)

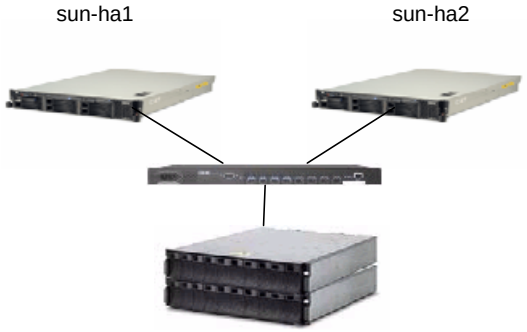


Diagram 1: Two-node cluster with shared disk

2.1 Configure IPMP

IP network multipathing (IPMP) was introduced in the Solaris 8 OE update 2 (10/00) software release and is present in all later releases of the Solaris operating system. With IPMP, network adapters function in the following modes:

- Active-Active
- Active-Standby

IP network multipathing coupled with multiple network connections per subnet provides a server with one or both of the following advantages:

- Resilience from network adapter failure

- Increased data throughput for outbound traffic

To provide resilience, IPMP detects the failure or repair of a network adapter, and switches the network address to or from an alternative adapter. If more than one network adapter is active, outbound packets are spread across adapters, thereby increasing data throughput. For full details, consult the relevant Solaris documentation. Here in this document, we provide a complete example for our system using IPMP in an Active-Standby mode to provide fault tolerance at the network adapter level. Note that IPMP must be configured (in either Active-Active or Active-Standby mode); otherwise, a loss of a single NIC may result in a system outage.

Ensure that all adapters have unique Media Access Control (MAC) addresses. Also ensure that two or more adapters (we will use exactly two in our example) are connected in the same subnet. Test IP addresses must also be reserved for each network adapter. These addresses are only used internally by IPMP and will not be known to external clients. In addition, one well-known IP address is required for each public network adapter group.

The values we will use are as follows. We choose to use the IPMP group name sc_ipmp0. For machine sun-ha1, adapter hme0 will be active and adapter ce0 will be the standby. The test IP address for adapter hme0 will be 9.26.49.160 with the test hostname of sun-ha1-hme0. For ce0, the test IP address will be 9.26.49.161 with test hostname of sun-ha1-ce0. For sun-ha2, hme0 will be active and ce0 will be standby. The test IP address for adapter hme0 will be 9.26.49.163 with test hostname of sun-ha2-hme0. For ce0, the test IP address will be 9.26.49.164 and the test hostname will be sun-ha2-ce0.

To achieve the described IPMP setup, perform the following steps.

Update /etc/hosts on sun-ha1 as follows:

```
# cat /etc/hosts
#
# Internet host table
#
127.0.0.1   localhost
9.26.48.254 sun-ha1 sun-ha1.torolab.ibm.com loghost
9.26.48.255 sun-ha2 sun-ha2.torolab.ibm.com
9.26.49.160 sun-ha1-hme0 \
            sun-ha1-hme0.torolab.ibm.com
9.26.49.161 sun-ha1-ce0  sun-ha1-ce0.torolab.ibm.com
9.26.49.163 sun-ha2-hme0 \
            sun-ha2-hme0.torolab.ibm.com
9.26.49.164 sun-ha2-ce0  sun-ha2-ce0.torolab.ibm.com
```

Update /etc/hosts on sun-ha2 as follows:

```
# cat /etc/hosts
#
# Internet host table
#
127.0.0.1   localhost
9.26.48.254 sun-ha1 sun-ha1.torolab.ibm.com
9.26.48.255 sun-ha2 sun-ha2.torolab.ibm.com loghost
9.26.49.160 sun-ha1-hme0 \
            sun-ha1-hme0.torolab.ibm.com
9.26.49.161 sun-ha1-ce0 \
            sun-ha1-ce0.torolab.ibm.com
9.26.49.163 sun-ha2-hme0 \
            sun-ha2-hme0.torolab.ibm.com
9.26.49.164 sun-ha2-ce0  sun-ha2-ce0.torolab.ibm.com
```

Ensure the /etc/netmasks file on each node contains the appropriate netmask information, for example (consult network administrator for the correct netmask for your environment):

```
# cat /etc/netmasks

9.0.0.0 255.255.254.0
```

Ensure that neither node is being used as a router. To do this, issue the following (at each node):

```
# touch /etc/notrouter
```

Plumb up the network interfaces that will be used (on each node). For example:

```
# ifconfig hme0 plumb
# ifconfig ce0 plumb
```

Then, at each node, create the IP multipathing group name:

```
# ifconfig ce0 group sc_ipmp0
# ifconfig hme0 group sc_ipmp0
```

Create the files /etc/hostname.hme0 and /etc/hostname.ce0 as follows.

For host sun-ha1:

```
# cat hostname.hme0
sun-ha1 netmask + broadcast + \
 group sc_ipmp0 up addif \
 sun-ha1-hme0 netmask + broadcast + \
 deprecated -failover up

# cat hostname.ce0
sun-ha1-ce0 netmask + broadcast + \
```

```
 deprecated group \
 sc_ipmp0 -failover standby up
```

For sun-ha2:

```
# cat hostname.hme0
sun-ha2 netmask + broadcast + \
 group sc_ipmp0 up addif \
 sun-ha2-hme0 netmask + broadcast + \
 deprecated -failover up

# cat hostname.ce0
sun-ha2-ce0  netmask + broadcast + \
 deprecated group \
 sc_ipmp0 -failover standby up
```

Reboot each node for the settings to take effect. Verify that IPMP was correctly configured by the scstat command. Output similar to this should be displayed for the Active-Standby case (it would differ slightly for the Active-Active case):

```
-- IPMP Groups --

Node Name           Group   Status  Adapter   Status
---------           -----   ------ -------    ------
IPMP Group: sun-ha2  sc_ipmp0 Online   hme0   Online
IPMP Group: sun-ha2  sc_ipmp0 Online   ce0    Standby

IPMP Group: sun-ha1  sc_ipmp0 Online   hme0   Online
IPMP Group: sun-ha1  sc_ipmp0 Online   ce0    Standby
```

3. DB2 UDB INSTALLATION AND INSTANCE CREATION

This is a two-stage process. First, the executable image is installed. Then, the instances are created.

3.1 DB2 UDB Installation on the Cluster nodes

DB2 UDB will be installed locally on both cluster nodes (sun-ha1 and sun-ha2). As the superuser, on each node, run the db2_install utility, and choose “DB2.ESE” to install the complete DB2 UDB V8.2 software. Take the default “/opt” as the base directory. The DB2 UDB software will then be installed in “/opt/IBM/db2/V8.1”.

3.1.1 Post Installation Tasks: Update Solaris Kernel Parameters:

To adapt kernel parameters required by DB2 UDB, follow the procedure described below. As the superuser, on each node, run the db2osconf utility to obtain the recommended DB2 kernel parameters. Update the /etc/system file with the entries given by the db2osconf tool. Note that

the values returned could be cut and pasted directly to the /etc/system file. To have the changes take effect, reboot the system.

For example,

```
# /opt/IBM/db2/V8.1/bin/db2osconf

set msgsys:msginfo_msgmax = 65535
set msgsys:msginfo_msgmnb = 65535
set msgsys:msginfo_msgmni = 3584
set msgsys:msginfo_msgtql = 3584
set semsys:seminfo_semmni = 4096
set semsys:seminfo_semmns = 8602
set semsys:seminfo_semmnu = 4096
set semsys:seminfo_semume = 240
set shmsys:shminfo_shmmax = 3752976384
set shmsys:shminfo_shmmni = 4096
set shmsys:shminfo_shmseg = 240

Total kernel space for IPC:
0.56MB (shm) + 2.79MB (sem) + 1.63MB (msg) == 4.98MB
```

Then, update the /etc/system file with these values and reboot the node with the following command:

```
# shutdown -y -g0 -i6
```

3.2 Create DB2 Instances

We work with instances named *db2instp* and *db2insts* in this example.

First, ensure that appropriate user and group IDs are created on each node of the cluster, and ensure that they are identical across each node.

On each physical node participating in the cluster, you need to create a separate group and user account for each of these:

? DB2 instance owner
? User ID that will execute fenced UDFs (user-defined functions)
? DB2 administration server (DAS)

If NIS or NIS+ is in use, groups and users must be created on the NIS server. If you are using local server authentication, repeat the following steps on all nodes in the cluster.

For example, use the following commands to create groups on each node in a local server authentication environment:

```
# groupadd -g 999 db2iadm1
# groupadd -g 998 db2fadm1
# groupadd -g 997 db2asgrp
```

You are then ready to create the user IDs as follows:

```
# useradd -g db2iadm1 -u 1005 \
   -d /global/global13/home/db2instp -m db2instp

# useradd -g db2iadm1 -u 1004 \
   -d /global/global13/home/db2insts -m db2insts

# useradd -g db2fadm1 -u 1003 -d \
   /global/global13/home/db2fenc1 -m db2fenc1

# useradd -g db2asgrp -u 1002 -d \
   /global/global13/home/db2as -m db2as
```

At this point, the appropriate user and group IDs exist at all nodes in the cluster.

Ensure that the instance is set up correctly. For example, if you are using local passwords, issue the following commands:

```
# rsh sun-ha1 "cat /etc/passwd | grep db2inst"
# rsh sun-ha2 "cat /etc/passwd | grep db2inst"
```

You should see output at each node similar to this:

```
db2instp:x:1005:999::/global/global13/home/db2instp:/bin/ksh
db2insts:x:1004:999::/global/global13/home/db2insts:/bin/ksh
```

Ensure that the /etc/services file is set up correctly (and the relevant entries are identical) across both nodes. Issue the following commands to verify those details:

```
# rsh sun-ha1 “cat /etc/services | grep inst”
# rsh sun-ha2 “cat /etc/services | grep inst”
```

You should see output similar to this:

```
xinstp      18817/tcp
xinsts      18818/tcp
hadrinstp   18819/tcp
hadrinsts   18820/tcp
```

Ensure that all nodes have identical entries. The name used for the services entry is not critical, but ensure that four entries exist with unique port numbers in the /etc/services file.

In the next step, we create the DB2 instances named db2instp and db2insts. To do this, issue the following commands:

```
# /opt/IBM/db2/V8.1/instance/db2icrt \
 -w 64 -u db2instp db2instp
```

```
# /opt/IBM/db2/V8.1/instance/db2icrt \
 -w 64 -u db2insts db2insts
```

The output of each command should be similar to this:

```
DBI1070I Program db2icrt completed successfully.
```

3.3 HA Enable DB2 Instance

This section provides instructions to enable the DB2 instances db2instp and db2insts for control by the SC3.1 infrastructure.

First, for instance db2instp, register the instance. Logon as root to node sun-ha1 and issue the following command:

```
# /opt/IBM/db2/V8.1/ha/sc30/util/regdb2udb \
 -a db2instp
```

Next, for instance db2insts, logon as root to node sun-ha2 and issue the following command:

```
# /opt/IBM/db2/V8.1/ha/sc30/util/regdb2udb \
-a db2insts
```

Next, bring each instance online.

For db2instp:

```
# /opt/IBM/db2/V8.1/ha/sc30/util/onlinedb2udb \
-a db2instp
```

For db2insts:

```
# /opt/IBM/db2/V8.1/ha/sc30/util/onlinedb2udb \
-a db2insts
```

Each instance is now highly available. The next steps involve creating an HADR pair and bringing the HADR state under cluster control.

4 MANAGE HADR WITH SUN CLUSTER 3.1

The following steps create a database and establish an HADR database.

4.1 Create a DB2 UDB database

We should now create a database under this instance. Logon to the server sun-ha1 as the instance owner db2instp and issue the following command:

```
% db2 create database hadrdb on /db
```

Note that the path at which the database is created can be any valid path on the primary machine.

Turn on LOGRETAIN for this database and take a database backup image as follows:

```
% db2 update db cfg for hadrdb using LOGRETAIN ON
% db2 backup database hadrdb
```

Transfer the database backup image to the machine sun-ha2 and perform a database RESTORE under the user ID of db2insts as follows:

```
% db2 restore database hadrdb
```

4.2 Configure Instances and Database

Also, ensure that the TCP/IP listener is started and listening on a well-known port. Issue the following commands as instance db2instp on node sun-ha1 and then as instance db2insts on node sun-ha2.

```
% db2set DB2COMM=tcpip
% db2 update dbm cfg using SVCENAME xinstp
```

Note that we will listen on the same port for both the primary and the standby instances.

Next, update the database-level configuration parameters required for an HADR pair to be established. For example, as instance db2instp, issue the following set of commands:

```
% db2 update db cfg for hadrdb using \
HADR_LOCAL_HOST sun-ha1
% db2 update db cfg for hadrdb using \
HADR_REMOTE_HOST sun-ha2
% db2 update db cfg for hadrdb using \
HADR_LOCAL_SVC 18819
% db2 update db cfg for hadrdb using \
HADR_REMOTE_SVC 18819
% db2 update db cfg for hadrdb using \
HADR_REMOTE_INST db2insts
```

Note that the values for the HADR_LOCAL_SVC and HADR_REMOTE_SVC were taken from the values placed in the /etc/services file earlier.

Now, update the parameters at instance db2insts as follows:

```
% db2 update db cfg for hadrdb using \
```

```
HADR_LOCAL_HOST sun-ha2
% db2 update db cfg for hadrdb using \
HADR_REMOTE_HOST sun-ha1
% db2 update db cfg for hadrdb using \
HADR_LOCAL_SVC 18819
% db2 update db cfg for hadrdb using \
HADR_REMOTE_SVC 18819
% db2 update db cfg for hadrdb using \
HADR_REMOTE_INST db2instp
```

Next, the alternate server values for each instance must be updated. In the following example, note that the IP address supplied as the hostname argument is exactly the IP address that was registered along with the instance db2insts, and that the port is the value used for the SVCENAME parameter at each instance. So, for instance db2instp, enter the following command:

```
% db2 update alternate server for \
database hadrdb using \
hostname sun-ha2 port 18817
```

As instance db2insts, issue the following command. Note that the argument supplied to the hostname is the IP used to register instance db2instp.

```
% db2 update alternate server for \
database hadrdb using \
hostname sun-ha1 port 18817
```

Then, as instance db2insts, issue the following command to start HADR as a standby database:

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

As instance db2instp, logon to node sun-ha1 and start the database there as an HADR primary database:

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

Once this is complete, ensure that the HADR pair is in peer state with the instance db2instp hosting the primary database hadrdb on the physical host sun-ha1 and the instance db2insts hosting the standby database hadrdb on the physical host sun-ha2. You can verify those details, for example, from the CLP using the DB2 snapshot command.

```
% db2 get snapshot for all on hadrdb
```

The output should be similar to the following example, taken from the primary node:

HADR Status

```
 Role              = Primary
 State             = Peer
 Synchronization mode  = Sync
 Connection status     = Connected,
07/12/2004 14:35:07.913742
 Heartbeats missed     = 0
 Local host          = sun-ha1
 Local service       = 18819
 Remote host         = sun-ha2
 Remote service      = 18819
 Remote instance     = db2insts
 timeout(seconds)    = 30
 Primary log position(file, page, LSN) = S0000000.LOG,
0, 00000000007D0000
 Standby log position(file, page, LSN) = S0000000.LOG,
0, 00000000007D0000
```

4.3 Create the HADR Resource Group

The resource group controlling the HADR state must be configured. To do this, we register the resource group at the node currently hosting the primary (for our example that is sun-ha1).

Issue the following command sequence:

```
# rlogin sun-ha1
# cd /opt/IBM/db2/V8.1/ha/sc30/util
#./reghadr -a db2instp -b db2insts -d hadrdb
```

Where the argument to *-a* is the name of the instance hosting the HADR primary database, the argument to *-b* is the name of the instance hosting the HADR standby database, and the argument to -d is the name of the HADR database itself.

The HADR pair state is now controlled by cluster manager. That is to say, the node at which the resource group is online is the node at which the database is an HADR primary state. Verify this with the following scstat command:

```
# scstat -g
```

```
-- Resource Groups and Resources --

        Group Name       Resources
        ----------       ---------
 Resources: db2_db2instp_0-rg  db2_db2instp_0-rs
 Resources: db2_db2insts_0-rg  db2_db2insts_0-rs
 Resources: db2hadr_hadrdb-rg  db2hadr_hadrdb-rs

-- Resource Groups --

        Group Name           Node Name     State
        ----------           ---------     -----
   Group: db2_db2instp_0-rg   sun-ha1       Online
   Group: db2_db2insts_0-rg   sun-ha2       Online

   Group: db2hadr_hadrdb-rg   sun-ha2       Offline
   Group: db2hadr_hadrdb-rg   sun-ha1       Online

-- Resources --

        Resource Name      Node Name     State     Status Message
        -------------      ---------     -----     --------------
  Resource: db2_db2instp_0-rs   sun-ha1  Online   Online
  Resource: db2_db2insts_0-rs   sun-ha2  Online   Online

  Resource: db2hadr_hadrdb-rs   sun-ha2  Offline  Online - HADR Standby, Peer
  Resource: db2hadr_hadrdb-rs   sun-ha1  Online   Online - HADR Primary, Peer
```

Now, if the physical host *sun-ha1* fails, the resource group *db2hadr_hadr-rg* will fail over to be hosted by physical host *sun-ha2*, which will cause the HADR primary database to be hosted now on *sun-ha2*.

4.4 Catalog the DB2 Instance

At each client, ensure that the server is cataloged.

```
% db2 catalog tcpip node hadrdb \
remote sun-ha1 server 18817
```

5. TESTING AND RESPONSE TO FAILURES

Controlled Failover Testing

Change the location of the node hosting the HADR primary database with the following command:

```
# scswitch -z -g db2hadr_hadrdb-rg -h sun-ha2
```

You should issue a scstat command and see that the HADR primary database is now hosted by the machine sun-ha2. To bring the HADR primary database back to being hosted by sun-ha1, issue the command:

```
# scswitch -z -g db2hadr_hadrdb-rg -h sun-ha1
```

Once scstat indicates that the HADR primary database is hosted by instance db2instp, issue a DB2 snapshot command. Verify that the instance db2instp is hosting the HADR primary database hadrdb and that db2insts is hosting the HADR standby database hadrdb. Further, verify that the databases are in peer state.

Testing Instance Failure: Primary Instance

Logon to the node sun-ha1 as the instance owner db2instp and issue the following command (this will simulate an instance failure):

```
% kill -9 $(ps -ef | grep db2sysc \
  | grep –v grep | awk ‘{print $2}’)
```

Issue a scstat command repeatedly and observe that the resource for the db2 instance goes offline briefly in response to the command and is quickly restarted on the same node. Verify that the HADR pair has reached Peer State

Testing Instance Failure: Standby Instance

Logon to the node sun-ha2 as the instance owner db2insts and issue the following command (this will simulate an instance failure):

```
% kill -9 $(ps -ef | grep db2sysc \
  | grep –v grep | awk ‘{print $2}’)
```

Issue a scstat command repeatedly and observe that the resource for the db2 instance goes offline briefly in response to the command and is quickly restarted on the same node. Applications will continue to process against the instance hosting the HADR primary database.

Testing Adapter Failure

Pull the cable on one of the network adapters. You should see at most a brief interruption of service while the IP address fails over to the other adapter or adapters in that IPMP group.

Testing Machine Failure

Turn off the power on the primary machine (or alternatively, issue as root the reboot -nf command). The resource group db2hadr_hadrdb-rg will fail over to node sun-ha2, causing instance db2insts to host the HADR primary for database hadrdb.

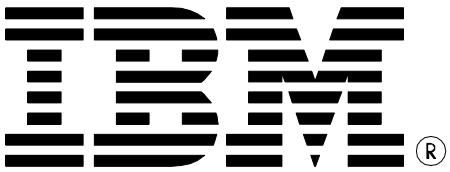
For this type of takeover, a takeover by force may have been issued by the resource group. If this has been the case, in order to prevent any possible split-brain scenario from occurring, the resource group controlling instance db2instp will be offlined. This will ensure that the instance db2instp will not come online without some operator intervention.

```
# scswitch -e -j db2_db2instp_0-rs
# scswitch -z -g db2_db2instp_0-rg -h sun-ha1
# su - db2instp
% db2 start hadr on db hadrdb as standby
```

The HADR pair should now be re-established. To bring the primary back to being hosted by the instance db2instp, move the HADR resource group to node sun-ha1 via the following command:

```
# scswitch -z -g db2hadr_hadrdb-rg -h sun-ha1
```

Note that successful reintegration is not assured after a TAKEOVER BY FORCE for sync modes other than SYNC. If the reintegration steps given above fail to establish an HADR pair in peer state, then it will be necessary to manually create a new standby database via an image backup of the current primary database.



IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

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.