High Availability NFS Server with Linux Heartbeat

Inner Pulse

Clusters or failover solutions are the typical answer to high availability on Linux. Linux Heartbeat makes light work of this with a redundant NFS Server and a shared hard disk. BY ANDRÉ BONHÔTE

Two machines, a few leads, a switch and three IPs are all you need to provide redundant network services. To be more precise, your choice of equipment could be two typical desktop computers with two network adapters each, a fiber channel interface and a preinstalled Red Hat 8.0. A SCSI disk array attached to the fiber channel interface and thus to both computers provides a shared storage pool. The active computer uses NFS to serve up the content of the disk array to an Intranet. In case of the active server failing, the redundant server immediately takes over the role of the NFS server transparently for the clients.

This scheme is based on the Linux HA project, the project's aim being to provide highly available Linux systems. In this article, we will be looking at the Heartbeat [1] subproject. Stonith (Shoot the other node in the head) additionally prevents both servers writing simultaneously to the shared hard disk. This approach allows the redundant machine to power down the machine it is replacing thus making doubly sure that the original machine will not attempt to write to the shared media.

Twin Sisters

The cluster is easy to configure if both systems are identically equipped. Configuring an NFS server involves installing the basic Red Hat server, as the *nfs-utils* package is installed by default. You will also need development tools like *gcc* and *make*, plus *glib-devel* and *libnet* [2]. If you want the system to handle SNMP requests, you will also need *net-snmp-devel*.

Table 1 shows the configuration for both nodes. The host names and IP addresses should be known to either the DNS or the */etc/hosts* files on both nodes. Figure 1 shows how the systems are networked.

The fiber channel card and disk array types will define how Linux accesses the shared disks in the array. After mastering this obstacle the admin user can go on to partition and format the disks. Our example uses an Ext3 partition called */dev/sdc1*. Various alternatives with LVM or RAID are possible and make sense. The external disk will later be used to store data and the NFS lockfiles. Note that */etc/fstab* will not be edited. Heartbeat will take care of mounting and dismounting the partitions later.

Stonith

Whenever two systems can access the same hard disk, it is vital to avoid simultaneous writes to a single partition. Normally, there is no way this can occur. If one server fails the other takes over and starts writing to the disk from that point on. Unfortunately, if one cluster node thinks the other node is down, although the other node is actually quite happily writing to disk, both machines might attempt to access the shared media. This will inevitably damage your file systems and necessitate tedious rescue operations.

Stonith, basically a hardware device positioned between the power socket and the server (see Figure 2), can help prevent this disaster. When the cluster software assumes that a machine has crashed, it uses Stonith to power down

Table 1: Cluster configuration

Role	Hostname	IP
Node 1 (Master)	one	192.168.0.201
Node 2	two	192.168.0.202
Shared IP	cluster	192.168.0.200
APC-Master	apc	192.168.0.100
Heartbeat 1	-	172.16.0.1
Heartbeat 2	-	172.16.0.2

THE AUTHOR

André Bonhôte is an IP Engineer and works for COLT Telecom in Zurich, Switzerland. He was allowed to use his father's DEC Professional for the first time at the tender age of twelve. André has been heavily involved with Linux since 1996. Both of his kids have at least one Tux and a Perl camel in their cuddly toy collections.

the failed server. After a pre-defined interval, the device will re-connect mains power allowing the Linux system to boot and possibly replace the second node in the cluster.

If both servers write to the disk despite these precautionary measures, a Stonith hack ensures that the forcibly downed server will not be able to issue an *umount*, so being unable to *sync*, which would mean writing data from cache to disk. In doing so Stonith sacrifices the cache data, but saves the filesystem.

Various Stonith models and types are commercially available, mostly referred to as Master or Power Switches, ranging from budget two-port switches with a single power outlet, to devices with redundant switching power supplies and eight or more outlets. We used an APC Master Switch with a single input port and eight outlets in our lab. This architecture has the disadvantage of downing the whole cluster if you happen to disconnect the wrong power lead, but then again it is a low-budget solution. If you can afford it, you might prefer to use a redundant system with one Stonith device per server (see Figure 3).

The APC Master Switch can be configured via a Web interface. It is important that the power outlets know the exact hostnames of the servers to which they are attached, as shown by *uname -n*. Failing to do so will mean that Heartbeat will be unable to power down a faulty machine. The servers must be capable of sending SNMP commands to the APC device; APC expects a *Write+* type command. Note: The Master Switch is the Achilles heel in this type of cluster. Removing the power supply is an efficient denial of service attack.

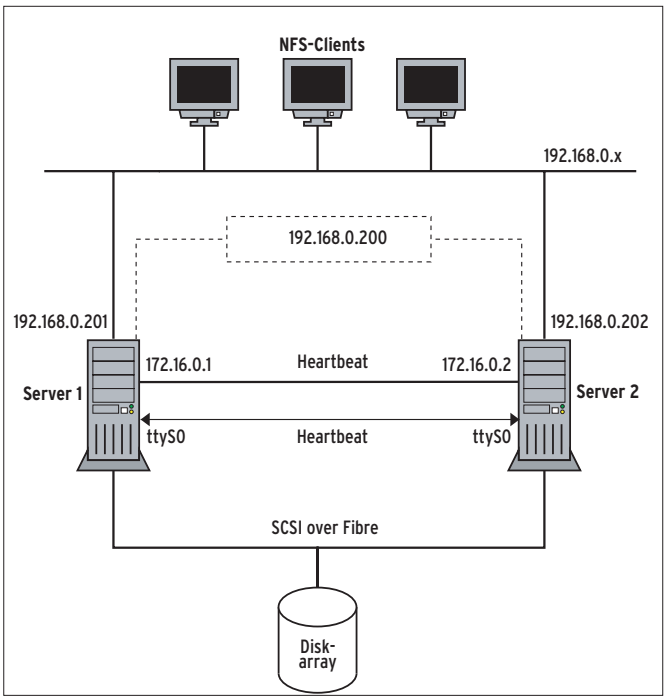

Figure 1: The NFS clients only see the *192.168.0.200* IP address, which the two servers share. Both servers access shared storage and use a heartbeat connection to check if their partner is alive

Umbilical Cords

Both machines need to know which system is currently active, and they use Heartbeat cables to do so. These connections should be configured redundantly, using an Ethernet crossover cable attached to *eth1* on both machines, and an additional serial nullmodem cable as a failsafe. A real IP address and the virtual IP for the cluster can then be assigned to *eth0*.

Two quick tests will ensure that both "umbilical cords" are working. After using *cat < /dev/ttyS0* to open the serial interface on one node, the other end can transmit data across this connection: *echo test > /dev/ttyS0*. If *test* appears at the other end, the serial connection is working in this direction at least. Of course, you might like to test the opposite direction too. If the transmission fails, or an error message is displayed, try the serial howto [3] for more tips. You can use ping to test the Ethernet connection. To do this both servers will need an IP in the same subnet. You can configure the first node as follows:

```
ifconfig eth1 172.16.0.1 up
```

Now assign an address to the second computer and ping the first machine:

```
ifconfig eth1 172.16.0.2 up
ping 172.16.0.1
```

If the ping works, you can add the eth1 configuration to */etc/sysconfig/network-scripts* for both interfaces. Listing 1 shows the configuration for node one.

The second machine can be configured identically, except for its IP, which will be 172.16.0.2. This ensures that both interfaces will have the right IP address when rebooted.

Heart Sounds

After installing Heartbeat (see the "Heart Transplant" box), both systems are ready for clustering. The Stonith device is the next component to configure and test. Heartbeat provides a *stonith* command. Entering *stonith* without any arguments will display a long and fairly cryptic list of available devices. Each of these devices has its own command set.

The Stonith program needs a */etc/ha.d/rpc.cfg* file with the IP address or hostname of the Stonith device, plus a valid login and password, or the SNMP community and port. You can type *stonith -t* devicename *dummy* to find out what

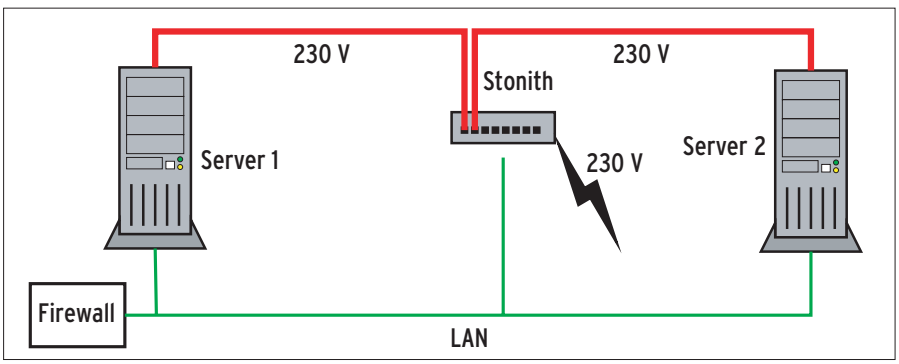

Figure 2: The traditional approach is to attach both servers to a central Stonith device, which can then be remote controlled

entries you need for which device. This command will also show you the required file format – the following lines are required for the APC Master Switch on both cluster nodes:

```
#IP      Port     Community
192.168.0.100   161      stw
```

The SNMP community will need *Write+* privileges to disconnect the power supply to individual ports. If the ports are named correctly, node two should be able to disconnect node one:

```
stonith -t apcmastersnmp one
```

The LED should be extinguished and the switch should interrupt the power supply, then restore the supply after a pre-defined interval.

Heartbeat additionally requires three files */etc/ha.d/ha.cf*, */etc/ha.d/haresources* and */etc/ha.d/authkeys*. The */usr/share/doc/heartbeat-1.0.1/* directory contains well-documented and self-explanatory samples.

Ready for Lift-Off

When launched, the Heartbeat daemon reads its defaults from the lengthy *ha.cf* file. This file does not need to be identical on both machines, one of the nodes might, for example, use a different serial port to listen for the heartbeat or log events to a different target. Listing 2 shows the default configuration without any comments.

The *haresources* file, which defines the shared services, the main node and the IP address for the cluster, must be identical at both ends. The sample file in */usr/share/doc/heartbeat-1.0.1/* is well documented and provides several practical examples. An NFS cluster requires only the following lines:

```
one 192.168.0.200 Filesystem
::/dev/sda1::/data::ext3
nfslock nfs
```

The first word in the line specifies the hostname (*uname -a*) of the master server, followed by the cluster IP, and then the services that Heartbeat will launch or terminate. When launched, Heartbeat searches */etc/init.d/* and */etc/ha.d/resource.d/* for the scripts listed in *haresources*. In our example this might be called */etc/ha.d/resource.d/Filesystem*. Heartbeat passes the *start* parameter to the script. Other colon-separated parameters can also be supplied. Heartbeat will pass them to the script before passing the *start* argument. The filesystem entry produces the following command:

```
/etc/ha.d/resource.d/Filesystem
  /dev/sda1 /data ext3 start
```

Services can be halted using similar syntax with the *stop* parameter.

The */etc/ha.d/resource.d/* directory contains scripts for specific services. For example, *Filesystem* mounts and unmounts partitions. The script sources are well documented and provide examples of practical applications.

The third file, *authkeys*, is required to authenticate the heartbeat over the Ethernet interface. This is hardly necessary if you are using a crossover cable, but if the heartbeat is transmitted across a non-trusted network, this option can protect your server from denial of service attacks. In our case, pseudo-authentication via CRC will do the trick:

```
auth 1
1 crc
```

For security reasons, Heartbeat requires *chmod 600 /etc/ha.d/authkeys*.

The links in the appropriate runlevels need to be modified to launch the HA software on rebooting (see Listing 3). Note that the network must be available before launching Heartbeat, as the machine will otherwise assume that the other node is down, and use Stonith to power the node down.

Duplicate NFS

So far, we have only mentioned NFS in passing. NFS is one of the services for which Heartbeat can provide a redundant configuration. In our example, the idea is to mount the */dev/sda1* partition in */data* and to export it via NFS. Both

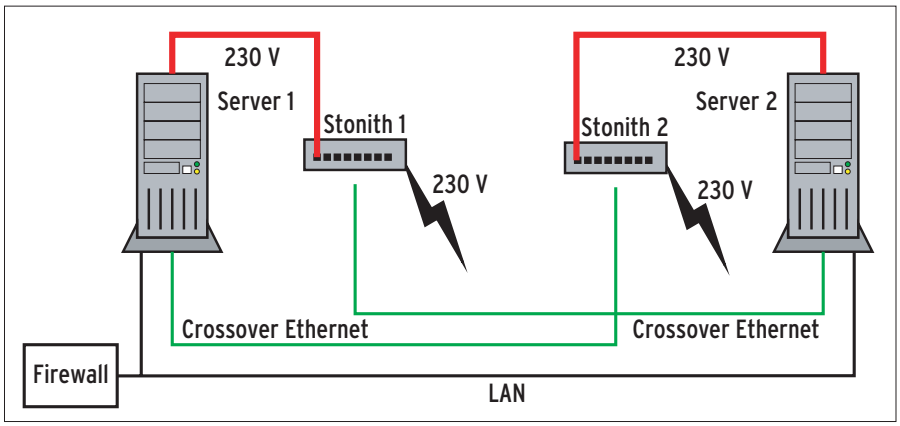

Figure 3: A redundant wiring scheme is preferable: Each server uses an Ethernet crossover cable to talk to the Stonith device attached to its partner

Listing 1: *ifcfg-eth1*

```
DEVICE=eth1
BOOTPROTO=static
IPADDR=172.16.0.1
NETMASK=255.255.255.0
ONBOOT=yes
```

Heart Transplant

The sources for Heartbeat are available from [4]. The site also provides a short howto to get you started. You will need Libnet, before you can compile Heartbeat:

```
wget
http://www.packetfactory.net/
libnet/dist/libnet.tar.gz

tar zxvf libnet.tar.gz
cd Libnet-latest
./configure
make
make install-strip
```

After installing Libnet, you should be able to configure and compile Heartbeat with a minimum of effort.

```
wget http://www.linux-ha.org/
download/heartbeat-1.0.1.tar.gz

tar zxvf heartbeat-1.0.1.tar.gz
cd heartbeat-1.0.1
./ConfigureMe make
make install-strip
```

machines can access this partition. You should note that an entry in */etc/fstab* must be avoided to prevent both systems attempting simultaneous writes. *mount* and *umount* operations will be handled by the *Filesystem* Heartbeat script.

NFS normally uses */var/lib/nfs* to manage its locking files. If */data* now fails over from node one to node two, the lockfiles will remain on the server originally used to manage them. This causes NFS a problem: the clients start to transmit error messages, and can no longer access the files they require.

To avoid this, the lockfiles also have to migrate to the new server. The easiest way of ensuring this is to use the shared partition. After mounting manually (*mount /dev/sda1 /data*), you can then move the directory: *mv /var/lib/nfs /data/varlibnfs*, and use a symlink to hand the directory back to the NFS daemon: *ln -sf /data/varlibnfs /var/lib/nfs*. You can now dismount the hard disk, allowing Heartbeat to mount it a few seconds later.

The *rpc.statd* process, which is initialized in the */etc/init.d/nfslock* start script, uses its hostname to identify itself. It will typically use the Libc function *gethostname()* to do so, but this will supply the name of the current node, rather than the cluster name.

statd provides the option *-n* to resolve this issue. You set the required value in the start script:

```
start() {
 ...
 echo -n "Starting NFS statd: "
 daemon rpc.statd -n cluster
 RETVAL=$?
 ...
}
```

NFS shares the directories listed in */etc/exports*. As lockfiles should be transparent to your users, admins should export the subdirectories below */data/*. The following entry should be sufficient for initial testing:

```
/data/userdata  *(rw)
```

The Home Straight

The preparatory work has been completed, both nodes in the cluster know what to do in case of a failure, and the Stonith device will cut the power supply if the worst comes to the worst. You can now launch Heartbeat on both machines: */etc/init.d/heartbeat start*

A quick look at */var/log/messages* and */var/log/ha-** confirms that the services on node one have been activated without any glitches. The shared IP address is pingable and has been assigned to *eth0:0* on node one.

If the admin user manually stops the Heartbeat software on the master node, by typing */etc/init.d/heartbeat stop*, node two will take over the hard disk and the services. The lines shown in Listing 4 then appear in */var/log/messages*.

When node one starts producing the required signals again, node two will drop the services, the disk and the IP, passing them back to the master.

Stonith has not needed to intervene so far. Heartbeat went down gracefully and reported the take over. If node one is overloaded to a point where it cannot perform a graceful handover, Heartbeat will cut the power supply. A short Bash script can simulate this scenario:

```
#!/bin/bash
:(){ :|:&};:
```

There maybe friendlier and more efficient methods, such as CPUburn, but this script does have an e legance.

After the dirty reboot, the machine will come back up and reinstate itself. Being transparent to your NFS clients.

Future

Heartbeat provides the foundations for far more complex scenarios than the one described in this article, such as active/active configurations, where both nodes work in parallel and perform load balancing. A quick look at */etc/ha.d/resource.d/* reveals services that can be added to *haresources*.

The mailing list at [5] is a useful source of help, if you are having trouble getting things running. The Heartbeat developers and a few extremely experienced admins monitor the list, and are often willing to lend a hand. ■

INFO

[1] Heartbeat homepage: *http://www.linux-ha.org/heartbeat/*

[2] Libnet: *http://www.packetfactory.net/libnet/dist/*

[3] Serial Howto: *http://www.ibiblio.org/pub/Linux/docs/HOWTO/Serial-HOWTO*

[4] Heartbeat download: *http://www.linux-ha.org/download/*

[5] Contact and mailing list: *http://www.linux-ha.org/contact/*

Listing 2: Heartbeat Configuration

```
# /etc/ha.d/ha.cf
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
deadtime 20
initdead 120
serial  /dev/ttyS0
baud    19200
udpport 694
bcast   eth1
stonith apcmastersnmp
/etc/ha.d/rpc.cfg
node    one
node    two
```

Listing 3: Heartbeat Start Scripts

```
cd /etc/rc0.d ; ln -s ../init.d/heartbeat K03heartbeat
cd /etc/rc3.d ; ln -s ../init.d/heartbeat S80heartbeat
cd /etc/rc5.d ; ln -s ../init.d/heartbeat S80heartbeat
cd /etc/rc6.d ; ln -s ../init.d/heartbeat K03heartbeat
```

Listing 4: Node two taking over

```
info: Taking over resource group 192.168.0.200
info: /sbin/ifconfig eth0:0 192.168.0.200 ...
info: Running /etc/ha.d/resource.d/Filesystem ...
info: Running /etc/init.d/nfslock  start
info: Running /etc/init.d/nfs  start
info: mach_down takeover complete for node one.
```