Introducing Heartbeat High Availability Clustering

The purpose of this document is to explain how to implement a 2-node Apache high availability cluster (HAC). The purpose of this HA cluster is to couple a production Apache node with a hot backup node. In the event of a failure on the primary node, the following actions will take place automatically on the secondary node:

- Reconfigure the production IP address of the primary node
- Mount the shared Apache installation directory
- Start all Apache processes

HAC Cluster Architecture

The HAC architecture is comprised of two Linux systems running data synchronization and monitoring software. In order to achieve data synchronization and monitoring, the HAC implements a redundant physical private connection between the two nodes in the cluster.

HAC Cluster Software Components

The Apache HA cluster solution requires additional software packages and scripts not bundled with many Linux distributions. These packages are described below:

- Monitoring - Heartbeat
- Data Synchronization - Disk Replicating Block Device (DRBD)
- Apache Recovery - SYSV RC init scripts

Monitoring Apache with Heartbeat

The Heartbeat package is the most common open source HA package available. It has a simple syntax and monitoring approach to HA. Two systems share a heartbeat over a private closed loop. The heartbeat consists of a sequence of simple messages that use checksums to ensure normal activity. If the heartbeat is lost between the two nodes, the secondary node acquires the resources from the primary node.

Heartbeat uses redundant physical private loops to monitor heartbeats between two systems. These loops consist of:

- Serial Ports – Heartbeat monitors two nodes using a null-modem serial cable plugged into both devices.
- Ethernet Ports – Heartbeat monitors two nodes on a secondary Ethernet port using a crossover cable and private IP address network between both hosts.

Although only one private loop is required, two are needed to prevent the heartbeat itself from being a single point of failure (SPOF). Heartbeat will continue to monitor a system as long as 1 heartbeat loop is still active.

Heartbeat Resource Group

A heartbeat resource group (HRG) consists of system components that are controlled by heartbeat. Heartbeat assumes full ownership of the management of these resources. The OS relinquishes control. An HRG is managed between the two nodes in the cluster. When a primary node fails, it relinquishes control of the HRG and the primary assumes control. The resources in an HRG consist of:

- IP addresses
- Filesystems
- Initialization of processes

If the primary node fails, the Heartbeat program itself executes a series of SYSV `init` compatible run control scripts on both nodes. These scripts are executed in a similar fashion to SYSV init scripts using the "`stop`" and "`start`" keywords. On the primary node, Heartbeat executes these `init` scripts to perform the following actions:

- Down all Heartbeat controlled Ethernet interfaces (not system controlled interfaces)
- Stop all Heartbeat controlled processes
- Unmount all Heartbeat controlled filesystems

The Heartbeat program on the secondary node, executes the same scripts in "`start`" mode to perform the following actions:

- Up all Heartbeat controlled Ethernet interfaces
- Start all Heartbeat controlled processes
- Mount all Heartbeat controlled filesystems

Since these functions are controlled by Heartbeat and not the system init process, special modifications to the system (described later) must be made to ensure that both init and Heartbeat do not compete for access to the same resources.

Synchronizing Data with DRBD

The DRBD software package provides point to point synchronization between two disks over a private Ethernet connection. This is accomplished by creating a virtual disk block device for both Apache nodes to write to. Instead of mounting and performing filesystem checks on physical devices such as `/dev/sda1`, DRBD provides a metadevice called `/dev/drbd0`.

The primary node in the HAC cluster mounts the DRBD metadevice:

```
ha1# mount /dev/drbd0 /apache
```

Any data written to the `/apache` directory at this point is replicated to the two underlying physical disks. The write on the primary node will occur on the local `/dev/sda` device file. The write to the secondary node will be dispatched over a closed loop private network (`10.0.0.x`) connection on the `eth1` Ethernet interface. A DRBD listener on the secondary node receives this write request and then passes it to the local `/dev/sda` device on the secondary node.

The DRBD package is often referred to as a RAID 1 (mirror) device over Ethernet. The DRBD package includes extremely robust checksums and state management to ensure that synchronization occurs in the appropriate directions. It also provides mechanisms to avoid read and write contention between the primary and secondary nodes. The HAC uses DRBD instead of a separate NAS/NFS (or GFS) device because of these advanced read/write locking features. DRBD also reduces the complexity of the deployment by not requiring a 3rd networked device.

DRBD only allows one device to be in write mode (primary). The secondary device is in read mode. Although the second device is in read mode, it still receives real time synchronization from the primary device. In the event that the primary node in the HAC fails, the secondary disk should have near millisecond synchronization with the primary disk when it mounts the DRBD device.

Recovering Apache – Custom RC Script

Since Heartbeat executes a SYSV `init` script upon the failure of a node, the recovery of Apache must be placed in an RC script. The HAC contains the standard Apache init script (httpd). The httpd script supports the following standard options.

- `start` – This option starts all Apache processes and also executes the recovery function.
- `stop` – This option stops all Apache processes.
- `status` – This option reports the status of the Apache processes
- `reload` – This option reloads all configuration parameters.

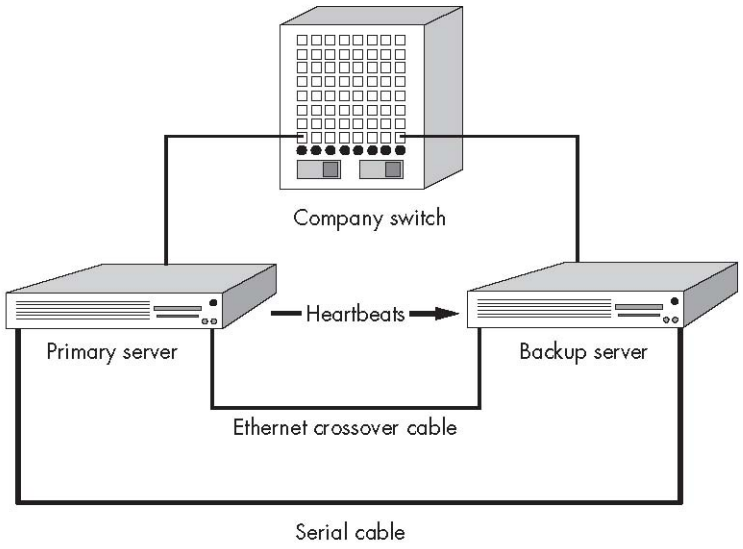
Heartbeat invokes the `httpd` script upon node failure with the `start` option.

Heartbeat High Availability Deployment

In order to successfully deploy a HAC, the following hardware resources and base configurations must be in place prior to implementing the HAC:

- Two X86 servers with a RedHat branch (CentOS and Fedora supported).
- Both eth0 interfaces configured with production IP addresses and connected to the network
- Both eth1 interfaces configured with a private network and directly connected by a crossover cable
- A null modem serial cable connected to the 9 pin serial ports on both servers

Figure 1: HAC Configuration



Installing Heartbeat/DRBD and Apache on Fedora Core 6

The Heartbeat source code is available at http://www.linux-ha.org. Various vendors and individuals have created packages for distributions. The Fedora Core branch contains pre-built versions of Heartbeat that can be installed through the Yum package interface.

To install Heartbeat, perform the following:

```
# yum install heartbeat
```

The DRBD package requires a little more work. The DRBD source code contains a SPEC file for building an RPM. Since DRBD contains a kernel module, the DRBD RPMS built will be relevant only to the specific kernel loaded.

To install DRBD perform the following steps:

1. You must first update your kernel to the latest revision.

```
# yum update kernel
```

2. You also need the kernel source to build the DRBD module.

```
# yum install kernel-devel
```

3. Reboot your system to load the new kernel.

```
# init 6
```

4. Once rebooted, grab the source code from the DRBD site.

```
# cd ~
# wget http://oss.linbit.com/drbd/0.7/drbd-0.7.23.tar.gz
# tar zxvf drdb*
```

5. The `Makefile` has a target to make RPMS. Simply enter the directory and run the `make` command.

```
# cd drbd*
# make rpm
```

6. The RPM packages are stored in a local directory called `dist`. Enter this directory and install the packages.

```
# cd dist/RPMS/i386
# rpm –ivh drbd-0*
# rpm –ivh drbd-km*
```

Configuring DRBD Components

The following section explains how to configure DRBD and Heartbeat in the appropriate order. Follow these steps closely as any misconfigurations will result in failure.

Perform the following steps:

1. DRBD provides a preconfigured DRBD configuration file. Most of the defaults are sufficient.

```
# cd /usr/share/doc/drbd-0.7.23/
# cp drbd.conf /etc
cp: overwrite `/etc/drbd.conf'? y
```

There are many configurable parameters in the `/etc/drbd.conf`. The following example lists the main configuration of the file:

```
on ha1 {
                device     /dev/drbd0;
                disk       /dev/sda3;
                address    10.0.0.1:7788;
                meta-disk  internal;
        }
```

The fields are as follows:

- on ***hostname*** - This directive requires the name of your host as per the uname –n command.
- `device /dev/drbd0` -This is the DRBD metadevice name you will use to mount your shared filesystem. All writes to this device will be replicated to the two underlying physical devices.
- `disk /dev/sda3` – This is the underlying physical device that the DRBD system will write to.
- `address 10.0.0.1:7788` – This is the IP address on which this node will try to connect to the secondary device. This IP address should be the address of the cross-over Ethernet connection on `eth1`.
- `metadisk internal` – This is the location of the DRBD metadevice status information. The internal keyword states that the last 128MB of the filesystem `/dev/sda3` will be reserved for DRBD metadata.

2. Edit the `on hostname` section of the `drbd.conf` file to reflect your system's hostname. This name should match the output of the `uname –n` command.

```
# vi /etc/drbd.conf

on yourhostname1 {
                device     /dev/drbd0;
                disk       /dev/sda3;
                address    10.0.0.1:7788;
                meta-disk  internal;
      }

on yourhostname2 {
                device     /dev/drbd0;
                disk       /dev/sda3;
                address    10.0.0.2:7788;
                meta-disk  internal;
      }
```

3. Delete all everything in the `drbd.conf` file from line 260 and down as these are other examples.
4. Configure the eth1 interface with the `10.0.0.x` (`10.0.0.1` for primary and `10.0.0.2` for secondary) address.

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth1


DEVICE=eth1
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.0.0.1
NETMASK=255.255.255.0
```

5. Restart the eth1 interface.

```
# ifdown eth1
# ifup eth1
```

6. Using `ifconfig`, verify that the IP address of the second interface:

```
# ifconfig eth1
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:15:C5:F1:93:69
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::215:c5ff:fef1:9369/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:404 (404.0 b)
          Interrupt:169 Memory:f8000000-f8011100
```

7. Modify the IPTables firewall ruleset to allow TCP port 7788. Insert the following line into the `iptables` file.

```
# vi /etc/sysconfig/iptables
```

CHANGE

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

TO

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 7788 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

8. Once finished, restart the IPTables firewall

```
# service iptables restart
```

9. Check to make sure that the eth1 interface is in full 100Tx mode.

```
# ethtool eth1
```

10. If the interface is not in full 100Tx mode, set it using `ethtool`. The eth1 interface only has to be at this speed for the initial sync.

```
# ethtool -s eth1 speed 100 duplex full autoneg off
```

Synchronizing DRBD Devices

Now that the DRBD package is installed on both systems, you can migrate the physical `/dev/sda3` devices on both `ha1` and `ha2` to the same shared logical device on both of `/dev/drbd0`.

1. Initialize the DRBD service on both nodes. You will eventually use LSB init scripts to control DRBD.

```
ha1# modprobe drbd
ha1# drbdadm up all
ha2# modprobe drbd
ha2# drbdadm up all
```

Run the following commands on the primary node only!

2. Neither device is mounted or writeable at this point. Both are in a secondary state. Since DRBD only allows one device to be in the primary state, force `ha1` to be in the primary state:

```
ha1# drbdsetup /dev/drbd0 primary –-do-what-I-say
```

3. Create a new filesystem on the `/dev/drbd0` device on the `ha1` system. The changes will start replicating down to the `/dev/drbd0` device on `ha2`.

```
ha1# mkfs –t ext3 /dev/drbd0
```

4. Once the filesystem is finished create a new directory to mount onto the `/dev/drbd0` device:

```
ha1# mkdir /apache
ha1# mount /dev/drbd0 /apache
ha1# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             9.9G  749M  8.7G   8% /
none                 1014M     0 1014M   0% /dev/shm
/dev/drbd0             56G   53M   53G   1% /apache
```

5. Modify the `/etc/fstab` on both systems so that the `/apache` directory uses the `/dev/drbd0` device. Make sure to disable the mount at boot option for `/apache` directory. The mounting of this device will be handled by the Heartbeat program at boot.

```
ha1# vi /etc/fstab

<snip>

/dev/drbd0         /apache       ext3    noauto          0 0


ha2# vi /etc/fstab

<snip>

/dev/drbd0         /apache        ext3    noauto          0 0
```

6. To complete the DRBD configuration, enable DRBD at boot using the `chkconfig` command:

```
ha1# chkconfig drbd on
ha2# chkconfig drbd on
```

Installing Apache on Primary Server

The Apache web server ships with every distribution of Linux. Almost every distribution places Apache in multiple locations across the files system. This makes it difficult for DRBD. The purpose of DRBD is to provide one instance of Apache that can run on two systems. In order to do this, you must copy every Apache file into a shared DRBD directory and then symbolically link the file back to its original spot. This makes it difficult to use package management to upgrade as you have changed the locations and links of files.

You can either figure out how to make this work OR install Apache from source. This document demonstrates the latter.

Configuring Apache

Download the latest Apache source tar ball from http://www.apache.org.

1. Create Apache's source directory in the DRBD shared directory.

```
# mkdir /apache/www
```

2. Unpack the Apache source.

```
# cd /tmp
# tar zxvf http*
```

3. Compile Apache from source. The following example shows a very basic install of Apache. If you want more robust features like DSO, SSL, and PHP/MySQL, you must do more research on how to compile these modules into Apache.

```
# cd httpd-2.2*
# ./configure –prefix=/apache/www; make; make install
```

4. Once Apache is done compiling, you must make an RC script that heartbeat can use to start and stop your Apache install. First, you must stop and disable the system version of Apache.

```
# chkconfig httpd off
# service httpd stop
# cd /etc/init.d
# mv httpd httpd.old
```

5. Create a new RC script called `apache` in the `/etc/init.d` directory.

```
# vi /etc/init.d/apache
#!/bin/bash
#
# httpd        Startup script for the Apache HTTP Server
#
# chkconfig: - 85 15
# description: Apache is a World Wide Web server.  It is used to serve \
#              HTML files and CGI.
# processname: httpd
# config: /etc/httpd/conf/httpd.conf
# config: /etc/sysconfig/httpd
# pidfile: /var/run/httpd.pid

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

# things -- attempting to start while running is a failure, and shutdown
```

```
# when not running is also a failure.  So we just do it the way init scripts
# are expected to behave here.
start() {
        echo -n $"Starting $prog: "
        /apache/www/bin/apachectl start
}

stop() {
        echo -n $"Stopping $prog: "
        /apache/www/bin/apachectl stop
}

# See how we were called.
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
    restart)
        stop
        start
        ;;

  *)
        echo $"Usage: $prog
{start|stop|restart|condrestart|reload|status|fullstatus|graceful|help|configtest}"
        exit 1
esac

exit $RETVAL
```

6. Make the apache file executable.

```
# chmod +x /etc/init.d/apache
```

7. Enable the apache service, but do not start it at boot.

```
# chkconfig –-add apache
# chkconfig apache off
```

Configuring Apache on the Secondary Server

Since the primary server currently has already setup Apache and the instance is shared, the only setup required on the secondary server is the run control script.

```
# scp ha2:/etc/init.d/apache /etc/init.d
# chmod +x /etc/init.d/apache
# chkconfig –add apache
# chkconfig apache off
```

Configuring Heartbeat – Primary and Secondary

The final step in configuring the Apache HAC is to configure Heartbeat. The Heartbeat package works by managing "resources". A resource is a grouping of functions on a system. This resource group migrates from the primary node to the secondary node in the event of a failure.

The resource group for the Apache HAC cluster consists of the following functions:

- Apache production IP addresses
- DRBD shared `/apache` drive
- Stopping and Starting Apache

Heartbeat requires 3 configuration files in the `/etc/ha.d` directory. These files consist of the `ha.cf`, `haresources`, and `authkeys`.

Perform the following steps on `ha1`. When complete, use `scp` to copy the files to `ha2`.

1. You must customize the `ha.cf` to add the names for your hosts. Modify the provided `ha.cf` to reflect your configuration.

```
ha1# vi /etc/ha.d/ha.cf

debugfile   /var/log/ha-debug
logfile     /var/log/ha-log
bcast       eth0 eth1   # Linux
auto_failback off
node    yourhostname1 # uname -n
node    yourhostname2 # uname -n
ping ip.address.of.router
```

2. You must customize the `haresources` file and configure the IP address(es) for that Heartbeat will monitor.

You need to modify this file to reflect your current configuration. This includes hostname, CIDR mask, interface, and broadcast.

```
ha1# vi /etc/ha.d/haresources
ha1 192.168.75.150/24/eth0/192.168.75.255 \ drbddisk::r0
Filesystem::/dev/drbd0::/apache::ext3 apache
```

The configuration parameters are as follows:

- `ha1` – This is the hostname of the primary node that manages this resource.
- `192.168.75.150/24/eth0/192.168.75.255` – This is the production IP address that Apache will use and that will fail over between the hosts. The fields are as follows:
  - IP Address
  - Subnet in CIDR notation
  - Ethernet Interface
  - Broadcast Address

- `drbddisk::r0` – This tells Heartbeat to make the DRBD resource `r0` (defined in `/etc/drbd.conf`) the primary. The `r0` name defines the `/apache` shared directory.
- `filesystem::/dev/drbd0::/apache::ext3` – This is a mount command issued by Heartbeat to mount the `/apache` directory.
- `apache` – This is a custom RC script in `/etc/init.d` that starts Apache

3. Change the permissions on the `authkeys` file.

```
ha1# vi /etc/ha.d/authkeys
auth 1
1 crc

ha1# chmod 600 /etc/ha.d/authkeys
```

4. Move the pre-configure apache start script provided by Heartbeat on both systems. It is not compatible with the custom compiled version of Apache.

```
# cd /etc/ha.d/resources.d
# mv apache apache.old
```

5. The `ha1` system is not configured. In order to configure `ha2`, copy the `ha*` files to the system.

```
ha1# cd /etc/ha.d
ha1# scp haresources ha.cf authkeys ha2:/etc/ha.d
```

6. You must allow the heartbeat port on both systems through the IPTables firewall. Edit the `iptables` file to include this port.

```
# vi /etc/sysconfig/iptables

ADD

-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 694 -j ACCEPT
```

7. Restart the IPTables firewall on both systems.

```
# service iptables restart
```

8. Enable the Heartbeat and DRBD services at boot on both systems as the final step.

```
# chkconfig heartbeat on
# chkconfig drbd on
```

If the system was already in production and had virtual interfaces already, be sure to remove these from the `/etc/sysconfig/network-scripts/` directory.

9. Reboot the `ha1` node first, followed by `ha2`.

```
ha1# init 6
ha2# init 6
```

HAC Monitoring

Both the DRBD and Heartbeat packages include monitoring scripts and log files to check on the health of the cluster. The following sections provide details on how to monitor the HAC after it boots.

HAC Log Files

Both the DRBD and the Heartbeat programs write to the Syslog messaging framework and additional log files. The `/var/log/messages` file is the first file to check for all DRBD and Heartbeat information. In the following example output, the Heartbeat program reports that both nodes are online after a boot:

```
# grep heartbeat /var/log/messages

Aug  5 12:16:06 ha1 heartbeat: [3303]: info: *************************
Aug  5 12:16:06 ha1 heartbeat: [3303]: info: Configuration validated. Starting heartbeat 2.0.6
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: heartbeat: version 2.0.6
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: Heartbeat generation: 5
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: G_main_add_TriggerHandler: Added signal manual handler
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: G_main_add_TriggerHandler: Added signal manual handler
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: Removing /var/run/heartbeat/rsctmp failed, recreating.
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: glib: Starting serial heartbeat on tty /dev/ttyS0 (19200
baud)
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: glib: UDP Broadcast heartbeat started on port 694 (694) i
nterface eth1
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: glib: UDP Broadcast heartbeat closed on port 694 interfac
e eth1 - Status: 1
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: glib: ping heartbeat started.
Aug  5 12:16:06 ha1 heartbeat: [3304]: info: G_main_add_SignalHandler: Added signal handler for signal
 17
Aug  5 12:16:07 ha1 heartbeat: [3304]: info: Local status now set to: 'up'
Aug  5 12:16:08 ha1 heartbeat: [3304]: info: Link ha2.example.net:/dev/ttyS0 up.
Aug  5 12:16:09 ha1 heartbeat: [3304]: WARN: G_CH_dispatch_int: Dispatch function for read child took
too long to execute: 950 ms (GSource: 0x9f87bc8)
Aug  5 12:16:09 ha1 heartbeat: [3304]: WARN: G_CH_dispatch_int: Dispatch function for read child was d
elayed 950 ms before being called (GSource: 0x9f87cf8)
Aug  5 12:16:09 ha1 heartbeat: [3304]: info: G_CH_dispatch_int: started at 429405964 should have start
ed at 429405869
Aug  5 12:16:09 ha1 heartbeat: [3304]: info: Link ha2.example.net:eth1 up.
Aug  5 12:16:09 ha1 heartbeat: [3304]: info: Status update for node ha2.example.net: status up
Aug  5 12:16:09 ha1 heartbeat: [3304]: info: Link 192.168.29.1:192.168.29.1 up.
Aug  5 12:16:09 ha1 heartbeat: [3304]: info: Status update for node 192.168.29.1: status ping
Aug  5 12:16:09 ha1 heartbeat: [3304]: info: Link ha1.example.net:eth1 up.
Aug  5 12:16:10 ha1 heartbeat: [3304]: info: Comm_now_up(): updating status to active
Aug  5 12:16:10 ha1 heartbeat: [3304]: info: Local status now set to: 'active'
Aug  5 12:16:10 ha1 heartbeat: [3304]: info: Starting child client "/usr/lib/heartbeat/ipfail" (500,50
0)
```

The Heartbeat package also provides its own debug and info log files. They are called `ha-debug` and `ha-log` and are located in the `/var/log` directory.

```
# cd /var/log
# ls ha*
ha-debug    ha-log
```

These contain many of the messages that are in the `/var/log/messages`, but their format is much simpler. The following example log entry from the `ha-log` shows a normal startup for Heartbeat on the primary node:

```
# tail –f /var/log/ha-log
heartbeat[3636]: 2006/08/04_23:37:42 info: *************************
heartbeat[3636]: 2006/08/04_23:37:42 info: Configuration validated. Starting heartbeat 2.0.6
heartbeat[3637]: 2006/08/04_23:37:42 info: heartbeat: version 2.0.6
heartbeat[3637]: 2006/08/04_23:37:42 WARN: No Previous generation - starting at 1
heartbeat[3637]: 2006/08/04_23:37:42 info: Heartbeat generation: 1
heartbeat[3637]: 2006/08/04_23:37:42 info: No uuid found for current node - generating a new uuid.
```

```
heartbeat[3637]: 2006/08/04_23:37:42 info: G_main_add_TriggerHandler: Added signal manual handler
heartbeat[3637]: 2006/08/04_23:37:42 info: G_main_add_TriggerHandler: Added signal manual handler
heartbeat[3637]: 2006/08/04_23:37:42 info: Creating FIFO /var/lib/heartbeat/fifo.
heartbeat[3637]: 2006/08/04_23:37:42 info: Removing /var/run/heartbeat/rsctmp failed, recreating.
heartbeat[3637]: 2006/08/04_23:37:42 info: glib: Starting serial heartbeat on tty /dev/ttyS0 (19200
baud)
heartbeat[3637]: 2006/08/04_23:37:42 info: glib: UDP Broadcast heartbeat started on port 694 (694)
interface eth1
heartbeat[3637]: 2006/08/04_23:37:42 info: glib: UDP Broadcast heartbeat closed on port 694
interface eth1 - Stat
us: 1
heartbeat[3637]: 2006/08/04_23:37:42 info: glib: ping heartbeat started.
heartbeat[3637]: 2006/08/04_23:37:42 info: G_main_add_SignalHandler: Added signal handler for
signal 17
heartbeat[3637]: 2006/08/04_23:37:42 info: Local status now set to: 'up'
heartbeat[3637]: 2006/08/04_23:37:43 info: Link 192.168.29.1:192.168.29.1 up.
heartbeat[3637]: 2006/08/04_23:37:43 info: Status update for node 192.168.29.1: status ping
heartbeat[3637]: 2006/08/04_23:37:43 info: Link ha1.example.net:eth1 up.
heartbeat[3637]: 2006/08/04_23:37:44 info: Link ha2.example.net:/dev/ttyS0 up.
heartbeat[3637]: 2006/08/04_23:37:44 info: Status update for node ha2.example.net: status active
heartbeat[3648]: 2006/08/04_23:37:44 debug: notify_world: setting SIGCHLD Handler to SIG_DFL
heartbeat[3637]: 2006/08/04_23:37:44 info: Link ha2.example.net:eth1 up.
harc[3648]:     2006/08/04_23:37:44 info: Running /etc/ha.d/rc.d/status status
heartbeat[3637]: 2006/08/04_23:37:44 info: Comm_now_up(): updating status to active
heartbeat[3637]: 2006/08/04_23:37:44 info: Local status now set to: 'active'
heartbeat[3637]: 2006/08/04_23:37:44 info: Starting child client "/usr/lib/heartbeat/ipfail"
(500,500)
heartbeat[3659]: 2006/08/04_23:37:44 info: Starting "/usr/lib/heartbeat/ipfail" as uid 500  gid 500
(pid 3659)
```

The Heartbeat program places the following entry in the `ha-log` file on the secondary node when it acquires the Example resources from the primary node.

```
# tail –f /var/log/ha-log
heartbeat[3637]: 2006/08/04_23:39:53 info: ha2.example.net wants to go standby [all]
heartbeat[3637]: 2006/08/04_23:40:22 info: standby: acquire [all] resources from ha2.example.net
heartbeat[3663]: 2006/08/04_23:40:22 info: acquire all HA resources (standby).
ResourceManager[3673]:  2006/08/04_23:40:22 info: Acquiring resource group: ha1.example.net 192.168.29.184/28/
eth0/192.168.29.255 drbddisk::sm1 Filesystem::/dev/drbd0::/apache::ext3 apache
IPaddr[3697]:   2006/08/04_23:40:22 INFO: IPaddr Resource is stopped
ResourceManager[3673]:  2006/08/04_23:40:22 info: Running /etc/ha.d/resource.d/IPaddr
192.168.29.184/28/eth0/192.
168.29.255 start
IPaddr[3898]:   2006/08/04_23:40:22 INFO: /sbin/ifconfig eth0:0 192.168.75.150 netmask 255.255.255.0 broadcast
192.168.29.255
IPaddr[3898]:   2006/08/04_23:40:22 INFO: Sending Gratuitous Arp for 192.168.75.150 on eth0:0 [eth0]
IPaddr[3898]:   2006/08/04_23:40:22 INFO: /usr/lib/heartbeat/send_arp -i 500 -r 10 -p
/var/run/heartbeat/rsctmp/s
end_arp/send_arp-192.168.29.184 eth0 192.168.29.184 auto 192.168.75.150 ffffffffffff
IPaddr[3816]:   2006/08/04_23:40:22 INFO: IPaddr Success
ResourceManager[3673]:  2006/08/04_23:40:22 info: Running /etc/ha.d/resource.d/drbddisk r0 start
Filesystem[4100]:       2006/08/04_23:40:22 INFO: Running status for /dev/drbd0 on /apache
Filesystem[4100]:       2006/08/04_23:40:22 INFO: /apache is unmounted (stopped)
Filesystem[4036]:       2006/08/04_23:40:22 INFO: Filesystem Resource is stopped
ResourceManager[3673]:  2006/08/04_23:40:22 info: Running /etc/ha.d/resource.d/Filesystem /dev/drbd0 /apache
ext3
start
Filesystem[4209]:       2006/08/04_23:40:22 INFO: Running start for /dev/drbd0 on /apache
Filesystem[4145]:       2006/08/04_23:40:22 INFO: Filesystem Success
ResourceManager[3673]:  2006/08/04_23:40:24 info: Running /etc/init.d/apache  start
heartbeat[3663]: 2006/08/04_23:41:52 info: all HA resource acquisition completed (standby).
heartbeat[3637]: 2006/08/04_23:41:52 info: Standby resource acquisition done [all].
heartbeat[3637]: 2006/08/04_23:41:53 info: remote resource transition completed.
```

All DRBD messages are located in the `/var/log/messages` file. The following sample log output shows the DRBD device starting on the secondary node.

```
Aug 23 14:01:11 ha2 kernel: drbd: initialised. Version: 0.7.20 (api:79/proto:74)
Aug 23 14:01:11 ha2 kernel: drbd: SVN Revision: 2260 build by root@ha2.example.net, 2006-07-21 16:12:22
Aug 23 14:01:11 ha2 kernel: drbd: registered as block device major 147
Aug 23 14:01:11 ha2 kernel: klogd 1.4.1, ---------- state change ----------
Aug 23 14:01:17 ha2 kernel: drbd0: resync bitmap: bits=14710885 words=459716
Aug 23 14:01:17 ha2 kernel: drbd0: size = 56 GB (58843540 KB)
Aug 23 14:01:17 ha2 kernel: drbd0: 0 KB marked out-of-sync by on disk bit-map.
Aug 23 14:01:17 ha2 kernel: drbd0: Found 6 transactions (324 active extents) in activity log.
Aug 23 14:01:17 ha2 kernel: drbd0: drbdsetup [23655]: cstate Unconfigured --> StandAlone
Aug 23 14:01:17 ha2 kernel: drbd0: drbdsetup [23668]: cstate StandAlone --> Unconnected
Aug 23 14:01:17 ha2 kernel: drbd0: drbd0_receiver [23669]: cstate Unconnected --> WFConnection
Aug 23 14:01:18 ha2 kernel: drbd0: drbd0_receiver [23669]: cstate WFConnection --> WFReportParams
Aug 23 14:01:18 ha2 kernel: drbd0: Handshake successful: DRBD Network Protocol version 74
Aug 23 14:01:18 ha2 kernel: drbd0: Connection established.
Aug 23 14:01:18 ha2 kernel: drbd0: I am(S): 1:00000002:00000001:00000019:00000003:00
Aug 23 14:01:18 ha2 kernel: drbd0: Peer(P): 1:00000002:00000001:0000001a:00000003:10
Aug 23 14:01:18 ha2 kernel: drbd0: drbd0_receiver [23669]: cstate WFReportParams --> WFBitMapT
Aug 23 14:01:18 ha2 kernel: drbd0: Secondary/Unknown --> Secondary/Primary
Aug 23 14:01:18 ha2 kernel: drbd0: drbd0_receiver [23669]: cstate WFBitMapT --> SyncTarget
Aug 23 14:01:18 ha2 kernel: drbd0: Resync started as SyncTarget (need to sync 64 KB [16 bits set]).
Aug 23 14:01:18 ha2 kernel: drbd0: Resync done (total 1 sec; paused 0 sec; 64 K/sec)
Aug 23 14:01:18 ha2 kernel: drbd0: drbd0_worker [23656]: cstate SyncTarget --> Connected
```

The most important line of output here is the Secondary/Primary line:

```
Aug 23 14:01:18 ha2 kernel: drbd0: Secondary/Unknown --> Secondary/Primary
```

This line states that the DRBD device on this node is in Secondary state and that its peer on the other node is in primary state. This is an expected condition for the secondary node. This should be the entry on the primary node:

```
Aug 23 14:00:48 ha1 kernel: drbd0: Primary/Unknown --> Primary/Secondary
```

It is impossible for the HAC to get into a state of Primary/Primary as only one device is allowed to be primary. The only time this will ever happen is if both the Serial and Ethernet heartbeats fail. This situation is called a “Split Brain” situation and can cause serious data corruption.

The only way this can happen is if only the Ethernet and Serial ports fail while both systems are running (hardware failure) or if someone pulls both the Ethernet and Serial cables.

The “Split Brain” occurs when the secondary node believes the primary is dead as a result of no heartbeats and it acquires the HA resources. The problem is that the primary node is still online and can’t communicate with the secondary node. So, the primary node keeps the same HA resources and marks the secondary node as dead even though the secondary node has also acquired the resources.

There is also a rare event when both devices boot into the secondary state. In this case, the /apache directory WILL NOT MOUNT. A device can’t be mounted while in secondary state. The log file entry on either system looks like this:

```
Aug 23 14:01:18 ha2 kernel: drbd0: Secondary/Unknown --> Secondary/Secondary
```

The only way to fix this is to force one of the devices to be primary. In the following example, the primary node forces itself to be primary for the DRBD device:

```
ha1# drbdadm primary all
```

The syntax of this command is covered in the following sections.

HAC Monitoring Commands

You can use both common UNIX and special HAC commands to monitor the cluster.

In the following example, the ps command lists that all of the Heartbeat processes are up and running:

```
# ps -ef | grep heartbeat
root      2966     1  0 Aug17 ?   00:00:20 heartbeat: master control process
nobody    3203  2966  0 Aug17 ?   00:00:00 heartbeat: FIFO reader
nobody    3308  2966  0 Aug17 ?   00:00:00 heartbeat: write: serial /dev/ttyS0
nobody    3309  2966  0 Aug17 ?   00:02:08 heartbeat: read: serial /dev/ttyS0
nobody    3310  2966  0 Aug17 ?   00:00:02 heartbeat: write: bcast eth1
nobody    3311  2966  0 Aug17 ?   00:00:17 heartbeat: read: bcast eth1
nobody    3312  2966  0 Aug17 ?   00:00:07 heartbeat: write: ping 192.168.75.1
nobody    3313  2966  0 Aug17 ?   00:00:23 heartbeat: read: ping 192.168.75.1
500       3326  2966  0 Aug17 ?   00:00:00 /usr/lib/heartbeat/ipfail
```

Even though the processes are all running, it does not guarantee that the heartbeats are coming across the wire. The Heartbeat program is using eth1 to transmit heartbeats. The HAC configuration states that these are sent over UDP port 694. The following example uses the tcpdump command to observe the heartbeats between the two nodes:

```
# tcpdump -ni eth1 port 694
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
16:06:01.235374 IP 10.0.0.1.32769 > 10.0.0.255.ha-cluster: UDP, length 193
16:06:01.235412 IP 10.0.0.1.32769 > 10.0.0.255.ha-cluster: UDP, length 185
16:06:01.235701 IP 10.0.0.1.32769 > 10.0.0.255.ha-cluster: UDP, length 193
16:06:03.131600 IP 10.0.0.2.32769 > 10.0.0.255.ha-cluster: UDP, length 184
16:06:03.232947 IP 10.0.0.1.32769 > 10.0.0.255.ha-cluster: UDP, length 185
```

Heartbeats also travel across the serial port via a null modem cable if configured. It is possible to listen to these heartbeats by opening the serial device file.

The following cat command opens the serial device file to listen for heartbeats:

```
# cat /dev/ttyS0
>>
t=stus
st=tive
d7530
ptocol=src=hastrongil.net(1)srcud=MUUuz6RZel+t22ix==
seqaffc
h=6
ts=4e31f1
=0.00 .00 0.01/74 294
ttl
auth=29e7016
<<<
```

The following ps output shows that DRBD is running:

```
# ps -ef | grep -i drbd
root      2802     1  0 Aug17 ?        00:00:20 [drbd0_receiver]
root     29674     1  0 Aug23 ?        00:00:00 [drbd0_worker]
root     29680     1  0 Aug23 ?        00:00:02 [drbd0_asender]
```

The DRBD kernel module opens a TCP socket connection on port 7788 on the `eth1` interface. The following `netstat` output demonstrates that that port is open in a bi-directional fashion. Both the `ha1` node and the `ha2` node are connected to each other's port 7788:

```
# netstat -anp | grep 7788
tcp    0   0 10.0.0.1:7788      10.0.0.2:32776              ESTABLISHED -
tcp    0   0 10.0.0.1:32990     10.0.0.2:7788               ESTABLISHED -
```

Using the `tcpdump` on the DRBD connection validates that DRBD traffic is flowing between the two endpoints:

```
# tcpdump -ni eth1 port 7788
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
16:26:23.186627 IP 10.0.0.1.7788 > 10.0.0.2.32776: . 2863387387:2863388835(1448) ack 1829943254 win 16022
<nop,nop,timestamp 616209899 615470053>
16:26:23.186641 IP 10.0.0.1.7788 > 10.0.0.2.32776: . 1448:2896(1448) ack 1 win 16022 <nop,nop,timestamp
616209899 615470053>
16:26:23.186651 IP 10.0.0.1.7788 > 10.0.0.2.32776: P 2896:4120(1224) ack 1 win 16022 <nop,nop,timestamp
616209899 615470053>
16:26:23.186832 IP 10.0.0.2.32776 > 10.0.0.1.7788: . ack 1448 win 16022 <nop,nop,timestamp 615477062 616209899>
16:26:23.186848 IP 10.0.0.2.32776 > 10.0.0.1.7788: . ack 2896 win 16022 <nop,nop,timestamp 615477062 616209899>
16:26:23.186853 IP 10.0.0.2.32776 > 10.0.0.1.7788: . ack 4120 win 16022 <nop,nop,timestamp 615477062 616209899>
16:26:23.186988 IP 10.0.0.1.7788 > 10.0.0.2.32776: . 4120:5568(1448) ack 1 win 16022 <nop,nop,timestamp
616209899 615477062>
16:26:23.186996 IP 10.0.0.1.7788 > 10.0.0.2.32776: . 5568:7016(1448) ack 1 win 16022 <nop,nop,timestamp
616209899 615477062>
```

Checking Connection State

The DRBD software comes with 2 command line utilities to extract state information from the system. The utilities are the `drbdsetup` and `drbdadm` commands.

Both nodes in the cluster should always be connected. This is the normal state. Any other state signifies a problem in the cluster and needs to be addressed. The two states that signify a problem are:

- `WFConnect` – This state occurs when one node can't contact another. This means that no communication is taking place on port 7788. Use `netstat` to see if port 7788 is in the `ESTABLISHED` state on both systems. You can also use `telnet` to try to connect to port 7788 on both systems.
- `StandAlone` – This state occurs when the primary node has given up on the secondary and simply chooses to not send any updates until it can communicate with the secondary. This means that either the secondary is down OR the primary can't communicate with it. Both `netstat` and `telnet` may be used here to check the connection.

The following `drbdadm` example displays that the DRBD device on the local system is in primary mode:

```
# drbdadm state all
Primary/Secondary
```

The following `drbdadm` example displays that the devices are connected:

```
# drbdadm cstate all
Connected
```

These commands read from the raw `/proc/drbd` file. This file provides 3 checks for health: state, mode, and consistency.

```
# cat /proc/drbd
version: 0.7.20 (api:79/proto:74)
SVN Revision: 2260 build by root@rhel4u4, 2006-11-15 05:22:25
 0: cs:Connected st:Primary/Secondary ld:Consistent
 ns:5004 nr:1184 dw:6188 dr:95413 al:0 bm:11 lo:0 pe:0 ua:0 ap:0
```

Anything else besides these three states presents a problem. In the following output, the `/proc/drbd` states that the secondary server is completely out of synch with the primary:

```
# cat /proc/drbd
version: 0.7.20 (api:79/proto:74)
SVN Revision: 2260 build by root@rhel4u4, 2006-11-15 05:47:18
 0: cs:WFConnect st:Secondary/Unknown ld:Inconsistent
 ns:5004 nr:1184 dw:6188 dr:95413 al:0 bm:11 lo:0 pe:0 ua:0 ap:0
```

The following `drbdsetup` command displays the configuration information of the `/dev/drbd0` device file on the `ha1` node that was read in from the `/etc/drdb.conf` at runtime.

```
# drbdsetup /dev/drbd0 show
Lower device: 08:03   (sda3)
Meta device: internal
Disk options:
 on-io-error = panic
Local address: 10.0.0.1:7788
Remote address: 10.0.0.2:7788
Wire protocol: C
Net options:
 timeout = 6.0 sec (default)
 connect-int = 10 sec (default)
 ping-int = 10 sec (default)
 max-epoch-size = 2048  (default)
 max-buffers = 2048  (default)
 unplug-watermark = 128  (default)
 sndbuf-size = 524288
 ko-count = 4
Syncer options:
 rate = 102400 KB/sec
 group = 1
 al-extents = 257
```

Performing Automatic Failovers of the HAC

The HAC enables the ability to perform a system administrator initiated failover. This process may be done automatically or manually.

The Heartbeat package contains a command called `hb_standby` to perform automatic failovers. It informs the primary node to give up resources and go into a standby state. It will then inform the secondary to acquire the resources and the secondary will then become primary.

```
ha1# cd /usr/lib/heartbeat
ha1# ls hb_standby
hb_standby
ha1# ./hb_standby
2006/08/25_10:47:12 Going standby [all].
```

Observe the output in the `ha-log` on the primary node:

```
ha1# tail –f /var/log/ha-log
heartbeat[2966]: 2006/08/25_10:47:13 info: ha1.apache.net wants to go standby [all]
heartbeat[2966]: 2006/08/25_10:47:13 info: standby: ha2.apache.net can take our all resources
heartbeat[14224]: 2006/08/25_10:47:13 info: give up all HA resources (standby).
ResourceManager[14234]: 2006/08/25_10:47:13 info: Releasing resource group: ha1.apache.net
192.168.29.184/28/eth0/192.168.29.255 drbddisk::sm1 Fi
lesystem::/dev/drbd0::/apache::ext3 apache
ResourceManager[14234]: 2006/08/25_10:47:13 info: Running /etc/init.d/apache  stop
ResourceManager[14234]: 2006/08/25_10:47:46 info: Running /etc/ha.d/resource.d/Filesystem /dev/drbd0 /apache
ext3 stop
Filesystem[14863]:      2006/08/25_10:47:46 INFO: Running stop for /dev/drbd0 on /apache
Filesystem[14863]:      2006/08/25_10:47:46 INFO: Trying to unmount /apache
Filesystem[14863]:      2006/08/25_10:47:47 INFO: unmounted /apache successfully
Filesystem[14799]:      2006/08/25_10:47:47 INFO: Filesystem Success
ResourceManager[14234]: 2006/08/25_10:47:47 info: Running /etc/ha.d/resource.d/drbddisk r0 stop
ResourceManager[14234]: 2006/08/25_10:47:47 info: Running /etc/ha.d/resource.d/IPaddr
192.168.75.150/28/eth0/192.168.75.255 stop
IPaddr[15040]:  2006/08/25_10:47:47 INFO: /sbin/route -n del -host 192.168.75.150
IPaddr[15040]:  2006/08/25_10:47:47 INFO: /sbin/ifconfig eth0:0 192.168.75.150 down
IPaddr[15040]:  2006/08/25_10:47:47 INFO: IP Address 192.168.75.150 released
IPaddr[14958]:  2006/08/25_10:47:47 INFO: IPaddr Success
heartbeat[14224]: 2006/08/25_10:47:47 info: all HA resource release completed (standby).
heartbeat[2966]: 2006/08/25_10:47:47 info: Local standby process completed [all].
```