Information integration software solutions
White paper

DB2 Information Management Software

IBM DB2 Universal Database and High Availability on Sun Cluster 3.x

By
Xin Chen — IBM Software Group
Steve Raspudic — IBM Software Group
Julie Mcdonald — Sun Microsystems Inc.

December 2003

Table of Contents

Overview

This white paper describes the implementation and design of highly available IBM® DB2® Universal Database™ (DB2 UDB) environments on the Sun™ Cluster platform. Also included is a detailed description of the high availability DB2 (DB2 UDB HA) agent for Sun Cluster 3.x*. We provide guidance and recommendations for highly available strategies using DB2 Universal Database V7 Enterprise Edition (EE) and DB2 Universal Database Enterprise-Extended Edition (EEE), and V8 with or without using the Database Partitioning Feature (DPF). Practical considerations regarding design, implementation, testing and performance work with Sun Cluster 3.x* are also covered.

Sun Cluster 3.x

Sun Cluster 3.x provides high availability by enabling application failover. The state of each individual node is periodically monitored. The cluster software automatically relocates a cluster-aware application from a failed primary node to a designated secondary node. When a failover occurs, clients may see a brief interruption in service and may need to reconnect after the failover has finished. However, clients are not aware of the physical server from which they are provided the application and data.

By allowing other nodes in a cluster to automatically host workloads when the primary node fails, Sun Cluster 3.x can significantly reduce downtime and increase productivity, providing high availability service to all users.

Sun Cluster has the following key components:
Hardware:

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

*At the time of writing this white paper, the coexisting release versions of Sun Cluster software are 3.0 and 3.1. The DB2 UDB HA agent for Sun Cluster has been certified to support both Sun Cluster 3.0 and 3.1. For the purpose of this article, we will refer to the Sun Cluster version as 3.x.

Software:

- Solaris operating environment
- Sun Cluster software
- Volume management (Solaris Volume Manager, Solstice DiskSuite or VERITAS Volume Manager)
- Data service application

Multihost Storage

Sun Cluster 3.x requires multihost disk storage—disks that can be connected to more than one node at a time. In the Sun Cluster 3.x environment, multihost storage allows disk devices to become highly available. Disk devices resident on the multihost storage can tolerate single node failures since there still exists a physical path to the data through the alternative server node.

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

A volume manager provides for mirrored or RAID5 configurations for data redundancy of the multihost disks. Currently, Sun Cluster 3.x supports Solstice DiskSuite and VERITAS Volume Manager as volume managers. When running with Solaris9, Sun Cluster 3.x also supports Solaris Volume Manager. Combining multihost disks with disk mirroring and striping protects against both node failure and individual disk failure.

Global Devices

Sun Cluster 3.x uses global devices to provide cluster-wide, highly available access to any device in a cluster, from any node, without regard to the device's physical location. All disks are included in the global namespace with an assigned device ID (DID) and are configured as global devices. Therefore, the disks themselves are visible from all cluster nodes.

Cluster File Systems/Global File Systems

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

For the current file system and volume manager support, please refer to the Sun Cluster product Web site.

Device Groups

In Sun Cluster 3.x, all multihost storage must be controlled by the Sun Cluster framework. Disk groups are first created on the multihost disk. Then they are registered as Sun Cluster disk device groups. A disk device group is a type of global device.

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

Resource Group Manager (RGM)

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

Data Services

The term data service is used to describe a third-party application that has been configured to run on a cluster rather than on a single server. A data service includes the application software and Sun Cluster 3.x software responsible for starting, stopping and monitoring the application. Sun Cluster 3.x supplies data service methods that are used to control and monitor the application within the cluster. These methods run under the control of the RGM, which uses them to start, stop, and monitor the application on the cluster nodes. These methods, along with the cluster framework software and multihost disks, enable applications to become highly available data services. As such, they can prevent significant application interruptions after any single failure within the cluster, regardless of whether the failure is to a node, an interface component or to the application itself.

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

Resource Types, Resources and Resource Groups

A resource type consists of three things:

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

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

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

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

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

Sun Cluster 3.x DB2 UDB HA Agent

The DB2 UDB HA agent developed by IBM and jointly certified by Sun Microsystems is a software product consisting of the methods required to start, stop, monitor and administer DB2 UDB in a Sun Cluster 3.x environment. This agent integrates the robust, highly available DB2 UDB database technology with Sun's clustering technology. The combined system architecture provides reliable, high performance data services in environments from a workgroup up to the entire enterprise.

The DB2 UDB HA Agent for Sun Cluster 3.x, as part of the DB2 release, is free of charge for customers holding a valid DB2 license. The DB2 UDB HA Agent of DB2 V7.2 and the DB2 UDB HA Agent of DB2 V8 support both Sun Cluster 3.0 and 3.1.

For DB2 UDB V7.2 (or V7.1 with FixPak 3), because of the early media production cut-off date, the HA agent for Sun Cluster 3.x is not included on the V7.2 production CD. It is distributed with FixPak 5 and above, and installed with the FixPak. For DB2 UDB V8, the HA agent for Sun Cluster 3.x is shipped with the V8 production CD, and installed with the binary installation.

The DB2 UDB HA agent is installed as part of DB2 binaries. For Version 7, it is installed in /opt/IBMdb2/V7.1/ha/sc30 and for Version 8 in /opt/IBM/db2/V8.1/ha/sc30. For a successful installation, the following directory structure should appear in ./ha/sc30 directory:

IBMdb2udb
README.db2udb
bin/
docs/
etc/
util/

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

The monitor method is implemented in the file db2udb_svc_probe.ksh. For DB2 UDB HA agent of DB2 V8, the monitor uses the db2gcf executable to determine the health of the particular instance by determining that the required processes and inter-process communication facilities (IPCs) exist and are active.

Cluster Topology

The DB2 UDB HA agent fully supports all three cluster topology classes supported by Sun Cluster 3.x, which are:

- Clustered Pairs / Idle Standby
- N+1 (or STAR)
- Pair+N

Further discussion on the topologies is covered later in the Discussion section.

Logical Hostname/IP Failover

A logical hostname, together with the IP address to which it maps, must be associated with a particular DB2 instance. Client programs will access the DB2 database instance using this logical hostname instead of the physical hostname of a server in the cluster. This logical hostname is the entry point to the cluster, and it shields the client program from addressing the physical servers directly. That is, this logical hostname/IP address is cataloged from the DB2 TCP/IP clients (using the catalog TCP/IP node DB2 command).

This logical hostname is configured as a logical hostname resource, and must be added to the same resource group as the instance resource. In the case of a failure, the entire resource group including the instance and the logical host name will be failed over to the backup server. This floating IP setup provides high availability DB2 service to client programs.

Ensure that this hostname maps to an IP address, and that this name-to-IP address mapping is configured on all nodes in the cluster, preferably in /etc/inet/hosts on each node. More information on configuration for public IP addresses can be found in the Sun Cluster 3.x Installation Guide.

Logical Hostname/IP Consideration for DB2 (single partition)

In the case of a single partition DB2 UDB instance the highly available logical hostname/IP must be collocated (that is, located within the same Sun Cluster 3.x resource group) with the DB2 UDB instance. This will guarantee that the logical hostname/IP address is always local to the DB2 instance. If they are not collocated, it is possible that a particular physical node will host the logical hostname/IP address, while a different physical node may host the DB2 instance itself.

If there are no TCP/IP clients, you may choose to assign no highly available hostname/IP address for a particular DB2 instance.

Logical Hostname/IP Consideration for DB2 (multiple partitions)

You may configure zero, one or N logical hostname resources for the use of a highly available multiple partition DB2 UDB instance.

One Logical Hostname

The DB2 UDB HA package will create one logical hostname resource for a particular DB2 Universal Database instance, and this logical hostname resource is added to the same resource group as the first partition in the instance (as defined by the first entry in the $INSTHOME/sqllib/db2nodes.cfg file). In this case, client programs will use the logical hostname to access this DB2 Universal Database ESE/EEE instance. Therefore, this partition will be the coordinator node (regardless of where that particular DB2 partition is physically hosted). This is the default installation behavior of the DB2 UDB HA package and is the most common configuration scenario.

Zero Logical Hostnames

You may choose to have no highly available logical hostname/IP addresses defined. The floating logical hostname resource can be removed from the DB2 UDB HA package using the following Sun Cluster 3.x administrative command.

```
scrgadm -r -j resource_name
```

Alternatively, you may choose not to initially configure a highly available logical hostname/IP address. However, there's more to this scenario.

DB2 Universal Database (multiple partition) is designed with symmetrical data access across partitions in the sense that client programs may access any partition node as an entry point to the instance and receive the same result sets from their queries, regardless of the coordinator node used to process the query. Thus, a DB2 Universal Database installation provides access redundancy (when the DB2 Universal Database instance exists initially on more than one physical node). Here, the client program can access the DB2 Universal Database instance using a round-robin approach through all available physical node names for the instance (or for a subset, provided the subset contains at least two distinct physical nodes). In the case of a failover, the multiple partition DB2 Universal Database instance can be accessed through any of the remaining healthy host names/IP addresses.

N Logical Hostnames

If the demands of the application require access to a particular DB2 Universal Database partition, a logical hostname can be associated with each partition. Using Sun Cluster 3.x administrative commands, each logical hostname resource can be added and grouped with its corresponding DB2 partition resource. Consequently, the logical hostname/IP address will fail over together with its associated DB2 Universal Database partition resource. Thus, connections to a particular logical hostname/IP address will always be associated with a connection to a particular DB2 Universal Database coordinator node.

Package Contents

With the package are supplied four scripts, or methods, that are used to control the registration, unregistration, onlining and offlining of DB2 Universal Database in a Sun Cluster 3.x environment. Note that while there are a number of other components in the DB2 UDB HA agent, only the four discussed here are called directly by an administrator. A man page is provided in the doc directory for each of the four supplied methods.

regdb2udb

This method will register appropriate resources and resource groups for a specified instance. Note that it will not attempt to bring online any resources. This will usually be the first script called, as it will perform all necessary steps to prepare DB2 Universal Database for Sun Cluster 3.x control.

unregdb2udb

This method will execute required Sun Cluster 3.x commands in order to remove DB2 UDB HA (including resources and groups registered for this instance) from the cluster. Essentially, this method is the inverse of regdb2udb, and will be generally called if the instance is no longer required to be highly available. This does not remove or otherwise uninstall the instance itself.

onlinedb2udb

This method will execute required Sun Cluster 3.x commands in order to bring a DB2 UDB HA instance online. It will not create any resources or resource groups.

offlinedb2udb

This method will execute required Sun Cluster 3.x commands in order to take a DB2 UDB HA instance offline. It will not remove any resources or groups from the Sun Cluster 3.x infrastructure.

DB2 UDB: Installation and Configuration in a Sun Cluster 3.x Environment

The installation of DB2 Universal Database in a Sun Cluster 3.x environment closely mirrors the installation that is documented in the relevant DB2 for UNIX® Quick Beginnings guide. The DB2 Product Technical Library (*http://www.ibm.com/software/data/db2/library/*) provides links to the manuals for DB2 Universal Database.

Installation checklist

To simplify the DB2 UDB HA installation in a Sun Cluster 3.x environment, the installation can be staged in the following seven phases. Each phase should be validated before moving onto the next phase.

1. Sun Cluster configuration (preferably with at least one global cluster file system configured to be used for instance home directory)
2. DB2 binary installation and post-installation configuration
3. DB2 instance creation (preferably on a cluster file system)
4. Manually start/stop the DB2 instance from either side of the cluster
5. Enable the DB2 instance for high availability
6. HAStoragePlus configuration for database storage devices as needed
7. Failover testing

The following steps highlight the sequence of events for a successful DB2 UDB HA implementation in a Sun Cluster 3.x environment. Detailed discussion and step-by-step examples for steps 3 to 7 are in the coming sections of this white paper. For steps 1 and 2, you should refer to Sun Cluster product Web site (*http://www.sun.com/software/cluster*) and the corresponding Sun Cluster documentations.

1. Prepare hardware:
   A. Physically configure hardware.
   B. Install Solaris OS on all nodes.
   C. Apply recommended and security patches.
   D. Apply specific Sun Cluster and DB2 patches if required.

2. Configure Sun Cluster - scinstall
3. Prepare system for DB2 install
   - Create cluster file system for DB2 instance home directory.
   - Create cluster file system for DB2 binaries if you have decided to place DB2 binaries globally. (Review the discussion on creating global mount points for /var/db2 & /var/lum.)
   - Create user/group entries for DB2 instance.

4. Install DB2 Binary (db2setup or db2_install) and any applicable FixPaks (instance creation is a separate task)
5. DB2 Instance creation
   A. /etc/system settings (db2osconf)
   B. .rhosts config (ESE or EEE only)
   C. /etc/services updates
   D. svcename updates

6. Enable DB2 instance for high availability
   A. Disable entries in /etc/inittab that autostart DB2 processes.

      In DB2 UDB V7:
      #db:234:once:/etc/rc.db2 > /dev/console 2>&1

      In DB2 UDB V8:
      #fmc:234:respawn:/opt/IBM/db2/V8.1/bin/db2fmcd

   B. Test to manually start/stop the DB2 instance from either side of the cluster.
   C. For DB2 V8 only, copy $INSTALLPATH/ha/sc30 directory to $INSTHOME/sqllib directory.
   D. Register the DB2 resource group (regdb2udb).
   E. Bring online the DB2 resource group (onlinedb2udb).
   F. Create HAStorage or HAStoragePlus resources for storage devices.

7. Cluster Failover Testing

Cluster File System Considerations

At this stage, it is presumed that Sun Cluster 3.x is configured according to the information given in the Sun Cluster 3.x installation guide. Note that at least one cluster file system must be defined and available, as it will be required in order to host the instance home directory for each soon-to-be installed DB2 instance.

In this document, we assume that the installation and configuration of DB2 are through the DB2 installer program, db2setup. Other methods are possible, but we believe the use of the db2setup installer is the most straightforward, and that is the method discussed in this document.

(Note: A sample installation process using "db2_install" is also illustrated later in this paper.)

You should review the Sun Cluster 3.x Data Services Installation and Configuration Guide chapter entitled Determining the Location of Application Binaries.

Briefly, there are two distinct DB2 binary location strategies: the binaries can exist either on a global file system or on each physical node of a local file system. We recommend that the DB2 installation binaries be placed on a global file system for the following reasons:

- Access to the binaries will be highly available
- Installation of binary images need only be performed once per set of cluster nodes, which share the global binary installation mount points. (Subsequent product installation and addition should be performed on the same server in order to keep the complete DB2 package installation information.)
- Installation of PTFs (fix packages) is only done once per set of cluster nodes, which share the global binary installation mount points. (DB2 FixPaks should be applied from the node on which the binary was originally installed, because the FixPak installation depends on the DB2 package installation information stored on the current host.)
- There is no chance of having mismatched binary code levels across the nodes of the cluster.
- New nodes have access to the DB2 binary.

There are certainly advantages to the installation of DB2 binaries on the local file system of each node. One of them is to protect the system from the rare event that the underlying HA disk experiences system failure. Another advantage is to reduce the maintenance windows for FixPak updates. The system can be running while the FixPak installation is taking place on the standby node. Then the primary can be failed over in a controlled fashion, and the instance will come up on the secondary node at the FixPak level that was installed there. The same FixPak could then be installed on this now idle node (formerly the primary node).

For DB2 Universal Database V8 ESE with DPF and V7 EEE, a cluster file system must be used to mount the instance home directory. For DB2 Universal Database EE (or for DB2 Universal Database EEE systems that are confined to a single physical node), we strongly recommend that the cluster file system be used as the instance home directory mount point; but it is not strictly required.. Should you decide to place the instance home directory on the FFS (Failover File System), HAStoragePlus resource would need to be defined to ensure the instance home directory is accessible from the cluster node on which the DB2 instance will be active.

User and Group Creation Considerations

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

- DB2 instance owner
- User who will execute fenced UDFs (user defined functions) or stored procedures
- Administration server

If NIS or NIS+ is in use, groups and users must be created on the NIS server prior to executing db2setup. If using local server authentication, repeat the following steps on all the nodes in the cluster:

1. Define the users and groups identically across all nodes in the cluster.
2. Ensure that the user ID, group ID, user home directory, account passwords and user shell are consistent across all nodes in the cluster.

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

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

Use the following Solaris commands to create user accounts:

```
# useradd -g db2iadm1 -u 1004 -d /global/home/db2inst1 -m -s /bin/ksh
  db2inst1
# useradd -g db2fadm1 -u 1003 -d /global/home/db2fenc1 -m –s /bin/
  ksh db2fenc1
# useradd -g db2asgrp -u 1002 -d /global/home/db2as -m -s /bin/ksh db2as
```

Note that for these commands, we presume that /global/home is the mount point of a cluster file system that will be utilized to host the DB2 UDB instance home directory.

Ensure that the account passwords are consistent across all nodes in the cluster.

Miscellaneous Considerations

At this point, you are advised to consult the appropriate DB2 Quick Beginnings for UNIX book. Note, however, that NFS has been effectively superseded by the CFS, so in a highly available DB2 Universal Database ESE with DPF or DB2 UDB EEE Sun Cluster 3.x environment, one would use the CFS instead of NFS to perform file system sharing among the various physical nodes in the cluster.

One other important procedure (unchanged from a typical DB2 UDB on Solaris installation) is to ensure that the correct kernel parameters have been applied at each physical node in the cluster.

Installation of DB2 Binary

The default DB2 binary installation path is /opt/IBMdb2/V7.1 for Version 7 and /opt/IBM/db2/V8.1 for Version 8. If the binaries will be installed on a global device, ensure that this mount point is on a global device. This can be accomplished by mounting this path directly, or providing a symbolic link from this path to a global mount point.

For example, for DB2 UDB Version 7, on one node, run:

```
# mkdir /global/scdg2/scdg2/opt/IBMdb2
```

For DB2 UDB Version 7 on every node, run:

```
# ln -s /global/scdg2/scdg2/opt/IBMdb2 /opt/IBMdb2
```

Ensure that the instance is not set to start on system reboot (using the db2iauto command); the instance start and stop should be under the control of the Sun Cluster infrastructure.

Additionally, the DB2 registry setting DB2SYSTEM should refer to the logical hostname rather than the physical hostname.

If the DB2 binary is installed on a global shared file system, you must also take steps to ensure that the license key is available in the case of failover. You can achieve this in either of these ways:

- Install the license key of each machine in the cluster using the db2licm tool.
- Mount the license key location as a global mount point. (This location, at the time of writing, is /var/lum.) Then install the license key on exactly one node in the cluster.

Configuration of a Particular DB2 Instance

The installation of the DB2 binaries and the creation of the specified DB2 instance have now been completed by the db2setup tool. Note that a particular DB2 instance is created only once. There is no need, nor is it even possible, to attempt to create an instance for each node of the cluster. If errors occur during installation, consult the log file specified by the db2setup to determine the cause of the errors.

For a DB2 Universal Database ESE and EEE instance configure the $INSTHOME/sqllib/db2nodes.cfg file, the $INSTHOME/.rhosts, and the /etc/services file in order to allow for communications between the various DB2 UDB partitions.

The /etc/services file reserves a range of ports required for DB2 UDB multiple partition communications. Ensure that the port range is sufficiently large to support all failover scenarios envisioned. For simplicity, we recommend that you configure the port range to be as large as the number of partitions in the instance. Configure the same port range for all nodes in the cluster.

For example, assume db2inst1 was an 8-partition ESE or EEE database instance. Place the following entries in the /etc/services files on each physical node of the cluster.

```
DB2_db2inst1 60000/tcp
DB2_db2inst1_END 60007/tcp
```

The entries that compose the db2nodes.cfg determine the logical-to-physical mapping of DB2 logical partition to the appropriate physical host. For each DB2 partition expecting to be subjected to significant disk activity, it is strongly recommended that each partition exist on a physical node with at least one locally mastered device group for database table spaces of that partition. The physical node should be the primary of the multihost HA device group, which can be used as raw devices or local failover file system with SUNW.HAStoragePlus, or the global cluster file system with SUNW.HAStorage or SUNW.HAStoragePlus. We will discuss the trade-offs between local failover file system (FFS) and the global cluster file system (CFS) in more detail later in this paper.

For DB2 UDB ESE/EEE, the cluster must be configured to give the user remote shell access from every node in the cluster to every other node in the cluster. Generally, this is accomplished through the creation of a .rhosts file in the instance home directory. (There are other methods, but this is the most common.) Remote shell commands should proceed unprompted, and the db2_all command should execute without error.

As instance owner, issue the following command:

% db2_all date

This should return the correct date and time from each node. If not, you'll have to troubleshoot and solve this problem before proceeding. This is also a good time to ensure that the system dates are reasonable, and are in sync.

For the instance in question (say, db2inst1), issue the following command (again as the instance owner):

% db2start

This should complete successfully, and for multiple partition DB2 UDB should work on all nodes. Failure likely indicates a configuration error. (In this case, review the *DB2 Universal Database Quick Beginnings Guide* and resolve the problem before proceeding.)

Next, attempt to stop the instance with the following command (again as the instance owner):

% db2stop

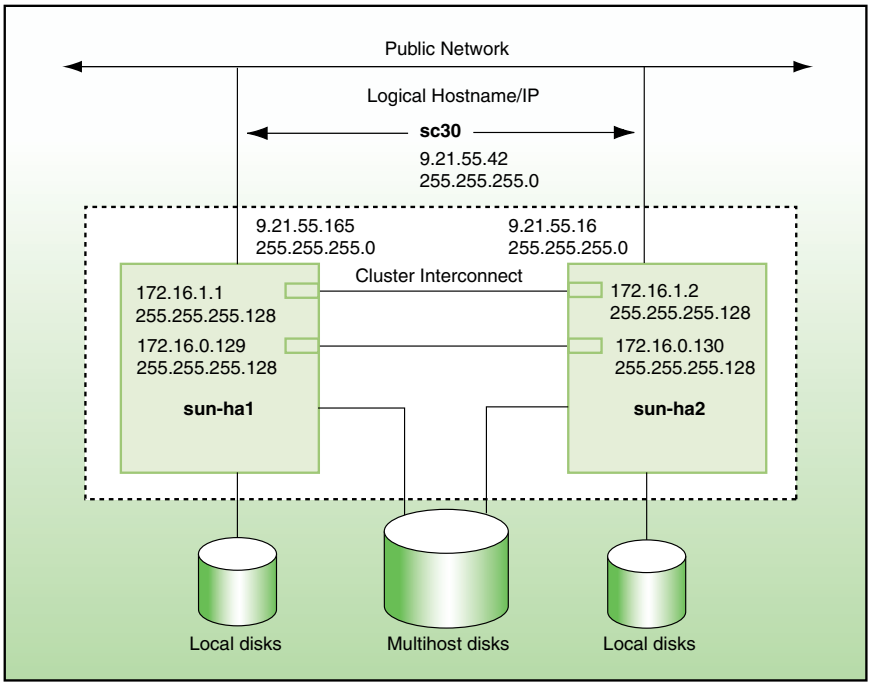
This should also complete successfully. If it does not, review the *DB2 Universal Database Quick Beginnings Guide* and resolve the errors before proceeding.

Once you have verified that the instance can be started and stopped, attempt to create the sample database with db2sampl (or an empty database, if you prefer). Create the database on the path of the global device that you plan to use for storage of the actual production database. When you're certain the create database command has completed successfully, remove the test or sample database using the drop database command. The instance is now ready to be made highly available.

Example 1: DB2 UDB V7 EEE Sun Cluster Implementation

In this example, we will create a highly available DB2 Universal Database EEE instance along with an associated highly available hostname/IP address. Our DB2 Universal Database EEE instance name is db2inst1 and the highly available hostname is sc30.

This is how our example cluster appears in diagrammatic form.



Before proceeding, ensure that:

- db2inst1 can be stopped and started successfully
- The home directory of db2inst1 resides on a global device accessible to all nodes in the cluster

Register the DB2 Instance (Checklist Step 6C)

Note: If registering a multi-partition EEE instance, make sure the content of db2nodes.cfg file reflect the logical-to-physical mapping of DB2 logical partition to the appropriate physical host. The DB2 UDB HA agent also supports the netname. The physical host and netname pairs are captured and stored during the registration process. During any failover, the db2nodes.cfg file will be reconfigured by the DB2 UDB HA agent automatically. Here is an example of the db2nodes.cfg file used for this example before the instance registration:

```
# cat ~db2inst1/sqllib/db2nodes.cfg
0    sun-ha1    0    172.16.194.6
1    sun-ha2    0    172.16.194.5
```

sun-ha1	Physical hostname of the machine 1
172.16.194.6	HA high speed interconnect interface on sun-ha 1
sun-ha2	Physical hostname of the machine 2
172.16.194.5	HA high speed interconnect interface on sun-ha1

First, use the regdb2udb utility to register the instance with the Sun Cluster 3.x infrastructure (we use the convention of boldface for input entered by the user):

```
# /opt/IBM/db2udb/V7.1/ha/sc30/util/regdb2udb -a db2inst1 -h sc30

About to register db2inst1 with Sun Cluster 3.x
Please hit any key to continue, break to terminate:
Adding IBM. db2udb as defined in file
/opt/IBMdb2/V7.1/ha/sc30/etc/IBM.db2udb...
db2inst1 appears to be a DB2 UNIVERSAL DATABASE EEE instance...
Processing partition 0...
Processing partition 1...
```

For Multi-partition DB2 ESE, update DB2_NUM_FAILOVER_NODES to the total number of logical partitions in the instance. For example, for a sixteen partition DB2 instance, enter the command as the instance owner:

```
% db2set DB2_NUM_FAILOVER_NODES=16
```

Use the **scstat** command to investigate the status of the cluster. We should see the necessary resources and resource groups registered.

```
# scstat -p
```

Cluster nodes

	Node Name	*Status*
Cluster node:	sun-ha1	Online
Cluster node:	sun-ha2	Online

Cluster transport paths

	Endpoint	*Endpoint*	*Status*
Transport path:	sun-ha1:hme3	sun-ha2:hme1	Path online
Transport path:	sun-ha1:hme2	sun-ha2:hme2	Path online

Quorum summary

Quorum votes possible: 3
Quorum votes needed: 2
Quorum votes present: 3

Quorum votes by node

	Node Name	*Present*	*Possible*	*Status*
Node votes:	sun-ha1	1	1	Online
Node votes:	sun-ha2	1	1	Online

Quorum votes by device

	Device Name	*Present*	*Possible*	*Status*	*Owner*
Device votes:	/dev/did/rdsk/d15s2	1	1	Online	sun-ha2 Device group servers

	Device Group	*Primary*	*Secondary*
Device group servers:	scdg2	sun-ha2	sun-ha1
Device group servers:	scdg3	sun-ha2	sun-ha1
Device group servers:	scdg4	sun-ha2	sun-ha1

Device group status

	Device Group	*Status*
Device group status:	scdg2	Online
Device group status:	scdg3	Online
Device group status:	scdg4	Online

Resource groups and resources

	Group Name	*Resources*
Resources:	db2_db2inst1_0-rg	sc30 db2_db2inst1_0-rs
Resources:	db2_db2inst1_1-rg	db2_db2inst1_1-rs

Resource groups

	Group Name	*Node Name*	*State*
Group:	db2_db2inst1_0-rg	sun-ha1	Offline
Group:	db2_db2inst1_0-rg	sun-ha2	Offline
Group:	db2_db2inst1_1-rg	sun-ha1	Offline
Group:	db2_db2inst1_1-rg	sun-ha2	Offline

Resources

	Resource Name	*Node Name*	*State*	*Status Message*
Resource:	sc30	sun-ha1	Offline	Offline
Resource:	sc30	sun-ha2	Offline	Offline
Resource:	db2_db2inst1_0-rs	sun-ha1	Offline	Offline
Resource:	db2_db2inst1_0-rs	sun-ha2	Offline	Offline
Resource:	db2_db2inst1_1-rs	sun-ha1	Offline	Offline
Resource:	db2_db2inst1_1-rs	sun-ha2	Offline	Offline

The result of the regdb2udb processing is that the appropriate DB2 resources and resource groups are created and registered with Sun Cluster 3.x.

Note the naming convention of the resources and resource groups and their structure. This instance that we have made highly available is a two partition DB2 Universal Database EEE instance. The partition numbers are 0 and 1 and the instance name is db2inst1. The naming system is rather mechanical, and is chosen to ensure name uniqueness regardless of the number of instances or partitions that are to be made highly available. For each partition, we can see that exactly one resource group is created. Within that resource group, there is exactly one resource for the partition of the DB2 instance resource. The highly available hostname/IP address (sc30) has been associated with partition 0 and added as a resource in the resource group db2_db2inst1_0-rg. The logical hostname/IP consideration for DB2 partitioned environment was discussed

earlier. Since the unit of a failover is a resource group, this allows for fine-grained control of the movement of each of the DB2 Universal Database EEE partitions independently across the physical nodes of the complex. One scenario would be to fail over multiple logical partitions of a failing node to multiple remaining running nodes in the cluster.

The naming convention is as follows:

- The string "db2_"
- Followed by the name of the instance
- Followed by the string "_"
- Followed by the partition number of the instance (note that for a EE instance, the partition number will be represented by 0)
- Followed by the string "-"
- Followed by the string "rs" to represent a resource, or the string "rg" to represent a resource group.

Note the one-to-one mapping of DB2 resources to DB2 resource groups (and of a particular DB2 instance's logical partition resources and the Sun Cluster 3.x resources).

In addition, there is one highly available hostname/IP address. This is the address that will be used by clients to catalog the databases in this instance. This hostname/IP address, if present, is always associated with the first DB2 resources group for the instance (first when reading top-to-bottom the db2nodes.cfg file) or for EE, is associated with the only DB2 resource group defined for that instance.

Note that if you want to enable multiple highly available DB2 instances, each highly available DB2 instance requires a unique highly available hostname/IP address.

Bring online the DB2 resource group (Checklist Step 6D)

Next, use the onlinedb2udb utility to bring online these registered resources. (We use the convention of boldface for input entered by the user. Note that some of the information returned by scstat has been edited for brevity.)

 # /opt/IBMdb2/V7.1/ha/sc30/util/onlinedb2udb -a db2inst1 -h sc30

db2inst1 appears to be a DB2 UNIVERSAL DATABASE EEE instance...
Processing partition 0...
Processing partition 1...

 # scstat -p

Cluster nodes

	Node name	*Status*
Cluster node:	sun-ha1	Online
Cluster node:	sun-ha2	Online

Cluster transport paths

	Endpoint	*Endpoint*	*Status*
Transport path:	sun-ha1:hme3	sun-ha2:hme1	Path online
Transport path:	sun-ha1:hme2	sun-ha2:hme2	Path online

Resource groups and resources

	Group Name	*Resources*
Resources:	db2_db2inst1_0-rg	sc30 db2_db2inst1_0-rs
Resources:	db2_db2inst1_1-rg	db2_db2inst1_1-rs

Resource groups

	Group Name	*Node Name*	*State*
Group:	db2_db2inst1_0-rg	sun-ha1	Online
Group:	db2_db2inst1_0-rg	sun-ha2	Offline
Group:	db2_db2inst1_1-rg	sun-ha1	Offline
Group:	db2_db2inst1_1-rg	sun-ha2	Online

Resources

	Resource Name	*Node Name*	*State*	*Status Message*
Resource:	sc30	sun-ha1	Online	Online
Resource:	sc30	sun-ha2	Offline	Offline
Resource:	db2_db2inst1_0-rs	sun-ha1	Online	Online
Resource:	db2_db2inst1_0-rs	sun-ha2	Offline	Offline
Resource:	db2_db2inst1_1-rs	sun-ha1	Offline	Offline
Resource:	db2_db2inst1_1-rs	sun-ha2	Online	Online

The result of the online processing is that the db2inst1 instance (and its associated highly available IP address) are online and under the control of Sun Cluster 3.x.

Because of this, you should see the resources brought online at the appropriate node (for example, on the physical hostname sun-hal, you should see that it hosts the highly available IP address for sc30, as well as the processes for the instance db2inst1, partition 0).

There are two more supplied scripts, which we will now discuss offlinedb2udb and unregdb2udb.

Typically, you may wish to take offline a DB2 instance in order to remove the DB2 resources from Sun Cluster 3.x control. For example, you may wish to take the database engine down for an extended period of time. Directly issuing the appropriate DB2 commands (for example, db2start, db2stop) will be ineffective, as Sun Cluster 3.x will interpret the absence of resources caused by the successful completion of the db2stop command as a failure and attempt to restart the appropriate database instance resources.

Instead, to accomplish this task, you must take offline the resources as follows.

```
# offlinedb2udb -a db2inst1 -h sc30

db2inst1 appears to be a DB2 UNIVERSAL DATABASE EEE instance...
Processing partition 0...
Processing partition 1...

# scstat -p
```

Cluster nodes

	Node name	*Status*
Cluster node:	sun-ha1	Online
Cluster node:	sun-ha2	Online

Cluster transport paths

	Endpoint	*Endpoint*	*Status*
Transport path:	sun-ha1:hme3	sun-ha2:hme1	Path online
Transport path:	sun-ha1:hme2	sun-ha2:hme2	Path online

Resource groups and resources

	Group Name	*Resources*
Resources:	db2_db2inst1_0-rg	sc30 db2_db2inst1_0-rs
Resources:	db2_db2inst1_1-rg	db2_db2inst1_1-rs

Resource groups

	Group Name	*Node Name*	*State*
Group:	db2_db2inst1_0-rg	sun-ha1	Online
Group:	db2_db2inst1_0-rg	sun-ha2	Offline
Group:	db2_db2inst1_1-rg	sun-ha1	Online
Group:	db2_db2inst1_1-rg	sun-ha2	Offline

Resources

	Resource Name	*Node Name*	*State*	*Status Message*
Resource:	sc30	sun-ha1	Offline	Offline
Resource:	sc30	sun-ha2	Offline	Offline
Resource:	db2_db2inst1_0-rs	sun-ha1	Offline	Online
Resource:	db2_db2inst1_0-rs	sun-ha2	Offline	Offline
Resource:	db2_db2inst1_1-rs	sun-ha1	Offline	Online
Resource:	db2_db2inst1_1-rs	sun-ha2	Offline	Offline

As you can see from the scstat output, all resources are now offline. No resources associated with the instance db2inst1 should be present on either node, nor will Sun Cluster 3.x take any action to protect this instance should it be brought online manually (that is, via db2start) and a failure occur.

Let's assume that you've decided to remove this instance permanently from Sun Cluster 3.x monitoring and control. For this task, you may use the unregdb2udb method. Note that this method merely interfaces with Sun Cluster 3.x in order to perform these tasks; the instance itself is neither dropped nor removed (user input in boldface below).

 # unregdb2udb -a db2inst1 -h sc30

Removing SC3.x resources and groups for db2inst1...

 # scstat -p

Cluster nodes

	Node name	***Status***
Cluster node:	sun-ha1	Online
Cluster node:	sun-ha2	Online

Cluster transport paths

	Endpoint	***Endpoint***	***Status***
Transport path:	sun-ha1:hme3	sun-ha2:hme1	Path online
Transport path:	sun-ha1:hme2	sun-ha2:hme2	Path online

Should there be any difficulty with the registration, unregistration, onlining or offlining processes, consult the system error log (typically in /var/adm/messages).

Advanced users trying to debug may also call the various start, stop and monitor methods directly and observe their behavior.

Example 2: DB2 UDB V8 Multi-partition ESE Sun Cluster Implementation

The example provides step-by-step instructions on how to install, create, and enable DB2 V8 64bit ESE multi-partition instance for high availability, called "db2mpi", in a sample two-node Sun Cluster 3.x environment. It also demonstrates how to work with HAStoragePlus to collocate CFS, FFS, and Raw device with a DB2 UDB HA instance. The hostname of the two physical boxes are "clust1" and "clust2". The "clustmp" is the HA-IP address for this multi-partition instance, "db2mpi".

Device Group, File System, and Raw Device

In this exercise, we utilized the built-in Solaris Volume Manager (SVM) with Solaris 9 to create disk sets and mirror volumes. We created disk sets cfs_*, ffs_*, and raw_* to demonstrate how DB2 works with Cluster File System, Failover File System (local file system on multihost HA device), and Raw device.

Here is the sample /etc/vfstab, common across all cluster nodes, for mounting the CFS and FFS file systems:

```
# DB2 global devices
/dev/md/home/dsk/d11 /dev/md/home/rdsk/d11 /global/home ufs 2 yes
global, logging
/dev/md/cfs_0/dsk/d10 /dev/md/cfs_0/rdsk/d10 /global/db2/cfs_0/d10 ufs
2 yes global, logging
/dev/md/cfs_1/dsk/d10 /dev/md/cfs_0/rdsk/d10 /global/db2/cfs_1/d10 ufs
2 yes global, logging

# DB2 ffs devices
/dev/md/ffs_0/dsk/d10 /dev/md/ffs_0/rdsk/d10 /db2/ffs_0/d10 ufs 2
no logging
/dev/md/ffs_1/dsk/d10 /dev/md/ffs_1/rdsk/d10 /db2/ffs_1/d10 ufs 2
no logging
```

We also created disk sets raw_0 and raw_1 with mirror volumes /dev/md/raw_0/rdsk/d10 and /dev/md/raw_1/rdsk/d10 to demonstrate using raw devices with HAStoragePlus.

DB2 UDB GA Binary Installation

The DB2 UDB binary will be installed locally on both cluster nodes (clust1 and clust2).

Note: The preferred method is to install DB2 binaries on CFS, as discussed previously.

As superuser, on each node, run "db2_install" and choose "DB2.ESE" to install the complete binary set. Take the default "/opt" as the base directory. The DB2 binary will be installed in "/opt/IBM/db2/V8.1".

 # db2_install

Specify one or more of the following keywords, separated by spaces, to install DB2 products.

Keyword	*Product Description*
DB2.ESE	DB2 Enterprise Server Edition for SUNOS
DB2.ADMCL	DB2 Administration Client for SUNOS
DB2.ADCL	DB2 Application Development Client for SUNOS

Enter "help" to redisplay product names.
Enter "quit" to exit.

****

DB2.ESE

Default directory for installation of products - /opt

****

Do you want to choose a different directory to install [yes/no]?
no
Directory for installation of products = /opt

****

Do you want to continue [yes/no]?
Yes

DB2 FixPak Installation

With DB2 binary installed locally on both cluster nodes, DB2 FixPak also needs to be installed on both cluster nodes.

Obtain the latest DB2 FixPak. As superuser, on each node, install the FixPak according to the installation instructions.

Update Solaris Kernel Parameters (Checklist Step 5A)

The Solaris kernel parameters on both cluster nodes (clust1 and clust2) need to be updated.

As superuser, on each node, run "db2osconf" utility to get the recommended DB2 kernel parameters. Then update the "/etc/system" with the appropriate entries. (You can cut and paste the "set parm=value" output from "db2osconf" to "/etc/system".) Reboot the system for changes to take effect.

The following is an example of db2osconf output:

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

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

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

The following command reboots the system:

 # shutdown -y -g0 -i6

Globalize /var/db2 and /var/lum (Checklist Step 3B):

Some profile registry values and environment variables are stored in the files in "/var/db2", and the DB2 license key is kept in "/var/lum". To keep information synchronized on both cluster nodes, "/var/db2" and "/var/lum" can be placed on the cluster file system.

When running DB2 UDB V8 ESE, each physical node has to have its own copy of /var/db2/global.reg to keep track its registry information. Therefore, the file /var/db2/global.reg has to be placed on the local file system even if the /var/db2 is shared among physical nodes. This file can be placed in /opt/IBM/db2/V8.1/ha/sc30 directory on each of the physical cluster nodes, where /opt/IBM/db2/V8.1/ha/sc30 is on the local file system on each of the physical cluster nodes.

Assuming "/global/home" is the mount point for a global mounted cluster file system, execute the following command as superuser on exactly one node in the cluster, for example: clust1.

```
# mkdir -p /global/home/var
# mv /var/db2 /global/home/var/.
# mkdir -p /global/home/var/lum
```

note: /var/lum does not exist before adding license file with "db2licm". If /var/lum exists, use the following command instead:

```
# mv /var/lum /global/home/var
```

Create a symbolic link from both cluster nodes. As superuser, execute the following commands on each node:

```
# rm -rf /var/db2
# ln -s /global/home/var/db2 /var/db2
# rm -rf /var/lum
# ln -s /global/home/var/lum /var/lum
```

Create a symbolic link from "/global/home" to point the "global.reg" back to a local file system. Execute this only once as superuser from "/global/home" directory.

```
# cd /global/home/var/db2
# ln -s /opt/IBM/db2/V8.1/ha/sc30/global.reg global.reg
```

Update the DB2 Product License Key

With "/var/lum" on the cluster file system, as superuser, execute the following command on exactly one node in the cluster to update the DB2 product license key.

```
# /opt/IBM/db2/V8.1/adm/db2licm -a $DBSW/db2/license/db2dlm.lic
```

Create DB2 Instance (Checklist Step 5)

Creating group and user IDs for a DB2 operation

As superuser, execute the following commands on each node.

```
# groupadd -g 101011 db2adm
# groupadd -g 101012 db2as
# groupadd -g 101013 db2udf
# useradd -u 145811 -g db2as -G db2adm -d /global/home/db2as –m \
      -s /bin/ksh db2as
# useradd -u 145814 -g db2adm -G db2as -d /global/home/db2mpi –m \
      -s /bin/ksh db2mpi
# useradd -u 145813 -g db2udf -d /global/home/db2udf -m \
      -s /bin/ksh db2udf
```

As superuser, set the password for each user. Make sure to use the same password for the same user ID on all cluster nodes.

Create the DB2 Instance

As superuser, on exactly one node, execute the following command to create a 64-bit DB2 ESE instance.

```
#/opt/IBM/db2/V8.1/instance/db2icrt -w 64 -u db2udf db2mpi
Sun Microsystems Inc. SunOS 5.9  Generic May 2002
DBI1070I Program db2icrt completed successfully.
```

Update "svcename" and "/etc/services" for DB2 instance

As superuser, su to DB2 instance owner and update the database manager port definition for client connections.

```
# su - db2mpi -c "db2 update dbm cfg using svcename db2mpi"
```

Update the /etc/services on both cluster nodes to reflect the port definition for client connections and the reserved FCM ports.

```
# DB2 Ports
db2mpi 50l00/tcp
DB2_db2mpi 60004/tcp
DB2_db2mpi_1 60005/tcp
DB2_db2mpi_2 60006/tcp
DB2_db2mpi_END 60007/tcp
```

.rhosts file for instance owner

Logon to one of the cluster nodes as the instance owner, and create .rhosts file in the instance owner's home directory with the following entries:

```
# cat .rhosts
clust1
clust2
clustmp
```

Enable the DB2 instance for high availability (Checklist Step 6)

The section provides instructions to enable the DB2 instance for high availability: db2mpi

1. Copy $INSTALLPATH/ha/sc30 to the instance home directory under ~db2mpi/sqllib. As superuser, on exactly one node (clust1), execute the following command:

   **# cd /opt/IBM/db2/V8.1root@clust1:/opt/IBM/db2/V8.1# tar -cvf - ha/sc30 **
   | (cd ~db2mpi/sqllib; tar -xvf -)

2. Register the instance (db2mpi) with the Sun Cluster and associate the HA-IP address (clustmp). As superuser, on exactly one node (clust1), execute the following command:

   # ./regdb2udb -a db2mpi -h clustmp

If registering a multi-partition EEE instance, make sure the content of db2nodes.cfg file reflect the logical-to-physical mapping of DB2 logical partition to the appropriate physical host. The DB2 UDB HA agent also supports the netname. The physical host and netname pairs are captured and stored during the registration process. During any failover, the db2nodes.cfg file will be reconfigured by the DB2 UDB HA agent automatically. Here is an example of the db2nodes.cfg file used for this example before the instance registration:

 # cat ~db2mpi/sqllib/db2nodes.cfg

```
0    clust1    0    sw1
1    clust2    0    sw2
```

clust1 *Physical hostname of the machine 1*
sw1 *HA high speed interconnect interface on clust1*

clust2 *Physical hostname of the machine 2*
sw2 *HA high speed interconnect interface on clust2*

If the instance is registered with SC commands instead of regdb2udb or if the user wants to override the registered netname, the following steps can be followed:

For Multi-partition DB2 ESE, record the netname value for each physical host. For example, assume the cluster consists of two physical hosts named clust1 and clust2, with high speed interconnects named sw1 and sw2, respectively.

Then on node clust1 issue the command:

 # cd /opt/IBM/db2/V8.1/bin
 # ./db2greg -addvarrec variable=switchname, value=sw1

Then on node clust2 issue the command:

 # cd /opt/IBM/db2/V8.1/bin
 # ./db2greg -addvarrec variable=switchname, value=sw2

3. Bring the instance (db2mpi) online.

 # ./onlinedb2udb -a db2mpi -h clustmp

You could experience errors if steps of /etc/hosts.equiv and /etc/services were not updated on both sides of the clusters. If this happens, complete those steps first and try again.

4. Verify resource group status using "scstat -g".

 # scstat -g

Resource Groups and Resources

	Group Name	*Resources*
Resources:	db2_db2inst1_0-rg	clust db2_db2inst1_0-rs
Resources:	db2_db2mpi_0-rg	clustmp db2_db2mpi_0-rs
Resources:	db2_db2mpi_1-rg	db2_db2mpi_1-rs

Resource Groups

	Group Name	*Node Name*	*State*
Group:	db2_db2inst1_0-rg	clust1	Online
Group:	db2_db2inst1_0-rg	clust2	Offline
Group:	db2_db2mpi_0-rg	clust1	Online
Group:	db2_db2mpi_0-rg	clust2	Offline
Group:	db2_db2mpi_1-rg	clust1	Offline
Group:	db2_db2mpi_1-rg	clust2	Online

Resources

	Resource Name	*Node Name*	*State*	*Status Message*
Resource:	clust	clust1	Online	Online - LogicalHostname online.
Resource:	clust	clust2	Offline	Offline - LogicalHostname offline.
Resource:	db2_db2inst1_0-rs	clust1	Online	Online
Resource:	db2_db2inst1_0-rs	clust2	Offline	Offline
Resource:	clustmp	clust1	Online	Online - LogicalHostname online.
Resource:	clustmp	clust2	Offline	Offline
Resource:	db2_db2mpi_0-rs	clust1	Online	Online
Resource:	db2_db2mpi_0-rs	clust2	Offline	Offline
Resource:	db2_db2mpi_1-rs	clust1	Offline	Offline
Resource:	db2_db2mpi_1-rs	clust2	Online	Online

5. Use scswitch to bring the resource group to the desired node.
   Depending on the contents of the $INSTHOME/sqllib/db2nodes.cfg file, the partitions of db2mpi were brought up onto various nodes. In the above example, partition 0 is running on clust1, and partition 1 is running on clust2. Now you can use "scswitch" to move the partition around, for example, to move partition 0 to clust2:

   # scswitch -z -g db2_db2mpi_0-rg -h clust2

Enable HAStorage+ for DB2 Instance (Checklist Step 6E)

In this example, using the built-in SVM (Solaris Volume Manager) of Solaris 9, the following disk sets (home, cfs_0, cfs_1, ffs_0, ffs_1, raw_0, and raw_1) were created with multihost storage devices. Disks from the two A5200 were configured as RAID1.

Diskset	RAID	SIZE	Disks (did)
home	RAID1	18G	d20/d42
cfs_0	RAID1	18G	d6/d28
ffs_0	RAID1	18G	d7/d29
raw_0	RAID1	18G	d8/d30
cfs_1	RAID1	18G	d10/d32
ffs_1	RAID1	18G	d11/d33
raw_1	RAID1	18G	d12/d34

The file systems are created on the disk sets, home, cfs_0/cfs_1, and ffs_0/ffs_1 utilizing the soft partition feature of SVM.

The following entries are extracted from the /etc/vfstab of this example. Before starting, make sure /etc/vfstab are updated on both cluster nodes (clust1 and clust2) with these entries.

```
# home directory cfs mount points
/dev/md/home/dsk/d11 /dev/md/home/rdsk/d11 /global/home ufs 2 yes
global, logging
```

```
# DB2 cfs global devices
/dev/md/cfs_0/dsk/d10 /dev/md/cfs_0/rdsk/d10 /global/db2/cfs_0/d10 ufs
2 yes global, logging
/dev/md/cfs_1/dsk/d10 /dev/md/cfs_0/rdsk/d10 /global/db2/cfs_1/d10 ufs
2 yes global, logging

# DB2 ffs devices
/dev/md/ffs_0/dsk/d10 /dev/md/ffs_0/rdsk/d10 /db2/ffs_0/d10 ufs 2 no
logging
/dev/md/ffs_1/dsk/d10 /dev/md/ffs_1/rdsk/d10 /db2/ffs_1/d10 ufs 2 no
logging
```

As superuser, on clust1, execute the following commands:

1. Take the instance offline if it is running.

   # /opt/IBM/db2/V8.1/ha/sc30/util/offlinedb2udb -a db2mpi

2. Register HAStoragePlus resource types.

   # scrgadm -a -t SUNW.HAStoragePlus

   (Note: If the "SUNW.HAStoragePlus" has already been registered, ignore the warning message: HAStoragePlus: resource type exists; cannot create.)

3. Add HAStoragePlus resource to the DB2 instance resource group.

   ```
   # scrgadm -a -j db2mpi_0-hasp-rs \
                -g db2_db2mpi_0-rg \
                -t SUNW.HAStoragePlus \
                -x FilesystemMountPoints=/global/db2/cfs_0/d10, /
                db2/ffs_0/d10 \
                -x GlobalDevicePaths=raw_0 \
                -x AffinityOn=true
   ```

This command adds an HAStoragePlus resource (db2mpi_0-hasp-rs) to the resource group (db2_db2mpi_0-rg), and will make the node hosting the resource group (db2_db2mpi_0-rg) the primary for the cluster file system (/global/db2/cfs_0/d10), the failover file system (/db2/ffs_0/d10), and disk set raw device (raw_0).

Note: all parameters for "-x FilesytemMountPoints" must be on the same line without white space; otherwise, you may get a "usage error."

Note: "-x FilesytemMountPoints" works with both CFS and FFS; "-x GlobalDevicePaths" is for raw devices.

```
# scrgadm -a -j db2mpi_1-hasp-rs \
                -g db2_db2mpi_1-rg \
                -t SUNW.HAStoragePlus \
                -x FilesystemMountPoints=/global/db2/cfs_1/d10, /
                      db2/ffs_1/d10 \
                -x GlobalDevicePaths=raw_1 \
                -x AffinityOn=true
```

This command does the equivalent for resource group (db2_db2mpi_1-rg).

4. Modify the DB2 instance resource (db2_db2tst_0-rs) to add "Resource_dependencies".

```
# scrgadm -c -j db2_db2mpi_0-rs -y Resource_dependencies=db2mpi_
                0-hasp-rs
# scrgadm -c -j db2_db2mpi_1-rs -y Resource_dependencies=db2mpi_
                1-hasp-rs
```

5. Bring resource groups online with scswitch:

```
# scswitch -Z -g db2_db2mpi_0-rg
# scswitch -Z -g db2_db2mpi_1-rg
```

Note: For Step 5, because "onlinedb2udb" can't predict what name the customer will assign to the newly created HAStoragePlus resource, you could see the following error message when using the out of box "onlinedb2udb" to bring the instance online:

scswitch: Cannot enable resource db2_db2mpi_0-rs because it depends on one or more disabled resources.

You can either use "scswitch" as shown in Step 5 or modify the "onlinedb2udb" by adding appropriate commands similar to those below to start the dependent HAStoragePlus resources first before bringing the DB2 instance resource online:

scswitch -e -j db2_db2mpi_0-hasp-rs
scswitch -e -j db2_db2mpi_i-hasp-rs

6. Verify resource group and device group status using "scstat":
   The following sample output shows partition 0 of db2mpi, its associated storage devices are collocated on "clust1", while partition 1 of db2mpi, and its associated storage devices are collocated on "clust2".

Device Group Servers

	Device Group	*Primary*	*Secondary*
Device group servers:	cfs_0	clust1	clust2
Device group servers:	ffs_0	clust1	clust2
Device group servers:	raw_0	clust1	clust2
Device group servers:	cfs_1	clust2	clust1
Device group servers:	ffs_1	clust2	clust1
Device group servers:	raw_1	clust2	clust1

Device Group Status

	Device Group	*Status*
Device group status:	cfs_0	Online
Device group status:	ffs_0	Online
Device group status:	raw_0	Online
Device group status:	cfs_1	Online
Device group status:	ffs_1	Online
Device group status:	raw_1	Online

Resource Groups and Resources

	Group Name	*Resources*
Resources:	db2_db2mpi_0-rg	clustmp db2_db2mpi_0-rs db2mpi_0-hasp-rs
Resources:	db2_db2mpi_1-rg	db2_db2mpi_1-rs db2mpi_1-hasp-rs

Resource Groups

	Group Name	*Node Name*	*State*
Group:	db2_db2mpi_0-rg	clust1	Online
Group:	db2_db2mpi_0-rg	clust2	Offline
Group:	db2_db2mpi_1-rg	clust1	Offline
Group:	db2_db2mpi_1-rg	clust2	Online

Resources

	Resource Name	*Node Name*	*State*	*Status Message*
Resource:	clustmp	clust1	Online	Online - LogicalHostname online.
Resource:	clustmp	clust2	Offline	Offline -LogicalHostname offline.
Resource:	db2_db2mpi_0-rs	clust1	Online	Online
Resource:	db2_db2mpi_0-rs	clust2	Offline	Offline
Resource:	db2mpi_0-hasp-rs	clust1	Online	Online
Resource:	db2mpi_0-hasp-rs	clust2	Offline	Offline
Resource:	db2_db2mpi_1-rs	clust1	Offline	Offline
Resource:	db2_db2mpi_1-rs	clust2	Online	Online
Resource:	db2mpi_1-hasp-rs	clust1	Offline	Offline
Resource:	db2mpi_1-hasp-rs	clust2	Online	Online

Cluster verification testing

Testing is an important aspect of a highly available cluster. The purpose of testing is to gain some confidence that the highly available cluster will function as envisioned for various failure scenarios. What follows is a set of minimum recommended scenarios for cluster testing and verification. These tests should be run regularly to ensure that the cluster continues to function as expected. Timing will vary depending on production schedules, the degree to which the cluster state evolves over time, and management diligence.

Cluster verification testing: test 1

In this test, we're performing Sun Cluster 3.x management commands to ensure that the db2inst1 instance can be controlled correctly.

First, verify that the instance is accessible from the clients (or locally), and that various database commands complete successfully (for example, create database).

Take the db2_db2inst1_0-rs resource offline, and take the sc30 resource offline using the following commands:

```
#scswitch -n -j db2_db2inst1_0-rs
#scswitch -n -j sc30
```

Observe that the DB2 instance resources no longer exist on any node in the cluster, and the highly available hostname sc30 is likewise inaccessible. From the perspective of a client of this instance, the existing DB2 connections are closed, and new ones start up, pending the bringing online of the appropriate DB2 and IP resources.

Cluster verification testing: test 2

To return the resources to their previous states, bring them online with the following Sun Cluster commands:

```
#scswitch -e -j db2_db2inst1_0-rs
#scswitch -e -j sc30
```

DB2 clients waiting in the previous test mode will be able to connect and resubmit transactions to pick up from the last failure. The client program must send retries to accomplish this.

Cluster verification testing: test 3

Test the failover of the DB2 instance and associated resources from sun-ha1 onto sun-ha2. At this point, the cluster is again at its initial state (that is, have the db2_db2inst1_0-rg hosted on sun-ha1, while db2_db2inst1_1-rg hosted on sun-ha2).

At this point the db2nodes.cfg file looks like:

```
0    sun-ha1
1    sun-ha2
```

or with netname:

```
0    sun-ha1    0    172.16.194.6
1    sun-ha2    0    172.16.194.5
```

Then move the containing resource group over to the secondary node using the following Sun Cluster 3.x command:

```
# scswitch -z -g db2_db2inst1_0-rg -h sun-ha2
```

You should now see the DB2 resource for db2inst1 and the associated hostname/IP address hosted by the secondary machine, sun-ha2. Verify this by executing the scstat -p command.

At this point the db2nodes.cfg file looks like:

```
0    sun-ha2
1    sun-ha2
```

or with netname:

```
0    sun-ha2    1    172.16.194.5
1    sun-ha2    0    172.16.194.5
```

Cluster verification testing: test 4

Here, test the failover capabilities of the Sun Cluster 3.x software itself. Bring the resources back into their initial state (that is, have the db2_db2inst1_0-rg hosted on sun-ha1).

Once the instance and its associated resources are hosted on the sun-ha1 machine, perform a power-off operation on that physical node. This will cause the internal heartbeat mechanism to detect a physical node failure, and the DB2 resources will be restarted on the surviving node.

Verify that the results are identical to those seen in Test 3 (that is, the DB2 resources should be hosted on sun-ha2 and the clients should behave similarly in both cases).

Cluster verification testing: test 5

Bring the cluster back into its initial state.

In this test, verify that the software monitoring is working as expected. To perform this test, you may issue commands as follows:

```
ps -ef | grep db2sysc
kill -9 <pid>

or

ps -ef | grep db2tcpcm
kill -9 <pid>
```

The Sun Cluster 3.x monitor should detect that a required process is not running and attempt to restart the instance on the same node. Verify that this in fact does occur. The client connections should experience a brief delay in service while the restart process continues.

Note that there are a large number of distinct testing scenarios, limited only by your resources and imagination. Those discussed here are the minimum that you should run to test the correct functioning of the cluster.

Discussion

Cluster topology

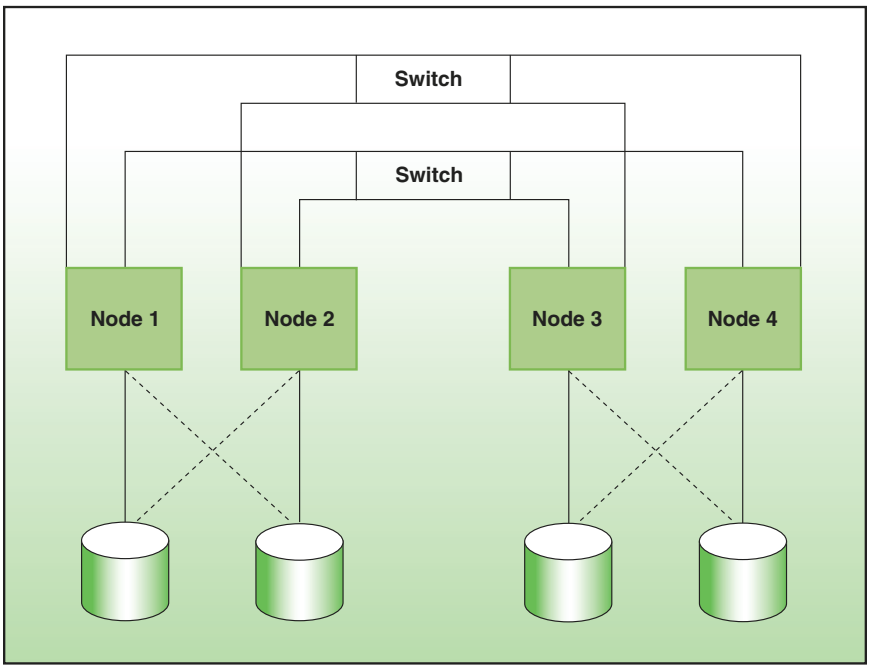
The DB2 UDB HA agent fully supports all three cluster topology classes supported by Sun Cluster 3.x:

- Clustered Pairs / Idle Standby
- N+1 (or STAR)
- Pair+N.

Idle standby

Idle standby is the simplest highly available cluster topology. In this scenario, the primary machine is hosting the production database instance and associated resources. A second idle machine is available to host the production database instance and associated resources in case a failure should occur on the primary machine. This is commonly used for DB2 EE instances. The second machine can also be running a workload (perhaps another DB2 instance) in order to maximize resource use.

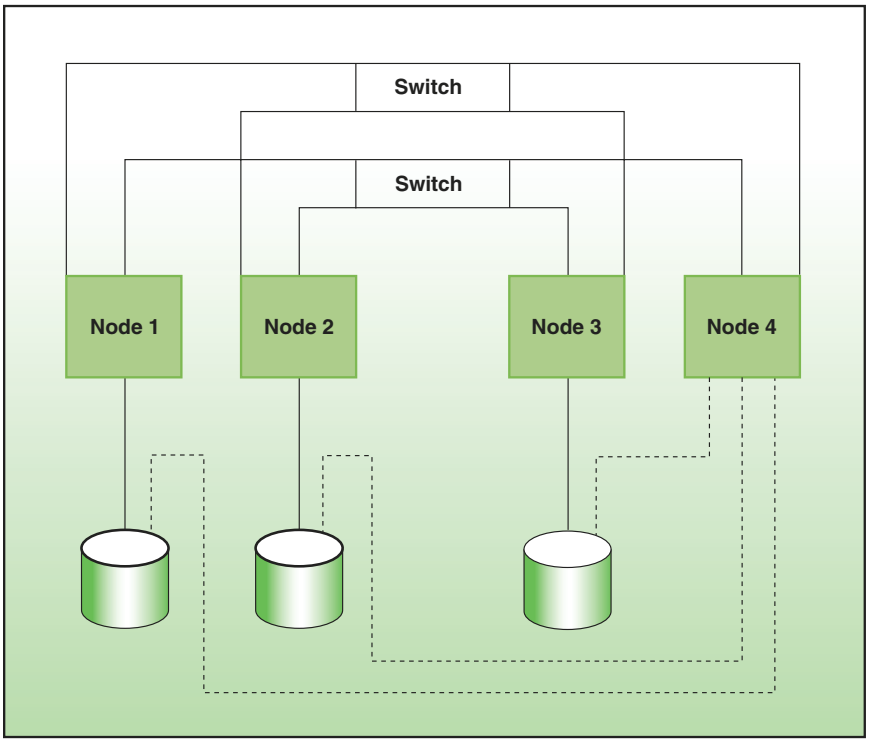
Clustered pairs



In the mutual takeover case, envision a cluster of N nodes as N/2 pairs of nodes. Node number n is responsible for failover support of node number n+1 (and vice versa), node number n+2 is responsible for failover support of node number n+3 (and vice versa), and so on until you reach the Nth node. Note that this scenario requires that N be an even number.

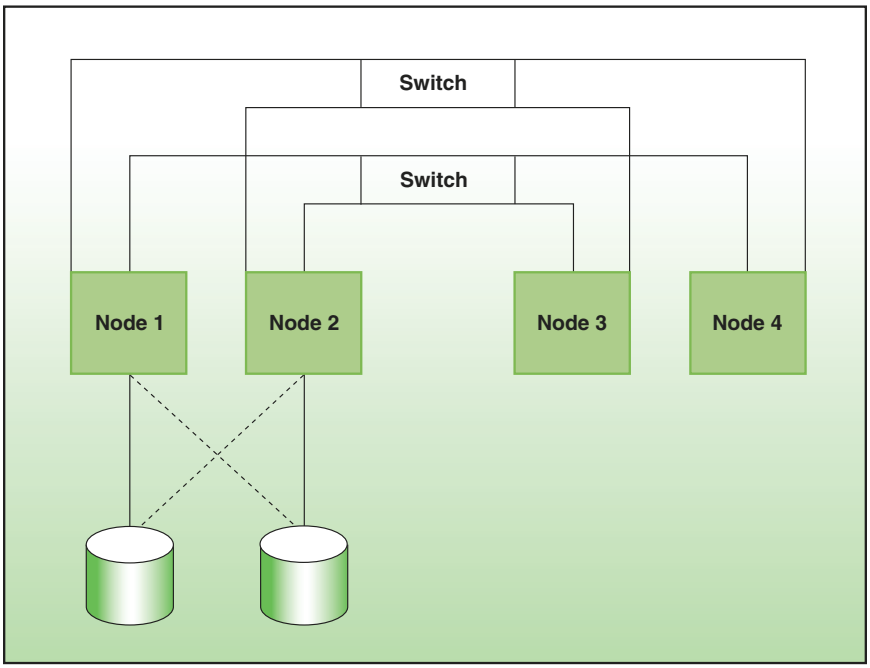
The advantage of this configuration is that in the normal (non-failure) case, all machines are hosting database resources and are performing productive work. The primary disadvantage is that in the failure case (the period after one of the hardware resources has failed and before its repair), there is one node that is required to support, on average, twice the workload of any other physical node. However, if failures are relatively rare and their duration short, this is a reasonable scenario, especially for DB2 Universal Database EEE configurations.

N+1 takeover



This case relies on an N-node cluster, with one defined node as the standby for all N nodes. The advantage of this scenario is that there is no performance degradation during the failure time (the time period after one of the hardware resources has failed and before its repair). The primary disadvantage is that approximately 1/ (N+1) of the aggregate physical computing resource goes unused during the normal operation.

Pair + N



This case relies on a pair+N cluster nodes, where N equals two in this example. Essentially this is the default cluster topology configured by the regdb2udb script. The number of physical nodes in the cluster is a pair+N, where N can be 0 to 6 and is limited by the number of nodes supported by the Sun Cluster 3.x. The prime advantage of this configuration is that the environment is fully redundant; up to N +1 node failures (given no double

failure in the pair) can be tolerated while still maintaining full database access (subject, of course, to increased query response times due to capacity constraints when there are fewer than N nodes in the cluster). In this way, DB2 used in conjunction with Sun Cluster 3.x ensures full database software redundancy and is most appropriate for environments requiring the highest degree of availability. The primary disadvantage of this configuration is potentially non-local access to data on the global device; this will be discussed later in the paper.

Customizing Sun Cluster 3.x resource properties for DB2 Universal Database

The DB2 UDB HA agent is registered with Sun Cluster 3.x as a defined resource type. Each DB2 Universal Database EE instance or DB2 Universal Database EEE partition (for a particular instance) is registered with Sun Cluster 3.x as a resource. For resource properties provided by Sun Cluster 3.x, you can change specific attributes using the Sun Cluster administrative commands.

The resource type properties and the resource properties are declared in the Resource Type Registration (RTR) file. The resource property declarations follow the resource type declarations in the RTR file. Entries begin with an open curly bracket and end with a closed curly bracket.

You can change resource properties by registering the resource with a modified RTR file:

/opt/IBMdb2/V7.1/ha/sc30/etc/IBM.db2udb

The Sun Cluster 3.x administrative command scrgadm can also be used to modify an existing configuration or add a new one.

Start_timeout, Stop_timeout

The <method>_timeout properties set the value in seconds after which the Sun Cluster 3.x Resource Group Manager (RGM) concludes invocation of the method has failed. The MIN value for all method timeouts is set to 60 seconds. This prevents administrators from setting shorter timeouts, which do not improve switchover/failover performance, and can lead to undesired RGM actions (false failovers, node reboot, or moving the resource group to ERROR_STOP_FAILED state, which requires operator intervention). Setting too-short method timeouts leads to a decrease in overall availability of the DB2 data service.

```
{
PROPERTY = Start_timeout;
MIN = 60;
DEFAULT = 600;
}
{
PROPERTY = Stop_timeout;
MIN = 60;
DEFAULT = 600;
}
```

Retry_Count, Retry_Interval

The Retry_Count and Retry_Interval properties define the number of retries (Retry_Count) to be done within the time interval (Retry_Interval) and before the Sun Cluster 3.x Resource Group Manager (RGM) concludes that the resource cannot be successfully started on the current node. The Retry_Interval needs to be set as a multiple of 60 since it is rounded up to minutes.

```
{
PROPERTY = Retry_Count;
MAX = 10;
DEFAULT = 10;
TUNABLE = ANYTIME;
}
{
PROPERTY = Retry_Interval;
MAX = 3600;
DEFAULT = 60;
TUNABLE = ANYTIME;
}
```

The property values can be added (-a) or changed (-c) using the Sun Cluster 3.x scrgadm command.

For a complete description of customizing the resource properties, refer to the Sun Cluster 3.x Data Services Installation and Configuration Guide (Sun Part No. 806-1421-10).

Optimizing Cluster File System (CFS) performance

Whenever possible, collocate the DB2 instance with the corresponding data and use locally hosted global file systems. If the global device is dual-hosted in the cluster, let the same node be the primary for the resource group and its corresponding device group, which will enable the active I/O path to support local access. During a failover, the backup node (secondary) will be promoted to the status of a primary for the resource and device group, which have been switched over.

We strongly recommend that the DB2 Universal Database instance (or, for DB2 Universal Database EEE, each logical partition) be collocated with its corresponding data in order to achieve the best performance in disk-intensive scenarios. This can be achieved using HAStorage or HAStoragePlus. This collocation is not significant for the instance home directory since it tends to be subjected to an extremely small I/O bandwidth by the database server.

CFS mount options

The recommended mount options for cluster file systems are as follows: global, logging, forcedirectio. A typical entry in the /etc/vfstab file would be:

/dev/vx/dsk/scdg2/vol05 /dev/vx/rdsk/scdg2/vol05 /global/scdg2/scdg2 ufs 2 yes global, logging, forcedirectio

The global mount option simply identifies this mount point as a Cluster File System (CFS). The logging option is always used implicitly for a CFS file system, whether or not it is directly specified on the mount line; but for clarity, we recommend that the option be explicitly stated.

The forcedirectio option essentially allows file system I/O to bypass the Solaris kernel page cache. This option is especially beneficial for DB2 Universal Database file system I/O, as DB2 performs all required I/O caching and derives no further benefit from additional caching in the Solaris kernel. We encourage its use, as it can reduce the CPU path length significantly, but it should be tested against the specific workload that the file system will support. Generally speaking, forcedirectio helps with large block consecutive I/O requests.

There is one additional CFS mount option that merits mention. The syncdir option only applies to global UFS mounts. Whether or not it is utilized depends on the relative importance placed on error semantics instead of on absolute performance. Briefly, the syncdir option is relevant in the case when a file is in the process of being extended. If it is set, notification of a "file system full" will be returned even if a failover of the primary occurs during the write call itself. This ideal error semantics behavior must be weighed against the performance degradation, which can be significant in the case of writes that occur on the non-primary node.

We recommend that you enable this option, in case all I/O should occur only local to the primary host. Otherwise, we recommend that you do not enable this option because of the performance penalty and the exceedingly small probability that a "file system full" will occur during a diskgroup failover.

To CFS or not to CFS

CFS (Cluster File System) is a highly available, distributed, cache-coherent file system that allows ufs, hsfs, and vxfs file systems to be concurrently accessed on multiple cluster nodes. It provides great convenience when working in a cluster environment. Using CFS can significantly reduce failover time. On the other hand, there may be some overhead cost with CFS. In areas where there is a large amount of I/O, such as table space container and logging devices, it may be worthwhile to investigate the use of FFS (Failover File System) for direct local access, especially for high-write bandwidth containers. In any case, it is always the best practice to utilize HAStoragePlus to collocate the primary of the disk device with the running processes of the resource group.

Since switching from CFS to FFS is as simple as changing the mount option (with or without "global") in addition to the HAStoragePlus implementation with FFS, it is acceptable to start with CFS for easy implementation, and to switch to FFS if there is observable I/O improvement.

We recommend the use of raw partitions for customers desiring the maximum performance possible with a global available device.

Failover time and how to minimize

To reduce the service interruption time, it is important to understand the discrete components that make up failover:

- Time for Sun Cluster 3.x to detect and respond to failure
- Time for DB2 Universal Database to recover from failure
- Time for TCP/IP clients to detect hostname switch over and reconnect.

In particular, you can influence (and reduce) the amount of time required for DB2 Universal Database failure recovery using several methods.

In the case of node failure in a DB2 Universal Database EEE environment, you reduce the failover interval through judicious use of the tcp_ip_abort_ cinterval parameter. This can minimize the time that the DB2 Universal Database EEE restart logic requires to detect a non-responsive node and transfer its partition resources to another node.

Here's an example of setting this parameter to allow this time window to be no longer than, for example, 10 seconds:

```
ndd -set /dev/tcp tcp_ip_abort_cinterval 10000
```

Also, you may reduce the time required for DB2 Universal Database restart recovery.

You may employ the standard techniques:

- Reducing the database configuration parameter (SOFTMAX)
- Increasing the number of configured page cleaners (NUM_IOCLEANERS)
- Decreasing the buffer pool threshold to trigger the page cleaners parameter (CHNGPGS_THRESH)
- Decreasing the log buffer size (LOGBUFSZ)
- Using applications that enable frequent commits.

While there are no specific configuration parameters required to allow highly available DB2 Universal Database on a Sun Cluster 3.x platform, we recommend that autorestart be enabled. This will allow the triggering of automatic restart recovery on an as-needed basis, and remove any requirement that the client applications perform special activities based on the state of the database.

Client issues

Applications that rely on a highly available DB2 instance must be able to reconnect in the event of a failover. Since the hostname and IP address of a logical host remain the same, there is no need to connect to a different hostname or catalog the database again.

Consider a cluster with two machines and one EE instance. The EE instance will normally reside on one of the machines in the cluster. Clients of the HA instance will connect to the logical IP address (or host name) of the logical host associated with the HA instance.

From the perspective of an HA client, there are two types of failovers. One type occurs if the machine hosting the HA instance crashes. The other type is when the HA instance is given an opportunity to shut down gracefully.

If a machine crashes and takes down the HA instance, both existing connections and new connections to the database will hang. The connections hang because there are no machines on the network that have the IP address that the clients were using for the database. If the database is shut down gracefully, a "db2stop force" forces off the connections to the database and then shuts down. In this case, existing connections to the database will receive an error because they were forced off.

During the failover, the logical IP address associated with the database is offline, either because the Sun Cluster 3.x software took it offline or because the machine hosting the logical host crashed. At this point, any new connections to the database will hang for a short period.

The logical IP address associated with the database is eventually brought up on another machine before DB2 is started. At this stage, a connection to the database will not hang, but will receive a communication error because DB2 has not yet been started on the system. DB2 clients still connected to the database will also begin receiving communication errors at this stage. Although the clients still believe they are connected, the new machine just started hosting the logical IP address and has no knowledge of any existing connections. The connections are simply reset and the DB2 client receives a communication error. After a short time, DB2 will be started on the machine and a connection to the database will succeed. At this point, the database may be inconsistent and the clients may have to wait for it to be recovered.

Even though this sounds like a complicated and time-consuming set of events, it is actually quite simple. When designing an application for an HA environment, it is not necessary to write special code for the stages where the database connections hang; the connections only hang for a short period of time while the Sun Cluster software moves the logical IP address. Any data service running on a Sun Cluster will experience the same hanging connections during this stage. No matter how the database comes down, the clients will receive an error and will simply need to try to reconnect until successful. From the client's perspective, it is as if the HA instance went down and was then brought back up on the same machine. In a controlled failover, it would appear to the client that it was forced off and that it could later reconnect to the database on the same machine. In an uncontrolled failover, it would appear to the client that the database server crashed, and was soon brought back up on the same machine.

The use of keep-alives

On the server side, using TCP keep-alives protects the server from wasting system resources for a down (or network-partitioned) client. If those resources are not cleaned up (in a server that stays up long enough), eventually the wasted resources grow unchecked as clients crash and reboot.

On the client side, using TCP keep-alives enables the client to be notified when a network address resource has failed over or switched over from one physical host to another. That transfer of the network address resource breaks the TCP connection. However, unless the client has enabled the keep-alive, it would not necessarily learn of the connection break if the connection happens to be quiescent at the time.

In the Solaris environment, the tcp_keepalive feature institutes a series of probe messages from the host to the client. The host sends a series of messages at specified intervals to the client. If no response returns, the host closes the channel. The default interval is 7.2 million milliseconds (2 hours), as well as in many other implementations. You can use ndd (1M) to change the value of tcp_keepalive_interval. In order for its value to be set every time the system boots, you can add an entry in /etc/init.d/inetinit script. The entry for setting tcp_keepalive_interval for one hour is as below.

```
# /usr/sbin/ndd -set /dev/tcp tcp_keepalive_interval 3600000
```

We recommend that the value of tcp_keepalive_interval not be set below 15 minutes.

In the Windows® NT® and Windows 2000 environments, the loss of client/server connection is also determined by using keep-alive probes. The following registry entries control keep-alive probe packet parameters on computers running on a Windows environment. Changing registry parameters affects all TCP/IP stream connections on the system.

KeepAliveInterval
KeepAliveTime
TcpMaxDataRetransmissions

The entries listed above can be found in the following registry location:

\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TCPIP\Parameters

Client retry

From the client's perspective, a failover appears to be a crash of the logical host followed by a fast reboot. In a controlled failover, it would appear to the client that it was forced off and that it could later reconnect to the database on the same machine. In an uncontrolled failover, it would appear to the client that the database server crashed, and was soon brought back up on the same machine.

During a failover, client applications are recommended to check the connection status (time out or communication errors as discussed earlier in this section), and to retry with new connections. Applications can also be implemented to notify the user that a long retry is in progress and allow him or her to choose whether to continue.

Recatalog DB2 Database on HA Logical Host

Since an HA-DB2 instance can float between either side of a cluster, to another in-cluster application that uses the same HA-DB2 instance as their DB2 client for local connection (for example, SAP, or MQSI), the HA-DB2 instance

may or may not be collocated with these "local" applications. Especially in a mutual take-over environment, DB2 and the in-cluster client application will be running on separate nodes most of the time.

To these in-cluster applications, the database directory can be either inaccessible if the DB2 instance is on FFS (Failover File System) or statically interpreted as a local database ("Directory entry type = Indirect") while the database is remotely running on the other node. A common error message would be:

"SQL1013N The database could not be found.".

There are two ways to resolve this issue. The bottom line is to catalog the DB2 UDB HA database as a remote database running on the HA logical host, and the in-cluster application needs to access the DB2 UDB HA database the same way as the remote clients outside of the cluster.

Solutions 1: Place the HA instance home directory on the CFS, and catalog the database on the HA logical node:

```
# su - db2inst1
% db2 catalog tcpip node hanode remote HAIP server 50000
% db2 catalog db sample as hasample at node hanode
```

Solutions 2: If the HA instance home directory is on the FFS, create the client instance on both side of the cluster and catalog the database on the HA logical node. Configure the in-cluster application to use this client instance as the DB2 client to access the HA instance.

```
% db2icrt -s client -u db2fenc1 clntinst
# su - clninst
% db2 catalog tcpip node hanode remote HAIP server 50000
% db2 catalog db sample as hasample at node hanode
```

For both of the above solutions, the in-cluster application will connect using "hasample" database alias.

Example to Support Mutual Takeover with SAP Central Instance

Following is an example to demonstrate Solution 1, and also still preserve the original database name as the database alias for SAP Central Instance to connect to DB2 UDB in a mutual takeover environment:

In order to maintain valid database directory information for SAP Central Instance to access <SID> (SID == TST in this example) database while they are running on or off the same physical node, the <SID> database needs to be recataloged onto DB2 UDB HA logical host.

1. Take SAP and DB2 resource group offline. (If SAP HA resource has not been created at this point, just make sure SAP is not running.)

   # scswitch -z -g sap-ci -h ""
   # scswitch -z -g db2_db2tst_0-rg -h ""

2. Verify on both nodes that you have added to your name service database all of the network resources that you use:

   ```
   # cat /etc/hosts
   127.0.0.1       localhost
   10.1.11.11      clust1      # cluster node 1
   10.1.11.12      clust2      # cluster node 2
   10.1.11.13      species     # SAP Application Server
   10.1.11.105     clustdb     # HA IP Address for HA DB2 instance
   10.1.11.106     clustci     # HA IP Address for SAP Central Instance
   ```

3. From either cluster node, logon as DB2 instance owner.

   # su - db2tst

   Sun Microsystems Inc. SunOS 5.9 Generic May 2002

4. Verify the existing database directory information.

   % db2 list db directory

   System Database Directory
   Number of entries in the directory = 1

   Database 1 entry:
   Database alias = TST
   Database name = TST
   Local database directory = /db2/TST
   Database release level = a.00
   Comment =
   Directory entry type = Indirect
   Catalog database partition number = 0

5. Recatalog the <SID> database (TST) onto the DB2 UDB HA logical host (clustdb).

   % db2 uncatalog db tst
   % db2 catalog db TST as TSTLOCAL on /db2/TST
   % db2 catalog tcpip node clustdb remote clustdb server sapdb2TST
   % db2 catalog db tstlocal as tst at node clustdb

   Make sure sapdb2TST matches the "svcename" dbm parameter for db2tst instance and it is resolved in /etc/services file on both sides of the cluster.

6. Verify the updated database directory information:

   % db2 list db directory

   System Database Directory
   Number of entries in the directory = 2

```
Database 1 entry:
Database alias                       = TSTLOCAL
Database name                        = TST
Local database directory             = /db2/TST
Database release level               = a.00
Comment                              =
Directory entry type                 = Indirect
Catalog database partition number    = 0

Database 2 entry:
Database alias                       = TST
Database name                        = TSTLOCAL
Node name                            = CLUSTDB
Database release level               = a.00
Comment                              =
Directory entry type                 = Remote
Catalog database partition number    = -1
```

DAS

For DB2 V7, the DAS is treated identically to a typical DB2 (single partition) instance. The shipped DB2 UDB agent is used. For DB2 V8, the fault monitor is responsible for maintaining a highly available DAS. For example, in the case of a software fault, the DAS will be restarted by the fault monitor demon as required.

Summary

We have discussed the design, implementation and verification of a highly available DB2 Universal Database cluster in a Sun Cluster 3.x environment. This combination produces an outstanding database management system with a high degree of availability, reliability and performance.

About the Authors

XIN CHEN
Systems Engineer, DB2 Ranger Team IBM Software Group

STEVE RASPUDIC
DB2 UDB Software Development, IBM Toronto Lab IBM Software Group

JULIE MCDONALD
Systems Engineer Sun Microsystems Inc

IBM Corporation
Silicon Valley Laboratory
555 Bailey Avenue
San Jose, CA 95141
U.S.A.

Produced in the United States of America
12-03