November 2005

DB2 Information Management Software

Automating DB2 Universal Database (DB2 UDB) HADR Failover using Heartbeat for Linux

Authors:

Mary Lynn Kitaura
Melody Ng
Steve Raspudic
Anand Subramanian
Jing Yang

IBM Toronto Software Lab

Table of Contents

1. Introduction and Overview

By completing the steps described in this paper, you will accomplish the implementation of an automated IBM® DB2® Universal Database™ (DB2 UDB) failover solution. This solution is based on a combination of the High Availability Disaster Recovery (HADR) feature in DB2 UDB Version 8.2 and the open source Linux-HA project (Heartbeat) on the Linux® platform. This paper also includes a detailed description of the installation, configuration, and test issues of HADR when failover is automated in the Heartbeat environment.

Target Audience for this Paper:
- DB2 UDB database administrators
- Linux system administrators

2. Before You Begin

Below you will find information about knowledge requirements, as well as hardware and software configurations used to set up the environment depicted in this paper. It is important that you read this section prior to beginning any setup.

2.1 Knowledge Requirements

- Basic knowledge of Linux system administration
- Basic knowledge of DB2 UDB and HADR*

*Information about DB2 UDB HADR can be found here: http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/core/c0011585.htm

2.2 Software Configuration Used

The minimum software requirement for running DB2 UDB on Linux is listed here: http://www.ibm.com/db2/linux/validate. For information about software requirements for running Heartbeat, refer to http://www.linux-ha.org.

Listed below are the actual software configuration used to set up the environment for this paper:

- Operating system : Red Hat Enterprise Linux AS 4 (kernel version 2.6.9-11.ELsmp)
- DB2 UDB product : DB2 UDB Enterprise Server Edition (ESE) Version 8.2
- Heartbeat : Heartbeat-1.2.3-2 built for Red Hat Enterprise Linux 3
- libnet : libnet-1.1.2.1
- mon : mon-0.99.2

2.3 Hardware Configuration Used

The minimum hardware requirements for implementing the solution described in this paper are the same as those documented for DB2 UDB here: http://www.ibm.com/db2/udb/sysreqs.html.

Listed below is the actual hardware configuration used to set up the environment for this paper:

- Processors used : Intel® Xeon® 2 CPU 2.80 GHz
- Memory : 2.60 GB
- Network adapters : Two Intel PRO/100 Ethernet Adapters

3. System Overview

3.1 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, reliability, and availability.

3.2 High Availability Disaster Recovery (HADR)

DB2 UDB HADR is a database 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 record shipping technology, where transactions that occur on the primary database are applied to the designated standby database. In the case that the primary database becomes inaccessible, clients can continue to submit transactions by automatically switching to the standby database.

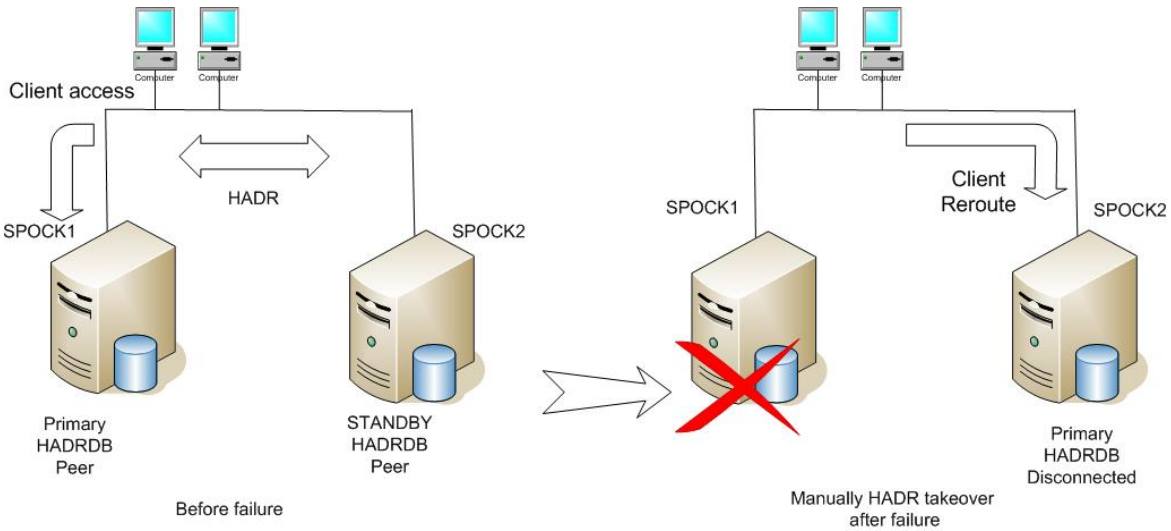

Fig 1. Typical HADR Environment

3.3 Automating HADR Takeover with Heartbeat

HADR does not automatically monitor the environment for a primary database server outage. Instead, you must monitor the HADR pair and manually issue appropriate takeover commands in the event of a primary database server failure. This is where Heartbeat can help.

Heartbeat is a fundamental part of the High-Availability Linux project [1]. It provides core cluster management services including membership, communication, resource monitoring and management services, and IP address takeover.

Heartbeat provides high availability by automating resources such as processes, applications, and IP addresses in Linux-based clusters. To automate an IT resource (for example, an IP address), the resource must be defined to Heartbeat.

Every application must be defined as a resource in order to be managed and automated with Heartbeat.

HADR can be viewed as a resource that is controlled by Heartbeat. When a primary database server outage occurs, Heartbeat will detect the node failure and fail over the resources hosted on that node (the HADR-enabled database) to the standby node, and thereby continue to provide service to the clients.

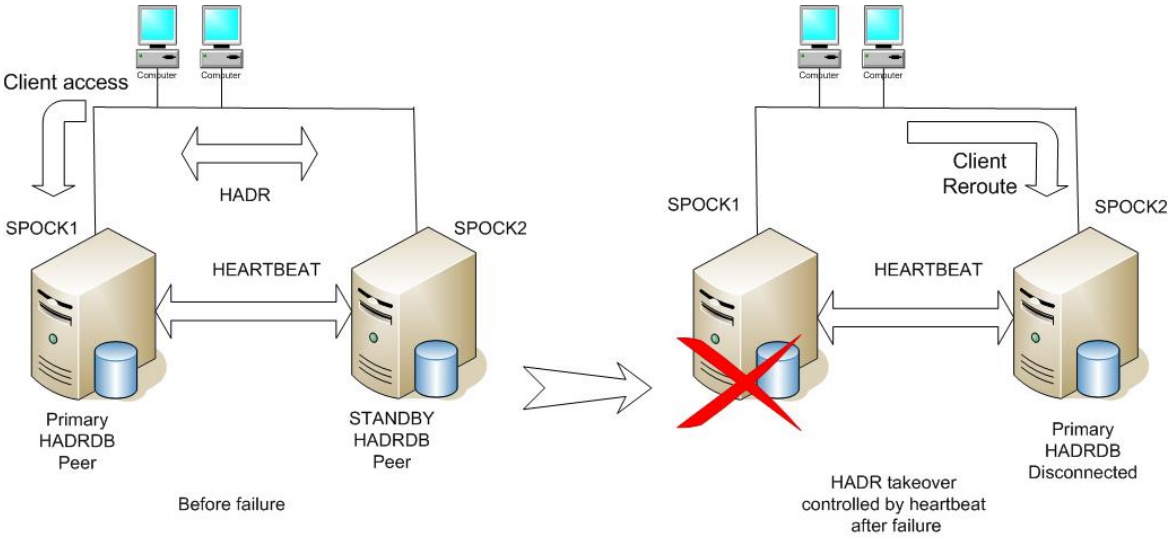

Fig 2. Typical HADR Environment with Heartbeat

3.4 IPfail

The design of Heartbeat is such that if any inter-node communication link is available, then the nodes consider each other to be available. However, this is not always the desired behavior. For example, if a pair of nodes have physical links on both the public and private network, it may be desirable for failover to occur only if the public adapter on the active node fails, because the client would no longer be able to access the primary HADR database in this case.

The IPfail for Heartbeat makes this possible by monitoring one or more external hosts, known as ping nodes. Typically this would be a router or gateway on the local network. The ping node is treated as a quorum device. If connectivity to a quorum

device is lost, then Heartbeat's resource manager may be configured to trigger a failover.

3.5 mon

Heartbeat Version 1.2.3 can only detect node failure. It cannot detect services failure, which would also cause the primary HADR database to be unavailable to the client. For example, Heartbeat Version 1.2.3 could not detect DB2 instance failure on the primary node. Instead, you can use the third-party free software package "mon" [2] to monitor services on the active node.

> **Note**:
> The recently released version of Heartbeat Version 2.0.0 provides support for the monitoring of resources (services) and supports larger size Linux clusters. However, the IPfail feature is not supported in Version 2.0.0 of Heartbeat, and will probably be supported in future releases.

mon is a general-purpose scheduler and alert management tool used for monitoring service availability and triggering alerts upon failure detection. It was designed to be open and extensible through supporting arbitrary monitoring facilities and alert methods via a common interface.

In the setup used for this paper, mon is working as a resource that is controlled by Heartbeat.

4. Overview of Topology Setup

This section details the topology in the environment setup used in this paper.

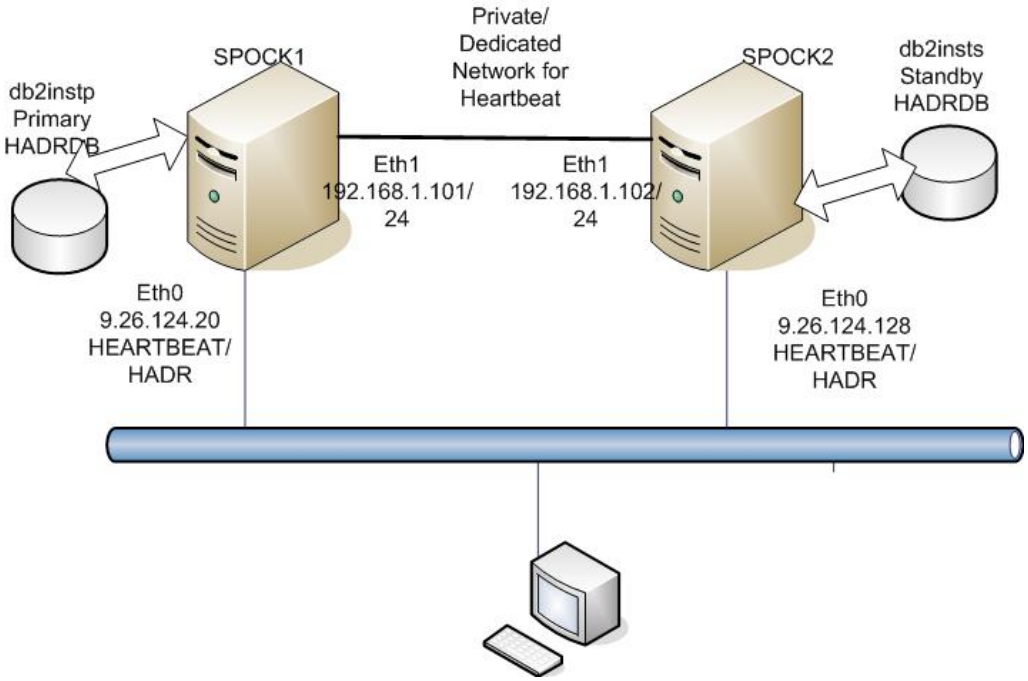

Fig 3. System Architecture

The setup for this paper uses two nodes, SPOCK1 and SPOCK2. SPOCK1 acts as the primary (master) node in the Heartbeat cluster and SPOCK2 acts as the standby (or slave) node.

The HADR database is the database that is being replicated across the two nodes through the DB2 UDB HADR technology. We refer to this database as SAMPLEDB for the purpose of this paper. Note that once the initial HADR pair is established, the HADR role of the database instance can change in response both to administrator commands and to external system state changes. Thus, it is possible that at a particular point in time, the instance db2instp on SPOCK1 might host the HADR standby database, or it might 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 named db2instp, and the DB2 instance that initially hosts the HADR standby database is named db2insts, consistently. The physical machine that hosts the db2instp instance is referred to as the primary node and physical machine that hosts the db2insts instance is referred to as the standby node.

The active node in this paper refers to the node currently hosting all the resources, and standby node refers to the node currently not hosting any resources.

Heartbeat uses heartbeat messages to monitor the status of the nodes in a cluster. The nodes periodically send each other signals (known as hearbeats) via a network. It is suggested that a redundant heartbeat path is provided by carrying the heartbeat through both the public LAN and a dedicated network. The dedicated network can be a dedicated LAN or a serial connection between the cluster nodes. In the setup for this paper, there are two network adapters on each node. One adapter is connected via the public network, and other one is connected via the dedicated private network for internal communication only. In case of local switching failure on the public network, the dedicated private heartbeat network will prevent a false diagnosis of heartbeat failure. Each node has its own IP address for the dedicated network.

5. Steps to Set Up the Topology

The following setup steps are required to successfully create the cluster and enable the resources to fail over between the cluster nodes when a failure happens:

- Set Up RSH
- Install DB2 UDB and Create an Instance
- Set Up the HADR database
- Install and Set Up Heartbeat (Including the IPfail Feature)
- Install and Configure mon and mon resources, which will be controlled by Heartbeat
- Configure the HADR resource, which will also be controlled by Heartbeat
- Start the Heartbeat
- Catalog the DB2 Instance at the Client Side
- Start Up the Client

> **Note:**
> The letters in front of a command in the following steps designate which node or nodes a command is to be issued on to properly set up the topology in Fig. 3 above. The order of the letters also designates the order that you should issue a command on each node:
>
> (P) = Primary Node (i.e., SPOCK1)
> (S) = Standby Node (i.e., SPOCK2)
> (C) = Client Node

> **Note:**
> Each command is shown as preceded by the command prompt "#" for the root user and "%" for the DB2 instance owner (i.e., db2instp and db2insts). A "\" in a command designates that the command text continues on the next line. **Do not** include the command prompt "#", "%", or the "\" when you issue the command.

5.1 Set Up RSH

Some of the commands in the shell scripts used in the environment require RSH to be set up on the primary and standby nodes. RSH allows a user from one node to run commands on another node remotely.

For detailed information about how to set up RSH on Red Hat Linux, refer to the following link:

http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/ref-guide/s1-pam-rexec.html

5.2 Install DB2 UDB and Create an Instance

This is a two-stage process. First, the DB2 UDB executable image is installed. Then, the DB2 instances are created.

a) Install DB2 UDB on the Primary and Standby Nodes

Install DB2 UDB V8.2 ESE on both of the cluster nodes (SPOCK1 and SPOCK2). Take the default "/opt" as the base directory. The DB2 UDB software will then be installed in "/opt/IBM/db2/V8.1". **Do not** create instances at this step.

For information about how to install DB2 UDB, refer to the product manual here:
ftp://ftp.software.ibm.com/ps/products/db2/info/vr82/pdf/en_US/db2ite81.pdf

b) Prepare the Primary and Standby Nodes for DB2 Instance Creation
Create appropriate groups and users to manage DB2 UDB on the primary and standby nodes, respectively.

1. As root, create the following groups for instance management on both SPOCK1 and SPOCK2 by issuing the following command on both nodes:

(P)(S) # groupadd –g 999 db2iadm1

2. As root, create a group for fenced users on both SPOCK1 and SPOCK2 by issuing the following command on both nodes:

   (P)(S) # groupadd –g 998 db2fadm1

3. As root, create the following users for instance management:

   On SPOCK1, issue the following command:

   (P) # useradd –g db2iadm1 –u 1005 –d \
   /misc/home/db2instp –m db2instp

   On SPOCK2, issue the following command:

   (S) # useradd –g db2iadm1 –u 1005 –d \
   /misc/home/db2insts –m db2insts

4. As root, create a fenced user on both SPOCK1 and SPOCK2, by issuing the following command on both nodes:

   (P)(S) # useradd –g db2fadm1 –u 1003 –d \
   /misc/home/db2fenc1 –m db2fenc1

c) Create the Primary and Standby Instances

Create the primary and standby instances (i.e., db2instp and db2insts, respectively) that will manage the HADR-enabled database (i.e., SAMPLEDB) by issuing the following commands:

1. Change directory on both SPOCK1 and SPOCK2 by issuing the following command:

   (P)(S) # cd /opt/IBM/db2/V8.1/instance

2. As root, create the primary and standby database manager instances as follows:

   On SPOCK1, issue the following command:

   (P) # ./db2icrt –w 32 –u db2fenc1 db2instp

   On SPOCK2, issue the following command:

   (S) # ./db2icrt –w 32 –u db2fenc1 db2insts

5.3 Set Up the HADR Database

1. Configure the /etc/services File

   DB2 UDB must start a HADR service on both the primary and standby nodes which ships logs between primary and standby database using the service port provided. As root, add the following entries to the /etc/services file on both nodes.

   On **(P)(S)**:
   ```
   hadrinstp  18819/tcp
   hadrinsts  18820/tcp
   ```

   Also, specify a TCP/IP port that a database server will use to await communications from remote client nodes. To do this, add the following entry to the /etc/services file on both nodes.

   On **(P)(S)**:
   ```
   xinstp  18817/tcp
   ```

   Ensure that both nodes have identical entries. The name used for the services entry is not critical, but make sure the port numbers used for these three entries are unique from all other entries in the /etc/services file.

2. Create a Standard Database SAMPLEDB

   To create a standard database under the DB2 instance db2instp on the primary node, log on to the primary node as the instance owner db2instp and issue the following command:

   (P) % db2 create database SAMPLEDB on /db

   Note that the path on which you create the database can be any valid path on the primary node.

3. Back up the Database SAMPLEDB

   As the instance owner db2instp, turn on LOGRETAIN for this database and take a database backup image as follows:

   ```
   (P) % db2 update db cfg for SAMPLEDB using LOGARCHMETH1 \
         LOGRETAIN
   (P) % db2 backup database SAMPLEDB to \
         <Directory_containing_backup_image>
   ```

   Note that the path on which you want to store the database backup can be any valid path on the primary node.

4. Restore the Database SAMPLEDB

   Transfer the database backup image to the standby machine and perform a database RESTORE as the user db2insts, as follows:

   (S) % db2 restore database SAMPLEDB from <restore_directory>

   Note that <restore_directory> is the directory where you store the database backup image on the standby node.

   > **Please Note:**
   > You should not perform a redirected restore. The database must be created using the same files, devices, or file systems if you are using absolute paths.

5. Configure Instances and Database

   We need to configure the instance to allow the client to communicate with it via the TCP/IP service port. Issue the following commands as user db2instp on the primary node and then as user db2insts on the standby node.

   (P)(S) % db2set DB2COMM=tcpip
   (P)(S) % db2 update dbm cfg using SVCENAME xinstp

   Note that xinstp is the service name we entered earlier in the /etc/services file.

   Next, update the database level configuration parameters required for a HADR pair to be established. As the user db2instp, issue the following commands:

   (P) % db2 update db cfg for SAMPLEDB using HADR_LOCAL_HOST \
   9.26.124.20
   (P) % db2 update db cfg for SAMPLEDB using HADR_REMOTE_HOST \
   9.26.124.128
   (P) % db2 update db cfg for SAMPLEDB using HADR_LOCAL_SVC \
   18819
   (P) % db2 update db cfg for SAMPLEDB using HADR_REMOTE_SVC \
   18820
   (P) % db2 update db cfg for SAMPLEDB using HADR_REMOTE_INST \
   db2insts

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

   Now, as user db2insts on the standby node, issuing the following command to update the database level configuration parameters there:

   (S) % db2 update db cfg for SAMPLEDB using HADR_LOCAL_HOST \
   9.26.124.128
   (S) % db2 update db cfg for SAMPLEDB using HADR_REMOTE_HOST \
   9.26.124.20

(S) % db2 update db cfg for SAMPLEDB using HADR_LOCAL_SVC \
18820
(S) % db2 update db cfg for SAMPLEDB using HADR_REMOTE_SVC \
18819
(S) % db2 update db cfg for SAMPLEDB using HADR_REMOTE_INST \
db2instp

6. Start HADR on Database SAMPLEDB

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

(S) % db2 start hadr on db SAMPLEDB as standby

For instance db2instp, log on to the primary node as user db2instp and start the database there as a HADR primary, by issuing the following command:

(P) % db2 start hadr on db SAMPLEDB as primary

Once this is completed, ensure that the HADR pair is in peer state with instance db2instp hosting the primary HADR database on the primary node and instance db2insts hosting the standby HADR database on the standby node. You can verify this through issuing the following DB2 snapshot command:

(P)(S) % db2 get snapshot for all on SAMPLEDB

The "HADR Status" section of 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
  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
……
```

5.4 Install and Set Up Heartbeat (Including the IPfail Feature)

This section takes you through the set up of Heartbeat in order to automate the failover of the HADR-enabled database from the primary HADR database server to the standby server in case of a primary outage.

5.4.1 Download and Install Heartbeat

Download Heartbeat and install it on both the primary and standby nodes. You will also need to install libnet, before you can compile Heartbeat.

Obtain the tarball for libnet from the following URL:

http://www.packetfactory.net/libnet/dist/libnet-1.1.2.1.tar.gz

As root, install libnet on both machines by issuing the following commands (in the order given):

(P)(S) # cd /<directory_containing_install_code>
(P)(S) # tar zxvf libnet-1.1.2.1.tar.gz
(P)(S) # cd libnet
(P)(S) # ./configure
(P)(S) # make
(P)(S) # make install

After installing libnet, you should be able to install and compile Heartbeat.

Obtain the tarball for Heartbeat from the following URL:

http://www.ultramonkey.org/download/heartbeat/1.2.3/heartbeat-1.2.3.tar.gz

Install Heartbeat on both machines by entering the following commands (in the order given):

(P)(S) # cd /<directory_containing_install_code>
(P)(S) # tar zxvf heartbeat-1.2.3.tar.gz
(P)(S) # cd heartbeat-1.2.3
(P)(S) # ./configure
(P)(S) # make
(P)(S) # make install

5.4.2 Configure Heartbeat

You must configure three files to get Heartbeat to work: authkeys, ha.cf, and haresources. For detailed information, refer to the Heartbeat Web site:

http://www.linux-ha.org/

The specific configuration used in the environment setup for this paper is as follows. It is important to make sure that all the configuration files are exactly the same on both nodes.

- Configure /usr/local/etc/ha.d/authkeys

This file determines your authentication keys for the cluster. The keys must be the same on both nodes. You can choose from three authentication schemes: crc, md5, or sha1.

The format of the file is as follows:

```
auth <number>
<number> <authmethod> [<authkey>]
```

The /usr/local/etc/ha.d/authkeys file in our setup is as follows.

/usr/local/etc/ha.d/authkeys configuration file:

```
auth 3
3 md5 Hello!
```

- Configure /usr/local/etc/ha.d/ha.cf

This file tells Heartbeat what types of media paths to use and how to configure them. This file also defines the nodes in the cluster and the interfaces that Heartbeat uses to verify whether or not a system is up. The relevant portion of the /usr/local/etc/ha.d/ha.cf file for the setup is as follows.

/usr/local/etc/ha.d/ha.cf configuration file:

```
……
#       Facility to use for syslog()/logger
logfacility     local0
#
#       keepalive: how long between heartbeats?
keepalive 2
#
#       deadtime: how long-to-declare-host-dead?
deadtime 5
#
#       What UDP port to use for bcast/ucast communication?
#
udpport 694
#       What interfaces to broadcast heartbeats over?
bcast   eth0 eth1               # Linux
#
auto_failback off
#
#       Tell what machines are in the cluster
#       node    nodename ...    -- must match uname -n
node    spock1
node    spock2
# Specified ping nodes for IPfail
ping_group group1 9.26.124.1
#
#Enable the IPfail feature
respawn hacluster /usr/local/lib/heartbeat/ipfail
```

Please Note:
In our setup, the auto failback option in the ha.cf file is set to off. This means a resource will not automatically fail back to its original "primary" node when the primary node comes back after failure, but will remain on whatever node is serving it until that node fails.

- Configure /usr/local/etc/ha.d/haresources

  This file describes the resources that are managed by Heartbeat.

  /usr/local/etc/ha.d/haresources configuration file:

```
spock1 9.26.124.44 mon Hadr
```

This line indicates that, on startup, Heartbeat will:

- Have SPOCK1 serves the IP address 9.26.124.44 (floating IP)
- Start the mon daemon
- Execute the HADR script with –start option

On shutdown, Heartbeat will:

- Execute the HADR script with –stop option
- Stop the mon daemon
- Give up the IP address 9.26.124.44

5.5 Configure mon and mon resources

To configure mon to monitor the DB2 instances db2instp and db2insts, perform the following steps on both the primary and standby nodes. Make sure all the configuration files are exactly the same on both nodes. (The IP address provided to hostgroup db2inst and hadr_stat in the mon.cf file will be different on each node.)

5.5.1 Download and Install mon

Obtain the mon package from the following URL and download it to the /etc/ha.d directory:

ftp://ftp.kernel.org/pub/software/admin/mon/

Install the mon package under the /etc/ha.d directory using the following commands.

(P)(S) # cd /etc/ha.d
(P)(S) # tar xzvf mon-0.99.2.tar.gz

mon requires the following Perl modules to work properly. You can obtain the packages from the CPAN Web site http://www.cpan.org:

- Time::Period
- Time::HiRes
- Convert::BER
- Mon::*

Unpack these modules and install them by issuing the following commands (in the order given):

(P)(S) # perl Makefile.pl
(P)(S) # make
(P)(S) # make test
(P)(S) # make install

5.5.2 Configure mon

In this setup, four resources are being monitored: the local DB2 instance, the remote DB2 instance, the HADR state, and the public/private network adapter. The method in which the resources are monitored is specified in the /etc/ha.d/mon-0.99.2/mon.cf configuration file.

> **Please Note:**
> The host specified in the hostgroup entry in the mon.cf file will be different on each node. For example, on node SPOCK1, the host specified should be "spock1", and on SPOCK2, the host specified should be "spock2".

- Monitoring the Local DB2 Instance

The following entry is an excerpt from the mon.cf configuration file in our test setup, which specifies how the DB2 instance is being monitored. It indicates that:

- Hostgroup "db2_inst" includes the node "spock1".
- mon will monitor the service "db2inst" for all the nodes in hostgroup "db2_inst".
- The monitor script "db2_inst.monitor" will be triggered every 10 seconds.
- If the return code of monitor script is not 0, which means the DB2 instance is not up and running, the alert script "db2_inst.alert" will be triggered.
- The alert script will only be triggered after 2 consecutive failures specified by the alertafter parameter.
- The alert script "db2_inst.alert" will restart the DB2 instance in place and activate the SAMPLEDB database.
- If the service continues to fail, the alert will only be triggered once every 10 minutes, which is specified by the alertevery parameter.

Relevant portion of mon.cf for monitoring the local DB2 instance is as follows:

```
hostgroup db2_inst  spock1

watch db2_inst
    service db2inst
        interval 10s
        monitor db2_inst.monitor
        allow_empty_group
        period wd {Mon-Sun}
            alert db2inst_down.alert
            alertevery 600s
                alertafter 2
```

For detailed information about the monitor and alert scripts for checking the local DB2 instance, refer to Section 10 "Appendix".

- Monitoring the Remote DB2 Instance

The remote DB2 instance state is basically monitored in the same way as the local DB2 instance is. The monitor script will periodically check the status of the remote instance, and will call the alert script if the instance on the remote side is not up and running. The alert script will restart the instance in place when triggered.

The relevant portion of mon.cf for monitoring the remote DB2 instance is as follows:

```
hostgroup db2_inst_remote  spock1

watch db2_inst_remote
    service db2inst_remote
        interval 10s
        monitor db2_inst_remote.monitor
        allow_empty_group
        period wd {Mon-Sun}
            alert db2inst_down_remote.alert
            alertevery 600s
                alertafter 2
```

For detailed information about the monitor and alert scripts for checking and restarting the remote DB2 instance, refer to Section 10 “Appendix”.

- Monitoring the HADR Pair

The HADR state is basically monitored the same way as a DB2 instance is. The monitor script “hadr.monitor” is triggered every 25 seconds to check the status of the HADR database (i.e., SAMPLEDB). The monitor has a dependency on the DB2 instance monitor, which means if the DB2 instance is down, the HADR monitor stops running until the instance becomes up and running again.

The relevant portion of mon.cf for monitoring the HADR state is as follows:

```
hostgroup hadr_stat spock1

watch hadr_stat
    service hadr_status
        interval 25s
        monitor hadr.monitor
        depend db2_inst:db2inst
        allow_empty_group
        period wd {Mon-Sun}
            alert hadr_down.alert
            alertevery 600s
                alertafter 2
```

For detailed information about the monitor and alert scripts for checking the HADR state, refer to Section 10 “Appendix”.

- Monitoring both Network Adapters Failure

In this setup, the heartbeat message is broadcasted through two network adapters, the public network adapter and the private network adapter. When both adapters on either node fail, for example, both adapters on the primary node fail, the standby node will think the primary node is down and take over all the resources.

Note that when the above scenario occurs, there is the possibility that both copies of your database will operate independently as primaries. (This is sometimes referred

to as split brain or dual primary.) In this case, each primary database may accept connections and perform transactions, and neither will receive and replay the updates made by the other. As a result, the two copies of the database will become inconsistent with each other. To prevent this from happening, we will need to shut down the failed node.

In our test setup, the mon daemon detects the failure of the above scenario as follows: mon is configured to periodically ping a group of network instances in and outside the cluster. If the ping fails in a given period of time, mon will shut down the node on which it is running. The ping nodes should be carefully selected. In this setup, two ping nodes are chosen, one is the default gateway of the public subnet, and another one is the private IP address of the other node. This ensures that the fail node will be shut down only when both the public and private adapters have failed.

The relevant portion of mon.cf for monitoring network adapter failure is as follows:

```
hostgroup gateway 9.26.124.1 192.168.1.102

watch gateway
service ping
        interval 10s
        monitor ping.monitor
        allow_empty_group
        period wd {Mon-Sun}
            alert nic_down.alert
            alertevery 600s
            alertafter 2
```

Where:

- The two IP addresses provided are the IP address of the default gateway of the public subnet and the private IP address of the other node in the cluster.
- The monitor script “ping.monitor” will be triggered every 10 seconds, and will ping the two provided IP addresses periodically. If it does not get any response from both the ping nodes, it will trigger the “nic_down.alert” script.
- The “nic_down.alert” script will complete the following tasks:
    1) Stop the mon daemon.
    2) Kill the DB2 instance
    3) Kill the heartbeat process and IPfail process
    4) Shut down the node.

> **Note:**
> The private IP address specified in the above entry should be different for each node in the cluster. On SPOCK1, the private IP address in mon.cf should read 192.168.1.102. On SPOCK2, the private IP address should read 192.168.1.101.

For detailed information about the monitor and alert scripts for checking the state of the network adapters, refer to Section 10 “Appendix”.

- Monitor and Alert scripts

Put all the monitor scripts mentioned in the /etc/ha.d/mon-0.99.2/mon.cf file under the directory /etc/ha.d/mon-0.99.2/mon.d
Put all the alert scripts referred to in the /etc/ha.d/mon-0.99.2/mon.cf file under the directory /etc/ha.d/mon-0.99.2/alert.d
Make sure all the scripts are identical on both nodes.
See Section 10 “Appendix” for detailed information on the monitor and alert scripts.

5.5.3 Configure the mon-start Script

Create the /etc/ha.d/mon-0.99.2/mon-start script to be called by Heartbeat later. The “mon-start” script will start the mon daemon if provided with the “start” option, and will kill the mon daemon if provided with the “stop” option.

/etc/ha.d/mon-0.99.2/mon-start script:

```
#!/bin/bash
MON_HOME=/etc/ha.d/mon-0.99.2

case "$1" in
    start)
        if [ -f $MON_HOME/mon.pid ]; then
                logger –i –p INFO –t $0 “INFO: mon already
started"
                exit
        fi
        logger –i -p INFO –t $0  "INFO: Starting mon…"
    $MON_HOME/mon -c $MON_HOME/mon.cf -L $MON_HOME -P
MON_HOME/mon.pid &
        ;;
    stop)
    if [ -f $MON_HOME/mon.pid ]; then
    logger –i –p INFO –t $0  "INFO: Stopping mon…"
        kill -9 `cat $MON_HOME/mon.pid`
        rm  -f $MON_HOME/mon.pid
        kill -9 $(ps –ef |grep  db2sysc|grep –v grep|awk
’{print $2}’)
    else
        logger –i –p ERROR –t $0  "ERROR: No server pid,
server doesn't seem to run"
    fi
    ;;
    *)
    echo "Usage: $0 {start|stop}"
    exit 1
esac
exit 0
```

At startup, Heartbeat looks in /etc/rc.d/init.d or /usr/local/etc/ha.d/resource.d for the resource start scripts. Create symbolic links in /usr/local/etc/ha.d/resource.d directory for the mon-start file by issuing the following commands on both nodes:

(P)(S) # cd /usr/local/etc/ha.d/resource.d
(P)(S) # ln –s /etc/ha.d/mon-0.99.2/mon-start mon

5.6 Configure the HADR resource

Create the HADR script under the /usr/local/etc/ha.d/resource.d directory, which will be called by Heartbeat.

Refer to Section 10 “Appendix” for detailed information about the HADR script.

5.7 Start the Heartbeat

> **Note:**
> Make sure that the primary HADR database will initially be running on the cluster node that is supposed to keep the resource when Heartbeat is started. For example, if you bring up the primary HADR database on SPOCK1, remember to start the Heartbeat service first on SPOCK1 as well, so that the resource will be brought online on that node.

As root, issue the following command on the primary and then the standby node to start the heartbeat service:

(P)(S) # /usr/local/etc/ha.d/heartbeat start

5.8 Catalog the DB2 Instance at the Client Side

- Usage of the Highly Available IP Address vs. Client Reroute

  The automatic client reroute (ACR) feature in DB2 UDB allows client applications to recover from a loss of communication with the database server so that they can continue to work with minimal interruption. If the client-to-server communication fails, then the client is automatically rerouted to an alternate server.

  ACR can be used with HADR to automatically establish client applications connections to the new primary database after a HADR takeover operation. If ACR is not enabled, client applications will receive SQL error message SQL30081, and no further attempts will be made to establish a connection with the server.

  Note that in environments where Client Reroute cannot be used or is not desired, it is possible to associate a highly available floating IP address with the HADR resource. Heartbeat provides a method to allow floating IP addresses in the cluster (refer to the /usr/local/etc/ha.d/haresources configuration file). It is possible to use either ACR or the cluster manager’s ability to move IP addresses between physical nodes in the cluster for redirecting the clients to the physical location of the new database server.

- Configure Automatic Client Reroute

> **Note:**
> If you configure the ACR in this step, you do not need to configure the HA floating IP described in the next section.
> Otherwise, if you plan to configure the HA floating IP, skip this part and go directly to the next section for configuring the HA floating IP.

In order to support the Automatic Client Reroute functionality that is available in DB2 UDB Version 8.2, the alternate server information for the HADR database must be updated at each instance.

As user db2instp, log on to the primary node and enter the following command:

```
(P) % db2 update alternate server for database SAMPLEDB using \
        hostname 9.26.124.128 port 18817
```

Note that the IP address supplied as the hostname argument is the public IP address of the standby node and the port is the value used for the SVCENAME parameter at each instance.

As instance db2insts, log on to the standby node and issue the following command:

```
(S) % db2 update alternate server for database SAMPLEDB using \
        hostname 9.26.124.20 port 18817
```

Note that the IP address supplied as the hostname argument is the public IP address of the primary node and the port is the value used for the SVCENAME parameter at each instance.

- Configure the HA floating IP

  On the server side, as the instance owner, configure the alternate server to point to the floating IP at both nodes:

```
(P)(S)% db2 update alternate server for db SAMPLEDB using \
          hostname 9.26.124.44 port 18817
```

  The IP address specified in the above command is the HA floating IP.

- Catalog the DB2 instance at the Client Side

  At each client, ensure that the server is cataloged via a command similar to the following, where the IP address specified is the active node hosting the primary HADR database:

```
(C) % db2 catalog tcpip node spocknode remote 9.26.124.20 server 18817
```

  In the setup for this paper, it is the IP address of SPOCK1. The port for the server is defined on both instance nodes and is the same value that is used for the SVCENAME configuration parameter on the server.

Catalog the HADR database at the client side by issuing the following command:

(C) % db2 catalog db SAMPLEDB at node spocknode authentication client

5.9 Set Up the Client

To shorten the time it takes for the client to re-establish a connection when a machine or adapter failure occurs on the server side, it is recommended that you modify the TCP/IP parameters on the client host by running the following shell script:

```
#!/bin/ksh
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 3000 > /proc/sys/net/core/netdev_max_backlog
echo 3000 > /proc/sys/net/core/somaxconn
# send tcp_keepalive_probes every
tcp_keepalive_intvl
#  after idling for tcp_keepalive_time
echo 2 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 2 > /proc/sys/net/ipv4/tcp_keepalive_intvl
echo 2 > /proc/sys/net/ipv4/tcp_keepalive_probes
#
echo 3 > /proc/sys/net/ipv4/tcp_retries1
echo 3 > /proc/sys/net/ipv4/tcp_retries2
/etc/init.d/xinetd restart
```

Also set the following DB2 Profile Registry on the client side by issuing the following command:

(C) %db2set DB2TCP_CLIENT_RCVTIMEOUT=10

6. Testing the Common Failures

For all of the following tests, it is recommended to have an active client load against the database system similar to what could be expected in the production environment.

6.1 Controlled Failover Test

Note:
In order to get the status for the HADR database SAMPLEDB when issuing the "/etc/init.d/heartbeat status" command, modify the /etc/init.d/heartbeat script and add the following lines to the file:

```
StatusHA() {
  $HA_BIN/heartbeat -s
  $HA_BIN/ResourceManager status Hadr # (the line added to call the \
                                      Hadr  script with "- status" argument)
}
```

First, verify that the HADR pair is in peer state. To do this, issue the following command as root at the primary node:

(P) # /etc/init.d/heartbeat status

The output should be similar to the following example:

```
# /etc/init.d/heartbeat status
heartbeat OK [pid 28523 et al] is running on spock1
[spock1]...
---------------------------------
Heartbeat resources are running on this node
---------------------------------
spock1

HADR Status
  Role                    = Primary
  State                   = Peer
  Synchronization mode    = Nearsync
  Connection status       = Connected, 07/05/2005
10:02:03.184911
  Heartbeats missed       = 0
  Local host              = 9.26.124.20
  Local service           = 18819
  Remote host             = 9.26.124.128
  Remote service          = 18820
  Remote instance         = db2insts
  timeout(seconds)        = 30
  Primary log position(file, page, LSN) =
S0000004.LOG, 0, 0000000001770000
  Standby log position(file, page, LSN) =
S0000004.LOG, 0, 0000000001770000
  Log gap running average(bytes) = 0
```

To test the controlled failover, enter the following command on SPOCK1:

(P) #/etc/init.d/heartbeat stop

The output should be similar to the following example:

```
# /etc/init.d/heartbeat stop
Stopping High-Availability services:
                                        [  OK  ]
```

Check the HADR database status on SPOCK2 after doing the controlled failover, and observe that the takeover is initiated on SPOCK2 (the takeover operation may take a moment to complete):

(S) # /etc/init.d/heartbeat status

The output should similar to the following example:

```
# /etc/init.d/heartbeat status
heartbeat OK [pid 24247 et al] is running on spock2
[spock2]...
---------------------------------
 Heartbeat resources running on this node
---------------------------------
spock2

HADR Status
  Role                    = Primary
  State                   = Disconnected
  Synchronization mode    = Nearsync
  Connection status       = Disconnected, 08/22/2005
12:34:05.962213
  Heartbeats missed       = 0
  Local host              = spock2
  Local service           = 18820
  Remote host             = spock1
  Remote service          = 18819
  Remote instance         = db2instp
  timeout(seconds)        = 120
  Primary log position(file, page, LSN) =
S0000001.LOG, 0, 0000000000BB8000
  Standby log position(file, page, LSN) =
S0000001.LOG, 0, 0000000000BB8000
  Log gap running average(bytes) = 0
```

Note:
Heartbeat will stop the DB2 instance on the node where the command "/etc/init.d/heartbeat stop" is issued. However, after the heartbeat on the standby node takes over all the resources, the new active node will try to restart the db2 instance remotely on the node where the heartbeat has been stopped earlier and start HADR as standby for the database SAMPLEDB on that node if the instance has been restarted.

To re-integrate the original primary node to the cluster, issue the following command as root:

(P) # /etc/init.d/heartbeat start

6.2 Instance Failure Test

- **Primary Instance Failure**

First make sure the HADR database SAMPLEDB is in peer state and the primary SAMPLEDB is running on the primary node.

On the primary node, issue the following command as root:

(P) # su – db2instp –c " db2_kill"

This will kill the DB2 instance on the primary node.
Now check the HADR status repeatedly by issuing the command "/etc/init.d/heartbeat status" and observe in a short period of time that the DB2 instance on the primary node will be restarted and the primary HADR database will come back and enter peer state.

- **Standby Instance Failure**

First make sure the HADR database SAMPLEDB is in peer state and the primary SAMPLEDB is running on the primary node.

On the standby node, issue the following command as root:

(S)# su – db2instp –c " db2_kill"

This will kill the DB2 instance on the standby node.
Now check the HADR status repeatedly by issuing the command "/etc/init.d/heartbeat status" and observe in a short period of time that the DB2 instance on the standby node will be restarted and the standby HADR database will come back and enter peer state.

6.3 Machine Failure Test

First make sure the HADR database SAMPLEDB is in peer state and the primary SAMPLEDB is running on the primary node.

Shut down the primary node by issuing the following command:

(P) # shutdown –Fr now

The resources HADR and mon will fail over to the standby node, causing the DB2 instance db2insts to become the HADR primary server for database SAMPLEDB.

Check the HADR status on the original standby node continuously and observe that the database role has been switched to Primary after a while.

The output should be similar to the following example:

```
#/etc/init.d/heartbeat status
heartbeat OK [pid 7895 et al] is running on spock2
[spock2]...
---------------------------------
 Heartbeat resources are running on this node
---------------------------------
spock2

HADR Status
  Role                   = Primary
  State                  = Disconnected
  Synchronization mode   = Nearsync
  Connection status      = Disconnected, 08/20/2005
16:03:08.108237
  Heartbeats missed      = 0
  Local host             = spock2
  Local service          = 18820
  Remote host            = spock1
  Remote service         = 18819
  Remote instance        = db2instp
  timeout(seconds)       = 120
  Primary log position(file, page, LSN) =
S0000001.LOG, 0, 0000000000BB8000
  Standby log position(file, page, LSN) =
S0000001.LOG, 0, 0000000000BB8000
  Log gap running average(bytes) = 0
```

To re-establish the HADR pair once the original primary node has come back online, issue the following commands at the that node as root:

(P) # /etc/init.d/heartbeat start

6.4 Adapter Failure Test

In the setup used for this paper, the heartbeat message is broadcasted through two network adapters, the public network adapter and the private network adapter dedicated for heartbeat communication only. If one of the adapters on either of the nodes fails, the heartbeat message can still be broadcasted through the other adapter, so the other node will not initiate a takeover. However, if the public network adapter on the active node fails, the client would not be able to access the primary database. This is where IPfail can help.

If the public adapter on the active node fails, Heartbeat on this node will lose communication to the ping nodes defined in the "/usr/local/etc/ha.d/ha.cf" file earlier. IPfail on the active node will then contact the other node via the private network to see if the ping nodes can be reached from the standby node. If the standby node can reach the ping nodes, then the heartbeat will perform a failover operation and move the resources to that node. If the standby node cannot reach the ping nodes either, the failover operation will not be initiated.

- **Public adapter failure on the primary node**

Make sure the HADR database SAMPLEDB is in peer state and the primary SAMPLEDB is running on the primary node.
Pull the Ethernet cable from the public network adapter on the primary node. Observe that the Heartbeat will perform a failover operation and move the primary HADR SAMPLEDB to the standby node.

- **Private adapter failure on the primary node**

Make sure the HADR database SAMPLEDB is in peer state and the primary SAMPLEDB is running on the primary node.
Pull the Ethernet cable from the private network adapter on the primary node. Check the system log and observe that Heartbeat has detected the private adapter failure but no resource takeover has occurred.

- **Public and private adapter failures on the primary node**

Make sure the HADR database SAMPLEDB is in peer state and the primary SAMPLEDB is running on the primary node.
Pull the Ethernet cable from both the public and private network adapters on the primary node. Check the system log on both nodes and observe that the standby node initiates a takeover in a short period of time (which causes the split-brain scenario), and mon on the original primary node has detected the adapter failures and triggered the alert script to shut down the original primary node.

- **Public and private adapter failures on the standby node**

Make sure the HADR database SAMPLEDB is in peer state and the primary SAMPLEDB is running on the primary node.
Pull the Ethernet cable from both the public and private network adapters on the standby node. Check the system log on both nodes and observe that the standby node initiates a takeover in a short period of time (which causes the split-brain scenario), and mon on the original standby node has detected the adapter failures and triggered the alert script to shut down the original standby node.

7. Brief Introduction to Heartbeat Version 2.0.0

Heartbeat Version 2.0.0 was released in August 2005. This version provides support for resources (services) monitoring and support for a larger cluster size (allowing the number of nodes in the cluster to be greater than two). However, the IPfail feature is not currently supported in this new version and will probably be supported in a future release.

The configuration process for Heartbeat Version 2.0.0 will be somewhat different from Version 1.2.3. For Heartbeat Version 2.0.0, the configuration file for the resource agent (previously /etc/ha.d/haresources found in Version 1.2.3) is in XML format. Also, resources can now be added and deleted dynamically via the “cibadmin” tool while the heartbeat is running.

The following basic steps outline how to configure the HADR resource for Heartbeat Version 2.0.0:

1) On both the primary and standby nodes, install Heartbeat from the source by issuing the following commands as root:
   (P)(S) # ./ConfigureMe configure
   (P)(S) # ./ConfigureMe make
   (P)(S) # ./ConfigureMe make install
2) As root, add the user and group to be used by Heartbeat on both the primary and standby nodes:
   (P)(S) # useradd hacluster
   (P)(S) # groupadd haclient
3) To enable the features of Version 2.0.0, edit the /etc/ha.d/ha.cf file on both nodes and add the following line to it:
   crm yes
   This enables the Cluster Resource Manager (CRM), new to Version 2.0.0 of Heartbeat.
4) As root, start Heartbeat on both nodes by issuing the following commands in the order given:
   (P) # /etc/init.d/heartbeat start
   (S) # /etc/init.d/heartbeat start
5) Edit the resource configuration file and define all the resources in that file in XML format. The following example shows a basic configuration in our HADR setup. (The file name used is file.xml)

```
<group id="MyHadr">
  <primitive id="HADRIP" class="ocf" type="IPaddr" provider="heartbeat">
    <instance_attributes>
      <attributes>
        <nvpair name="ip" value="9.26.124.44"/>
      </attributes>
    </instance_attributes>
  </primitive>
  <primitive id="Hadr" class="heartbeat" type="Hadr" >
    <operations>
         <op id="1" name="stop"  timeout="3s"/>
         <op id="2" name="start" timeout="5s"/>
         <op id="3" name="monitor" interval="30s"  timeout="30s"/>
    </operations>
    <instance_attributes>
      <attributes>
      </attributes>
    </instance_attributes>
  </primitive>
  <primitive id="checkInstRemote" class="heartbeat" type="checkInstRemote" >
    <operations>
         <op id="1" name="stop"  timeout="3s"/>
         <op id="2" name="start" timeout="5s"/>
         <op id="3" name="monitor" interval="30s"  timeout="30s"/>
    </operations>
    <instance_attributes>
      <attributes>
      </attributes>
    </instance_attributes>
  </primitive>
</group>
```

Here we define a resource group called “MyHadr”. This resource group contains three resources: HADRIP, Hadr and checkRemoteInst. Basically, HADRIP defines the floating IP address serving for the primary HADR database. The checkRemoteInst resource will periodically check to see if the remote DB2 instance is up and running.

> **Note:**
> “Hadr” and “checkInstRemote”, defined as primitive IDs in the XML configuration file, are the names of the scripts that are found in the directory /etc/ha.d/resource.d/.
> The script should be able to pass positional argument like “start”, “stop”, and “status”. When the script is called by Heartbeat with “-monitor” argument, it will execute the “status” part of the script.

Register the resource group to the CIB (cluster information base) using the following command on one of the nodes in the cluster as root:

```
# cat file.xml |cibadmin –C –o resource
```

1) The configuration file will then be generated and will be put under the directory /var/lib/heartbeat/crm. The configuration file name is cib.xml.
2) The CIB will populate the configuration to other nodes in the cluster.
3) You can also add the resources dynamically by using the “cibadmin” tool without having to stop the heartbeat. For detailed information, visit the following Web site:

   http://www.linux-ha.org/ClusterInformationBase_2fUserGuide

8. Conclusion

This paper described the automated failover solution for DB2 UDB HADR on the Linux platform using Heartbeat and mon. This solution provides high availability in two aspects. First, the transaction log shipping provided by HADR ensures the redundancy and disaster recovery of the HADR-enabled database. Secondly, Heartbeat and mon together provide failure detection for DB2 instances, network adapters, and machines. The combined solution provides high availability and automated failover for your database in mission-critical product environments.

9. References

[1] http://www.linux-ha.org
[2] http://www.kernel.org/software/mon/
[3] http://www.geocities.com/latompa/ha/apache_heartbeat.html
[4] http://www.ultramonkey.org/3/ipfail.html
[5] http://www.ibm.com/developerworks/library/l-halinux/?ca=dnt-541

10. Appendix

Note:
For each of the following monitor and alert scripts, edit the following lines, replacing the database name, hostname, and instance name with the actual names used in your environment:

```
DB2HADRDBNAME=SAMPLEDB
# Get the local instance name
tempHost=$(echo $HOSTNAME)
case $tempHost in
       spock1)
       DB2INST=db2instp;;
       spock2)
       DB2INST=db2insts;;
esac
```

/usr/local/etc/ha.d/resource.d/Hadr:

```
#!/bin/ksh
# This script will implement the HADR takeover operation if
# provided with “- start” argument and current local node hosts
# a standby HADR database.
# It will redirect local HADR database information to the
# standard output if provided with “- status” argument.
# It will do nothing if provided with “- stop” argument.
#
# Define the temp file which the all the heartbeat resource
# status information will redirect to this file.
temp_stat_file=/heartbeat/scriptTemp.stat

#Create the temp files if they do not exist
touch $temp_stat_file > /dev/null 2> /dev/null
chmod a+w $temp_stat_file > /dev/null 2> /dev/null
#
DB2HADRDBNAME=SAMPLEDB
# Get the local instance name
tempHost=$(echo $HOSTNAME)
 case $tempHost in
       spock1)
       DB2INST=db2instp;;
       spock2)
       DB2INST=db2insts;;
 esac

# Argument parsing
case "$1" in
# If the argument provided is "start", get the HADR Role on this node.,
# If the role is standby, then do a HADR takeover; in case
# the takeover is not successful, do a “takeover by force”.
 start)
    hadrRole=$(su - $DB2INST -c "db2 get snapshot for all on
    $DB2HADRDBNAME" |grep -i "Role" |awk '{print $3 }')
   if [[ $hadrRole = "Standby" ]];
```

```
   then
     logger -i -p notice -t $0 "su - $DB2INST -c db2 takeover hadr on
      db $DB2HADRDBNAME"
     su - $DB2INST -c "db2 takeover hadr on db $DB2HADRDBNAME" >
      /dev/null
     if [[ $? = "0" ]];
     then
       logger -i -p info -t $0 "Info: db2 takeover succeed "
     else
     logger -i -p notice -t $0 "su - $DB2INST -c db2 takeover hadr on
      db $DB2HADRDBNAME by force"
       su - $DB2INST -c "db2 takeover hadr on db $DB2HADRDBNAME by
      force" > /dev/null
       if [[ $? = "0" ]];
       then
       logger -i -p notice -t $0 "Notice: Takeover by force is succeed"
       else
       logger -i -p err -t $0 "ERROR: Takeover by force failed"
       fi
     fi
# If the HADR Role is primary, then do not issue a takeover.
   else
    logger -i -p notice -t $0 "Notice:  Primary SAMPLEDB is running on
    this node"
   fi;;
# If the argument is "stop", then do nothing.
 stop)
     logger -i -p notice -t $0 "Notice: stop the heartbeat";;
# If the argument provided is "status", then show customer the
# heartbeat information on this node.
 status)
     logger -i -p notice -t $0 "Notice: get $DB2HADRDBNAME status....."
       echo "---------------------------------" >> $temp_stat_file
       # If the mon pid is on this node, it indicates all the Heartbeat
       # resources are running on this node; otherwise, all the
       # resources are running on the other node
       if [[ -f /etc/ha.d/mon-0.99.2/mon.pid ]];
       then
       echo " Heartbeat resources running on this node " >>
      $temp_stat_file
       else
       echo " Heartbeat resources running on the other node " >>
      $temp_stat_file
       fi
       echo "---------------------------------" >> $temp_stat_file
       echo  $HOSTNAME >> $temp_stat_file
       echo "" >> $temp_stat_file
       # Get the HADR information on this node
       su - $DB2INST -c "db2 get snapshot for all on SAMPLEDB" |grep -
A14
      "HADR Status" >> $temp_stat_file
       # Output all the information to the standard output
       cat $temp_stat_file 1>&2
       rm -f $temp_stat_file;;
esac
exit 0
```

/etc/ha.d/mon-0.99.2/mon.d/db2_inst.monitor:

```
# This script monitors the local DB2 instance activity
#!/bin/ksh
#
DB2HADRDBNAME=SAMPLEDB
# Get the local instance name
tempHost=$(echo $HOSTNAME)
   case $tempHost in
       spock1)
       DB2INST=db2instp;;
       spock2)
       DB2INST=db2insts;;
   Esac
# Check if the local instance is up; return 0 if it is up
# Return 1 if the instance is down
INSTUP=$(su - $DB2INST -c "db2pd -inst" |grep -i "Database Partition"
|grep -c "Active")
   if [ $INSTUP -eq 0 ]
   then
      logger -i -p error -t $0 "ERROR: DB2 INSTANCE $DB2INST IS DOWN"
      exit 1
   else
      logger -i -p info -t $0 "INFO: DB2 INSTANCE $DB2INST IS UP "
      exit 0
   fi
```

/etc/ha.d/mon-0.99.2/alert.d/db2inst_down.alert:

```
#!/bin/ksh
# This script tries to restart the local DB2 instance if it is down
#
DB2HADRDBNAME=SAMPLEDB
# Get the local instance name
tempHost=$(echo $HOSTNAME)
 case $tempHost in
       spock1)
       DB2INST=db2instp;;
       spock2)
       DB2INST=db2insts;;
 esac

logger -i -p notice -t $0 "NOTICE: TRYING TO START THE INSTANCE"
# Restart the DB2 instance
su - $DB2INST -c "db2start"
if [[ $? = "0" ]];
then
    logger -i -p info -t $0 "INFO: DB2 INSTANCE HAS BEEN RESTARTED"
    su - $DB2INST -c "db2 activate db $DB2HADRDBNAME"
else
    logger -i -p ERROR -t $0 "ERROR: DB2 INSTANCE CAN NOT BE STARTED"
fi
```

Note:
For each of the following monitor and alert scripts, edit the following lines, replacing the database name, hostname, and instance name with the actual names used in your environment:

```
DB2HADRDBNAME=SAMPLEDB
# Get the local instance name
tempHost=$(echo $HOSTNAME)
case $tempHost in
       spock1)
       DB2INST=db2instp;;
       spock2)
       DB2INST=db2insts;;
esac
```

/etc/ha.d/mon-0.99.2/mon.d/db2_inst_remote.monitor:

```
#!/bin/ksh
# This script monitors the instance activity on remote node
# DB2HADRDBNAME=SAMPLEDB
# Get the remote hostname
tempHost=$(echo $HOSTNAME)
  case $tempHost in
         spock1)
         REMOTEHOST=spock2
      DB2INT=db2insts;;
         spock2)
         REMOTEHOST=spock1
      DB2INST=db2instp;;
  esac
# Check if the remote instance is up; if it is up,
# Return 0; otherwise, return 1
  INSTUP=$(rsh $REMOTEHOST ps -ef |grep -c db2sysc )
  if [ $INSTUP -eq 0 ]
  then
    logger -i -p error -t $0 "ERROR: DB2 INSTANCE ON REMOTE HOST IS
    DOWN"
      sleep 50
      exit 1
  else
    logger -i -p info -t $0 "INFO: DB2 INSTANCE ON REMOTE HOST IS UP"
    exit 0
  fi
```

/etc/ha.d/mon-0.99.2/alert.d/db2inst_down_remote.alert:

```
#!/bin/ksh
#This script tries to restart the remote instance and start SAMPLEDB
# as standby on remote side if the instance restarts successfully.
HADRDBNAME=SAMPLEDB
# Get the remote instance name and hostname
   tempHost=$(echo $HOSTNAME)
      case $tempHost in
         spock1)
              REMOTEHOST=spock2
                  DB2INST=db2insts;;
         spock2)
              REMOTEHOST=spock1
                  DB2INST=db2instp;;
     esac
logger -i -p notice -t $0 "NOTICE: TRYING TO START THE REMOTE INSTANCE"
      rsh $REMOTEHOST "su - $DB2INST -c 'db2start'"
      if [[ $? = "0" ]];
      then
  logger -i -p info -t $0 "INFO: REMOTE DB2 INSTANCE HAS BEEN RESTARTED"
      sleep 10
      rsh $REMOTEHOST "su - $DB2INST -c 'db2 start hadr on db
      $HADRDBNAME as standby'"
      else
  logger -i -p ERROR -t $0 "ERROR: REMOTE DB2 INSTANCE CAN NOT BE
  STARTED"
      fi
```

/etc/ha.d/mon-0.99.2/mon.d/hadr.monitor:

```
#!/bin/ksh
This script gets the status of SAMPLEDB and redirect to the syslog
DB2HADRDBNAME=SAMPLEDB
#Get the local DB2 instance name
tempHost=$(echo $HOSTNAME)
   case $tempHost in
       spock1)
       DB2INST=db2instp;;
       spock2)
       DB2INST=db2insts;;
   esac
HADRROLE="UNKNOWN"
HADRSTAT="UNKNOWN"
HADRROLE=$(su - $DB2INST -c "db2pd -hadr -db $DB2HADRDBNAME" |grep -A2
"HADR Information" |grep -v HADR |grep -v Role | awk '{print $1}')
HADRSTAT=$(su - $DB2INST -c "db2pd -hadr -db $DB2HADRDBNAME" |grep -A2
"HADR Information" |grep -v HADR |grep -v Role | awk '{print $2}')
   if [[ $? = "0" ]];
   then
      logger -i -p info -t $0 "Info: $DB2HADRDBNAME ROLE IS $HADRROLE,
      state is in $HADRSTAT state"
      exit 0
   else
      logger -i -p error -t $0 "ERROR: CAN NOT GET INFORMATION FOR
      $DB2HADRDBNAME "
      exit 1
   fi
```

/etc/ha.d/mon-0.99.2/alert.d/hadr_down.alert:

```
#!/bin/ksh
#Alert script when get HADR status failed
logger -i -p err -t $0 "ERROR: Can not get SAMPLEDB info "
```

/etc/ha.d/mon-0.99.2/mon.d/ping.monitor:

```
#!/bin/ksh
# This script will try to ping all the nodes provided as arguments
# Return 1 if all the nodes cannot be reached, otherwise return 0
plat=`uname`
p=`which ping`
if [ ! -x "$p" ]
then
    echo ping.monitor error, could not find ping
    exit 1
fi

if [ "$#" = 0 ]
then
    echo ping: no hosts found
    exit 1
fi

case "$plat" in
    Linux)
      PING="ping -c 1"
      ;;
    SunOS)
      PING="/usr/sbin/ping"
      ;;
    NetBSD|OpenBSD)
      PING="/sbin/ping -c 1"
      ;;
    *)
      echo "unknown plat <$plat>"
      exit 1
      ;;
esac

failed=""

for h in "$@"
do
    if $PING $h >/dev/null 2>/dev/null
    then
      :
    else
      if [ "$failed" = "" ]
      then
          failed="$h"
      else
          failed="$failed $h"
      fi
    fi
done
```

```
if [[ "$failed" = "$@" ]];
then
    logger -i -p error -t $0 " Failed to ping $failed"
    logger -i -p error -t $0 "ERROR: NIC MAYBE DOWN..."
    exit 1
else
    logger -i -p INFO -t $0 "INFO: Service adapter is up..."
fi
```

/etc/ha.d/mon-0.99.2/alert.d/nic_down.alert:

```
#!/bin/ksh
# This script will try to shut down the node if all the adapters
# on it fail. Before shutting down the node, stop the mon daemon,
# kill the DB2 instance and kill the heartbeat.
logger -i -p ERROR -t $0 "ERROR: NIC DOWN, STOP THE MON DAEMON...."
/etc/ha.d/mon-0.99.2/alert.d/mon-stop
logger -i -p INFO -t $0 "INFO: MON DAEMON STOPPED...."
logger -i -p ERROR -t $0 "ERROR: NIC DOWN, STOP THE DB2 INSTANCE..."
kill -9 $(ps -ef |grep db2sysc |grep -v grep |awk '{print $2}')
logger -i -p ERROR -t $0 "ERROR: NIC DOWN, STOP THE HEARTBEAT/IPFAIL.. "
kill -9 $(ps -ef |grep heartbeat  |grep master |awk '{print $2}')
kill -9 $(ps -ef |grep heartbeat  |grep ipfail |awk '{print $2}')
shutdown -Fr now
```

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

Printed in United States of America
11-05