How to Build a Beowulf HPC System Using the FedoraLiveCD Project

Build a Beowulf cluster without disks to optimize cost and reliability, and simplify software maintenance.

HOWARD POWELL

The FedoraLiveCD Project allows anyone to create a custom bootable CD or PXE boot image with little effort. For large HPC systems, this greatly simplifies the creation of diskless compute nodes, leading to higher reliability and lower costs when designing your cluster environment. The network and CPU overhead for a diskless setup are minimal, and the compute nodes will run entirely from an initial ramdisk, so they will exhibit very good I/O for normal OS disk operations.

The cluster I've designed is set up for MPI-based computation. The master node runs a queue system where jobs can be submitted and farmed out to the compute nodes to run within the allotted resources. Because my compute nodes are diskless, the goal is to

Listing 1. Example dhcpd.conf File

```
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#
ddns-update-style interim;
allow booting;
allow bootp;
option dns-domain-search-list code 119 = string;

subnet 10.0.0.0 netmask 255.255.0.0 {
         default-lease-time 604800;
         max-lease-time 1209600;
         option routers 10.0.0.1;
         option ip-forwarding off;
         option subnet-mask 255.255.0.0;
         range dynamic-bootp 10.0.0.100 10.0.0.254;
}

subnet 10.1.0.0 netmask 255.255.0.0 {
         default-lease-time 604800;
         max-lease-time 1209600;
         option routers 10.1.0.1;
         option ip-forwarding off;
         option ntp-servers 10.1.0.1;
         option subnet-mask 255.255.0.0;
         option domain-name-servers 10.1.0.1;
         option time-offset -5;
         option domain-name "cluster";
         option interface-mtu 9000;
}

class "pxeclients" {
         match if substring(option vendor-class-identifier, 0, 9) =
         ➥"PXEClient";
         next-server 10.1.0.1;
         filename "pxelinux.0";

}


host c0 {
         hardware ethernet A4:BA:DB:1E:71:2D;
         fixed-address 10.1.0.254;
         option host-name "c0";
}

host c1 {
         hardware ethernet A4:BA:DB:1E:71:3A;
         fixed-address 10.1.0.253;
         option host-name "c1";
}

host c2 {
         hardware ethernet A4:BA:DB:1E:71:47;
         fixed-address 10.1.0.252;
         option host-name "c2";
}

host c3 {
         hardware ethernet A4:BA:DB:1E:71:54;
         fixed-address 10.1.0.251;
         option host-name "c3";
}
```

produce a simple and streamlined operating system with as few libraries and utilities as necessary to get the nodes to interact with the master job scheduler. Software that is needed by jobs (such as the MPI libraries) can be shared via NFS from the master node. The compute nodes simply have a kernel and the basic libraries needed to start a job. User account information can be shared via a local LDAP service running on the master node or by any method you already may have available in your environment.

To prepare a diskless cluster, your master node will need some amount of reasonably fast local disk storage and at least 10/100 Ethernet, preferably gigabit Ethernet. Your diskless nodes will need Ethernet hardware that can PXE boot from a network interface; most modern hardware supports this. These nodes will need to be on the same physical subnet, or you will have to configure your dhcpd service to respond or relay between subnets. Your diskless nodes also should have sufficient physical memory (RAM) to hold the OS image plus have enough room to run your programs—a few gigabytes of RAM should be sufficient if you keep your OS image simple.

For the rest of this article, I assume your cluster is based on a Red Hat-derived distribution, as this is based on a Fedora-specific tool. I'm going to demonstrate an environment where all of the cluster nodes can communicate with the master on a private Ethernet subnet.

Your boot server needs to run just two services for diskless booting: DHCP and TFTP. DNSMasq can be substituted for DHCP and TFTP, but I demonstrate using separate DHCP and TFTP services because that's how I set up my own cluster. For convenience, you may choose to install bind or some other DNS to make communication between nodes more friendly. To deploy custom rpm files quickly, you may want to have access to a local repository shared via Apache or another Web service. Local rpm repositories also are a viable method to deploy custom rpm files.

First, install DHCP via yum:

```
yum -y install dhcp tftp-server syslinux
```

The file /etc/dhcpd.conf should be created, and in this config file, you need to define your subnet and a pxeclients class that simply locates the bootable pxelinux image on disk. You also need

Listing 2. Example tftp File

```
service tftp
{
   socket_type     = dgram
   protocol        = udp
   wait            = yes
   user            = root
   server          = /usr/sbin/in.tftpd
   server_args     = -s /tftpboot
   disable         = no
   bind            = 10.1.0.1
   per_source      = 11
   cps             = 100 2
   flags           = IPv4
}
```

Listing 3. Example nodes-ks.cfg

```
### System  language
lang en_US.UTF-8


### System keyboard
keyboard us


### System timezone
timezone America/New_York


### Root password
rootpw abcd1234


### System authorization information
auth  --useshadow  --enablecache


### Firewall configuration
# Firewalls are not necessary in a cluster, usually
firewall --disabled


### Disables Selinux
selinux --disable


### Repositories
repo --name=Your-Custom-Repo --baseurl=
    http://your.custom.repo/
repo --name=base --baseurl=
    http://mirror.centos.org/centos/5/os\$basearch/
repo --name=newrepo --baseurl=file:///tmp/localrepo


### Enable and disable some services
services --enabled=gpm,ipmi,ntpd --disabled=nfs


### Package install information
%packages
bash
kernel
syslinux
passwd
policycoreutils
chkconfig
authconfig
rootfiles
comps-extras
xkeyboard-config
nscd
nss_ldap
autofs
gpm
ntp
compat-gcc-34-g77
compat-libf2c-34
compat-libstdc++-296
compat-libstdc++-33
dapl
dapl-utils
dhcp
dmidecode
hwloc
iscsi-initiator-utils
libXinerama
libXmu
libXpm
libXp
libXt
man
mesa-libGL
nfs-utils
openssh
openssh-clients
openssh-server
pciutils
sysklogd
tvflash
vim-minimal
vim-enhanced


### Pre-install scripts

### Post-install scripts
%post

### Here you can run any shell commands you wish to
### further customize your nodes.

### Sets up DHCP networking on the compute nodes
cat << EOF > ifcfg-eth0
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
MTU=1500
EOF

mv ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0
```

If you have multiple network interfaces on your master node, you can choose to bind TFTP to one interface by using the bind command.

to define the diskless hosts definition for each node by associating the bootable MAC address of each node with a static IP that you define for that node. I also chose to include the host-name option, so that my diskless hosts will know a name other than localhost.localdomain once they are booted.

Next, you need to enable the TFTP dæmon. Red Hat systems launch TFTP via xinetd—I simply needed to enable the /etc/xinetd.d/tftp config file and start xinetd. If you have multiple network interfaces on your master node, you can choose to bind TFTP to one interface by using the bind command.

Once configured, both services should be added to the default runlevel and started:

```
chkconfig dhcpd on
chkconfig xinetd on
service dhcpd start
service xinetd start
```

Now for the fun part—creating the OS image. RPMForge hosts a version of the livecd-tools package, which can be installed via yum:

```
yum install livecd-tools
```

The live CD tools require a Red Hat kickstart file—templates can be found via Google and as part of the livecd-tools package. A template kickstart is generated by anaconda on any freshly installed system in the root home directory as /root/anaconda-ks.cfg.

Of particular interest here are the %packages and the %post sections. In %packages, you can choose exactly which programs you need or want installed on the initial ramdisk image and available to the OS at boot. I recommend choosing as little as you can in order to keep the initrd small and streamlined. In %post, you can add any shell commands you need in order to customize your compute nodes further—for example, by editing config files for needed services. The example kickstart provided here works with a RHEL- or CentOS 5.5-based distribution.

If you review my example kickstart file, you'll notice that I've specified DHCP as the boot protocol for the network on each of the compute nodes. Because the dhcpd service already knows about the Ethernet MAC address of my diskless compute nodes, the nodes simply will re-request an IP address during boot and be reassigned the same one. Remember that no unique information is stored on the node's OS image, so using DHCP is the easiest way to assign IPs to each diskless node.

One special situation to note: because the compute nodes are diskless, each time SSH starts on a node, it generates a new set of host keys. When the node reboots, it generates a new set of different keys, leading to an impossible-to-maintain situation for SSH users. To solve this, I have generated a template host key that I then deploy copies of to each of my diskless compute nodes via an rpm file. To build your own version of this rpm, you need to create a spec file (see the example) and copy the host keys from /etc/ssh to the location specified by BuildRoot in the spec file. The rpmbuild command generates the rpm, and this rpm can be included in a local yum repository by specifying its name to the %packages section of your kickstart:

```
rpmbuild -bb sshkeys.spec
```

By setting up SSH with the same host key on each node, I've defeated some of the security of SSH by allowing the possibility of man-in-the-middle attacks between my master node and compute nodes. However, in my cluster environment where compute nodes communicate on a private and dedicated channel and do not have a direct connection to the outside

Listing 4. Example cluster-ssh-keys.spec

```
%define name    cluster-ssh-keys
%define version 1.0
%define release 1

Summary: ssh keys for cluster compute nodes
Name: %{name}
Version: %{version}
Release: %{release}
Group: System Environment/Base
License: GPL
BuildArch: noarch
BuildRoot: %{_builddir}
URL: http://your.custom.url
Distribution: whatever
Vendor: You
Packager: your email

%description
This provides the ssh keys necessary for compute
nodes on a diskless cluster.

%prep
exit 0

%build
exit 0

%install
exit 0

%clean
exit 0

%files
%defattr(-,root,root)
/etc/ssh
```

Listing 5. Example mknodes.sh

```
#!/bin/bash

/etc/init.d/nscd stop

cd /local-disk/nodes/

livecd-creator --config=/local/nodes/nodes-ks.cfg \
  --fslabel=cluster -t /local-disk/nodes/

livecd-iso-to-pxeboot /local-disk/nodes/cluster.iso

rsync -av /local-disk/nodes/tftpboot/ /tftpboot/

rm /local-disk/nodes/cluster.iso
rm -rf /local-disk/nodes/tftpboot

/etc/init.d/nscd start
```

world, this shouldn't be a problem.

Another idea that might simplify your SSH environment is to consider enabling host-based SSH authentication (so users don't have to generate private and public keys while on your cluster). The root SSH environment is hardened against SSH host-based authentication, so you'll either have to work around this security measure or set up SSH public/private key-chains for the root account on your new cluster. Normal users should have no problems with host-based SSH authentication, so long as the UIDs are common among the entire cluster.

Once your kickstart has been customized to your liking, the rest of the setup is simple. Just run the livecd-creator script to generate an ISO image, then use the livecd-isto-to-pxe script to convert that into something TFTP can use.

When compiling the OS image, some active dæmons may interfere with the build process. Of particular note, SELinux must be permissive or disabled, and if you use the nameserver cache dæmon (nscd), you may need to disable it temporarily while the build process runs or else risk a corrupted image:

```
setenforce 0
service nscd stop
livecd-creator --config=nodes-ks.cfg --fsla-
bel=Compute_nodes
livecd-iso-to-pxe Compute_nodes.iso
rsync -av tftpboot/ /tftpboot/
service nscd start
```

I've chosen to write all of this into a handy shell script that creates the image and cleans up any temporary files for me.

Listing 6. Example exports File

```
/local-disk 10.0.0.0/255.0.0.0(rw,async)
```

Listing 7. Example fstab File

```
master:/local-disk  /local-disk  nfs  _netdev  0 0
```

Once the files have been copied to tftpboot, it's time to boot a compute node. If all goes well, the diskless client will request a DHCP address, and your DHCP server will respond with an IP and the location of the TFTP server and image to download. The client then should connect to the TFTP server, download the image and launch the OS you just created.

Problems with the PXE boot process can be diagnosed by using any network protocol analyzer, such as Wireshark. Once the image is loaded and the kernel is alive, you should see the normal boot process on the screen of the diskless compute node.

As noted before, specialized user-level software (such as the MPI libraries in my case) can be distributed to your nodes via standard NFS shares. On your NFS server (it can be the same as your master node), simply define a new share in /etc/exports and enable NFS:

```
chkconfig nfs on
service nfs start
```

Your nodes need to add an entry for the NFS server either to their local fstab files or via some other method like autofs.

User home directories can be shared via NFS or via a high-performance, cluster-based filesystem, such as PVFS2 or Lustre. NFS is reliable when disk I/O is not very intensive or mostly read-only, but it breaks down if your code relies heavily on large numbers of files or heavy, simultaneous I/O operations to disk.

Please keep in mind that any customizations of the environment on the diskless nodes are not maintained between reboots. In fact, it's perfectly okay to cold-reset a diskless node; the OS image cannot be corrupted like it could be if it were on a local disk. This simplifies troubleshooting strange node problems. If a reboot doesn't clear the problem (and assuming no other diskless nodes show the same problem), it's almost certainly a hardware bug—this alone can save hours of time when working with a large cluster.■

User home directories can be shared via NFS or via a high-performance, cluster-based filesystem, such as PVFS2 or Lustre.

Howard Powell is the sole sysadmin at the University of Virginia Astronomy Department. He's built three generations of Linux-based high-performance computing clusters to support the Virginia Institute of Theoretical Astronomical, which are used to study cool things like what's happening around black holes in our universe. He lives near Charlottesville, Virginia.