Return to CACR Home

About CACR

News & Events

Research

Computing Resources

Publications

Contact Information/ People

Visitor's Center

Help Desk

# Building a Beowulf System

Jan Lindheim, Caltech
Updated 05/03/05

## 1. Introduction

With the power and low prices of today's off-the-shelf PCs and the availability of 100 Mb/s ethernet interconnect, it makes sense to combine them to build High-Performance-Computing and Parallel Computing environment. This is the concept behind the Beowulf parallel computing system we will describe. With free versions of Unix and public domain software packages, no commercially available parallel computing system can compete with the price of the Beowulf system. The drawback to this system is, of course, there will exist no support center to call when a problem arises. But there does exist a wealth of good information available through FTP sites, web sites and newsgroups.

The examples you find in this document will apply to using the RedHat or Mandrake Linux operating system. For other distributions of Linux, please check the relevant documentation. The different Linux distributions can be found at http://www.linuxhq.com/dist-index.html.

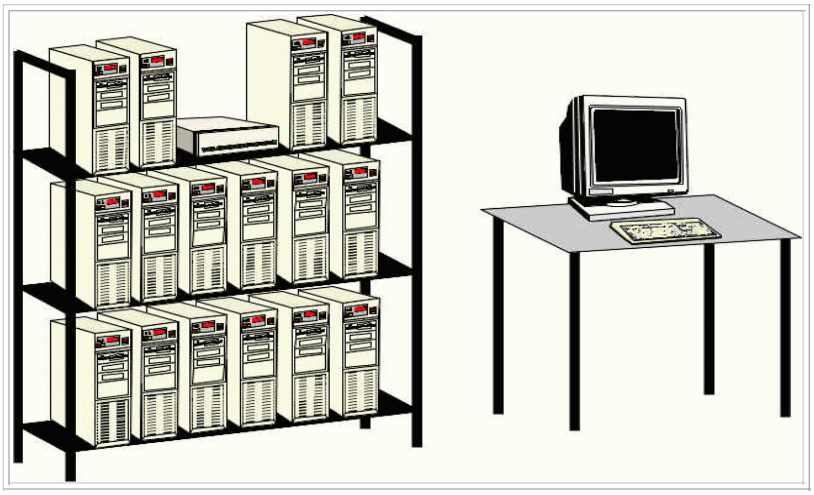

Figure 1: Example Beowulf System Setup

## 2. Planning the system

Today, there's a wide range of switches available, ranging from 8 to over 100 ports, some with one or more Gigabit modules that lets you build large systems by interconnecting such switches or by using them with a Gigabit switch. Switches have become inexpensive enough that there's not much reason to build your network by using cheap hubs or by connecting the nodes directly in a hypercube network.

An example of parts to buy for a 16-node system + front-end system, where each node has a 400 MHz Pentium II processor, 256 MB RAM, a 10 GB disk drive, a 100 Mb/s network adapter and one SVGA adapter, could be something like:

| | |
|---|---|
| 24-port network switch | 1 |
| Motherboard | 17 |
| 400 MHz Pentium II | 17 |
| PC cases | 17 |
| 128 MB SDRAM SIMMs | 34 |
| 100 Mb/s ethernet adapters | 18 |
| 10 GB disks | 17 |
| 1.44 MB Floppy drive | 17 |
| SVGA adapter | 17 |
| CD-ROM | 1 |
| Keyboard | 1 |
| 17" Multisync Monitor | 1 |
| Monitor/keyboard 4 way switch boxes | 5 |
| keyboard extension cables | 16 |
| video extension cables | 16 |
| 6' twisted pair network cables | 18 |
| 6-Outlet Surge protectors | 3 |

Table 1: Shopping list

A typical setup, will be to configure the nodes for a private beowulf network. The front-end machine will have two network adapters, one configured with a real IP address, the other configured for the private beowulf subnet. The suggested range of addresses for a private network is from 192.168.0.0 to 192.168.254.0. It does not matter which one you choose, since nobody will be able to connect directly to a compute node from outside this network anyway. The front-end system will act like a bridge from other machines to your Beowulf system. The monitor and the keyboard will be shared among all PCs, by using video/keyboard switch boxes.

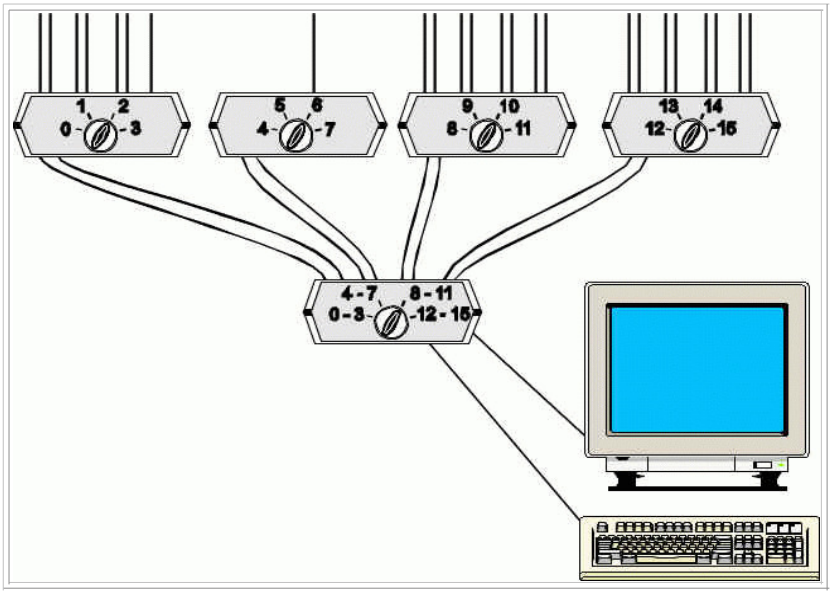


Figure 2: Connections of Video/Keyboard Switch Boxes

All the parts in the PCs are standard off-the-shelf components. The switch would typically have to be ordered through a distributor of the brand you decide to use.

### 3. Software Installation

To make the system as easy as possible to maintain, plan on making all compute nodes identical, i.e. same partition sizes, same software and same configuration files. This makes the software a whole lot easier to install and maintain. For partition sizes, I like to give 128 MB to root, 2X memory for swap partition for less than 128 MB RAM, 1X memory if you have 128 MB or more RAM. Some people would say, "Why have any swap space at all?" For most usage, swapping should not happen on the compute nodes, but accidents happen. For /usr partition it really depends on the number of packages you are planning to install, about 800 MB should go a long way for most systems.

I recommend installing at least the following, if you are installing

RedHat or Mandrake Linux:

- Basic System Software, including networking software
- Kernel Sources
- C, C++, g77 compilers and libraries
- X11 with development libraries
- xntp or another time synchronizer
- autofs
- rsync

To install these packages, please follow the standard instructions that come with the Linux distribution you choose to use. You want to try to install all you need, before you start the cloning process. There are a few files that you probably want to edit before cloning the system:

- /etc/hosts

- /etc/hosts.equiv
- /etc/shosts.equiv
- /etc/fstab
- /etc/pam.d/rlogin
- /etc/ntp.conf
- /etc/lilo.conf
- /etc/exports
- /etc/auto.master
- /etc/auto.beowulf

For the hosts file, you need to list all the names, with aliases, for each node. All nodes need to be listed in hosts.equiv and shosts.equiv to let users use rsh commands without creating their own .rhosts and .shosts files. In your /etc/pam.d/rlogin file, you want to make sure you do not require secure tty when logging into the nodes. To fix this, I just switch the order of the modules pam_securetty.so and pam_rhosts_auth.so., and end up with the following:

```
auth            sufficient      /lib/security/pam_rhosts_auth.so
auth            required        /lib/security/pam_securetty.so
auth            required        /lib/security/pam_pwdb.so shadow
nullok
auth            required        /lib/security/pam_nologin.so
account         required        /lib/security/pam_pwdb.so
password        required        /lib/security/pam_cracklib.so
password        required        /lib/security/pam_pwdb.so shadow
nullok use_authtok
session         required        /lib/security/pam_pwdb.so
```

When setting up ntp, have the front-end machine (could be node 0) sync up to the outside world and broadcast the time to the private network. The nodes will be configured to sync their time to the front-end machine. If you don't use a private network, you can have all machines sync up to any ntp server.

Also, setting up your front-end machine to do masquerading will let you open graphics output from a compute node to your workstation. For more documentation on masquerading, please see: http://atlantic2000.com/ip-masq/index.html

It's always a good idea to have a good backup kernel that lilo can find in case you install a new kernel that does not work. Our lilo.conf looks like:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
promt
timeout=20
read-only
root=/dev/hda1
image=/boot/vmlinuz
            label=linux
image=/boot/vmlinuz.old
             label=linux.old
image=/boot/vmlinuz.clone
             label=linux.clone
image=/boot/vmlinuz.save
            label=linux.save
```

We decided to export scratch partitions from all nodes and use autofs to mount these from any node upon request. Our exports file looks like:

```
/scratch n???.cacr.caltech.edu(rw,no_root_squash)
```

The no_root_squash option lets root on one node operate as root on any of the mounted files systems.

To set up autofs, add a line like:

```
/data        /etc/auto.beowulf          --timeout 600
```

to the file /etc/auto.master and create /etc/auto.beowulf with entries like:

```
n000 -fstype=nfs n000:/scratch
n001 -fstype=nfs n001:/scratch
n002 -fstype=nfs n002:/scratch
n003 -fstype=nfs n003:/scratch
```

On the nodes, you will also want to automatically mount home directories from the front-end machine, so you'll need to have:

```
home -fstype=nfs front-end-machine-name:/home
```

added to the /etc/auto.beowulf file as well.

On the front-end machine I create a directory named "local" on the large partition used for home directories, move everything from /usr/local to this directory and make a symbolic link to this directory from the /usr directory (ex. ln -s /home/local /usr/local). For software packages like MPICH and PVM, I install them under /usr/local/mpich and /usr/local/pvm on the front-end machine On the nodes /usr/local will will be a symbolic link to the NFS mounted file system on the front-end machine. This way I only have to maintain the packages in one place. PVM can be found at http://www.epm.ornl.gov/pvm and MPICH can be found at http://www.mcs.anl.gov/mpi/mpich.

### 4. The Cloning Process

There are different ways to clone your first system. One way is to physically connect a drive to an already configured system and then copy the content from one drive to another. Another, maybe more elegant method, is to boot the nodes like diskless clients the first time and let setup scripts partition and copy the system from tar files or another already running system.

The first method is simpler, but it requires a lot of reboots and power shut downs. This is the only way to do it if you don't have a floppy drive in every system. When copying from one drive to another there are two simple ways of doing this:

1. Use dd on the entire drive. If your configured drive is hda and the unconfigured is hdb, you would do "dd if=/dev/hda of =/dev/hdb". This will take care of partitions as well as boot sector and files.

2. If you don't use the same type of drives in your nodes, create tar files of the root and usr partition of the configured system that you set up:

```
# mkdir /usr/images
# tar cvflz /usr/images/root.tar.gz /
# tar dvflz /usr/images/usr.tar.gz /usr --exclude /usr/images
```

The "1" switch is important when doing the tar, to only archive files within the partition. You can now connect drives that will go into the other nodes one by one. Remember to power off the system before connecting and disconnecting drives. Partition and format the drives as you did with the first system. You will need to reboot after partitioning to make sure the drive gets reset properly with the new partitioning tables. After creating the file systems (ex. mkfs -t ext2 /dev/hdc1), you can extract the file systems for the root and usr partition. You are now ready to initialize lilo to update the boot sector. To do this, you need to have your cloned disk seen as the first disk (/dev/hda for IDE and /dev/sda for SCSI). When booting from floppy, at the boot prompt type "*linux root=/dev/hda1*" for IDE drives and "*linux root=/dev/sda1*" for SCSI drives. When the system is running, log in as root and type "*lilo*". This is also a good time to edit the network configuration files listed below.

After creating each clone, they need to be given their own identity. For RedHat and Mandrake Linux, you will have to edit the files:

- /etc/HOSTNAME
- /etc/sysconfig/network
- /etc/sysconfig/network-scripts/ifcfg-eth0

When booting diskless to clone the nodes, you need to use bootp or dhcp. Information about how to set up bootp can be found at http://cesdis.gsfc.nasa.gov/beowulf/howto/bootp.html . The package, NodeCloner.tar.gz, will help you setting up NFS root on the front-end machine. When adding a new node, all we now have to do is add the node to the NodeDatabase file under the NFS-root tree and add the hardware address for its ethernet adapter to /etc/bootptab, send an hangup signal to the bootpd, that should be running on the front-end machine, and we're ready to boot the new node diskless. As soon as the new node is done booting from the floppy, an init script, found in the sbin directory under the NFS-root tree, is run and does the whole cloning process. This script finally reboots the machine when the process is done and the node is ready to compute.

### 5. Where can I find more information?

- Linux links at http://www.linuxlinks.com
- Linux Start at http://www.linuxlinks.com
- Kernelnotes.org at http://kernelnotes.org
- The Beowulf Underground at http://beowulf-underground.org
- The Linux HOWTO Index at http://sunsite.unc.edu/mdw/HOWTO
- RedHat Linux at http://www.redhat.com
- Mandrake Linux at http://www.linux-mandrake.com
- RAID Solutions for Linux at http://linas.org/linux/raid.html
- Linux security at http://www.luv.asn.au/overheads/security/index.html
- Linux Software for Scientists at http://www.llp.fu-berlin.de/baum/linuxlist-a.html
- Scientific Applications on Linux at http://SAL.KachinaTech.COM

- Linux Gazette at http://www.linuxgazette.com/
- Linux Journal's Linux Resources at http://www.ssc.com/linux