
Clustering with Redhat

Neil H. Watson

October 20, 2006

Abstract

High availability clusters use redundant servers to provide a service, such as a database, that will be available even if one of the servers fails. In addition, maintenance can be performed on a clustered service without making the service unavailable. Be aware that the service level of a cluster can vary greatly. What software and hardware is made available to the cluster will determine what service level expectations can be met.

This report presents an overview of how to build a DB2 service cluster. This procedure can be useful in learning how to setup clusters for other services.

Contents

1	Cluster Options	4
1.1	Redhat Cluster Software	4
1.1.1	Good	4
1.1.2	Bad	4
1.2	Load Balancer	4
1.2.1	Good	4
1.2.2	Bad	5
2	Installation	6
2.1	Install Redhat Linux AS 4 on primary node:	6
2.2	Install DB2 on primary node:	8
2.3	Prepare additional nodes	8
3	Configuring the Cluster Suite	9
3.1	Configuring the cluster	9
3.2	Additional service considerations	10
4	Testing the Cluster	11
4.1	Turn off the network	11
4.1.1	Expected results	11
4.1.2	Actual results	11
4.2	Unmount the shared partition	11
4.2.1	Expected results	11
4.2.2	Actual results	11
4.3	Stop DB2	12
4.3.1	Expected results	12
4.3.2	Actual results	12
4.4	Unplug a single network cable	12
4.4.1	Expected results	12
4.4.2	Actual results	12
4.5	Shutdown node	12
4.5.1	Expected results	12
4.5.2	Actual results	12
4.6	Unplug fiber cable	13
4.6.1	Expected results	13

4.6.2	Actual results	13
4.7	Turn off the network during a DB2 write	13
4.7.1	Expected results	13
4.7.2	Actual results	13
4.8	Unplug fiber cable during a DB2 write	13
4.8.1	Expected results	13
4.8.2	Actual results	13
5	Managing the cluster	14
5.1	Manual fail over	14
5.1.1	Fail over node	14
5.1.2	Remove node from cluster	14
5.1.3	Perform off line maintenance	15
5.1.4	Return node to cluster	15
5.2	Checking cluster status	15
5.2.1	X Windows GUI	15
5.2.2	Command line interface	15
5.3	Log files	16
5.3.1	Status check	16
5.3.2	Failover	16
6	Conclusions	17
A	References and Additional Reading	18
B	DB2 init script	19

Chapter 1

Cluster Options

After consideration it was decided to use the Redhat Cluster Suite.

1.1 Redhat Cluster Software

The Redhat Cluster Suite offers a prepackaged front-end to setup and manage high availability Linux clusters. Using this method, the cluster nodes share a single copy of the data. A 'fence' is used to ensure that only a single node at a time is using the data.

1.1.1 Good

- Software is already available at no extra cost.
- Redhat support is available at no extra cost.

1.1.2 Bad

- If the SAN fails the cluster fails.
- All nodes could fail if the shared data is corrupted.

1.2 Load Balancer

Each cluster node has its own copy of the data. The data is replicated either synchronously or asynchronously. Service requests are directed to the load balancer which decides which node to direct the request to.

1.2.1 Good

- Data redundancy.
- Speed could be improved if all nodes are active.

1.2.2 Bad

- Load balancing hardware will need to be purchased.
- If the SAN fails the cluster fails.

Chapter 2

Installation

The chief reference for this procedure is the Redhat Cluster Suite documentation provided by Redhat (see appendix). It should be read fully in order to understand this article and to be sure there are no syntax changes between revisions of this paper and Redhat's documentation.

2.1 Install Redhat Linux AS 4 on primary node:

1. In the `/etc/hosts` file make entries for each node in the cluster:

```
172.16.1.203    hadrian.example.com hadrian
172.16.1.204    caesar.example.com caesar
```

2. Bond two of the four Ethernet cards. Be sure that the network switch is set to auto negotiate. Forcing protocols can break bonding.
3. In `/etc/modprobe.conf` add these lines:

```
install bond0 /sbin/modprobe bonding -o bond0 mode=1 miimon=100
```

4. Edit `/etc/sysconfig/network-scripts/ifcfg-eth0` to:

```
DEVICE=eth0
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

5. Edit `/etc/sysconfig/network-scripts/ifcfg-eth1` to:

```
DEVICE=eth1
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

6. Create the file `/etc/sysconfig/network-scripts/ifcfg-bond0` to:

```
DEVICE=bond0
USERCTL=no
ONBOOT=yes
BROADCAST=176.16.1.255
NETWORK=172.16.1.0
NETMASK=255.255.255.0
GATEWAY=172.16.1.1
IPADDR=172.16.1.203
```

Activate the bonding by restarting the network service. Test for fail over.

7. Stop all unneeded services from starting at boot.

```
chkconfig --del <service>
```

8. Install the NTP service and point it to 209.50.68.2.
9. Install Redhat Cluster Suite by subscribing the system to the channel and running `up2date`.

```
up2date --installall rhel-x86_64-as-cluster
```

10. Create the shared partition on a SAN. Format it to ext3. Ensure that the server can mount the shared partition but do not allow the server to mount this partition automatically. Make `fstab` entry with the `noauto` option to prevent mount and boot time.
11. Install HP Insight management software suite.
12. Install Perl SSL module for iLO management
`up2date -i perl-Crypt-SSLeay`

2.2 Install DB2 on primary node:

1. Certain memory related kernel paramters must be set. Add these lines to `/etc/sysctl.conf`. (see Appendix A).

```
kernel.sem=250 256000 32 1024
#Example shmmx for a 64-bit system
kernel.shmmax=1073741824
#Example shmall for 90 percent of 16 GB memory
kernel.shmall=3774873
kernel.msgmax=65535
kernel.msgmnb=65535
```

2. Unpack required DB2 binaries. These may be different for each install.
3. Run `db2install` program.
4. Choose directory `/opt/ibm/db2/V9.1`.
5. Choose type 'ESE'.
6. Create group `db2fence`.
7. Create group `db2inst`.
8. Create user `db2fence` with `db2fence` group membership and with the shell `/sbin/nologin`.
9. Create user `dwapinst` with `db2inst` membership. Users will access this with `sudo` only.
10. Issue the command:
`db2icrt -a SERVER -p 30001 -s ese -u db2fence dwapinst.`
11. Disable fault monitor to prevent DB2 from spawning without cluster control: `/opt/ibm/db2/V9.1/bin/db2fmcu -d`.
12. Create an init script to stop and start the DB2 service (see Appendix B). Ensure that the service does not start at boot time.

2.3 Prepare additional nodes

Repeat the above steps on all participating cluster nodes. Be sure to change the IP address each time. One could also clone the nodes and then change the IP addresses and host names afterwards. The second node will have address 172.16.1.204.

Chapter 3

Configuring the Cluster Suite

3.1 Configuring the cluster

1. SSH to the primary node using the -YC switches for X forwarding.
2. Run the system-config-cluster application.
3. The application will prompt the user to create a new configuration.
4. Choose the dlm lock method.
5. Name the cluster 'dcdb'.
6. Create iLO fence devices for each node.
7. Add all nodes to the cluster. Configure the iLO fence for each node.
8. Add the shared file system to the resources section. Check the 'force unmount' and the 'reboot host node of unmount fails' boxes.
9. Add the cluster's floating IP (172.16.1.205) to the resources section. The floating IP is the IP address that is used to access the service that is being clustered. The Cluster Suite will assign this IP address to the active node.
10. Add the DB2 init script (see Appendix) to the resources section.
11. Add the DB2 service to the cluster and include all the shared resources above. Set the recovery policy to 'restart'. Check the 'autostart this service' box.
12. Propagate the cluster configuration by copying the file `/etc/cluster/cluster.conf` to each node.

3.2 Additional service considerations

A cluster may have additional services that are not clustered but, must be aware of the clustering environment. Cron is a good example. Suppose a cron job has to be run to help maintain the clustered service or to provide some sort of reporting. If the cron job were to run on a standby node then it would fail. If the cron job were running on the active node during a fail over what would happen?

If a cron job has to be run only on the active node then some checks should be performed before the job is run. For example, the job could check for the presence of share resources. Is the shared partition mounted? Is the DB2 service running? Multiple checks are better than just one. It may also be useful to store the scripts on the shared partition. Although cron would call them, they would not be run on non-active nodes since the mount would not be valid. Additionally, the job should be able to check for completion and report any error that may have been caused by fail over.

Finally, remember that service related cron table entries will need to be duplicated across each node. Other information that may need duplication across nodes are user accounts and group IDs.

Chapter 4

Testing the Cluster

Prior to deploying the cluster into service it should be tested to ensure fail over happens as and when expected. Prepare test cases for the node. The current expected results for each test are only estimates. Proper testing will clarify how the cluster suite manages these problems.

4.1 Turn off the network

4.1.1 Expected results

The cluster should fail over to the standby node. The elapsed time for this may impact the SLA for the cluster.

4.1.2 Actual results

The cluster's standby node activated in about 2 minutes. Within 4 minutes the failed node had been rebooted and instated as the standby node.

4.2 Unmount the shared partition

4.2.1 Expected results

The cluster should fail over to the standby node. The elapsed time for this may impact the SLA for the cluster.

4.2.2 Actual results

The cluster remounted the shared partition and restarted the DB2 service. The restart was completed in less than one minute.

4.3 Stop DB2

4.3.1 Expected results

The cluster should fail over to the standby node. The elapsed time for this may impact the SLA for the cluster. The status of the cluster can be tested with: `clustat`.

4.3.2 Actual results

The cluster stops all the shared resources, including DB2 even though it is already stopped. The cluster then restarted all the shared resources on the same node. The service was returned to normal in under one minute.

4.4 Unplug a single network cable

4.4.1 Expected results

Since two Ethernet cards are bonded in fail over mode this should not impact service at all. The kernel should start the standby card within a few seconds.

4.4.2 Actual results

The results were as expected.

4.5 Shutdown node

The node is powered off without issuing a software shutdown.

4.5.1 Expected results

The cluster should fail over to the standby node. The elapsed time for this may impact the SLA for the cluster. The status of the cluster can be tested with: `clustat`.

4.5.2 Actual results

The cluster activated the standby node within two minutes. The cluster, using the iLO fence, turned the power on to the failed node. The node then booted as normal and was returned to the cluster as a standby node.

4.6 Unplug fiber cable

4.6.1 Expected results

The cluster should fail over to the standby node. The elapsed time for this may impact the SLA for the cluster.

4.6.2 Actual results

The cluster failed over as expected. Once the fiber cable was plugged again in the failed node was returned to the cluster as a standby node.

4.7 Turn off the network during a DB2 write

4.7.1 Expected results

The cluster should fail over to the standby node. The elapsed time for this may impact the SLA for the cluster.

4.7.2 Actual results

The cluster failed over to the standby node in less than two minutes. The failed node was then rebooted and returned to standby service automatically. The data being written to DB2 was completed up to the last commit. Uncommitted data was lost.

4.8 Unplug fiber cable during a DB2 write

4.8.1 Expected results

The cluster should fail over to the standby node. The elapsed time for this may impact the SLA for the cluster.

4.8.2 Actual results

The cluster moved the service to the standby node. The previously active node was left as is. The data being written to DB2 was complete up to the last commit. Uncommitted data was lost.

Chapter 5

Managing the cluster

During the life of the cluster, the administrator will need to perform certain tasks regularly. The following is a list of those tasks and how to complete them.

5.1 Manual fail over

Bringing a node down for maintenance is a normal operation that allows administrators to maintain servers without affecting service. Also, such maintenance can be performed during normal working hours.

5.1.1 Fail over node

If the node to be brought down is the currently running node then the service must be moved to a standby node. On the running node issue this command:

```
/usr/sbin/clusvcadm -r db2 -m caesar
```

This instructs the cluster to move the service (db2) to another node (caesar). The cluster will then stop the service and restart it on the other node. Downtime while the service is moved should be less than thirty seconds.

5.1.2 Remove node from cluster

The node that is now down must be marked out of service to the cluster. This will prevent the cluster from attempting to use it if the running node should fail. Stop these services, in order, and then ensure that they do not start at boot time:

```
service rgmanager stop
service fenced stop
service cman stop
service ccsd stop
```

```
chkconfig --del rgmanager
```

```
chkconfig --del fenced
chkconfig --del cman
chkconfig --del ccsd
```

5.1.3 Perform off line maintenance

Now the node is now not part of the cluster. It can be upgraded, rebooted or anything else that is required without affecting the cluster.

5.1.4 Return node to cluster

The node must now be added back to the cluster.

```
chkconfig --add rgmanager
chkconfig --add fenced
chkconfig --add cman
chkconfig --add ccsd
```

```
service ccsd start
service cman start
service fenced start
service rgmanager start
```

5.2 Checking cluster status

The current status of the cluster can be checked in two ways.

5.2.1 X Windows GUI

You must have a working X server to use this option.

1. SSH to any node using the `-YC` switches for X forwarding.
2. Run the `system-config-cluster` application.

5.2.2 Command line interface

The command `clustat` will return the status of the cluster. For example:

```
[root@hadrian ~]# clustat
Member Status: Quorate
```

Member Name	Status
-----	-----
hadrian	Online, Local, rgmanager
caesar	Online, rgmanager

Service Name	Owner (Last)	State
-----	-----	-----
db2	hadrian	started

5.3 Log files

The cluster suite logs events in the file `/var/log/messages`. The cluster logs status events and fail overs. For example:

5.3.1 Status check

```
Aug 16 15:09:40 caesar clurgmgrd: [5159]: <info> Executing /etc/rc.d/init.d/db2 status
Aug 16 15:09:40 caesar su(pam_unix)[28369]: session opened for user dwapinst by (uid=0)
Aug 16 15:09:40 caesar su:
Aug 16 15:09:40 caesar su: Instance : dwapinst
Aug 16 15:09:40 caesar su: DB2 State : Available
Aug 16 15:09:40 caesar su(pam_unix)[28369]: session closed for user dwapinst
Aug 16 15:09:40 caesar db2: succeeded
```

5.3.2 Failover

```
Aug 16 15:03:26 hadrian kernel: end_request: I/O error, dev sda, sector 65
Aug 16 15:03:26 hadrian kernel: EXT3-fs: unable to read superblock
Aug 16 15:03:26 hadrian clurgmgrd: [4679]: <err> 'mount -t ext3 /dev/sda1 /db2' failed, error=32
Aug 16 15:03:26 hadrian clurgmgrd[4679]: <notice> start on fs "db2" returned 2 (invalid argument(s))
Aug 16 15:03:26 hadrian clurgmgrd[4679]: <warning> #68: Failed to start db2; return value: 1
Aug 16 15:03:26 hadrian clurgmgrd[4679]: <notice> Stopping service db2
Aug 16 15:03:26 hadrian clurgmgrd: [4679]: <info> Executing /etc/rc.d/init.d/db2 stop
Aug 16 15:03:26 hadrian su(pam_unix)[27368]: session opened for user dwapinst by (uid=0)
Aug 16 15:03:26 hadrian su:
Aug 16 15:03:26 hadrian su: Instance : dwapinst
Aug 16 15:03:26 hadrian su: DB2 State : Operable
Aug 16 15:03:26 hadrian su(pam_unix)[27368]: session closed for user dwapinst
Aug 16 15:03:26 hadrian db2: failed
Aug 16 15:03:26 hadrian db2: succeeded
Aug 16 15:03:26 hadrian clurgmgrd: [4679]: <info> /dev/sda1 is not mounted
Aug 16 15:03:31 hadrian clurgmgrd[4679]: <notice> Service db2 is recovering
Aug 16 15:03:31 hadrian clurgmgrd[4679]: <warning> #71: Relocating failed service db2
Aug 16 15:03:35 hadrian clurgmgrd[4679]: <notice> Service db2 is now running on member 1
```

Chapter 6

Conclusions

The Redhat Cluster Suite can offer a significant amount of redundancy above that of a single, well built server. Additionally, down time is reduced since cluster nodes can be maintained at leisure with little impact on service delivery. A chain is only as strong as its weakest link. An important proverb when building a cluster. While the nodes are redundant, external hardware may not be. Are all the nodes connected to the same network switch? Are they connected to the same UPS? Do the nodes rely on the same gateway router? The cluster in this paper does not include a redundant infrastructure

It should also be noted that the client must be able to account for a fail over. During the write tests, some data was lost. The client must be able to confirm its writes, know when a failure has occurred and be able to continue after its last success. Similarly, if a client is reading from the cluster it should be able to, where possible, detect a connection loss and repeat the request.

Although this paper discusses a DB2 cluster, it could be easily applied to cluster other services like Apache or Bind. It is even possible to have a cluster provide more than one service. Clusters can also be configured with load balancing instead of or, in addition to redundancy.

Appendix A

References and Additional Reading

1. Redhat Cluster Suite
<http://www.redhat.com/docs/manuals/csgfs/browse/rh-cs-en/index.html>
2. Linux Ethernet Bonding
<http://linux-net.osdl.org/index.php/Bonding>
3. High Availability Linux
<http://www.linux-ha.org/>
4. Kernel Parameters for Linux
<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.uprun.doc/doc/t0008238.htm>

Appendix B

DB2 init script

```
#!/bin/bash
#
#----- /etc/init.d/db2 -----
# db2
#
# description: Start up the db2 service

# Source function library.
. /etc/rc.d/init.d/functions

# Application owner
USER=dwapinst

RETVAL=0
prog="db2"

start() {
    echo -n "Starting $prog:"
    initlog -c "/bin/su - $USER -c 'db2start'" && success || failure
    RETVAL=$?
    echo ""
}

status() {
    initlog -c "/bin/su - $USER -c 'db2gcf -s'" && success || failure
    RETVAL=$?
    echo ""
}

stop() {
    echo -n "Stopping $prog:"

    # Is DB2 already stopped?
    status
    if [ $RETVAL -gt 0 ]; then

        # Already stoped return 0
        echo "Already stopped"
        success
        RETVAL=0
        echo ""
    else
        # DB2 must still be running. Stop it.
        initlog -c "/bin/su - $USER -c 'db2 force application all'" && success || failure &
        sleep 5
        initlog -c "/bin/su - $USER -c 'db2stop force'" && success || failure &
        sleep 5
        initlog -c "/bin/su - $USER -c 'db2_kill'" && success || failure &
        echo ""
    fi
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    status)
        status
        ;;
    restart)
```

```
stop      sleep 3
start
        ;;
*)
    echo $"Usage: $0 {start|stop|restart}"
    RETVAL=1
    ;;
esac
exit $RETVAL
```