July 2004

DB2 Information Management Software

High Availability for partitioned databases using Tivoli System Automation

Authors:
Enrico Joedecke, IBM Boeblingen Lab
Steve Raspudic, IBM Toronto Lab
Baris Naciterhan, IBM Toronto Lab

Contents

1. OVERVIEW

1.1 Objective

In this paper, we describe the implementation and design of highly available IBM® DB2® Universal Database™ (DB2 UDB) environments on the IBM Tivoli® System Automation for Multiplatforms (Tivoli SAM). Also included is a detailed description of the DB2 UDB agent for Tivoli SAM. We provide guidance and recommendations for highly available strategies using DB2 UDB Version 8.x Enterprise Server Edition (ESE) with the Database Partitioning Feature. Practical considerations regarding design, implementation, testing, and maintenance work with the system are also discussed.

1.2 Overview: DB2 UDB

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. This combination, DB2 UDB coupled with Tivoli SAM, forms a highly available computing environment. For more information, see *www.software.ibm.com/data*.

1.3 Overview: IBM Tivoli System Automation for Multiplatforms

IBM System Automation for Multiplatforms provides high availability (HA) by automating the control of IT resources such as processes, file systems, IP addresses, and other resources in Linux®-based clusters. It facilitates the automatic switching of users, applications, and data from one system to another in the cluster after a hardware or software failure. A complete highly available setup includes many parts, one of which is the HA software. As well as tangible items such as hardware and software, a good HA solution includes planning, design, customizing, and change control.

An HA solution reduces the amount of time that an application is unavailable by removing single points of failure. For more information, see *www.software.ibm.com/tivoli/products/sys-auto-linux*.

1.4 Overview: Minimum software and hardware levels

DB2 UDB ESE - Database Partitioning Feature V 8.1.6 or later
Tivoli System Automation for Multiplatforms Version 1.2 or later
Linux on the IBM eServer™, zSeries®, xSeries®, or pSeries® platforms, or any 32-bit Intel®-based server.

See http://www.ibm.com/software/tivoli/products/sys-auto-linux/platforms.html for the latest information pertaining to hardware supported by Tivoli SAM.

See http://www.ibm.com/software/data/db2/linux/validate/ for the latest information concerning Linux distributions supported by DB2 UDB.

The Linux distribution chosen must meet the requirements for both Tivoli SAM and DB2 UDB in order to be supported for the purposes of automating DB2 UDB with Tivoli SAM.

2. DB2 UDB IN AN HA ENVIRONMENT

In a single-partition DB2 UDB setup, a single DB2 UDB instance is running on a server. This DB2 UDB instance has local access to data (its own executable image as well as databases owned by the instance). If this DB2 UDB instance is made accessible to remote clients, an (unused) IP address must be assigned to this DB2 UDB instance.

The DB2 UDB instance, the local data, and the IP address are considered resources, which must be automated by Tivoli SAM. Since these resources are closely related (for example, they collectively run on the same node at the same time), they are called a *resource group*.

The entire resource group is collocated on one node in the cluster. In the case of a failover, the entire resource group is started on another node.

There are the following dependencies among the resources in the group:

The DB2 UDB instance must be started after the local disk
The DB2 UDB instance must be stopped before the local disk
The HA IP address must be collocated with the instance

2.1 Disk Storage

DB2 UDB can utilize these resources for local data storage:

Raw disk (for example, /dev/sda1)
Logical volume managed by Logical Volume Manager (LVM)
File system (for example, ext3, jfs)

DB2 data can be stored either entirely on one or more raw disks, entirely on logical volumes, entirely on file systems, or on a mixture of all three. Of course, the executables will need to reside on a file system of some sort.

2.1.1 Disk Storage: Raw disk

For high availability, the use of raw disk for data storage is the most straightforward approach. A raw disk is simply a disk partition that does not host a file system. The identical path to raw disk (for example, /dev/sda1) must appear across all nodes in the cluster that are physically connected to the storage. Moreover, this same path is always available across all nodes that are physically attached to the storage, thus eliminating the need for any start or stop methods to make the disk visible to the host. A monitor method is likewise unnecessary.
Note that the true benefit of raw disk is that a possibly time-consuming **fsck** is never required, even in scenarios where the machine is halted abruptly by a power failure, for example.

2.1.2 Disk Storage: Logical Volume Manager

There are currently several logical volume managers (LVMs) available for Linux. We will discuss the LVM currently available as part of the Linux kernel. It is important to remember that LVM is not currently cluster aware. The implication of this is that, as root, it is very easy to cause an unrecoverable data loss in shared disk environments.

In shared disk environments, (the most common of which is shared SCSI for small two-node clusters and shared fiber channel for larger clusters), more than one machine has physical access to a set of disks. Access to these disks must be serialized, however.

It is critical that LVM be shut down on all nodes other than the node from which any LVM administration is done. It is also critical that no more than one node have a file system mounted on a given logical volume (at any given time).

2.1.3 Disk Storage: File System

File systems hosted by raw disk require a resource to mount, unmount, and monitor them. This support is written as a resource of the class *IBM.Application*, and can be added to the resource group containing the DB2 resource, and the IP address of DB2 UDB (collocated and with appropriate dependencies on relationship setup). File systems that are mounted on top of LVM-managed logical volumes are handled identically.

2.2 Disk Protection

From the point of view of DB2 UDB, the disk protection is sufficient if the data on the raw disk, file system, or logical volume is accessible for read and write operations. The DB2 engine itself ensures that only one instance accesses a database at any one time. DB2 UDB data is stored in logical entities called *table spaces*. A table space is composed of one or more table space containers. A table space container is either a file, file system mount point, or a raw disk partition. DB2 UDB does extensive checking when these containers are created and accessed. In particular, it is impossible to have another instance or database concurrently access the same raw disk, file system, or file system mount point.

Disk protection (in the sense of protecting a file system from being mounted concurrently from two different physical hosts) is provided, if used, at the logical volume manager layer, but only for cluster-aware LVMs. If raw disk storage is chosen, care must be taken that the system administrator does not inadvertently concurrently mount the raw disk across multiple hosts. All file system mounting and unmounting tasks (for shared file systems) must be controlled by Tivoli SAM.

2.3 DB2 UDB requirements for the HA IP address

DB2 UDB has no special requirements for the IP address. That is to say, it is not necessary to define a highly available IP address in order for the instance to be considered highly available. However, it is important to remember that the IP address that is protected (if any) is the client's access point to the data, and as such must be well known by all clients. In practice, this implies that this IP address is the one used by the clients in their CATALOG TCPIP NODE commands.

Note these requirements (which are not specific for HA IP addresses but must be remembered by anyone wanting to make a DB2 server accessible to remote TCP/IP clients):

> SVCENAME must be defined in /etc/services at all hosts
> DB2COMM must be set to TCPIP

It is also necessary to define the network interfaces that will be able to host these HA IP address. This is especially important if a node has more than one network interface and only a subset of the network interfaces is allowed to host this IP address.

2.4 DB2 UDB ESE – Database Partitioning Feature in a Tivoli SAM Environment

With DB2 UDB ESE (with the database partitioning feature), there is an additional requirement. The instance home directory must be shared across all nodes participating in the cluster. This will be done via Network File System (NFS) and the HA-NFS package provided in conjunction with Tivoli SAM. In this environment, each individual DB2 partition is controlled by its own distinct resource group (similar to the single partition case described earlier, but extended to many nodes). This approach has many advantages, chiefly that it allows the administrator to specify individually the location of each DB2 partition wherever desired on the cluster.

3. IBM Tivoli SYSTEM AUTOMATION FOR MULTIPLATFORMS

IBM Tivoli System Automation for Multiplatforms (Tivoli SAM) is a product that provides high availability by automating resources such as processes, applications, IP addresses, and others in Linux-based clusters. To automate an IT resource (such as an IP address), the resource must be defined to Tivoli SAM. Furthermore, these resources must all be contained in at least one resource group. If these resources are always required to be hosted on the same machine, they should all be placed in the same resource group. Further information on Tivoli SAM can be found in the *IBM Tivoli System Automation for Multiplatforms, Guide and Reference* manual.

Every application needs to be defined as a resource in order to be managed and automated with Tivoli SAM. Application resources are usually defined in the generic resource class *IBM.Application*. In this resource class, there are several attributes that define a resource, but at least three of them are application-specific:

StartCommand
StopCommand
MonitorCommand

These commands may be scripts or binary executables.

There are some requirements for these commands:

You must ensure that the scripts are well tested, and will produce the desired effects within a reasonable period. This is necessary since these commands are the only interface between Tivoli SAM and the application.

The monitoring script has to be efficient, as it is running at the frequency (in seconds) set in MonitorCommandPeriod, and therefore can consume

a lot of CPU time if written without efficiency in mind.

The StartCommand and StopCommand must have a return code of 0 (zero) for successful completion; otherwise, an error is assumed and the resource is set to 'Failed Offline'.

For the HA IP address there's another resource class *IBM.ServiceIP.* This resource class has special attributes to define an IP address and a net mask. In addition, there can be a defined equivalency of network interfaces, which will define the network interfaces that will be able to host an IP address. The resource class for equivalencies is *IBM.Equivalency*.
After the resources for the equivalency and the IP address are defined correctly, a DependsOn relationship must be created from the IP address to this equivalency.

3.1 The DB2 for Tivoli SAM Agent

The agent consists of the set of scripts necessary for the control of DB2 instances, the DB2 administration server (DAS), and file system mount points, as well as utilities that simplify management of the cluster. Ensure that the agent is included with your version of DB2 UDB; for example, for DB2 UDB v8, the agent can be found in the install path `/opt/IBM/db2/V8.1/ha/salinux.`

4. IBM Tivoli SYSTEM AUTOMATION FOR MULTIPLATFORMS AND DB2 UDB SETUP

This section describes the detailed setup of DB2 UDB and Tivoli SAM. Diagram 1 is an illustration of the sample two-node cluster to which we reference later:

HA FAStT storage shared between nodes
Two-node cluster on xSeries hardware, node1, node2
DB2 instance running DB2 UDB Version 8.x
Sufficient LUNs prepared to host all database data
DB2 instance home directory shared via NFS

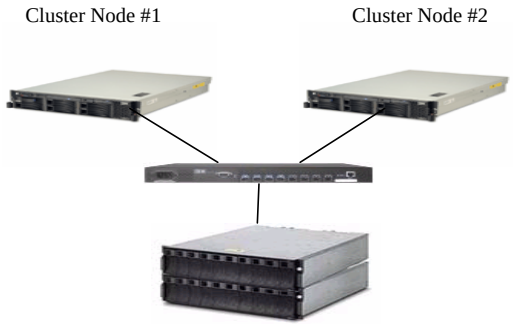


Diagram 1: two-node cluster with shared disk

4.1 IBM Tivoli SAM: Setup Overview

The setup consists of the following steps:

1. Set up the hardware
2. Install DB2 UDB
3. Install IBM Tivoli System Automation for Multiplatforms
4. Set up Tivoli SAM to manage the NFS
5. Create a DB2 instance
6. Set up Tivoli SAM to manage the DB2 instance

4.2 Hardware resources

If redundancy is required, it is important to ensure that you have adequate hardware resources to provide redundancy. Consult the *IBM Tivoli System Automation for Multiplatforms, Guide and Reference* for detailed and current platform requirements.

4.3 Install DB2 UDB

DB2 UDB must be installed on all nodes that will potentially host a DB2 instance or the DAS. Generally, follow the *Quick Beginnings for DB2 Servers* manual. When installation is complete, the DB2 UDB software is installed (by default) in /opt/IBM/db2/V8.1 (for Version 8). Do not create instances at this step; that task will be deferred to a subsequent step.

4.4 Install IBM Tivoli System Automation for Multiplatforms

Tivoli SAM must be installed on both machines locally. To install Tivoli SAM, follow the installation procedure described in *IBM System Automation for Multiplatforms, Guide and Reference*.

4.5 Set up IBM Tivoli System Automation for Multiplatforms to manage the DB2 instance and NFS Server

First, all nodes that will comprise the cluster must be prepared. This includes some security setup, without which the following commands will not work. This command must be executed on every node. In this example, the nodes are named db2tsa01 and db2tsa02:

```
# /usr/sbin/rsct/bin/preprpnode db2tsa01 db2tsa02
```

Create the cluster domain:

```
# /usr/sbin/rsct/bin/mkrpdomain \
  db2tsa12 db2tsa01 db2tsa02
```

Therefore, the domain created is named db2tsa12, and consists of the nodes db2tsa01 and db2tsa02. All Tivoli SAM commands and resources are relative to the active domain; all Tivoli SAM commands can now be issued at any node of the domain. To start a domain (or to cause it to become active), issue the following commands:

Start the domain:

```
# /usr/sbin/rsct/bin/startrpdomain db2tsa12
```

Ensure the domain is online:

```
# /usr/sbin/rsct/bin/lsrpdomain
```

You should see output similar to this:

```
Name OpState RSCTActiveVersion MixedVersions TSPort
db2tsa12 Online  2.3.3.1       No            12347
```

Ensure all nodes in the domain are online:

```
# lsrpnode
```

You should see output similar to this:

```
Name            OpState          RSCTVersion
db2tsa01        Online           2.3.3.1
db2tsa02        Online           2.3.3.1
```

For any two-node cluster, a tiebreaker disk is required. This tiebreaker is used exclusively for cluster quorum duties (in other words, a disk that cannot be used to support any file system or data storage). Note that this can be a very small Logical Disk of a disk array, but it must be devoted to the tiebreaker function. To create the tiebreaker disk, some background knowledge of SCSI device numbering is required.

SCSI devices can be identified with four integers: one each represents the host, the channel, the SCSI ID, and the LUN. Note that these numbers may not necessarily be consistent across nodes, even if the target device to which they refer is the same physical disk.

For example, if a SCSI device is connected to two nodes (for example, db2tsa01 and db2tsa02) and has the following SCSI identifiers at the respective nodes:

db2tsa01: ID=2 LUN=0 HOST=1 CHAN=0
db2tsa02: ID=2 LUN=0 HOST=1 CHAN=0

Note that these numbers can be obtained with the dmesg command, or from the contents of /proc/scsi/scsi. For example, on either node, run the following command:

```
# cat /proc/scsi/scsi  | grep Channel

Host: scsi0 Channel: 00 Id: 00 Lun: 00
Host: scsi0 Channel: 00 Id: 08 Lun: 00
Host: scsi1 Channel: 00 Id: 00 Lun: 00
Host: scsi1 Channel: 00 Id: 01 Lun: 00
Host: scsi1 Channel: 00 Id: 02 Lun: 00
Host: scsi1 Channel: 00 Id: 03 Lun: 00
Host: scsi1 Channel: 00 Id: 04 Lun: 00
Host: scsi1 Channel: 00 Id: 05 Lun: 00
Host: scsi1 Channel: 00 Id: 08 Lun: 00
Host: scsi1 Channel: 00 Id: 09 Lun: 00
Host: scsi1 Channel: 00 Id: 10 Lun: 00
Host: scsi1 Channel: 00 Id: 11 Lun: 00
Host: scsi1 Channel: 00 Id: 13 Lun: 00
Host: scsi1 Channel: 00 Id: 14 Lun: 00
Host: scsi1 Channel: 00 Id: 15 Lun: 00
```

You may run this command at each node in the cluster to confirm that the LUNs are visible across all of the cluster nodes. We will choose to make the following LUN our TieBreaker:

Host: scsi1 Channel: 00 Id: 02 Lun: 00

Then, create the TieBreaker object as follows:

```
# mkrsrc IBM.TieBreaker Name="tb" Type="SCSI" \
     DeviceInfo="HOST=1 CHAN=0 ID=2 LUN=0" \
     HeartbeatPeriod=5

# chrsrc -c IBM.PeerNode OpQuorumTieBreaker="tb"
```

We will now create the NFS server resource to enable hosting of the instance home directory.

4.6 Create the Highly Available NFS Server

The reader must understand the white paper describing the steps in configuring highly available NFS from the URL:

ftp://ftp.software.ibm.com/software/tivoli/products/sys-auto-linux/ITSAMP-NFS-Server-v1.1.pdf

The white paper is entitled "Highly available NFS server with Tivoli System Automation for Linux".

Furthermore, the RPMs must be installed, and must include the policy and script samples for making the NFS server highly available. These are also available at the same site, via the following URL:

ftp://ftp.software.ibm.com/software/tivoli/products/sys-auto-linux/sam.policies-1.2.1.0-0.i386.rpm

4.6.1 Configure the HA NFS Server Package

This is a brief overview of all necessary steps to enable HA NFS in our cluster. For further information, consulting the white paper cited above is strongly recommended.

First, create a partition of about 10 MB on a shared disk (for example, /dev/sdb1). Then, format the partition with a file system (for example, ext3 or jfs). Note that you should avoid using a non-logged file system (such as ext2) for this or any other mount point that you may require in the cluster. Create a mount point /varlibnfs at each node of the cluster. Next, place an entry in /etc/fstab for this mount point (do this at each node in the cluster), for example:

```
/dev/sdb1 /varlibnfs  ext3  noauto      0 0
```

Perform the following steps at one node only:

```
# mount /varlibnfs
# cp /var/lib/nfs/* /varlibnfs
# umount /varlibnfs

Then, at each node:

# cd /var/lib
# mv nfs nfs.org
# ln –s /varlibnfs nfs
```

Ensure that the sam.policies RPM is installed at each node with:

```
# rpm –q sam.policies
```

If the sam.policies rpm is not installed, get it from

ftp://ftp.software.ibm.com/software/tivoli/products/sys-auto-linux/sam.policies-1.2.1.0-0.i386.rpm

and install it.

4.6.2 Edit sa-nfsserver.conf

You must change the following lines in the sa-nfsserver.conf file (located in /usr/sbin/rsct/sapolicies/nfsserver). The remainder of the file (that is, those lines that we do not reference herein), can remain unchanged.

The names of the set of nodes that can host the NFS server:

```
nodes="db2tsa01 db2tsa02"
```

The IP address and net mask, which will be used by clients to mount the file system:

```
ip_1="9.26.166.36,255.255.254.0"
```

The set of Ethernet adapters that are equivalent for purposes of hosting the NFS server IP address:

```
nieq_1="eth0:db2tsa01,eth0:db2tsa02"
```

The name of the directory that will locally mount the instance home directory:

```
data_work="/shared_home"
```

Ensure that this file is copied to all nodes in the cluster; for example, if the file was originally edited on db2tsa01, you can use the following command now (from db2tsa02) to replicate it to db2tsa02:

```
# rcp \
db2tsa01:/usr/sbin/rsct/sapolicies/nfsserver/* .
```

Then, ensure that there is a line in the /etc/fstab similar to this one:

```
/dev/sdb2 /shared_home   ext3  noauto           0 0
```

Also, ensure that the following line exists in the /etc/exports file on all nodes that may host the NFS server:

```
/shared_home    *(rw,no_root_squash,sync)
```

Now you are ready to create the NFS resources with the following command:

```
#/usr/sbin/rsct/sapolicies/nfsserver/cfgnfsserver -p
```

Bring the NFS resource group up as follows:

```
# chrg -o Online SA-nfsserver-rg
```

4.6.3 Mount Client NFS mount points

Either you can choose to use automounter or you can manually mount the client NFS mount points. If you are manually mounting the client NFS mount points, ensure a line similar to the following one exists in the /etc/fstab for each node of the cluster. Note that this is one long line that may be wrapped in this text but should be entered as one line in the /etc/fstab file:

```
9.26.166.36:/shared_home  /home nfs
rw,bg,vers=2,tcp,timeo=600,retrans=2,hard,nointr,noac,
suid,nolock 0 0
```

When no automounter is used, it is the responsibility of the system administrator to mount the client NFS mount point before attempting to use the machine as a host for a DB2 partition.

4.7 Create a DB2 instance

We will work with an instance named db2tsa in this example.

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

On each physical node participating in the cluster, three separate groups and user accounts need to be created for the:

DB2 instance owner
User ID that will execute fenced user-defined functions (UDFs)
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 1004 \
   -d /home/db2tsa -m db2tsa

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

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

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

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

```
# rsh db2tsa01 "cat /etc/passwd |grep db2tsa"
# rsh db2tsa02 "cat /etc/passwd |grep db2tsa"
```

You should see output at each node similar to the following line:

```
db2tsa:x:1004:999::/home/db2tsa:/bin/ksh
```

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

```
# rsh db2tsa01 “cat /etc/services | grep db2tsa”
# rsh db2tsa02 “cat /etc/services | grep db2tsa”
```

You should see output similar to the following lines:

```
DB2_db2tsa         60000/tcp
DB2_db2tsa_1       60001/tcp
DB2_db2tsa_2       60002/tcp
DB2_db2tsa_END     60003/tcp
xdb2tsa            60004/tcp
```

Ensure that all nodes have identical entries. One important point to remember is to ensure that sufficient entries are allocated so that each partition in the instance can be brought up on each host. That means, for example, that the port range given above (60000 to 60003 inclusive) is sufficient for a four-partition DB2 UDB ESE - Database Partitioning Feature instance. The range will have to be made larger if more partitions are required.

Ensure that the IP address that you want to associate with this instance is not currently in use. We use the IP address 9.26.166.123 in the examples that follow. Also, ensure that you have unprompted rsh access (as root) to the other nodes in this cluster. Unprompted rsh access to the other nodes in the domain is only required for the duration of the registration process and can be removed once the registration script has completed processing.

Notice that for this instance, the instance home directory will be mounted under /home. Issue the following commands at some node in the cluster:

```
# rsh db2tsa01 "cat /etc/fstab | grep /home”
# rsh db2tsa02 “cat /etc/fstab | grep /home”
```

For DB2 UDB ESE - Database Partitioning Feature instances, you should see output similar to the following lines at each node:

```
9.26.166.36:/shared_home  /home nfs
rw,bg,vers=2,tcp,timeo=600,retrans=2,hard,nointr,noac,
suid,nolock 0 0
```

After checking the fstab file, ensure that the instance home directory is available by executing the following commands:

```
# rsh db2tsa01 mount | grep /home/
# rsh db2tsa02 mount | grep /home/
```

The output should be similar to the following lines:

```
9.26.166.36:/shared_home on /home type nfs
(rw,bg,vers=2,tcp,timeo=600,retrans=2,hard,nointr,noac
,nolock,addr=9.26.166.36)
```

In the next step, we will create a DB2 instance named ‘db2tsa’ and configure the database partitions. To do this, execute the following commands:

```
# /opt/IBM/db2/V8.1/instance/db2icrt \
 -w 64 -u db2tsa db2tsa
```

The output of the command should be similar to the line below:

DBI1070I Program db2icrt completed successfully.

To verify the instance, log on as the instance user and start the db2 service:

```
# su - db2tsa
% db2start
```

The output should be similar to the following lines:

```
0 0 SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

After verifying the instance, the next step is to configure the database partitions. To do this, we will modify the content of the db2nodes.cfg file. However, before configuring the partitions, stop the DB2 service. Issue the following commands to configure the database partitions:

```
% db2stop

% vi ~/sqllib/db2nodes.cfg
```

Add the following line:

1 db2tsa02 0

The final content of the db2nodes.cfg file should be similar to the lines below:

```
0 db2tsa01 0
1 db2tsa02 0
```

To verify the partitions, start the DB2 service:

```
% db2start
```

The output should be similar to the following lines:

```
0   0   SQL1063N DB2START processing was successful.
1   0   SQL1063N DB2START processing was successful.
```

Now, for our two-partition DB2 instance, we want exactly one local mount point to store database data (per partition). Create entries in the /etc/fstab of the following format:

```
/dev/sdb3 /db/db2tsa/NODE0000  ext3       noauto 0 0
/dev/sdb4 /db/db2tsa/NODE0001  ext3       noauto 0 0
```

Ensure that this is consistent across each node in the cluster. Here, the entry for NODE0000 (the first entry of the two) will correspond to data storage for partition 0 of the instance, and the entry for NODE0001 will correspond to data storage for partition 1 of the instance.

If these mount points do not already exist, create them with the following commands on each node:

```
# mkdir –p /db/db2tsa/NODE0000
# mkdir –p /db/db2tsa/NODE0001
```

Ensure that the instance owner ID has read and write permission to this mount point.

Note that if these mount points do not exist at the time of instance registration, then no local mount points will be created by the registration process. Local HA mount points for data storage can be created later via standard Tivoli SAM administration commands.

You have the option at this point to create an equivalency group between the public adapters set to host the highly available IP address. In our cluster, we will consider the adapters eth3 on db2tsa01 and eth3 on db2tsa02 as equivalent for the purposes of hosting the HA IP address. To express this in Tivoli SAM terms, issue the following command:

```
# mkequ virpubnic \
```

```
IBM.NetworkInterface:eth3:db2tsa01,eth3:db2tsa02
```

Note that the name of the equivalency must be virpubnic; the other parameters are optional, depending on the node names and the NIC names for your particular cluster.

If you choose not to create this equivalency group, a NIC failure alone will not cause the resource group to fail to another node, which may lead to an outage window.

After creating the instance and configuring the partitions, we need to execute the following command to protect each of the DB2 partitions and the well-known IP address 9.26.166.123.

```
# cd /opt/IBM/db2/V8.1/ha/salinux
# ./regdb2salin -a db2tsa -i 9.26.166.123
```

This may take a few moments, but the result will be a highly available instance. To verify this, issue the following command:

```
# ./getstatus
```

```
-- Resource Groups and Resources --

            Group Name           Resources
            ----------           ---------
          SA-nfsserver-rg   SA-nfsserver-server
          SA-nfsserver-rg   SA-nfsserver-ip-1
          SA-nfsserver-rg   SA-nfsserver-data-varlibnfs
          SA-nfsserver-rg   SA-nfsserver-data-work
                  -                 -
          db2_db2tsa_0-rg  db2_db2tsa_0-rs
          db2_db2tsa_0-rg  db2_db2tsa_0-rs_mount
          db2_db2tsa_0-rg  db2_db2tsa_0-rs_ip
                  -                 -
          db2_db2tsa_1-rg   db2_db2tsa_1-rs
          db2_db2tsa_1-rg   db2_db2tsa_1-rs_mount
                  -                 -


-- Resources --

             Resource Name          Node Name         State
             -------------          ---------         -----
         SA-nfsserver-server         db2tsa01        Online
         SA-nfsserver-server         db2tsa02        Offline
                  -                 -          -
         SA-nfsserver-ip-1           db2tsa01        Online
```

```
SA-nfsserver-ip-1                 db2tsa02        Offline
-                                 -               -
SA-nfsserver-data-varlibnfs       db2tsa01        Online
SA-nfsserver-data-varlibnfs       db2tsa02        Offline
-                                 -               -
SA-nfsserver-data-work            db2tsa01        Online
SA-nfsserver-data-work            db2tsa02        Offline
-                                 -               -
db2_db2tsa_0-rs                   db2tsa01        Offline
db2_db2tsa_0-rs                   db2tsa02        Online
-                                 -               -
db2_db2tsa_0-rs_mount             db2tsa01        Offline
db2_db2tsa_0-rs_mount             db2tsa02        Online
-                                 -               -
db2_db2tsa_0-rs_ip                db2tsa01        Offline
db2_db2tsa_0-rs_ip                db2tsa02        Online
-                                 -               -
db2_db2tsa_1-rs                   db2tsa01        Online
db2_db2tsa_1-rs                   db2tsa02        Offline
-                                 -               -
db2_db2tsa_1-rs_mount             db2tsa01        Online
db2_db2tsa_1-rs_mount             db2tsa02        Offline
```

While this is a large amount of output, it can be easily understood by looking at it one resource group at a time. The first section lists the resource groups along with the resources that are contained within it; in this case, we see there are three resource groups, named SA-nfsserver-rg, db2_db2tsa_0-rg, db2_db2tsa_1-rg.

SA-nfsserver-rg contains all resources necessary to automate failover of the NFS server, db2_db2tsa_0-rg contains all resources necessary to automate and control partition 0 of instance db2tsa, and db2_db2tsa_1-rg contains all resources necessary to automate and control partition 1 of instance db2tsa. Note that contained within each of the DB2 resource groups is both the partition control as well as a local mount point that will be needed to store database data for that instance. In addition, we see there is an additional resource in the db2_db2tsa_0-rg resource group, and this is the resource that controls the location of the well-known IP address (well known by the DB2 client applications). It will always be collocated with the location of partition 0 for the instance db2tsa.

4.9 Create a DB2 database

We should now create a database under this instance. Log on to the server as the instance owner and issue the following command:

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

Two things to note: first, the partition at which you create the database will also be the catalog node for the duration of the existence of this database. Second, the database is created on the local disks that are associated with the db2tsa database instance; these local disks are specified in the /etc/fstab. Notice particularly that the path clause on the CREATE DATABASE references the path '/db'; this is exactly the prefix to the string $DB2INSTANCE_NAME/NODE000x given in the /etc/fstab file. This prefix can be replaced by any prefix, as long as they are used consistently in both the /etc/fstab and the CREATE DATABASE command.

4.10 Catalog a DB2 Instance

```
% db2 catalog tcpip node db2tsaha \
remote 9.26.166.123 server 60004
```

Note that the IP address specified is identical to the IP address supplied as the argument to the registration script and that the port for the server is defined on both instance nodes and is the value of SVCENAME at the server.

5. TESTING HA DB2 UDB ESE - DATABASE PARTITIONING FEATURE INSTANCES WITH Tivoli SAM

Implementing an HA cluster is a four-stage process. The stages are as follows:

1. Carefully plan the cluster.
2. Implement the cluster.
3. Test the cluster.
4. If the test results are not satisfactory, return to Step 1.

Planning the proper setup of an HA cluster is not within the scope of this document. See *IBM System Automation for Multiplatforms, Guide and Reference* for advice on the setup of the cluster. We will provide some testing guidelines that are relevant for the use of DB2 UDB in such an environment.

Ideally, every critical point of failure (in other words, a failure that has the potential to affect production operation) should be tested. It may not be practical to shut down a building's power supply in order to test backup power supplies, but a single source of electricity is a single point of failure, and until the system has been physically tested, it will never be entirely certain that the cluster will behave in practice as it does in theory.

We will concentrate on failures internal to the cluster for the rest of this discussion. We will also assume a client load is being run on the system and monitored to ensure successful failover and failover timings.

Controlled Failover Testing

Controlled failover refers to the use of operator-driven commands to induce a change in the distribution of resources across the cluster. For this, we will use the Tivoli SAM command rgreq (Tivoli SAM 1.2 required) and assume the initial distribution given in the output to the getstatus command described above. That is, we consider the original two-node domain consisting of nodes db2tsa01 and db2tsa02. The NFS resource group initially resides on node db2tsa01, the resource group for partition 0 resides initially on node db2tsa01, and the resource group for partition 1 resides initially on node db2tsa02.

Failover partition 1 from db2tsa01 to db2tsa02

```
# rgreq -o Move -n db2tsa01 db2_db2tsa_1-rg
```

Issue the getstatus command and observe that the new location of the db2_db2tsa_1-rg resource group is no longer node db2tsa01 but rather node db2tsa02. Further, observe the behavior of the client program and verify that it is still active.

Failover partition 1 from db2tsa02 to db2tsa01

```
# rgreq -o Move -n db2tsa02 db2_db2tsa_1-rg
```

Issue the getstatus command and verify that the distribution of the partitions for instance db2tsa corresponds to that originally given.

Failover partition 0 from db2tsa01 to db2tsa02

```
# rgreq -o Move -n db2tsa01 db2_db2tsa_0-rg
```

Issue the getstatus command and observe that both partitions are hosted on the node db2tsa02.

Failover partition 0 from db2tsa02 to db2tsa01

```
# rgreq -o Move -n db2tsa02 db2_db2tsa_0-rg
```

Observe that the original distribution of partitions is restored.

Now, fail the NFS server from its original host with the command:

```
# rgreq -o Move -n db2tsa01 SA-nfsserver-rg
```

Observe that the location of the NFS server has changed to the node db2tsa02. Fail back the NFS server to its original host with the command:

```
# rgreq -o Move -n db2tsa02 SA-nfsserver-rg
```

Testing Instance Failure

Log on to the node db2tsa01 as the instance owner db2tsa and issue the following command (which will simulate an instance failure):

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

Issue a getstatus 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.

Log on to the other node db2tsa02 and issue the command given immediately above. You should see similar results at this node.

Testing Machine Failure

Log on to the node db2tsa02 as root and issue the command:

```
# reboot -nf
```

This should cause all resources hosted on node db2tsa02 to fail to db2tsa01. Observe that this, in fact, does occur and that the client notices a brief interruption of service.

Wait until the node db2tsa02 comes back online. At that point, ensure that the NFS client mount is mounted, and then issue the following command to bring the partition 1 of instance db2tsa online at the node:

```
# rgreq -o Move -n db2tsa01 db2_db2tsa_1-rg
```

Now, fail node db2tsa01 with the command:

```
# reboot -nf
```

This should cause all resources hosted on node db2tsa01 to fail to db2tsa02. Observe that this, in fact, does occur and that the client notices a brief interruption of service.

Wait until the node db2tsa01 comes back online. At that point, ensure that the NFS client mount is available and mounted, and then issue the following command to bring the partition 1 of instance db2tsa online at the node:

```
# rgreq -o Move -n db2tsa02 db2_db2tsa_1-rg
```

6. CAPACITY ON DEMAND

The combination of DB2 UDB ESE - Database Partitioning Feature and Tivoli SAM can provide a powerful solution to adding and removing computing resources on demand. During periods of high database use, additional computing resources can be added to the cluster; and these resources will host DB2 UDB ESE - Database Partitioning Feature partitions that can provide increased computing throughput. If at some point these additional resources are no longer required, they can be removed from the cluster.

Here is an example. It begins with a three-partition DB2 UDB ESE - Database Partitioning Feature database instance spread over two computing nodes. To increase throughput, one additional computing node is added to the Tivoli SAM cluster. DB2 UDB ESE – Database Partitioning Feature partitions are then started on these nodes to take advantage of their additional CPU, RAM, and disk resources of the newly added node.

We will begin with an initial state as follows. The Tivoli SAM domain we create is named halin_domain:

```
# lsrpdomain
Name OpState RSCTActiveVersion MixedVersions TSPort
halin_domain Online  2.3.3.1   No            12347
```

This domain is composed of the following nodes:

```
# lsrpnode
Name   OpState RSCTVersion
halin2 Online  2.3.3.1
halin3 Online  2.3.3.1
```

The steps to configure an NFS server given in 4.6 have been followed to create the following initial distribution of resources and resource groups. The state of the NFS and NFS-related resources are as follows:

```
# getstatus
```

```
-- Resource Groups and Resources --

            Group Name           Resources
            ----------           ---------
          SA-nfsserver-rg      SA-nfsserver-server
          SA-nfsserver-rg       SA-nfsserver-ip-1
          SA-nfsserver-rg SA-nfsserver-data-varlibnfs
          SA-nfsserver-rg    SA-nfsserver-data-work
                   -                    -


-- Resources --

            Resource Name            Node Name         State
            -------------            ---------         -----
          SA-nfsserver-server           halin3        Online
          SA-nfsserver-server           halin2        Offline
                   -                    -             -
           SA-nfsserver-ip-1            halin3        Online
           SA-nfsserver-ip-1            halin2        Offline
                   -                    -             -
    SA-nfsserver-data-varlibnfs         halin3        Online
    SA-nfsserver-data-varlibnfs         halin2        Offline
                   -                    -             -
      SA-nfsserver-data-work            halin3        Online
      SA-nfsserver-data-work            halin2        Offline
                   -                    -             -
```

The DB2 UDB ESE – Database Partitioning Feature instance we will use is *stevera*, and its initial distribution of partitions across our two nodes is given by its db2nodes.cfg file (from $INSTHOME/sqllib/db2nodes.cfg):

```
$ cat ~/sqllib/db2nodes.cfg
0 halin2 0 halin2
1 halin2 1 halin2
2 halin3 0 halin3
```

Therefore, we see that the instance is a three-partition instance, with two partitions spread across node halin2, and the remaining partition spread across node halin3. It is important to wisely choose the number of logical partitions; choose a sufficient number of partitions so that there is at least one logical partition per the maximum number of physical nodes that you anticipate adding to the Tivoli SAM domain.

Now, create the highly available DB2 instance via the regdb2salin script, for example, from halin2:

```
# ./regdb2salin -a stevera
```

The distribution of resources and resource groups across the cluster is now as follows:

```
# getstatus

-- Resources --

           Resource Name             Node Name         State
           -------------             ---------         -----
        SA-nfsserver-server            halin2         Offline
        SA-nfsserver-server            halin3          Online
                  -                 -              -
         SA-nfsserver-ip-1             halin2         Offline
         SA-nfsserver-ip-1             halin3          Online
                  -                 -              -
    SA-nfsserver-data-varlibnfs        halin2         Offline
    SA-nfsserver-data-varlibnfs        halin3          Online
                  -                 -              -
      SA-nfsserver-data-work           halin2         Offline
      SA-nfsserver-data-work           halin3          Online
                  -                 -              -
          db2_stevera_0-rs             halin2          Online
          db2_stevera_0-rs             halin3         Offline
                  -                 -              -
          db2_stevera_1-rs             halin2          Online
          db2_stevera_1-rs             halin3         Offline
                  -                 -              -
          db2_stevera_2-rs             halin2         Offline
          db2_stevera_2-rs             halin3          Online
                  -                 -              -
```

Note that for clarity, and to reduce the amount of output from the getstatus command, no local disk resources or HA IP addresses are being shown in this example.

Now, let us assume that node halin4 is being added to the Tivoli SAM domain. Ensure that whatever local disk resource are required for each partition are accessible on the new node, and that the /etc/fstab on the new nodes is identical to the existing nodes in order to facilitate the mounting of the local disk resource. You must also mount the instance home directory via NFS or automounter (same as for the other nodes in the cluster). Naturally, you must install the Tivoli SAM RPMs on any new nodes you want to participate in the domain.

Then, add the new node to the domain with the following command:

```
# addrpnode halin4
```

Start the new node in the domain with the following command:

```
# startrpnode halin4
```

Issue the lsrpnode command and wait until all the nodes are online to the domain, for example, you should see output similar to the following lines:

```
# lsrpnode
Name   OpState RSCTVersion
halin2 Online  2.3.3.1
halin3 Online  2.3.3.1
halin4 Online  2.3.3.1
```

Extend the set of nodes the DB2-based resources can participate in with the following command:

```
# chrsrc -s 'Name like "db2%" && ResourceType=1 ' \
IBM.Application \
NodeNameList="{'halin2','halin3','halin4'}"
```

Note that if an HA IP address was added to any of the DB2 resource groups; issue the following Tivoli SAM command to allow the IP resource to take advantage of the current set of nodes:

```
# chrsrc -s 'Name like "db2%" && ResourceType=1 ' \
IBM.ServiceIP \
NodeNameList="{'halin2','halin3','halin4'}"
```

If an equivalency was defined for the network interfaces on nodes halin2 and halin3, you must extend that to encompass the network interface on node halin4. For example, if the network interface is named virpubnic and the network interface on node halin4 is named eth3:

```
# chequ -u a virpubnic \
IBM.NetworkInterface:eth3:halin4
```

Fail over the desired resource group to the new node via:

```
# rgreq -o Move -n halin2 db2_stevera_2-rg
```

Issue the getstatus command, you should see the DB2 resource groups migrate to the new node, with the net result of one DB2 resource group per partition.

```
-- Resources --
```

```
      Resource Name           Node Name         State
      -------------           ---------         -----
SA-nfsserver-server             halin2         Offline
SA-nfsserver-server             halin3          Online
                -               -               -
 SA-nfsserver-ip-1              halin2         Offline
 SA-nfsserver-ip-1              halin3          Online
                -               -               -
SA-nfsserver-data-varlibnfs     halin2         Offline
SA-nfsserver-data-varlibnfs     halin3          Online
                -               -               -
  SA-nfsserver-data-work        halin2         Offline
  SA-nfsserver-data-work        halin3          Online
                -               -               -
     db2_stevera_0-rs           halin2         Offline
     db2_stevera_0-rs           halin3          Online
     db2_stevera_0-rs           halin4         Offline
                -               -               -
     db2_stevera_1-rs           halin2          Online
     db2_stevera_1-rs           halin3         Offline
     db2_stevera_1-rs           halin4         Offline
                -               -               -
     db2_stevera_2-rs           halin2         Offline
     db2_stevera_2-rs           halin3         Offline
     db2_stevera_2-rs           halin4          Online

                -               -               -
```

Now, let us assume that the node halin4 is not required to participate in this cluster.

Issue the following commands to restrict the resource groups to only their original nodes (that is, halin2 and halin3):

```
# chrsrc -s 'Name like "db2%" && ResourceType=1 ' \
 IBM.Application NodeNameList="{'halin2','halin3'}"
```

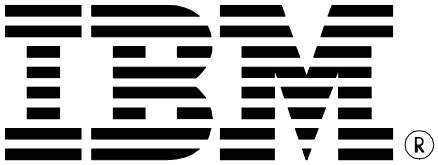
Note that if an HA IP address was added to any of the DB2 resource groups, issue the following Tivoli SAM command to allow the IP resource to take advantage of the current set of nodes:

```
# chrsrc -s 'Name like "db2%" && ResourceType=1 ' \
 IBM.ServiceIP NodeNameList="{'halin2','halin3'}"
```

Then, remove the node halin4 from the Tivoli SAM domain with the following command:

```
# stoprpnode halin4
# rmrpnode halin4
```

The original layout of the partitions will have been restored.



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