NETAPP TECHNICAL REPORT

DB2 9 for Linux: High Availability Using Tivoli System Automation and NetApp FAS or IBM N series Storage System

Jawahar Lal and Roger Sanders, NetApp, Inc.

Andy Beaton, IBM

Updated February, 2010 | TR-3492

EXECUTIVE SUMMARY

This paper describes how IBM® Tivoli® System Automation (TSA) and a NetApp® storage system (clustered) can mean high availability for your DB2® 9 database environment. Installation and configuration steps for a TSA cluster are discussed in detail. Also, common single-point failure scenarios are tested.

TABLE OF CONTENTS

1 OVERVIEW

1.1 PURPOSE AND SCOPE

This paper explains how to achieve high availability for your DB2 database by using Tivoli System Automation (TSA) and a NetApp FAS or IBM N series storage system. Basic configuration and setup steps for a two-node TSA cluster in a DB2, Linux®, and NetApp FAS or IBM N series storage system environment are described in detail.

Organizations need high-availability (HA) solutions to make their enterprise data highly available. The IT infrastructure of an organization has various components such as network, server, storage systems, and application processes that can become bottlenecks for data availability. High availability can be achieved by eliminating the single point of failure (SPOF) in the infrastructure. The failover cluster is one of the most commonly used architectures to eliminate the SPOF and improve the availability of a system or a component of a system. Failover cluster architecture uses redundant hardware and/or software components. For example, a server failover cluster has a group of nodes (servers) in which one node actively provides the service and, if that node fails, another node is made active in its place. Without this behavior, the service would be unavailable until the failed node was restarted. Having active nodes waiting in case of failure improves the uptime of each service and the system as a whole.

TSA combined with a NetApp FAS or IBM N series storage system (cluster) delivers a robust enterprise HA solution. TSA provides resource availability on the nodes, and the storage system cluster eliminates the SPOF at the storage level.

1.2 TIVOLI SYSTEM AUTOMATION

TSA is a software-based high-availability solution for mission-critical applications that provides a self-healing infrastructure. It allows customers to automate control of IT resources such as application processes, IP addresses, and file systems or disks. The key features of TSA are:

- Automation of resource control and monitoring
- Automation based on rules and policies
- Logical grouping of resources and resource relationships
- Self-healing architecture

TSA controls resources based on the policies and relationships among resources that are defined by the users. Application resources are usually defined by using a generic resource class named *IBM.Application*. This resource class has several attributes. The attributes specifically used for TSA are:

- StartCommand
- StopCommand
- MonitorCommand

These attributes determine start, stop, and monitor rules for an application resource that is controlled by using TSA. In case of failure, TSA stops the resource on the failed node and restarts it on another node in the cluster. In a TSA environment for DB2, a DB2 instance and storage disks or file systems that are used for the databases are defined as application resources. In order to achieve high availability for an IP address, another resource class named *IBM.ServiceIP* is used. For more information on the TSA, refer to the TSA documentation on the IBM Website (*http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.1.html*).

1.3 STORAGE CLUSTER FAILOVER

The cluster configuration for NetApp FAS or IBM N series storage systems comes with two controllers. During normal operation, each controller serves its own workload while monitoring the health of its "partner" controller. If one of the controllers fails or is taken offline, the second controller takes over the workload of the failed controller. The workload takeover process is automatic (no manual intervention is required at any point) and is transparent to end users and applications. For more information on the storage cluster failover configuration, refer to the *Cluster Installation and Administration Guide* on NOW™ (NetApp on the Web).

1.4 HIGH AVAILABILITY FOR A DB2 ENVIRONMENT

A typical DB2 environment consists of the following components:

- One or more database servers
- A DB2 instance (single or multinode)
- One or more databases within the DB2 instance
- A network IP address that serves the client applications
- One or more file systems for database storage

To achieve high availability of a DB2 environment, you must eliminate all single points of failure by using cluster failover architecture. Normally, cluster failover tasks such as takeover, giveback, and resource monitoring are performed manually, using complicated scripts. With TSA, you can automate all the cluster failover operations in a few easy steps for high availability of your DB2 environment. In a TSA configuration, DB2 environment components are defined as TSA resources. Also, you need to define resource dependencies and policies to control them. Resource dependencies are defined as the relationships between resources. TSA resources can be logically grouped together in a resource group. Once resource groups are defined and policies are created, TSA automatically controls the resources without any manual intervention. Figure 1 illustrates the TSA resources, resource relationships, and resource groups that we created for our test environment.

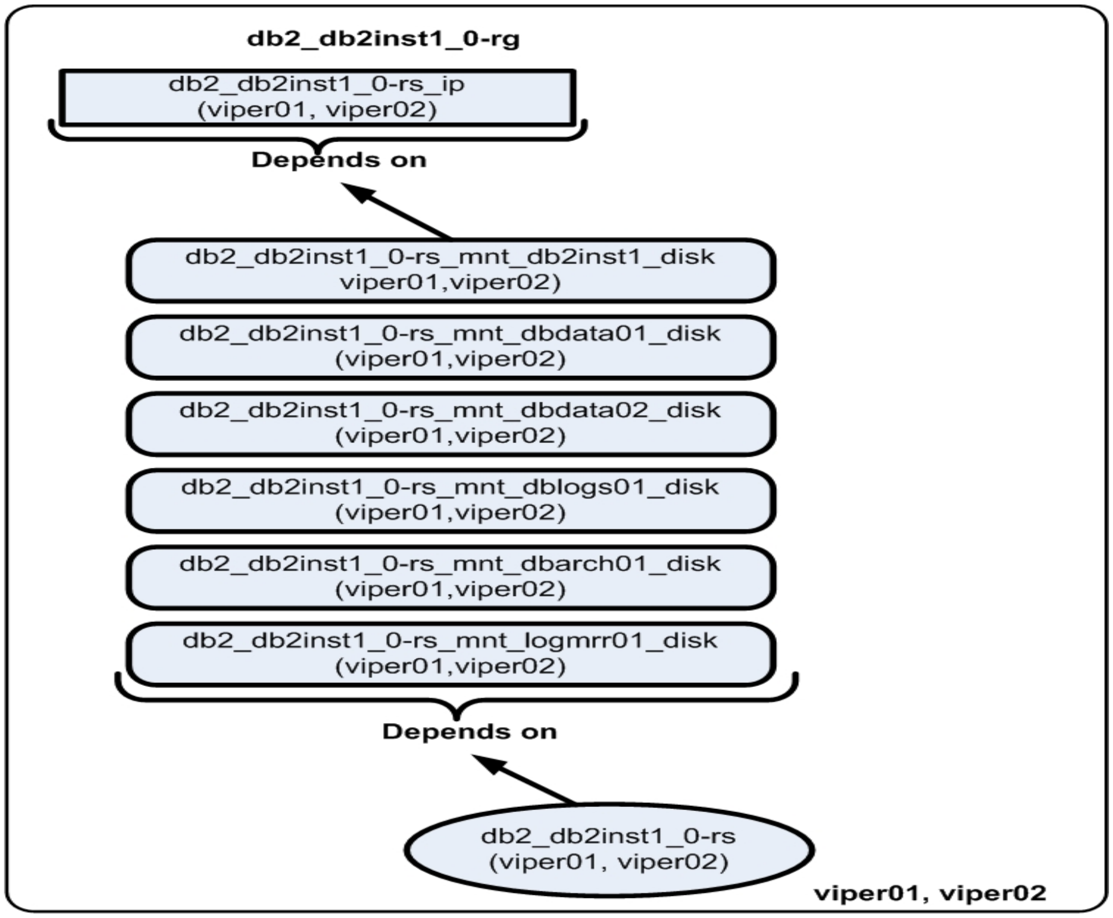
db2_db2inst1_0-rg
db2_db2inst1_0-rs_ip (viper01, viper02)
Depends on
db2_db2inst1_0-rs_mnt_db2inst1_disk viper01,viper02)
db2_db2inst1_0-rs_mnt_dbdata01_disk (viper01,viper02)
db2_db2inst1_0-rs_mnt_dbdata02_disk (viper01,viper02)
db2_db2inst1_0-rs_mnt_dblogs01_disk (viper01,viper02)
db2_db2inst1_0-rs_mnt_dbarch01_disk (viper01,viper02)
db2_db2inst1_0-rs_mnt_logmrr01_disk (viper01,viper02)
Depends on
db2_db2inst1_0-rs (viper01, viper02)
viper01, viper02

Figure 1) TSA resource groups, resource relationships, and resources in the test environment.

TSA starts resources and resource groups based on the user-defined relationships. For example, in our test environment, when resource group db2_db2inst1_0-rg is brought online, the IP address resource (db2_db2inst1_0-rs_ip) is started online first, followed by all the disk resources. The DB2 instance resource (db2_db2inst1_0-rs) starts last. If any resource that precedes db2_db2inst1_0-rs is not available, then the DB2 instance is not started.

1.5 ARCHITECTURE

To develop this paper, we used a single partition DB2 instance and a TSA cluster that has two nodes. To further enhance the high availability, we used a *clustered* NetApp FAS3050 storage system for database storage. Figure 2 illustrates the architecture used in our test environment.

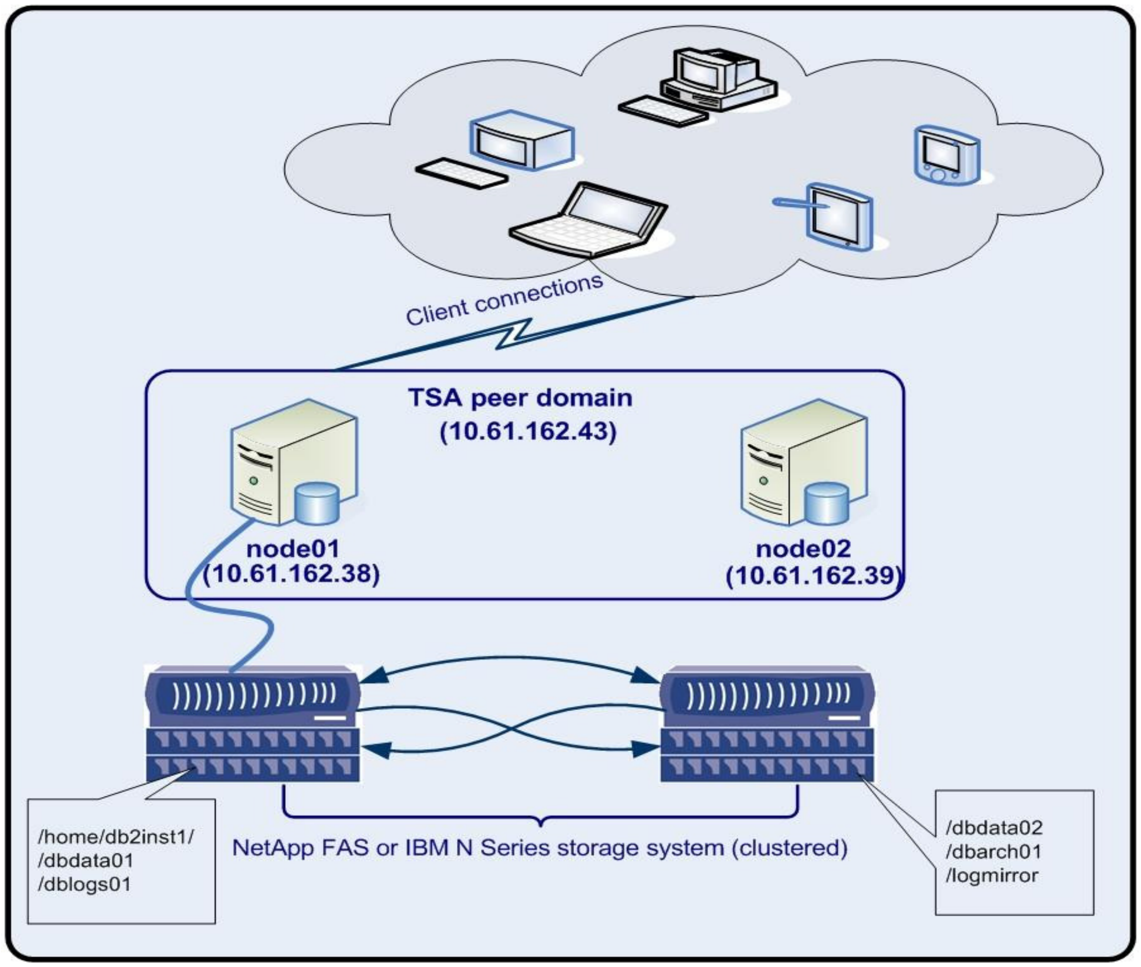

Figure 2) HA environment with a single partition DB2 instance, TSA cluster with two nodes, and storage system (cluster configuration).

If your TSA cluster has an even number of nodes, you must define a tiebreaker to protect your critical resources, in case your cluster falls apart into two or more subclusters. Using a tiebreaker configuration resource manager determines which subcluster has the majority of nodes and starts the critical resource on that subcluster. For information on other tiebreaker types, refer to the TSA base component and reference guide.

2 BUILD THE ENVIRONMENT

2.1 ASSUMPTIONS

To take maximum advantage of the information in this paper, you should have a fair understanding and knowledge of the following products and technologies:

- Linux operating system
- Host cluster concept and Tivoli System Automation
- DB2 Enterprise Server Edition (ESE) V9
- Data ONTAP®

You should also have verified the system prerequisites for TSA, DB2 ESE 9, and Data ONTAP and have product manuals available for reference.

2.2 INSTALL TSA FOR MULTIPLATFORM

To install the TSA base component, you need to complete the following basic steps on all of the nodes:

1. Log in as the user `root` and execute the following command to set up the `CT_MANAGEMENT_SCOPE` environment variable (all nodes):

```
echo "CT_MANAGEMENT_SCOPE=2" >>  /root/.bashrc
```

2. Obtain the latest TSA software CD or download the software from the IBM Website (*www14.software.ibm.com/webapp/download/home.jsp*).

3. If the software file is in compressed form, uncompress it by using the following command:

```
tar –xvf [TarFileName]
```

Where

`TarFileName` identifies the TSA software file name to be uncompressed.

Note: In this document, parameters shown in angle brackets (< >) are optional; parameters or options shown in square brackets ([ ]) are required and must be provided; a comma followed by ellipses (…) indicates that the preceding parameter can be repeated multiple times.

For example, to uncompress a file named `C86P9ML.tar`, you would execute the following command on the node:

```
tar –xvf C86P9ML.tar
```

Upon execution, the `tar` command uncompresses the software in a new directory (`SAM2200Base`) that is created in the current working directory.

4. Perform a prerequisites check by executing the following command:

```
./SAM2200Base/prereqSAM
```

If your system didn't pass the prerequisites check, correct any problem that has been detected before proceeding to the next step.

5. Install the TSA base component by executing the following command:

```
./SAM2200Base/installSAM
```

After you accept the license agreement, the installation program completes the TSA installation. For further information on the installation process, refer to the product's installation and configuration guide.

6. Repeat steps 1 through 5 on all the nodes that will be members of the cluster.

2.3 CONFIGURE NODES AND STORAGE SYSTEM

Before you create a DB2 instance, you must complete the following configuration steps on the storage system and on all the nodes that are involved in the TSA cluster.

A. Steps to be performed on the storage system:

1. TSA supports both remote shell (`rsh`) and secure shell (`ssh`) access among the cluster nodes and storage systems. For security reasons, we used `ssh` in our test environment. To enable `ssh` on the storage systems used for your TSA environment, follow the steps described in Appendix C, "Enable SSH for the Node and Storage Systems."

2. Add a user named `db2inst1` on the storage system by executing the following command:

```
useradmin user add db2inst1 –g Administrators
```

Note: The user name and password should match the user name and password on the TSA nodes.

3. Create volumes on the storage systems necessary for your database environment. A volume can be created by executing the following command on the storage system:

```
vol create [VolName] [AggrName] [VolSize]
```

Where

VolName identifies the name of the volume to be created.

AggrName identifies the name of the aggregate that contains the volume.

VolSize identifies the size of volume in KB, MB, or GB.

For example, to create a volume named `db2_home` within an aggregate named `dbaggr01` and a size of 750MB, you would execute the following command on the storage system:

```
vol create db2home dbaggr01 750m
```

For our test environment, the names of two storage systems used were Mystique and Iceman. We distributed the database storage on the storage cluster to achieve the highest availability. We created the following volumes.

Volume Name	Storage System Name	Purpose – DB2 Storage
/tsadisk	Mystique	Tiebreaker disk
/db2home	Mystique	Binaries
/dbdata01	Mystique	Data and indexes
/dbdata02	Iceman	Data and indexes
/dblogs01	Mystique	Transaction logs
/dbarch01	Iceman	Archive logs
/logmirror	Iceman	Transaction logs mirrored

4. By default, a volume has a schedule defined for Snapshot™ copies. NetApp does not recommend using this default schedule for the volumes that are used for the database. The default Snapshot schedule can be turned off by executing the following command:

```
vol options [VolName] nosnap on
```

Where

VolName identifies the name of the volume that default Snapshot turned off.

For example, to turn off the default Snapshot schedule for a volume named `db2home`, you would execute the following command on the storage system:

```
Vol options db2home nosnap on
```

5. If you are using a UNIX® operating system, make sure that each volume used for the DB2 database has its security style set to `UNIX`. The security style of a volume can be updated by executing the following command on the storage system:

`qtree security [`*`VolPath`*`] unix`

Where

`VolPath` identifies the volume path on the storage system that is used for the database.

For example, to update the security style of a volume named `dbdata01`, you would execute the following command on the storage system:

```
qtree security /vol/dbdata01 unix
```

6. Create a LUN on the volume named `tsadisk` and make the LUN accessible from all the nodes. For detailed information on how to make LUNs accessible on a node, refer to technical report "DB2 Enterprise Server Edition 9 for UNIX: Integrating with a NetApp Storage System" (www.netapp.com/library/tr/3531.pdf).

Note: The igroup that you create for TSA must have WWNN for all the nodes involved in the TSA cluster. The `igroup create` command should look similar to the following:

```
igroup create –I –t linux viper01_02_tsa_igp
iqn.198705.com.cisco:01.234764568c99
iqn.198705.com.cisco:01.94e8bc9a97bd
```

7. For our test environment, we used NFS to access the storage system. To achieve a high-availability DB2 database, the volumes that are used for the database must be accessible from all the nodes in the TSA cluster. Therefore you need to add an export entry to the `/etc/exports` file for each volume by executing the following command:

`exportfs -p sec=sys,rw=[`*`NodeList`*`],root= [`*`NodeList`*`] [`*`Volpath`*`]`

Where

`NodeList` identifies the list of cluster member nodes.

`VolPath` identifies the path for the volume on the storage system.

For example, to add the export entry for a volume named `dbdata01` and make it exportable on the nodes named `viper01` and `viper02`, you would execute the following command on the storage system:

```
exportfs -p sec=sys,rw=viper01:viper02,root=viper01:viper02
/vol/dbdata01
```

B. Steps to be performed on the TSA cluster nodes:

1. Add users and user groups required for the database operation by executing the following commands:

```
-- Add groups
groupadd -g 999 db2iadm1
groupadd -g 998 db2fadm1
groupadd -g 997 db2asgrp

-- Add users
useradd -g db2iadm1 -u 1005 -d /mnt/db2inst1 -m db2inst1
useradd -g db2fadm1 -u 1003 -d /home/db2fenc1 -m db2fenc1
useradd -g db2asgrp -u 1001 -d /home/db2as    -m db2as
```

Note: The home directory for the user `db2inst1`, who is going to be the owner of the DB2 instance, is on a volume that resides on the storage system.

2. Enable `ssh` on each node and copy the public key files to each storage system used for the database storage. For the steps, refer to Appendix C, "Enable SSH for the Node and Storage Systems."

3. Create mount points for the tiebreaker disk, DB2 binaries (DB2 home), data files, transaction log files, mirror log files, and archive log files by executing the following command:

```
mkdir -p /mnt/tsadisk /mnt/db2inst1 /mnt/dbdata01 /mnt/dbdata02
/mnt/dblogs01
/mnt/dbarch01 /mnt/logmrr01
```

4. For each volume that is used for database storage, add an entry to the `/etc/fstab` file. The entry should specify the mount point name and mount options. The entry format should look similar to the following:

```
[StorageSystemName]:[VolPath] [MountPoint] nfs
hard,rw,nointr,rsize=32768,wsize=32768,bg,vers=3,tcp 0 0
```

Where

StorageSystemName identifies the name of the storage system the volume resides on.

VolPath identifies the volume path for the volume on the storage system.

MountPoint identifies the name of the directory where the volume is mounted.

For example, the entry for a volume named `db2home` that resides on a storage system named `mystique` should look similar to the following:

```
mystique:/vol/db2home  /mnt/db2inst1 nfs
hard,rw,nointr,rsize=32768,wsize=32768,bg,vers=3,tcp 0 0
```

5. Using the `sanlun` utility, provided by NetApp, identify the tiebreaker disk's device name and add an entry for the disk to the `/etc/fstab` file. The entry should look similar to the following:

```
/dev/sdb /mnt/tsadisk ext2 defaults 0 0
```

2.4 INSTALL IBM DB2 9 AND CREATE A DB2 INSTANCE

To install DB2 and create a DB2 instance, you must complete the following steps:

1. Obtain DB2 Enterprise Server Edition 9 software from the product CD or electronic distribution (www-306.ibm.com/software/data/db2/udb/edition-ese.html). If you obtained the software from the IBM Web site, you must uncompress the downloaded software file by executing the following command:

```
tar -xvf [TarFileName]
```

Where

TarFileName identifies the compressed DB2 software file name.

For example, to uncompress TSA for a Linux software file named `DB2_Enterprise_Svr_Ed_Linux_x86.tar`, you would execute the following command on the node:

```
tar -xvf DB2_Enterprise_Svr_Ed_Linux_x86.tar
```

Upon execution, the `tar` command uncompresses the software in a new directory (`./db2`) that is created in the current working directory and places the `db2_install` program in the current directory.

2. Install DB2 by using the installation program that is located in the current directory:

```
./db2_install
```

After you answer a few questions, the installation program completes the DB2 installation. By default, DB2 is installed in the `/opt/IBM/db2/V9.x` directory and the installation process log is placed in the `/tmp/db2_install.log$$` file (where `$$` is the process ID). Check the log file to verify the installation.

3. For each mount point, grant full privileges to the user named `db2inst1` by executing the following commands:

```
chown -R [UserName]:[UserGroup] [MountPoint]
```

Where

UserName identifies the name that is to be assigned to the new database once it has been created.

UserGroup identifies the name of the user group to which the user belongs.

MountPoint identifies the mount point location whose ownership is changed.

For example, to change ownership of a mount point named `/mnt/db2home` to a user named `db2inst1`, you would execute the following command:

```
chown -R db2inst1:db2iadm1 /mnt/db2home
```

4. Repeat steps 1 through 3 on all the cluster nodes.

5. Log in as the user `root` to a TSA node and mount all the file systems that are used for the database by executing the following command:

mount [*MountPoint*]

Where

MountPoint identifies the mount point location where the new database is to be created.

For example, to mount a file system named `/mnt/db2home`, you would execute the following command on the node:

```
mount /mnt/db2home
```

6. Create a DB2 instance by executing the following command:

[*DB2Dir*]/instance/db2icrt -u [*FencedID*] [*InstanceName*]

Where

DB2Dir identifies the directory where the DB2 software was installed.

On all other UNIX-based operating systems, the installation directory for version 9.x is `/opt/IBM/db2/V9.x`.

FencedID identifies the ID of the user under which fenced user-defined functions and fenced stored procedures will run.

InstanceName identifies the name that is to be assigned to the new instance.

For example, to create a database instance named `db2inst1`, you would execute the following command on the database server:

```
/opt/IBM/db2/V9.1/instance/db2icrt -u db2fadm1 db2inst1
```

7. Log in as the database instance owner and execute the following command on the database server:

db2 "create database [*DatabaseName*] on [*MountPoint*]"

Where

DatabaseName identifies the name that is to be assigned to the new database once it has been created.

MountPoint identifies the location of the file system where the new database is to be created.

For example, to create a database named `mydb` on a file system that is mounted on a mount point named `/mnt/dbdata01`, you would execute the following command on the database server:

```
 db2 "create database mydb on /mnt/dbdata01"
```

8. It is not a good practice to leave transaction log files on their default location. In fact, when a DB2 database is stored on a NetApp FAS or IBM N series storage system volume, the transaction log files for the database should be stored on a separate volume. To change the location where transaction log files are stored, execute the following command on the database server:

db2 update db cfg for [*DatabaseName*] using NEWLOGPATH [*NewLogLocation*]

Where

DatabaseName identifies the name assigned to the database whose log file storage location is to be changed.

NewLogLocation identifies the new location where the database's transaction logs are to be stored.

For example, to change the log directory from the default location to a directory named `/mnt/dblogs01`, you would execute the following command on the database server:

```
db2 update db cfg for mydb using NEWLOGPATH /mnt/dblogs01
```

Note: In order to make new log path settings effective, disconnect all the users and deactivate the database. When the first connection is made after reactivating the database, the database manager moves the transaction log files to the new location.

9. Create a regular SMS table space with two containers (each on a separate drive), 64-page extent size, and 32-page prefetch size. Execute the following command on the node:

```
db2 "CREATE TABLESPACE mydbtbs03 PAGESIZE 4k MANAGED BY SYSTEM USING
('/mnt/dbdata01/db2inst1/NODE0000/MYDB/mydbtbs03',
'/mnt/dbdata02/db2inst1/NODE0000/MYDB/mydbtbs03') EXTENTSIZE 64
PREFETCHSIZE 32"
```

10. By default, DB2 uses a circular logging mechanism for the database. However, most production databases run in archive logging mode to support roll-forward recovery. Switching a DB2 database from circular logging to archive logging is easy, and it can be done by updating the primary log archive method configuration parameter (named `LOGARCHMETH1`). The `LOGARCHMETH1` configuration parameter can be updated by executing the following command on the database server:

db2 update db cfg for [*DatabaseName*] using LOGARCHMETH1 DISK:[*ArchiveDir1*]

Where

DatabaseName identifies the name assigned to the database whose logging method is to be changed.

ArchiveDir1 identifies the directory (location) where archived transaction log files are to be stored.

For example, to enable archive logging for a DB2 database named `mydb` and place the archive log file in a directory named `/mnt/dbarch1/mydb`, you would execute the following command on the database server:

```
db2 update db cfg for mydb using LOGARCHMETH1 DISK:/mnt/dbarch01
```

If you want to retain a second copy of archive log files on another disk, you need to update the secondary log archive method (`LOGARCHMETH2`) configuration parameter by executing the following command on the database server:

db2 update db cfg for [*DatabaseName*] using LOGARCHMETH2 DISK:[*ArchiveDir2*]

Where

DatabaseName identifies the name assigned to the database for which duplex logging is to be enabled.

ArchiveDir2 identifies the directory (location) where the second copy of the archived transaction log files is to be stored.

For example, to maintain a second set of archive logs for a DB2 database named `mydb` in a directory named `/mnt/dbarch2/mydb`, you would execute the following command on the database server:

```
db2 update db cfg for mydb using LOGARCHMETH2 DISK:/mnt/dbarch02/mydb
```

11. After you change the logging method from circular to archival, the database is placed into Backup Pending state and cannot be used until a full offline backup copy of the database is created. You can create an offline backup copy of the database by executing the following commands:

db2 force application all

db2 backup database [*DatabaseName*] to [*BackupDir*]

Where

DatabaseName identifies the name assigned to the database that is to be backed up.

BackupDir identifies the directory (location) where backup images are to be stored.

For example, to create an offline backup copy of the database named `mydb`, you would execute the following command on the database server:

```
db2 force application all
db2 backup database mydb to /dbbackup
```

After completing these steps, the DB2 database is ready and you can create database objects.

2.5 CREATE A TSA CLUSTER

To make DB2 environments highly available, you need to create a TSA cluster (domain), resource groups, resources, and resource relationships for the DB2 environment by completing the following steps:

1. The nodes used for the TSA cluster should be able to communicate with each other. You can prepare nodes for intracommunication by executing the following command:

preprpnode [*NodeList*]

Where

NodeList identifies the list of node names that will be part of the TSA cluster.

For example, to prepare nodes named `viper01` and `viper02`, you would execute the following command on each node:

```
preprpnode viper01 viper02
```

2. Create the TSA cluster by executing the following command on one of the nodes:

mkrpdomain [*DomainName*] [*NodeList*]

Where

DomainName identifies the name of the domain to be created.

NodeList identifies the list of node names that will be part of the TSA cluster.

For example, to create a TSA cluster named `db2tsa_cluster`, running on two nodes named `viper01` and `viper02`, you would execute the following command from any node:

```
mkrpdomain db2tsa_cluster viper01 viper02
```

3. Check the TSA cluster status by executing the following command:

```
Lsrpdomain
```

Output from this command should look similar to the following:

```
Name          OpState RSCTActiveVersion MixedVersions TSPort GSPort
db2tsa_cluster Offline  2.4.3.1           No            12347  12348
```

This output shows that the cluster is defined but is offline.

4. Bring the cluster online by executing the following command:

startrpdomain [*DomainName*]

Where

DomainName identifies the TSA cluster name to be brought online.

For example, to bring a TSA cluster named `db2tsa_cluster` online, you would execute the following command:

```
startrpdomain db2tsa_cluster
```

2.6 CREATE A TIEBREAKER

For our test environment, we used a SCSI tiebreaker. The primary requirement for the SCSI tiebreaker is that the disk that is used as a tiebreaker must be accessible from all the TSA cluster nodes. We created a LUN on the storage system to serve as a tiebreaker and made the LUN accessible to all the TSA cluster nodes using iSCSI. For detailed information on how to create and make a LUN accessible from a host, refer to the technical report "DB2 Enterprise Server Edition 9 for UNIX: Integrating with a NetApp Storage System" (www.netapp.com/library/tr/3531.pdf).

Once the SCSI disk is accessible from all the nodes, you must complete the following steps to create the SCSI tiebreaker:

1. Execute the following command to add an entry to the tiebreaker disk to the `/etc/fstab` file:

```
echo "/dev/sdb /mnt/tsadisk  ext3 _netdev 0 0">> /etc/fstab
```

2. Find the host, channel, ID, and LUN ID for the iSCSI disk by executing the following command:

```
cat /proc/scsi/scsi
```

The output from this command should look similar to the following:

```
Host: scsi2 Channel: 00 Id: 00 Lun: 00
         Vendor: NETAPP   Model: LUN              Rev: 0.2
         Type:   Direct-Access                    ANSI SCSI revision: 04
```

3. Create a tiebreaker resource by executing the following command from any TSA cluster node:

```
mkrsrc IBM.TieBreaker Name="TSA_TieBreaker" Type="SCSI" DeviceInfo="HOST=2
CHAN=00 ID=00 LUN=00" HeartbeatPeriod=2
```

4. Check the list of current tiebreakers by executing the following command:

```
lsrsrc –c IBM.PeerNode OpQuorumTieBreaker
```

The output from this command should look similar to the following:

```
Resource Class Persistent Attributes for: IBM.PeerNode
resource 1:
          OpQuorumTieBreaker = "Operator"
```

5. Make the tiebreaker resource you just created active by executing the following command:

```
chrsrc –c IBM.PeerNode OpQuorumTieBreaker="TSA_TieBreaker"
```

6. Check the tiebreaker status by executing the following command:

```
lsrsrc –c IBM.PeerNode
```

The output from this command should look similar to the following:

```
lsrsrc -c IBM.PeerNode
Resource Class Persistent Attributes for IBM.PeerNode
resource 1:
        CommittedRSCTVersion = ""
        ActiveVersionChanging= 0
        OpQuorumOverride     = 0
        CritRsrcProtMethod   = 1
        OpQuorumTieBreaker   = "TSA_TieBreaker"
        QuorumType           = 0
        QuorumGroupName      = ""
        Fanout               = 32
```

2.7 ASSOCIATE DB2 RESOURCES TO THE TSA CLUSTER

1. Append the DB2 service name and port to all the nodes. The name and port should match on all the nodes. For example, to make the DB2 service consistent in our test environment, we appended the following lines to the `/etc/services` file on all TSA cluster nodes:

```
DB2_db2inst1      60000/tcp
DB2_db2inst1_1    60001/tcp
DB2_db2inst1_2    60002/tcp
DB2_db2inst1_END  60003/tcp
```

2. Copy the TSA script from `/opt/ibm/db2/V9.1/ha/tsa` to the `/usr/sbin/rsct/sapolicies/db2` directory by executing the following command:

```
cp /opt/ibm/db2/V9.1/ha/tsa/* /usr/sbin/rsct/sapolicies/db2/.
```

3. Register the DB2 instance and its resources with the TSA cluster by using a TSA script (`regdb2salin`), supplied with DB2 software. You can find this script in the DB2 installation directory (`[DB2 install directory]/ha/tsa`). Execute the script:

regdb2sanlin -a [*DB2InstanceName*] -I [*IPAddress*] -s -t [*InstanceType*] -n <*NetMask*> -m <*DiskList*>

Where

DB2InstanceName identifies the name of the DB2 instance to be registered with the TSA cluster.

IPAddress identifies the IP to be made HA.

-s identifies that ssh is used instead of default `rsh`.

InstanceType identifies the instance type: a db2 instance or das. The default is db2.

NetMask identifies the HA IP address's NetMask.

DiskList identifies the list of disks (mount or raw lv) to make HA (comma separated, no spaces).

For example, to register a DB2 instance named `db2inst1` with a TSA domain that is running on the nodes named `viper01` and `viper02`, you would execute the following command:

```
./regdb2salin -a db2inst1 -i 10.61.162.43 -n 255.255.255.0 -s -t db2 -m
/mnt/db2inst1,/mnt/dbdata01,/mnt/dbdata02,/mnt/dblogs01,/mnt/dbarch01,/mnt
/logmrr01
```

In our test environment, the `regdb2salin` script was located in the `/opt/ibm/db2/V9.1/ha/tsa/` directory.

The output from the script execution should look similar to the following:

```
Distributing policy to node viper01 via ssh...
Distributing policy to node viper02 via ssh...
Warning: no netmask specified, will use default

About to register db2inst1 with TSA
Checking cluster validity...
Stopping instance db2inst1...

Checking resources able to host instance db2inst1 (partition 0)
 Checking viper01 ...
  Checking /mnt/db2inst1 ...
  Checking /mnt/dbdata01 ...
  Checking /mnt/dbdata02 ...
  Checking /mnt/dblogs01 ...
  Checking /mnt/dbarch01 ...
  Checking /mnt/logmrr01 ...
 Checking viper02 ...
  Checking /mnt/db2inst1 ...
  Checking /mnt/dbdata01 ...
  Checking /mnt/dbdata02 ...
  Checking /mnt/dblogs01 ...
  Checking /mnt/dbarch01 ...
  Checking /mnt/logmrr01 ...

Making resource group  db2_db2inst1_0-rg  ...
Making disk resource  db2_db2inst1_0-rs_mnt_db2inst1_disk ...
Making disk resource  db2_db2inst1_0-rs_mnt_dbdata01_disk ...
Making disk resource  db2_db2inst1_0-rs_mnt_dbdata02_disk ...
Making disk resource  db2_db2inst1_0-rs_mnt_dblogs01_disk ...
Making disk resource  db2_db2inst1_0-rs_mnt_dbarch01_disk ...
Making disk resource  db2_db2inst1_0-rs_mnt_logmrr01_disk ...
Making IP resource  db2_db2inst1_0-rs_ip ...
Making DB2 resource  db2_db2inst1_0-rs  ...
Making relationship db2_db2inst1_0-rg_mnt_db2inst1_disk_do ...
Making relationship db2_db2inst1_0-rg_mnt_dbdata01_disk_do ...
Making relationship db2_db2inst1_0-rg_mnt_dbdata02_disk_do ...
Making relationship db2_db2inst1_0-rg_mnt_dblogs01_disk_do ...
Making relationship db2_db2inst1_0-rg_mnt_dbarch01_disk_do ...
Making relationship db2_db2inst1_0-rg_mnt_logmrr01_disk_do ...
Making relationship db2_db2inst1_0-rg_IP_do...

Online resource group db2_db2inst1_0-rg ...

db2_db2inst1_0-rg is now online

Done, instance is now HA
```

Important: After registering DB2 instances with the TSA cluster, DO NOT USE `db2stop` or `db2start`. Instead, to stop or start the instance, use the following command:

`chrg –o offline|online [`*`ResourceGroup`*`]`

Where:

`ResourceGroup` identifies the name of the DB2 instance resource group (for example, `db2_db2inst1_0-rg`).

4. Check the status of the DB2 instance and resource by executing the following command:

```
/opt/ibm/db2/V9.1/ha/tsa/getstatus
```

The output from this command should look similar to the following:

```
-- Resource Groups and Resources --
       Group Name            Resources
       -----------------    ----------------------
       db2_db2inst1_0-rg    db2_db2inst1_0-rs_ip
       db2_db2inst1_0-rg    db2_db2inst1_0-rs_mnt_db2inst1_disk
       db2_db2inst1_0-rg    db2_db2inst1_0-rs
                                    -                    -

-- Resources --
       Resource Name                       Node Name       State
      --------------------                ------------     --------
       db2_db2inst1_0-rs_ip                viper01           Online
      db2_db2inst1_0-rs_ip                viper02          Offline
             -                               -                -

d2_db2inst1_0-rs_mnt_db2inst1_disk        viper01          Online
db2_db2inst1_0-rs_mnt_db2inst1_disk       viper02          Offline
              -                              -                -
      db2_db2inst1_0-rs                   viper01          Online
      db2_db2inst1_0-rs                   viper02          Offline
```

3 HIGH-AVAILABILITY TEST SCENARIO

3.1 CONTROL FAILOVER—MAINTENANCE MODE

Using control failover, you can bring a node off the cluster and perform regular maintenance operations such as upgrading the operating system, the application, or the database. In this scenario, the TSA stops all the active resources on the node and starts them on another node in the cluster. In order to perform a control failover, you must execute the following command on the node:

`rgreq –o Move –n p[NodeName] [`*`ResourceGroup`*`]`

Where

NodeName identifies the name of the node to be taken off the TSA cluster.

ResourceGroup identifies the name of the resource group to be moved to another node.

For example, to control failover of a TSA cluster node named `viper01`, which has an active DB2 instance, you would move the resource group named `db2_db2inst1_0-rg` by executing the following command:

```
rgreq –o move –n viper01 db2_db2inst1_0-rg
```

Upon execution of the `rgreq` command, the specified resource group is moved to another node in the TSA cluster. You can check the status of the resources by executing the following command:

```
/opt/ibm/db2/V9.1/ha/tsa/getstatus
```

The output from this command should look similar to the following:

```
-- Resource Groups and Resources --
       Group Name            Resources
       -----------------    ----------------------
       db2_db2inst1_0-rg    db2_db2inst1_0-rs_ip
       db2_db2inst1_0-rg    db2_db2inst1_0-rs_mnt_db2inst1_disk
       db2_db2inst1_0-rg    db2_db2inst1_0-rs
                                    -                    -
```

```
-- Resources --


       Resource Name                        Node Name       State
       --------------------                 ------------     --------
        db2_db2inst1_0-rs_ip                 viper01          Offline
       db2_db2inst1_0-rs_ip                 viper02          Online
            -                                  -                -

d2_db2inst1_0-rs_mnt_db2inst1_disk          viper01          Offline
db2_db2inst1_0-rs_mnt_db2inst1_disk         viper02          Online
             -                                 -                -
      db2_db2inst1_0-rs                     viper01          Offline
      db2_db2inst1_0-rs                     viper02          Online
                                                                           …
```

Once active resources are moved to another node, you can exclude the node from the TSA cluster by executing the following command:

stoprpnode [*NodeName*]

Where

NodeName identifies the name of the node that needs control failover.

For example, to exclude a node named `viper01` from the TSA cluster, you would execute the following command:

```
stoprpnode viper01
```

After performing maintenance, you can add the node to the TSA cluster by executing the following command:

startrpnode [*NodeName*]

Where

NodeName identifies the name of the node that needs control failover.

For example, to add a node named `viper01` to the TSA cluster, you would execute the following command:

```
startrpnode viper01
```

3.2 HARDWARE FAILURE

A database environment can have many hardware components such as server, storage, and network components. This section covers two of the most common hardware failure scenarios: node failure and storage failure.

NODE FAILURE

A node can become unavailable for various reasons, such as power failure, network failure, or server hardware failure. For our test, we tested a power failure scenario by rebooting the node. To reboot a node that is running an active database instance, you would execute the following command:

```
shutdown –r now
```

After executing this command, you would need to monitor the status of the TSA cluster from another node by executing the following command:

```
/opt/ibm/db2/V9.1/ha/tsa/getstatus
```

The output from this command should look similar to the following:

```
[root@viper01 tsa]# /opt/ibm/db2/V9.1/ha/tsa/getstatus

Resource Groups and Resources -

                              Group Name            Resources

. . .                                       -                    -

-- Resources -

                           Resource Name            Node Name             State

                           -------------            ---------             -----
                    db2_db2inst1_0-rs_ip              viper01            Online
                    db2_db2inst1_0-rs_ip              viper02    Failed_Offline
                                       -                    -                 -
     db2_db2inst1_0-rs_mnt_logmrr01_disk              viper01            Online
     db2_db2inst1_0-rs_mnt_logmrr01_disk              viper02    Failed_Offline
                                       -                    -                 -
     db2_db2inst1_0-rs_mnt_dbarch01_disk              viper01            Online
     db2_db2inst1_0-rs_mnt_dbarch01_disk              viper02    Failed_Offline
                                       -                    -                 -
     db2_db2inst1_0-rs_mnt_dblogs01_disk              viper01            Online
     db2_db2inst1_0-rs_mnt_dblogs01_disk              viper02    Failed_Offline
                                       -                    -
```

Upon failure to reboot in this scenario, all the TSA recourses that were active on the failed node are started on another node in the TSA cluster. The database instance becomes available on another node within a few seconds, and the application can reconnect to the database without any manual intervention. On failure, the status of the resources that were on the failed node becomes `Failed_Offline`. Once the node is repaired or becomes available, TSA changes the status of the resources from `Failed_offline` to `offline`.

NODE FAILURE

In a database environment, not only the nodes and network components but also storage are likely targets for single point of failure (SPOF). To avoid SPOF at the storage level, you can use a storage system cluster configuration. For our test, we used a NetApp FAS3050C storage system. To simulate storage system failure, you can do one of the following:

- Reboot or halt one of the storage system controllers.
- Pull the power or network cable from one of the storage system controllers.

For our test, we rebooted one of the storage system controllers while a load was running on the database. You can reboot the storage system controller by executing the following command:

```
reboot -t 0
```

After rebooting the storage system controller, you need to monitor the TSA cluster status by executing the following command:

```
/opt/ibm/db2/V9.1/ha/tsa/getstatus
```

The output from this command should look similar to the following.

```
[root@viper01 tsa]# /opt/ibm/db2/V9.1/ha/tsa/getstatus

-- Resource Groups and Resources --

. . .

 -- Resources --
                            Resource Name            Node Name            State

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

```
. . .
                                           -                     -               -

     db2_db2inst1_0-rs_mnt_dbarch01_disk              viper01          Online
     db2_db2inst1_0-rs_mnt_dbarch01_disk              viper02          Offline
                                           -                     -               -
     db2_db2inst1_0-rs_mnt_dblogs01_disk              viper01          Unknown
     db2_db2inst1_0-rs_mnt_dblogs01_disk              viper02          Offline
                                           -                     -               -
     db2_db2inst1_0-rs_mnt_dbdata02_disk              viper01          Online
     db2_db2inst1_0-rs_mnt_dbdata02_disk              viper02          Offline
                                           -                     -               -
     db2_db2inst1_0-rs_mnt_dbdata01_disk              viper01          Unknown
     db2_db2inst1_0-rs_mnt_dbdata01_disk              viper02          Offline
                                           -                     -               -
     db2_db2inst1_0-rs_mnt_db2inst1_disk              viper01          Unknown
     db2_db2inst1_0-rs_mnt_db2inst1_disk              viper02          Offline
                                           -                     -               -
```

Immediately after reboot, the status of the file system resources that were located on the rebooted storage system becomes `unknown`. But the storage system is using cluster configuration; therefore, after a few seconds, the second controller in the storage system cluster takes over the load of the failed controller and makes the file system available. The takeover is transparent to the applications, and the application connection remains active.

3.3 SOFTWARE FAILURE

A DB2 database may become unavailable because of software failure. To simulate this scenario, log in as the user instance owner and kill the DB2 processes by executing the following command:

```
db2_kill
```

Another way of killing DB2 processes is to log in as the user `root` and use the `kill -9` command to kill the parent process named `db2sysc` in the DB2 process tree.

After killing DB2 processes, monitor the TSA cluster status by executing the following command:

```
/opt/ibm/db2/V9.1/ha/tsa/getstatus
```

The output from this command should look similar to the following:

```
[root@viper01 tsa]# /opt/ibm/db2/V9.1/ha/tsa/getstatus

-- Resource Groups and Resources --
. . .

-- Resources --
                           Resource Name            Node Name           State
                           -------------            ---------           -----
                    db2_db2inst1_0-rs_ip              viper01          Online
                    db2_db2inst1_0-rs_ip              viper02          Offline
                                           -                     -               -
     db2_db2inst1_0-rs_mnt_logmrr01_disk              viper01          Online
     db2_db2inst1_0-rs_mnt_logmrr01_disk              viper02          Offline
     db2_db2inst1_0-rs_mnt_db2inst1_disk              viper01          Online
     db2_db2inst1_0-rs_mnt_db2inst1_disk              viper02          Offline

               db2_db2inst1_0-rs               viper01          Offline

               db2_db2inst1_0-rs               viper02          Offline
```

Pay attention to the status of the resource that is associated with your DB2 instance. The resource becomes `offline` just after you kill the DB2 processes. But after a couple of seconds, TSA restarts the DB2 processes and the instance becomes accessible without any manual intervention.

4 SUMMARY

High availability of information is vital for business success. IBM Tivoli System Automation, combined with a clustered NetApp storage system, provides a robust, high-availability solution for your DB2 environment.

5 TECHNICAL REFERENCES

- IBM Tivoli System Automation for Multiplatforms: Product page

(www-306.ibm.com/software/tivoli/products/sys-auto-linux)

- RSCT Technical Documentation

(www-03.ibm.com/systems/clusters/library/linux.html)

- Tivoli System Automation for Multiplatforms: Installation and Configuration Guide

(publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html)

- Tivoli System Automation for Multiplatforms: Base Component Administrator’s and User’s Guide

(publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html)

- DB2 Product Documentation IBM Website

(www-306.ibm.com/software/data/db2/udb/support/manualsv9.html)

- NetApp Cluster Installation and Administration Guide

(now.netapp.com/NOW/knowledge/docs/ontap/rel711)

- Technical report: “DB2 Enterprise Server Edition 9 for UNIX: Integrating with a NetApp Storage System”

(www.netapp.com/library/tr/3531.pdf)

- White paper: “Highly Available DB2 Universal Database Using Tivoli System Automation for Linux”

(ftp.software.ibm.com/software/data/pubs/papers/halinux.pdf)

REVISION HISTORY

Date	Author	Comments
March 2006	Jawahar Lal and Roger Sanders	Original draft
Feb 2010	Bobby Oommen	Updated Data ONTAP, DB2, Controller Type, OS Version and Template.

APPENDIX A: TSA TERMS

cluster/peer domain

The terms TSA cluster and peer domain are used interchangeably. The TSA cluster is a group of two or more host systems whose resources Tivoli System Automation manages.

node

A host system that is part of a TSA cluster is called a node. TSA v1.2 supports up to 32 nodes within a cluster.

resource

A resource is any piece of hardware or software that can be defined in TSA. Resources have characteristics, or attributes that can be defined. For example, when considering an IP address as a resource, attributes include the IP address itself and the net mask.

resource attributes

A resource attribute describes some characteristics of a resource. There are two types of resource attributes: persistent attributes and dynamic attributes. The attributes of the IP address (the IP address itself and the net mask) are examples of persistent attributes. They describe enduring characteristics of a resource. Although it is possible to change the IP address and net mask, these characteristics are, in general, stable and unchanging. Dynamic attributes represent changing characteristics of the resource. For example, a dynamic attribute of an IP address is its operational state.

resource class

A resource class is a collection of resources of the same type.

resource group

A resource group is a logical container for a collection of resources. This container enables the control of multiple resources as a single logical entity. Resource groups are the primary mechanism for operations within TSA.

managed resource

A managed resource is a resource that has been associated with a TSA cluster. To make this association, the resource is added to a resource group, at which time it becomes manageable through TSA.

nominal state

The nominal state of a resource group indicates to TSA whether the resources within the group should be online or offline at this point in time. Setting the nominal state to Offline indicates that you want TSA to stop the resources in the group, and setting the nominal state to Online indicates that you want to start the resources in the resource group. You can change the value of the `NominalState` resource group attribute, but you cannot set the nominal state of a resource directly.

equivalency

An equivalency is a collection of resources that provide the same functionality. For example, equivalencies are used for selecting network adapters that should host an IP address. If one network adapter goes offline, TSA selects another network adapter to host the IP address.

relationships

TSA enables the definition of relationships between resources in a cluster. There are two relationship types:

- *Start-stop* relationships are used to define start and stop dependencies between resources. You can use the `StartAfter`, `StopAfter`, `DependsOn`, `DependsOnAny`, and `ForcedDownBy` relationships to achieve this. For example, if a resource must be started only after another resource was started, you can define this by using the policy element `StartAfter` relationship.

- *Location relationships* are applied when resources must or should, if possible, be started on the same or a different node in the cluster. TSA provides the following location relationships: `Collocation`, `AntiCollocation`, `Affinity`, `AntiAffinity`, and `IsStartable`.

quorum

The main goal of quorum operations is to keep data consistent and to protect critical resources. Quorum can be seen as the number of nodes in a cluster that are required to modify the cluster definition or to perform certain cluster operations. There are two types of quorum:

- *Configuration quorum* determines when configuration changes in the cluster will be accepted. Operations affecting the configuration of the cluster or resources are allowed only when the absolute majority of nodes are online.
- *Operational quorum* is used to decide whether resources can be safely activated without creating conflicts with other resources. In case a cluster split, resources can only be started in the subcluster that has a majority of nodes or that has obtained a tiebreaker.

tiebreaker

Clusters with an even number of nodes require a method to ensure operational quorum in the event than no subcluster has a majority of nodes. The tiebreaker is used to determine which subcluster will have an operational quorum. A tiebreaker ensures that exactly one subcluster wins the tiebreaker and all others lose.

There are three main groups of tiebreakers in IBM TSA for Multiplatforms V2.1: disk tiebreakers (SCSI, ECKD, and so on), the operator tiebreaker, and the network tiebreaker.

APPENDIX B: BASIC TSA CLUSTER ADMINISTRATION COMMANDS

preprpnode

This command prepares the member nodes for the TSA cluster. Upon execution, member nodes exchange the public key and the RMC access control list is modified to enable access to the cluster resources by all the nodes in the cluster. Syntax:

`preprnode [`*`NodeName...`*`]`

Where

`NodeName` identifies the name of the node in the TSA cluster.

For example, to prepare nodes named `viper01` and `viper02` for a node TSA cluster, you would execute the following command from each node:

```
preprnode viper01 viper02
```

mkrpnode

The command is used to create a new TSA cluster. Syntax:

`mkrpdomain [`*`DomainName`*`][`*`NodeName...`*`]`

Where

`DomainName` identifies the name of the TSA cluster being created.

`NodeName` identifies the name of the nodes that are included in the TSA cluster.

```
mkrpdomain db2tsa_cluster viper01 viper02
```

lsrpdomain

This command is used to list the TSA cluster information. Syntax:

`lsrpdomain <`*`DomainName`*`>`

Where

`DomainName` identifies the name of the TSA cluster being created.

For example, to list information for a TSA cluster named `db2tsa_cluster`, you would execute the following command on a node:

```
lsrpdomain db2tsa_cluster
```

The output from this command should look similar to the following:

```
lsrpdomain db2tsa_cluster

Name           OpState RSCTActiveVersion MixedVersions TSPort GSPort
db2tsa_cluster Online   2.4.3.1           No            12347  12348
```

startrpdomain

This command is used to start a cluster domain. Syntax:

`startrpdomain [`*`DomainName`*`]`

Where

`DomainName` identifies the name of the TSA cluster being created.

For example, to start a cluster named `db2tsa_cluster`, you would execute the following command from a node:

```
startrpdomain db2tsa_cluster
```

stoprpdomain

This command is used to stop a cluster domain. Syntax:

`stoprpdomain [`*`DomainName`*`]`

Where

`DomainName` identifies the name of the TSA cluster being created.

For example, to stop a TSA cluster named `db2tsa_cluster`, you would execute the following command from a node:

```
stoprpdomain db2tsa_cluster
```

lsrpnode

This command is used to list the member nodes of the TSA cluster and their operating state. Syntax:

`lsrpnode`

The output from this command should look similar to the following:

```
Name    OpState RSCTVersion

viper01 Online  2.4.3.1
viper02 Online  2.4.3.1
```

rmrpdomain

This command is used to remove a TSA cluster. Syntax:

`rmrpdomain [`*`DomainName`*`]`

Where

`DomainName` identifies the name of the TSA cluster that is being removed.

For example, to remove a TSA cluster named `db2tsa_cluster`, you would execute the following command from a node:

```
rmrpdomain db2tsa_cluster
```

rmrpnode

The command is used to remove a node from a TSA cluster. Syntax:

`rmrpnode [`*`NodeName]`*

Where

`NodeName` identifies the name of node that is to be removed from the TSA cluster.

For example, to remove a node named `viper01` from a cluster, you would execute the following command from the node:

```
rmrpnode viper01
```

APPENXDIX C: ENABLE SSH FOR THE NODES AND STORAGE SYSTEMS

C.1 ENABLE SSH FOR THE NODES

TSA version 2.1 or later supports node access by using `rsh` or `ssh`. For security reasons, we chose to use `ssh`. To enable password-less logins to all nodes from each node, you need to exchange `ssh` keys between nodes. To enable `ssh` access, complete the following steps on each node:

1. Log in as the user `root` and generate `ssh-keygen` keys by executing the following command:

```
/usr/bin/ssh-keygen -f /root/.ssh/id_dsa -q -t dsa -N ""

/usr/bin/ssh-keygen -f /root/.ssh/id_rsa -q -t rsa -N ""
```

2. Place the generated keys in the `/root/.ssh/authorized_keys2` file:

```
cat /root/.ssh/id_dsa.pub >> /root/.ssh/authorized_keys2

cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys2
```

3. Create public keys for the user `db2inst1` by completing steps 1 and 2.
4. Create a copy of the public key file by executing the following command:

```
cp  /root/.ssh/authorized_keys2 /root/.ssh/public_keys2
```

5. You need to exchange the public keys among all the nodes. For example log in on `viper01` and `ssh` to all other nodes and append the public keys for those nodes to the public key file of `node01`, as shown in the following command:

```
ssh viper02 cat /root/.ssh/public_key2 >> /root/.ssh/authorized_keys2
```

When this command is executed,, the public key for the user `root` is appended on the node `viper02` to the public key for the user `root` on the node `viper01`.

C.2 ENABLE SSH ON NETAPP FAS OR IBM N SERIES STORAGE SYSTEM

To copy `ssh` on the storage system, complete the following steps:

1. Set up `ssh` by executing the following command:

```
secureadmin setup ssh
```

Accept all the default values and complete the setup process.

2. Enable `ssh` by executing the following command:

```
secureadmin enable {ssh1|ssh2}
```

Use `ssh1` to enable `ssh` service for SSH 1.*x* clients or use `ssh2` to enable `ssh` service for SSH 2.0 clients.

3. Mount the root volume (`/vol/vol0`) of the storage system on a node by executing the following command:

```
mount [StorageSystemName]:/vol/vol0 [MountPoint]
```

Where

StorageSystemName identifies the name of the storage system.

MountPoint identifies the mount point name on the node.

For example, to mount the root volume of a storage system named `iceman` to a mount point named `/mnt/iceman`, you would execute the following command:

```
mount iceman:/vol/vol01 /mnt/iceman
```

4. Create a directory in the `/etc/sshd/<user_name>/.ssh` directory for the user whose public keys are copied. For example, you would execute the following command to create a directory for the user `root`:

```
mkdir -p /mnt/iceman/sshd/root/.ssh
```

5. Copy the public keys by executing the following command on the node:

```
cat /root/.ssh/authorized_keys2 >>
/mnt/iceman/sshd/root/.ssh/authorized_keys2
```

Repeat steps 1 through 5 for each storage system.

Next you should test `ssh` access from each node to all other nodes. On the first connection to the remote node, `ssh` prompts you to accept the key because it is not yet known. It is useful to invoke `ssh` for all interfaces and nodes (short and fully qualified) to get all the required keys and to avoid being prompted for a password. You should do this on all the nodes in the cluster.