

Documentation

July 20, 2004

Pentium 4 Beowulf Cluster

Max Ehammer

ecmax@bgnet.bgsu.edu

Harald Roeck

hrock@bgnet.bgsu.edu

Academic Supervisor

Hassan Rajaei, Ph.D

Department of Computer Science
Bowling Green State University, Ohio

Contents

1	Design	3
1.1	Hardware	3
1.2	Operating System	4
1.3	Future Hardware Configuration	5
1.4	Batch-system	7
2	Server	8
2.1	Partition	8
2.2	Base system	9
2.3	Kernel	9
2.4	Configuration Files	9
2.5	Bootloader	10
2.6	System Logger	11
2.7	Reboot	11
2.8	Additional software	11
2.9	DHCP	11
2.10	TFTP	12
2.11	NFS	12
2.12	PBS/Torque	14
3	Nodes	17
3.1	Base System	17
3.2	Shared Root	18
3.3	PXEgrub	20
3.4	BIOS and Partition	21
3.5	PBS/Torque	22
3.6	mpiexec	25
4	Maintenance	26
4.1	User management	26
4.2	Group management	26

4.3	Software	27
4.4	Useful Scripts	28
4.5	Backup	28
4.6	Installed Tools	30
4.6.1	Printer	30
4.6.2	Message of the Day	30
5	Security	31
5.1	Firewall/Router	31
5.2	User Restrictions	32
A	Usefull Links and Resources	34
B	Kernel	35
B.1	Server Kernel	35
B.2	Nodes Kernel	38
C	Metalog	41
D	DHCP	44
E	PBS/Torque	47

Chapter 1

Design

1.1 Hardware

This cluster is built with 16 computation nodes and 1 server connected to a Ethernet switch. The computation nodes are Dell Optiplex GX270 desktop machines and the server is a Dell Precision 450 desktop. The hardware specification is as follows:

Server:

- 2.4 GHz Xeon CPU, with an additional free socket for a second processor
- 1 GB RAM
- 40 GB IDE Hard Drive
- two 1Gb Ethernet cards

Nodes:

- 2.8 GHz Pentium4 CPU
- 512 MB RAM
- 40 GB IDE Hard Drive
- 1Gb Ethernet

The server needs two NICs, because it is the only machine which is connected to the CS network and which has a public IP address. The first interface is connected to the internal switch, whereas the second is public. The server uses SNAT (source network address translation) to grant Internet access for the nodes. However, the nodes are not visible and not reachable by the public.

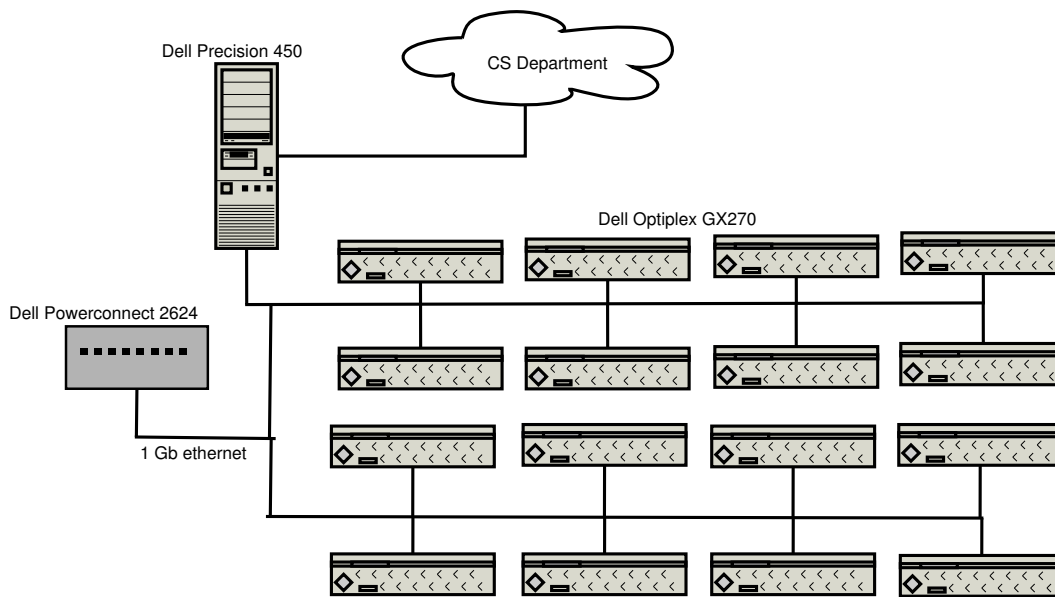


Figure 1.1: Beowulf Cluster

1.2 Operating System

We use the Linux operating system(<http://www.kernel.org>). However, Linux is only the kernel and we have to decide which distribution we want. We use Gentoo www.gentoo.org. Gentoo is a source based distribution, thus we have to download the source code and compile it on our machines.

Gentoo uses a BSD like port system, called `portage`, to assist with compilation and installation of new software. The following is a description of `portage` from the Gentoo web page <http://www.gentoo.org/main/en/about.xml>:

Portage is the heart of Gentoo Linux, and performs many key functions. For one, Portage is the software distribution system for Gentoo Linux. To get the latest software for Gentoo Linux, you type one command: `emerge sync`. This command tells Portage to update your local "Portage tree" over the Internet. Your local Portage tree contains a complete collection of scripts that can be used by Portage to create and install the latest Gentoo packages. Currently, we have nearly 6000 packages in our Portage tree, with new ones being added all the time.

Portage is also a package building and installation system. When you want to install a package, you type `emerge package-name`, at which point Portage automatically builds a custom version of the package to your exact specifications, optimizing it for your hardware and ensuring

that the optional features in the package that you want are enabled – and those you don't want aren't.

We use only two operating systems images. The server has its own OS image, and the nodes share one OS image. Both systems are located on the server hard disk, and the server has to provide the kernel image to the nodes. The reason to use only one image for all nodes is to ease the maintenance of the system. It is implicitly ensured that the nodes are homogenous and identically. When we install new software for the nodes we do it only once and it is immediately available for the user. To guarantee consistency and that the nodes do not interfere, the filesystem must be "read-only" for the computation nodes. We can't allow a node to write on the shared filesystem, because it would affect the other nodes and it would result in undefined and unpredictable behavior of the nodes. The shared "read-only" root filesystem introduces new problems to the system, especially during booting, because some information, e.g. hostname, is unique related to one node. To circumvent these problems we have to change the boot up procedure and we need some local storage at each node. We use about 1GB of the local hard disk and mount it as `/var`. `/var` contains variable data files. This includes spool directories and files, administrative and logging data, and transient and temporary files. Hence, some portions of `/var` are not shareable between different systems. The rest of the local hard disk (38GB) is available for the user as `/tmp`. Additionally the `/usr/local` directory of the server is mounted by the nodes to ease the installation of non-gentoo software on all cluster machines.

The nodes root directory is located on the server under `/opt/diskless/default`, the kernel image and the boot loader is in `/opt/diskless/boot`. We use a separate 2GB partition mounted as `/opt/diskless` to store all data which belongs to the nodes. We reserve 10GB for the server operating system and software. The `/home` directory is another partition with about 26GB available for users, which is shared among all nodes.

1.3 Future Hardware Configuration

This is only a small cluster of Linux machines. We propose a new configuration for a better and a more sophisticated solution. The services of the server should be distributed among different computers, and additionally the nodes should be SMP machines with 2 or more CPUs. Myrinet or Infiniband is a new technology used to connect cluster nodes and to implement low-latency message passing. These interfaces are very expensive, hence several CPUs should share one interface. Instead of one server we should use at least two or more. One login machine where the users

```
/ root partition – 10GB  
/boot partition with kernel and boot-loader – 20MB  
/home partition for user data – 26GB  
/opt/diskless partition for node’s OS – 2GB  
/mnt/usb.disk external hard disk for backups (not mounted by default) – 160GB
```

Figure 1.2: server filesystem hierarchy

```
/ root mounted read-only via nfs from the server directory /opt/diskless/default  
/home user data mounted via nfs from the server  
/usr/local non-gentoo software mounted from the server  
/usr/portage portage tree from the server (not mounted by default)  
/var partition for variable data on the local hard disk – 1GB  
/tmp temporary data on the local hard disk
```

Figure 1.3: node filesystem hierarchy and location

can implement, compile and schedule their jobs, and another file server where the home and the root directories of the other machines are stored. This would result in better performance of file transfer, because the load of the user would not interfere with the file access.

1.4 Batch-system

A user is not allowed to login at the computation nodes directly, instead we are using the server machine as login server too. The user connects to the server `beowulfpub.cs.bgsu.edu` and creates a job file, writes and compiles their programs and finally submits jobs to the batch system.

We are using the batch system Torque to schedule new jobs for the nodes (<http://www.supercluster.org/projects/torque/>).

Torque (Tera-scale Open-source Resource and QUEUE manager) is a resource manager providing control over batch jobs and distributed compute nodes. Torque is based on Open-PBS version 2.3.12 and incorporates scalability, fault tolerance, and feature extension patches provided by NCSA, OSC, the U.S. Dept of Energy, Sandia, PNNL, U of Buffalo, TeraGrid, and many other leading edge HPC organizations.

The use of a batch system is necessary to manage the nodes and to ensure that only one user at a time is using a node. Thus, if a job requests 4 nodes it has to wait until 4 nodes are free. When the job is started the system assigns the job 4 nodes and there are no other jobs, processes or users on these nodes during the execution of the job. This enables the user to test its algorithms and to calculate an exact speedup.

The rest of this documentation is organized as follows: The next chapter will focus on the installation of the server. After that we show how the nodes are setup, and in the last two chapters we will give the necessary information about maintenance of the system and how the system is secured.

Chapter 2

Server

The server is a standard Linux system and we followed the installation documentation of the Gentoo project, which can be found at:

<http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=1>.

This document describes only the choices we do during the installation and not the details about the packages or gentoo in general. For details about the installation procedure we refer to the installation manual.

2.1 Partition

After booting we partition the hard disk for our needs:

File: partition table

Disk /dev/hda: 40.0 GB, 40000000000 bytes

255 heads, 63 sectors/track, 4863 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1		1	3	24066	83	Linux
/dev/hda2		4	128	1004062+	82	Linux swap
/dev/hda3		129	1345	9775552+	83	Linux
/dev/hda4		1346	4863	28258335	5	Extended
/dev/hda5		1346	1595	2008093+	83	Linux
/dev/hda6		1596	4863	26250178+	83	Linux

We use hda1 as boot partition where the kernel and boot loader is located. The second partition, hda2, is a swap partition with 1GB. Hda3 is the root of the filesystem. The diskless OS is located in hda5 and the home partition is hda6.

The used filesystems are ext2 for the boot partition, ext3 for the root and diskless system, and reiserfs for the home directory.

2.2 Base system

After mounting and downloading the hardened stage1 tar-ball we extract the data and set the compiler options to `CFLAGS="-O2 -march=pentium4 -fomit-frame-pointer -pipe"`.

When the necessary data is on the hard disk we chroot to `/mnt/gentoo` and start an emerge sync. This takes a view minutes, but at the end we have a updated portage tree with the latest software and packages available. We used the hardened stage1, therefore it is not necessary to adjust the `USE` variable. We insert only the `-java` flag to speedup the bootstrap process. After bootstrapping is finished we can go on to the second stage and start compiling the system.

2.3 Kernel

We install the new kernel version 2.6.5 of the gentoo distribution, called `gentoo-dev-sources`. Menuconfig is used to configure the kernel. The `.config` file for our kernel is printed in appendix [B](#).

2.4 Configuration Files

The following are the adjusted configuration files for the system. After adjusting the necessary file we have to add the appropriate init scripts with `rc-update`.

File: `/etc/fstab`

```
/dev/hda1 /boot ext2 noauto,noatime 1 1
/dev/hda3 / ext3 noatime 0 0
/dev/hda2 none swap sw 0 0
/dev/hda6 /home reiserfs notail,noatime 0 0
/dev/hda5 /opt/diskless ext3 noatime 0 0
none /proc proc defaults 0 0
none /dev/shm tmpfs defaults 0 0
```

File: `/etc/hostname`

```
beowulf
```

File: /etc/dnsdomainname

```
cs.bgsu.edu
```

File: /etc/conf.d/net

```
iface_eth1="129.1.64.130 broadcast 129.1.64.255 netmask 255.255.255.0"  
iface_eth0="192.168.1.1 broadcast 192.168.1.255 netmask 255.255.255.0"  
  
gateway="eth1/129.1.64.254"
```

File: /etc/hosts

```
127.0.0.1 localhost  
129.1.64.130 beowulfpub.cs.bgsu.edu beowulfpub  
192.168.1.1 beowulf.cs.bgsu.edu beowulf  
192.168.1.101 node01.cs.bgsu.edu node01  
192.168.1.102 node02.cs.bgsu.edu node02  
192.168.1.103 node03.cs.bgsu.edu node03  
192.168.1.104 node04.cs.bgsu.edu node04  
192.168.1.105 node05.cs.bgsu.edu node05  
192.168.1.106 node06.cs.bgsu.edu node06  
192.168.1.107 node07.cs.bgsu.edu node07  
192.168.1.108 node08.cs.bgsu.edu node08  
192.168.1.109 node09.cs.bgsu.edu node09  
192.168.1.110 node10.cs.bgsu.edu node10  
192.168.1.111 node11.cs.bgsu.edu node11  
192.168.1.112 node12.cs.bgsu.edu node12  
192.168.1.113 node13.cs.bgsu.edu node13  
192.168.1.114 node14.cs.bgsu.edu node14  
192.168.1.115 node15.cs.bgsu.edu node15  
192.168.1.116 node16.cs.bgsu.edu node16
```

2.5 Bootloader

We use grub to load the kernel. Here is the configuration file:

File: /boot/grub/grub.conf

```
timeout 3
default 0

title GNU/Linux
root (hd0,0)
kernel /kernel-2.6.5 root=/dev/hda3

title Change the colors
color light-green/brown blink-red/blue
```

2.6 System Logger

The system logger for the server is `metalog`. The configuration file, which can be found in appendix C. We have to adjust this file because we are running several different daemons and a firewall on this server.

2.7 Reboot

Before we reboot we have to set the root password with `passwd`, exit the `chroot` environment and unmount the filesystems.

2.8 Additional software

After the reboot, we go ahead and install some basic software tools. We need `reiserfsprogs` because our `/home` partition is a reiser filesystem. Additionally we install our favorite editors, e.g. `vim` or `emacs`, to edit the upcoming configuration files, and a `ssh` daemon and `ssh` tools from the package `openssh`.

After these steps our server machine is running, and we install the necessary daemons and tools to provide the nodes with a kernel and a filesystem. After the diskless system is running we need to install the job scheduler.

2.9 DHCP

DHCP is needed to provide the nodes with their IP address and an initial PXE image to start the network boot process. We install the `dhcp` package and edit the config-

ration file `/etc/dhcp/dhcpd.conf`. The file is printed in appendix D. Important are the lines

```
File: /etc/dhcp/dhcpd.conf
allow booting;
allow bootp;

option option-150 code 150 = text;
group
  option option-150 "/grub.conf";
  filename "/pxegrub";
...
```

because the nodes download the boot loader and kernel from this server and dhcpd has to provide the ip address as well as the pxe image.

2.10 TFTP

All necessary data and images to boot a diskless system must be provided by a TFTP server. "TFTP is a simple protocol to transfer files, and therefore was named the Trivial File Transfer Protocol or TFTP" (RFC 783). We are using a port of the OpenBSD server, called `tftp-hpa`. The configuration file is:

```
File: /etc/conf.d/in.tftpd
INTFTPD_PATH="/opt/diskless/boot"
INTFTPD_USER="nobody"
INTFTPD_ADDR="192.168.1.1"
INTFTPD_OPTS="-u $INTFTPD_USER -r blksize -l -a $INTFTPD_ADDR \
              -vv -p -c -s $INTFTPD_PATH"
```

This means that the root path for the tftp server is `/opt/diskless/boot` the user of the process is `nobody` and the address to listen for connections is the private interface. The init file is called `in.tftpd` which should be added to the default run level.

2.11 NFS

We are sharing several different portions of our directory tree, e.g. `/home`, with the nodes. The best supported and stable method to use is NFS. We already enabled `nfs`

server support in the kernel, but additionally we need the package `nfs-utils`. The shared directories and option are configured in the file `/etc/exports`:

```
File: /etc/exports
/opt/diskless/default node01(sync,rw,no_root_squash,\
    no_all_squash,no_subtree_check)
/opt/diskless/default node??(sync,ro,no_root_squash,\
    no_all_squash,no_subtree_check)
/home node??(sync,rw,no_root_squash,\
    no_all_squash,no_subtree_check)
/usr/portage node??(sync,rw,no_root_squash,\
    no_all_squash,no_subtree_check)
```

The directory `/opt/diskless/default` is the root directory of the computation nodes. We allow `node01` to mount this directory with read and write access, but all other nodes are not allowed to write on this directory. The standard mount option for `node01` is still read-only, but when we install new software on the computation nodes we remount the root filesystem read-write.

To allow the nodes, and only the nodes, to connect to the portmap daemon we create the following files:

```
File: /etc/hosts.deny
portmap: ALL
```

```
File: /etc/hosts.allow
portmap: 192.168.1.*
```

Additionally to this settings we set the process count of the `nfs` daemon to 16:

```
File: /etc/conf.d/nfs
RPCNFSDCOUNT=16
RPCMOUNTDOPTS=""
RPCSTATDOPTS="-n beowulf"
```

The init script for this daemon is called `nfs` and we add it again to the default runlevel.

When the `dhcpd`, the `in.tftpd` and the `nfs` daemons are running we can start installing the node operating system. For a diskless client these three are the services which must be running on the server.

Now we can reboot the server to test if each service is starting correctly. This isn't necessary, we could just start all services manually, but it's a good test of the server boot procedure. After a successful reboot all services should run and wait for connections from the nodes.

2.12 PBS/Torque

We download the latest stable sources of the torque project at: <http://www.supercluster.org/downloads/torque/>. Additional information about Torque and OpenPBS can be found at the Torque homepage: <http://www.supercluster.org/projects/torque/>

The package is extracted at a directory which is available at all machines(/home). The configuration and compilation is done as follows:

Code:

```
> ./configure --enable-server --enable-clients --enable-mom \  
    --mandir=/usr/share/man --set-server-home=/var/spool/PBS  
> make  
> make install
```

After successful compilation the package is ready to install. We have to install PBS on the nodes and on the server. With the last line we installed it on the server. The binaries are in /usr/local/bin and /usr/local/sbin the configuration files and everything else is in /var/spool/PBS. /usr/local/bin should be added to the users and roots path, and the /usr/local/sbin is added to the roots path only.

Torque consists of three different applications: pbs_server, pbs_sched, and pbs_mom.

We create the environment for the pbs_server with:

Code:

```
> pbs_server -t create
```

This initializes various files, and must be executed only once.

The configuration of the server is done with qmgr and can be found in appendix E. This configuration is based on the OSC system, hence we use three execution queues: parallel, serial and test. The batch queue is only a routing queue and the default queue. If a job requests more than one node it is inserted to the parallel queue, otherwise it's a serial job. The third queue, test, is only for special users. This queue has a higher priority and no limits. The other queues have a time limit of 24 hours and a standard time of one hour.

The scheduler configuration is the file /var/spool/PBS/sched_priv/sched.config. This is the standard module which is shipped with PBS/Torque. It is designed for small environments and as a start configuration. Most of the supercomputer centers are using their own scheduler, e.g. OSC uses Maui. We could switch to this scheduler later on, if we need a more sophisticated scheduler.

We are using a simple *shortest_job_first* algorithm with higher priorities for starving jobs. A job is considered starving if it is more then 12 hours in the queue.

File: /var/spool/PBS/sched_priv/sched.config

```
round_robin: true      all
by_queue: True       prime
by_queue: True       non_prime
strict_fifo: false   ALL
fair_share: false    ALL
help_starving_jobs   true    ALL
sort_queues    true   ALL
load_balancing: false ALL
sort_by: shortest_job_first    ALL
log_filter: 256
dedicated_prefix: ded
max_starve: 12:00:00
half_life: 24:00:00
unknown_shares: 10
sync_time: 1:00:00
```

The `pbs_mom` module is only started at the computation nodes and not on the server. To start Torque at boot time we create an init script and add it to the default boot level:

File: /etc/init.d/pbs

```
#!/sbin/runscript

server=/usr/local/sbin/pbs_server
sched=/usr/local/sbin/pbs_sched
qterm=/usr/local/bin/qterm

depend() {
  need net
}

start() {
  ebegin "Starting pbs_server, pbs_sched and pbs_mom"

  $server
  $sched

  eend $?
}

stop() {
  ebegin "Stopping pbs_server, pbs_sched and pbs_mom"

  start-stop-daemon --stop --quiet --name pbs_sched
  $qterm -t quick

  eend $?
}
```

Chapter 3

Nodes

The computation nodes operating system is installed on the server too. We do not start the nodes at all, until we are finished with the installation of the system on the server. We start with point 5 "Installing the Gentoo Installation Files" of the gentoo documentation. Our target directory is `/opt/diskless/default` instead of `/mnt/gentoo`.

3.1 Base System

We do not need to download all packages again, instead we share the directory of the server. After unpacking the stage-tar-ball we include the server's portage tree by binding it to the new directory with the following command:

Code:

```
> mount --bind /usr/portage /opt/diskless/default/usr/portage
```

This makes the same directory available at two different places. Now we can ignore the parts where we synchronize the portage tree. We basically use the same `make.conf` file as for the server, i.e. we copy that file from the server. Now we proceed as usual: remount the `proc` filesystem and copy the `/etc/resolv.conf` file and finally make a `chroot` to `/opt/diskless/default`. From here we continue installing a new system as we would do on a normal computer. We built a special kernel for the nodes. This kernel should not have any modules and supports only the necessary features; especially important is the NIC driver and the `nfs` root support. The configuration is printed in appendix [B.2](#). We leave the kernel image where it is for now, hence we don't copy it to the standard directory. We are still in the `chroot` environment of our nodes system. The next step is to update the necessary configuration files.

The filesystem information is as follows:

```
File: /etc/fstab
beowulf:/opt/diskless/default / nfs noatime,ro 0 0
beowulf:/home /home nfs noatime 0 0
beowulf:/usr/portage /usr/portage nfs noatime,noauto 0 0
/dev/hda1 none swap sw 0 0
/dev/hda2 /var reiserfs noatime,notail 0 0
/dev/hda3 /tmp reiserfs noatime,notail 0 0

none /proc proc defaults 0 0
none /dev/shm tmpfs defaults 0 0
```

Additionally we substitute `/etc/mtab` with a symlink to `/proc/mounts`. We make sure that only a limited amount of services is started at boot time. The directory `/etc/runlevels/boot` should look like the following:

```
Code:
> ls /etc/runlevels/boot
bootmisc clock hostname keymaps localmount net.lo urandom
```

If there are more or other services in that directory we delete them with `rc-update del service`, where `service` is the name of the file.

3.2 Shared Root

As mentioned before the root filesystem of the nodes is readonly, therefore we have to change the boot procedure. The first step was to change the `mtab` to a symlink to `/proc/mounts`. We change the order and the number of critical boot services. This is done by creating the following file:

```
File: /etc/runlevels/boot/.critical
localmount
hostname
```

Now we edit the file `/etc/init.d/localmount` at line 15. This line mounts all local file systems. We add the `-n` option to the `mount` command. This is necessary to prevent writing to `/etc/mtab`.

```
File: /etc/init.d/localmount -- line 15
mount -n -at nocoda,nonfs,noproc,noncpfs,nosmbfs,noshm >/dev/null
```

Next we change `/sbin/env-update.sh` to be a non executable.

Code:

```
> chmod -x /sbin/env-update.sh
```

The reason for this is, that this script would change some files in `/etc` and we can't allow that.

The file `/etc/hostname` is used to set the hostname of the system. This is of course different for each node, therefore we make a symlink from `/etc/hostname` to `/var/hostname`. Later on we create this file on the local hard disk for each node.

We have to change the init script `/etc/init.d/hostname` too. We comment the lines 53-58 out.

We need a ssh daemon to enable the remote access to the nodes. We restrict the access via ssh in the following file:

File: `/etc/ssh/sshd_config`

```
PermitRootLogin yes
AllowGroups wheel
StrictModes no

RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys

Subsystem sftp /usr/lib/misc/sftp-server
```

We allow only root and the group wheel direct access to the cluster nodes. For details on group management and security issues see chapter 5.

The default runlevel should contain the following modules:

Code:

```
> ls /etc/runlevels/default
domainname local metalog netmount ntpd sshd
```

The `domainname` script must be adjusted otherwise it tries to write to `/etc`.

File: `/etc/init.d/domainname` -- lines 48-64

```

if checkconfig_dns
then
    mydnsdomain="$(< /etc/dnsdomainname) "

    [ ! -f /etc/resolv.conf ] && touch /etc/resolv.conf
    ebegin "Setting DNS domainname to ${mydnsdomain}"
    gawk -v DOMAIN="${mydnsdomain}" \
        'BEGIN { print "domain " DOMAIN }
        $0 !~ /^[[:space:]]*domain/ { print }' \
        /etc/resolv.conf > /tmp/resolv.conf.new
    retval2=$?
    [ "${retval2}" -eq 0 ] \
        && (rm -f /tmp/resolv.conf.new)
    #(mv -f /tmp/resolv.conf.new /etc/resolv.conf) \
    #          || (rm -f /tmp/resolv.conf.new)
    eend ${retval2} "Failed to set the DNS domainname"
fi

```

In addition to this changes we remove `checkroot` from the dependancy list, because the nodes never check the root file system.

The basic system should be installed and we can exit `chroot`. After exiting `chroot` the kernel image of the nodes is copied to `/opt/diskless/boot`, where we store all data which is necessary to boot the nodes.

Code:

```

> cp /opt/diskless/default/usr/src/linux/arch/i386/boot/bzImage \
    /opt/diskless/boot/kernel-2.6.5

```

3.3 PXEgrub

In this section we compile and configure the PXE image for the network booting. We have to download the sources for `grub-0.93`. This version doesn't support our network cards directly, therefore we need some additional patches which can be downloaded at: http://savannah.gnu.org/bugs/?func=detailitem&item_id=6690. We need the package `grub-diskless-patch-2.tar.gz`. Now we go ahead and configure `grub`.

Code:

```
> tar xvzf grub-0.93.tar.gz
> tar xvzf grub-diskless-patch-2.tar.gz
> cd grub-0.93
> patch -p1 < ../grub-diskless-patch-2/grub-0.93-0.94-1.patch
> patch -p1 < ../grub-diskless-patch-2/grub-0.94-1-0001.patch
> patch -p1 < ../grub-diskless-patch-2/grub-0.94-1-0002.patch
> ./configure --enable-diskless --enable-e1000
> make
```

After executing these commands we find the pxe image in the subdirectory `stage2`. The name of the image is `pxegrub`. We copy this image to the boot directory: `/opt/diskless/boot`.

We have to write the configuration file for `pxegrub`:

```
File: /opt/diskless/boot/grub.conf
default 0
timeout 3

title=diskless node gentoo
root (nd)
kernel /kernel-2.6.5 ip=dhcp root=/dev/nfs \
        nfsroot=192.168.1.1:/opt/diskless/default
```

In this file we tell grub where the kernel is and provide the kernel with the necessary parameters. We have to specify the root directory as `nfs` and where it is located. The ip address for the node is provided by the dhcp server. With this configuration each node asks the server twice for its ip address. First, the network card itself needs dhcp to download `pxegrub`, and second, when the operating system loads the network driver.

3.4 BIOS and Partition

Before we actually start the nodes we have to change the BIOS. First, we activate the network boot option, after that, we turn on `wake on lan`.

After changing the bios we boot with a gentoo cd and partition the hard disk with `fdisk`:

File: Partition Table

```
Disk /dev/hda: 40.0 GB, 40000000000 bytes
255 heads, 63 sectors/track, 4863 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1            1           62     497983+   82  Linux swap
/dev/hda2            63          187     1004062+   83  Linux
/dev/hda3           188         4863     37559970   83  Linux
```

The last thing before we reboot is creating the filesystems, we use `reiserfs`, and creating the `hostname` file at the second partition.

Code:

```
> mkreiserfs /dev/hda2
> mkreiserfs /dev/hda3
> mount /dev/hda2 /mnt/gentoo
> echo "nodexx" > /mnt/gentoo/hostname
```

Now we can reboot the node, remove the install disc and start the diskless boot process. If everything goes fine the node downloads the kernel from the server and mounts its root filesystem via `nfs`.

3.5 PBS/Torque

Torque is configured on the server, where `pbs_server` and `pbs_sched` are running. On the nodes we need the `pbs_mom` module. All files for Torque are in `/var/spool/PBS` and therefore not shared across the nodes. We start installing Torque on `node01`. After login to `node01` we change the root filesystem to be `rw`, change to the directory where we built Torque before and install the binaries.

Code:

```
> mount / -o remount,rw
> cd /home/torque-1.0.1.p6
> make install
```

We create a directory in `/etc` where we store the configuration of PBS for the nodes. We need a `config` file and we use an `epilogue` and a `prologue` script. These scripts are executed by root after and before a job is started.

File: `/etc/PBS/config`

```
$clienthost beowulf
$clienthost node01
$clienthost node02
$clienthost node03
$clienthost node04
$clienthost node05
$clienthost node06
$clienthost node07
$clienthost node08
$clienthost node09
$clienthost node10
$clienthost node11
$clienthost node12
$clienthost node13
$clienthost node14
$clienthost node15
$clienthost node16
$logevent 0x1ff

#usecp to remove rcp from file delivery
#usecp beowulf.cs.bgsu.edu:/home /home
```

The following files are based on the OSC scripts, which can be found at www.osc.edu/~troy/pbs.

File: /etc/PBS/prologue

```
#!/bin/sh
# Create TMPDIR on all the nodes
jobid=$1
user=$2
group=$3
nodefile=/var/spool/PBS/aux/$jobid
if [ -r $nodefile ] ; then
    nodes=$(sort $nodefile | uniq)
else
    nodes=localhost
fi
tmp=/tmp/pbstmp.$jobid
for i in $nodes ; do
    ssh $i mkdir -m 700 $tmp \&\& chown $user.$group $tmp
done
exit 0
```

```
File: /etc/PBS/epilogue
#!/bin/sh
# Clear out TMPDIR
jobid=$1
nodefile=/var/spool/PBS/aux/$jobid
if [ -r $nodefile ] ; then
    nodes=$(sort $nodefile | uniq)
else
    nodes=localhost
fi
tmp=/tmp/pbstmp.$jobid
for i in $nodes ; do
    ssh $i rm -rf $tmp
done
exit 0
```

These scripts create a unique directory on each node for temporary storage. The directory is deleted when the job is finished. To provide the user with an environment variable `TMPDIR` we add the following lines to `/etc/profile`:

```
File: /etc/profile
if [ -n "$PBS_ENVIRONMENT" ]
then
    if [ "$PBS_ENVIRONMENT" = PBS_BATCH -o "$PBS_ENVIRONMENT" \
        = PBS_INTERACTIVE ]
    then
        export TMPDIR=/tmp/pbstmp.$PBS_JOBID
    fi
fi
```

This directory is meant as a temporary storage during the job processing.

Finally we create symlinks from `/var/spool/PBS/mom_priv` to the three files in `/etc/PBS`:

```
Code:
> cd /var/spool/PBS/mom_priv
> ln -s /etc/PBS/config
> ln -s /etc/PBS/prologue
> ln -s /etc/PBS/epilogue
```

To distribute the configuration we make a package of the directory `/var/spool/PBS` and copy it to all nodes.

Similar to the server, we create an init script and add it to the default runlevel:

```
File: /etc/init.d/pbs
#!/sbin/runscript
# start and stop the pbs server

mom=/usr/local/sbin/pbs_mom
depend() {
    need net
}

start() {
    ebegin "Starting pbs_mom ..."

    $mom

    eend $?
}

stop() {
    ebegin "Stopping pbs_mom ..."

    start-stop-daemon --stop --quiet --name pbs_mom

    eend $?
}
```

3.6 mpiexec

`mpiexec`, developed by OSC, is a replacement and more for `mpirun`. It interacts with PBS and starts the parallel program on the computation nodes. The installation is easy, because it is include in the portage tree of gentoo:

Code:

```
> emerge mpiexec
```

Chapter 4

Maintenance

4.1 User management

The following shell commands are used to add, to change and to maintain the users on a Linux/Unix system:

useradd Create a new user or update default new user information

userdel Delete a user account and related files

usermod Modify a user account

newusers update and create new users in batch

passwd change user password

chfn change real user name and information

chsh change login shell

The last three commands can be executed by the user itself to change its settings(e.g. login shell). We set an alias for `useradd` to `useradd -m` so that the user home directory is always created. For details and how to use these commands we refer to the man pages.

4.2 Group management

This commands are used to manage user groups on the system.

groupadd Create a new group

groupdel Delete a group

groupmod Modify a group

newgrp log in to a new group

The last command is used by a user to change the actual group id of the current session. For details see the man pages.

These commands are executed on the server. After the user information has changed the script `synccluster` must be executed, which updates the user information on the computation nodes. Password changes for normal users are not necessary to be forwarded to the nodes, because they are not allowed to login to the nodes anyways.

File: `/usr/sbin/synccluster`

```
#!/bin/bash
# syncs passwd, shadow, group with cluster nodes
rsync /etc/passwd /opt/diskless/default/etc/passwd
rsync /etc/shadow /opt/diskless/default/etc/shadow
rsync /etc/group /opt/diskless/default/etc/group
rsync /etc/hosts /opt/diskless/default/etc/hosts
```

4.3 Software

The installation of new software on the cluster needs two basic steps. First, the installation on the server, and then on the nodes.

If we install software supported by Gentoo we use the portage command `emerge`. Useful information about portage and related software can be found at the Gentoo documentation home-page <http://www.gentoo.org/doc/>.

The server will automatically build a binary package, because of the `buildpkg` flag in the portage configuration file `/etc/make.conf`. When the server is finished with compiling and installing the packages, we log in to `node01`. We execute the script `setinstall`, which makes the root directory writable and incorporates the server's portage tree. If we use `emerge -k` to install a package, portage will use a pre-built binary package from the server and extract it. After this is done we execute `unsetinstall` to finish the installation.

Software which is not available in the portage tree is installed by compiling the package and then copying it to the system folders in `/usr/local`. This directory is shared among all nodes and the server. Thus, if we install new software on the server under `/usr/local` it is available on all machines.

File: /usr/sbin/setinstall

```
#!/bin/bash
if [[ `hostname` == node01 ]]
then
    mount /usr/portage
    mount / -o remount,rw
else
    echo only node01 can install new software
fi
```

File: /usr/sbin/unsetinstall

```
#!/bin/bash
if [[ `hostname` == node01 ]]
then
    mount / -o remount,ro
    umount /usr/portage
else
    echo only node01 can install new software
fi
```

4.4 Useful Scripts

To ease maintenance we write a couple of scripts. The first one executes a command on all nodes: `forall`. If no arguments are provided it checks just the status of the nodes, otherwise it executes the arguments.

`shutdownall` is used to shutdown all cluster nodes, whereas `wakeall` can be used to start them.

It's important to shut down the computation before the server, otherwise the nodes would stop working and we have to push the power button on each machine.

4.5 Backup

We use an external USB disc and the program `rdiff-backup` to backup the data. `fcron` is a cron daemon which executes some commands regularly (e.g. daily, weekly) on our server. To schedule a command or script we have to add it to the corresponding directory. e.g. `/etc/cron.daily/` for our backup script.

The user directories are copied to the external disc every night. `rdiff-backup` uses reverse diffs to keep track of the changes in the directories, so we can still recover files lost some time ago. The following script is executed every night at 3:00 am:

```
File: /etc/cron.daily/backup.sh
#!/bin/bash
#backup all home directories to the external disk
#and delete backups older than 3 weeks

mount /mnt/usb_disk &> /dev/null
cd /home
for i in *
do
    rdiff-backup -v0 $i /mnt/usb_disk/home/$i
done

cd /mnt/usb_disk/home
for i in *
do
    rdiff-backup -v0 --remove-older-than 3W $i
done

umount /mnt/usb_disk &> /dev/null
```

This script copies each user directory separately to the USB disc, and deletes old copies, in this case copies older than 3 weeks. If a user and their directory is deleted, a copy of the last data will be available on the usb disc. To restore a directory from the backup the option `-r time` is used. `time` can be an exact time or a interval, e.g. `-r 3D` will restore the directory as it was three days ago.

An example to restore the user directory of hroeck to `/tmp` as it was three days ago:

Code:

```
rdiff-backup -r 3D /mnt/usb_disk/home/hroeck /tmp/hroeck
```

The details can be found in the man pages.

If the root directory should be saved we have to exclude several directories, e.g. `/tmp` or `/dev`. `rdiff-backup` supports the flag `--exclude-filelist filename`. On the external disk is a file `exclude-files`, which contains the directories and files which should not be backed up. To backup the root directory we execute the following commands:

Code:

```
> mount /mnt/usb_disk
> cd /mnt/usb_disk
> rdiff-backup --exclude-filelist exclude-files / server/
> cd
> umount /mnt/usb_disk
```

4.6 Installed Tools

4.6.1 Printer

The printer spooler `lprng` is used to print on the server. We copy the `printcap` file of `bgunix` to `/etc/printcap` and adjust it to our configuration. We change only the location of the spool directory to `/var/spool/lpd`. Hence, all printers are available the same way as at `bgunix`.

4.6.2 Message of the Day

The contents of `/etc/motd` are displayed after a successful login.

Chapter 5

Security

5.1 Firewall/Router

We use shorewall to manage the routing and network filter capabilities of the Linux kernel. All configuration files are located in `/etc/shorewall`.

The server is the only machine with a public IP address. It performs SNAT (Source Network Address Translation) if a node connects to another machine outside the cluster environment. Furthermore, the kernel will drop all packets from outside the cluster which aren't addressed to the tcp port 22(ssh) or to the udp port 123(ntp). On the other side, the access from the the server or nodes to any other machine is not restricted.

Important configuration files are:

File: `/etc/shorewall/interfaces`

#ZONE	INTERFACE	BROADCAST	OPTIONS
net	eth1	129.1.64.255	
loc	eth0	192.168.1.255	dhcp

File: `/etc/shorewall/rules`

#ACTION	SOURCE	DEST	PROTO	DEST PORT
ACCEPT	net	fw	tcp	22
ACCEPT	net	fw	udp	123

File: `/etc/shorewall/policy`

#SOURCE	DEST	POLICY	LOG	LEVEL
loc	net	ACCEPT		
loc	fw	ACCEPT		
fw	net	ACCEPT		
fw	loc	ACCEPT		
net	all	DROP	info	
all	all	REJECT	info	

File: /etc/shorewall/masq

```
#Use this file to define dynamic NAT (Masquerading) and to
#define Source NAT(SNAT).
#INTERFACE          SUBNET          ADDRESS
eth1                 eth0             129.1.64.130
```

Shorewall reads those configuration files and with the help of the iptables utility, Shorewall configures Netfilter to match the requirements.

5.2 User Restrictions

We follow the Gentoo design and introduce three different security levels for users. We have normal users, a privileged group called `wheel`, and the superuser account `root`.

Normal users are not allowed to login to the computation nodes and they underlie several restrictions. First of all a normal user is not allowed to perform `su` to change the user ID. The maximum number of logins at a time is 8 and the number of processes is limited to 64. These limits are defined in the file `/etc/security/limits.conf`. Additional login definitions/restrictions for all users are set in `/etc/login.defs`. This file is read only by `useradd`, hence, changes to this file will take affect to new users only. A more sophisticated method to control user access and application limits is to use PAM(Pluggable Authentication Modules for Linux), but we use the standard configuration of PAM coming with Gentoo.

A member of the group `wheel` has more rights. It's allowed to use `su` to change the user ID or to become super user. `root` isn't allowed to login remotely, thus, `su` is necessary for a restricted group of users. Additionally it's possible to use `ssh` to connect to the computation nodes. Thus, when a member of the `wheel` group changes the password on the server it's necessary to execute the `synccluster` script. The users in the `wheel` group should be limited to these people who are responsible for maintenance.

The superuser(`root`) is allowed to login only to the virtual console 1(`vc/1`), but this

should not be necessary. We suggest to login as a normal user and use `su` to become superuser. This is controlled in the file `/etc/securetty`.

Appendix A

Usefull Links and Resources

www.gentoo.org Gentoo homepage

- forums.gentoo.org
- www.gentoo.org/doc/en
- www.gentoo.org/doc/en/handbook

www.kernel.org Linux Kernel

- www.kernel.org/pub/linux/libs/pam

www.osc.edu Ohio Supercomputer Center

- oscinfo.osc.edu
- www.osc.edu/~troy/pbs

www.supercluster.org/torque PBS – Torque Resource Manager

www-unix.mcs.anl.gov/mpi/mpich MPI implementation

rdiff-backup.stanford.edu Backup Tool

www.shorewall.net Firewall script

Appendix B

Kernel

B.1 Server Kernel

CONFIG_X86=y
 CONFIG_MMU=y
 CONFIG_UID16=y
 CONFIG_GENERIC_ISA_DMA=y

CONFIG_EXPERIMENTAL=y
 CONFIG_CLEAN_COMPILE=y
 CONFIG_STANDALONE=y

CONFIG_SWAP=y
 CONFIG_SYSVIPC=y
 CONFIG_SYSCTL=y
 CONFIG_LOG_BUF_SHIFT=15
 CONFIG_HOTPLUG=y
 CONFIG_IKCONFIG=y
 CONFIG_KALLSYMS=y
 CONFIG_FUTEX=y
 CONFIG_EPOLL=y
 CONFIG_IOSCHED_NOOP=y
 CONFIG_IOSCHED_AS=y
 CONFIG_IOSCHED_DEADLINE=y
 CONFIG_IOSCHED_CFQ=y

CONFIG_MODULES=y
 CONFIG_MODULE_UNLOAD=y
 CONFIG_OBSOLETE_MODPARM=y
 CONFIG_KMOD=y
 CONFIG_STOP_MACHINE=y

CONFIG_X86_PC=y

CONFIG_MPENTIUM4=y
 CONFIG_X86_CMPXCHG=y
 CONFIG_X86_XADD=y
 CONFIG_X86_L1_CACHE_SHIFT=7
 CONFIG_RWSEM_XCHGADD_ALGORITHM=y
 CONFIG_X86_WP_WORKS_OK=y
 CONFIG_X86_INVLPG=y
 CONFIG_X86_BSWAP=y
 CONFIG_X86_POPAD_OK=y
 CONFIG_X86_GOOD_APIC=y
 CONFIG_X86_INTEL_USERCOPY=y
 CONFIG_X86_USE_PPRO_CHECKSUM=y
 CONFIG_SMP=y
 CONFIG_NR_CPUS=2
 CONFIG_PREEMPT=y
 CONFIG_X86_LOCAL_APIC=y
 CONFIG_X86_IO_APIC=y
 CONFIG_X86_TSC=y
 CONFIG_X86_MCE=y
 CONFIG_X86_MCE_NONFATAL=y
 CONFIG_X86_MCE_P4THERMAL=y

CONFIG_NOHIGHMEM=y
 CONFIG_MTRR=y
 CONFIG_IRQBALANCE=y
 CONFIG_HAVE_DEC_LOCK=y

CONFIG_PM=y

CONFIG_ACPI_BOOT=y

CONFIG_APM=m

```

CONFIG_APM_REAL_MODE_POWER_OFF=y      CONFIG_SCSI_QLA2XXX=y

CONFIG_PCI=y                            CONFIG_NET=y
CONFIG_PCI_GOANY=y
CONFIG_PCI_BIOS=y                       CONFIG_PACKET=y
CONFIG_PCI_DIRECT=y                    CONFIG_UNIX=y
CONFIG_PCI_MMCONFIG=y                  CONFIG_INET=y
CONFIG_PCI_LEGACY_PROC=y               CONFIG_IP_MULTICAST=y
CONFIG_PCI_NAMES=y

CONFIG_NETFILTER=y

CONFIG_BINFMT_ELF=y
CONFIG_BINFMT_AOUT=y
CONFIG_BINFMT_MISC=y

CONFIG_IP_NF_CONNTRACK=m
CONFIG_IP_NF_TFTP=m
CONFIG_IP_NF_QUEUE=m
CONFIG_IP_NF_IPTABLES=m
CONFIG_IP_NF_MATCH_LIMIT=m
CONFIG_IP_NF_MATCH_IPRANGE=m
CONFIG_IP_NF_MATCH_MAC=m
CONFIG_IP_NF_MATCH_PKTTYPE=m
CONFIG_IP_NF_MATCH_MARK=m
CONFIG_IP_NF_MATCH_MULTIPORT=m
CONFIG_IP_NF_MATCH_TOS=m
CONFIG_IP_NF_MATCH_RECENT=m
CONFIG_IP_NF_MATCH_ECN=m
CONFIG_IP_NF_MATCH_DSCP=m
CONFIG_IP_NF_MATCH_AH_ESP=m
CONFIG_IP_NF_MATCH_LENGTH=m
CONFIG_IP_NF_MATCH_TTL=m
CONFIG_IP_NF_MATCH_TCPMSS=m
CONFIG_IP_NF_MATCH_HELPER=m
CONFIG_IP_NF_MATCH_STATE=m
CONFIG_IP_NF_MATCH_CONNTRACK=m
CONFIG_IP_NF_MATCH_OWNER=m
CONFIG_IP_NF_FILTER=m
CONFIG_IP_NF_TARGET_REJECT=m
CONFIG_IP_NF_NAT=m
CONFIG_IP_NF_NAT_NEEDED=y
CONFIG_IP_NF_TARGET_MASQUERADE=m
CONFIG_IP_NF_TARGET_REDIRECT=m
CONFIG_IP_NF_TARGET_NETMAP=m
CONFIG_IP_NF_TARGET_SAME=m
CONFIG_IP_NF_NAT_TFTP=m
CONFIG_IP_NF_MANGLE=m
CONFIG_IP_NF_TARGET_TOS=m

CONFIG_BLK_DEV_FD=y
CONFIG_BLK_DEV_LOOP=m

CONFIG_IDE=y
CONFIG_BLK_DEV_IDE=y

CONFIG_BLK_DEV_IDEDISK=y
CONFIG_IDEDISK_MULTI_MODE=y
CONFIG_BLK_DEV_IDECD=y
CONFIG_BLK_DEV_IDESCSI=m
CONFIG_IDE_TASKFILE_IO=y

CONFIG_IDE_GENERIC=y
CONFIG_BLK_DEV_CMD640=y
CONFIG_BLK_DEV_IDEPCI=y
CONFIG_IDEPCI_SHARE_IRQ=y
CONFIG_BLK_DEV_GENERIC=y
CONFIG_BLK_DEV_OPTI621=y
CONFIG_BLK_DEV_IDEDMA_PCI=y
CONFIG_IDEDMA_PCI_AUTO=y
CONFIG_BLK_DEV_ADMA=y
CONFIG_BLK_DEV_PIIX=y
CONFIG_BLK_DEV_IDEDMA=y
CONFIG_IDEDMA_AUTO=y

CONFIG_SCSI=y

CONFIG_BLK_DEV_SD=m
CONFIG_CHR_DEV_SG=m

```

CONFIG_IP_NF_TARGET_ECN=m	CONFIG_LEGACY_PTY_COUNT=256
CONFIG_IP_NF_TARGET_DSCP=m	
CONFIG_IP_NF_TARGET_MARK=m	CONFIG_RTC=m
CONFIG_IP_NF_TARGET_CLASSIFY=m	CONFIG_GEN_RTC=m
CONFIG_IP_NF_TARGET_LOG=m	CONFIG_GEN_RTC_X=y
CONFIG_IP_NF_TARGET_ULOG=m	
CONFIG_IP_NF_TARGET_TCPMSS=m	CONFIG_AGP=y
CONFIG_IP_NF_ARPTABLES=m	CONFIG_AGP_INTEL=y
CONFIG_IP_NF_ARPFILTER=m	
CONFIG_IP_NF_ARP_MANGLE=m	CONFIG_VIDEO_SELECT=y
CONFIG_NETDEVICES=y	CONFIG_VGA_CONSOLE=y
	CONFIG_DUMMY_CONSOLE=y
CONFIG_DUMMY=m	
	CONFIG_SPEAKUP_DEFAULT="none"
CONFIG_E1000=m	
CONFIG_E1000_NAPI=y	CONFIG_USB=m
CONFIG_INPUT=y	CONFIG_USB_DEVICEFS=y
CONFIG_INPUT_MOUSEDEV=y	CONFIG_USB_EHCI_HCD=m
CONFIG_INPUT_MOUSEDEV_PSAUX=y	CONFIG_USB_UHCI_HCD=m
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024	
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768	CONFIG_USB_PRINTER=m
	CONFIG_USB_STORAGE=m
CONFIG_SOUND_GAMEPORT=y	
CONFIG_SERIO=y	CONFIG_EXT2_FS=y
CONFIG_SERIO_I8042=y	CONFIG_EXT3_FS=y
	CONFIG_JBD=y
CONFIG_INPUT_KEYBOARD=y	CONFIG_REISERFS_FS=m
CONFIG_KEYBOARD_ATKBD=y	
CONFIG_INPUT_MOUSE=y	CONFIG_ISO9660_FS=y
CONFIG_MOUSE_PS2=y	CONFIG_JOLIET=y
CONFIG_VT=y	CONFIG_PROC_FS=y
CONFIG_VT_CONSOLE=y	CONFIG_PROC_KCORE=y
CONFIG_HW_CONSOLE=y	CONFIG_DEVFS_FS=y
	CONFIG_DEVFS_MOUNT=y
CONFIG_SERIAL_8250=y	CONFIG_DEVPTS_FS_XATTR=y
CONFIG_SERIAL_8250_NR_UARTS=4	CONFIG_TMPFS=y
	CONFIG_RAMFS=y
CONFIG_SERIAL_CORE=y	
CONFIG_UNIX98_PTYS=y	CONFIG_NFS_FS=m
CONFIG_LEGACY_PTYS=y	CONFIG_NFS_V4=y

```

CONFIG_NFSD=m
CONFIG_LOCKD=m
CONFIG_EXPORTFS=m
CONFIG_SUNRPC=m
CONFIG_SUNRPC_GSS=m
CONFIG_RPCSEC_GSS_KRB5=m

CONFIG_MSDOS_PARTITION=y

CONFIG_NLS=y
CONFIG_NLS_DEFAULT="iso8859-1"
CONFIG_NLS_CODEPAGE_437=y
CONFIG_NLS_ISO8859_1=y
CONFIG_NLS_ISO8859_15=y

CONFIG_EARLY_PRINTK=y
CONFIG_X86_FIND_SMP_CONFIG=y
CONFIG_X86_MPPARSE=y

CONFIG_SECURITY=y
CONFIG_SECURITY_CAPABILITIES=y

CONFIG_CRYPTOD=y
CONFIG_CRYPTOD_MD5=m
CONFIG_CRYPTOD_DES=m

CONFIG_CRC32=m
CONFIG_X86_SMP=y
CONFIG_X86_HT=y
CONFIG_X86_BIOS_REBOOT=y
CONFIG_X86_TRAMPOLINE=y
CONFIG_PC=y

CONFIG_STANDALONE=y
CONFIG_BROKEN_ON_SMP=y

CONFIG_SWAP=y
CONFIG_SYSVIPC=y
CONFIG_SYSCTL=y
CONFIG_LOG_BUF_SHIFT=14
CONFIG_KALLSYMS=y
CONFIG_FUTEX=y
CONFIG_EPOLL=y
CONFIG_IOSCHED_NOOP=y
CONFIG_IOSCHED_AS=y
CONFIG_IOSCHED_DEADLINE=y
CONFIG_IOSCHED_CFQ=y

CONFIG_X86_PC=y
CONFIG_MPENTIUMII=y
CONFIG_X86_CMPXCHG=y
CONFIG_X86_XADD=y
CONFIG_X86_L1_CACHE_SHIFT=5
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_X86_GOOD_APIC=y
CONFIG_X86_INTEL_USERCOPY=y
CONFIG_X86_USE_PPRO_CHECKSUM=y
CONFIG_PREEMPT=y
CONFIG_X86_TSC=y
CONFIG_X86_MCE=y
CONFIG_X86_MCE_NONFATAL=y

CONFIG_NOHIGHMEM=y
CONFIG_MTRR=y
CONFIG_HAVE_DEC_LOCK=y

CONFIG_X86=y
CONFIG_MMU=y
CONFIG_UID16=y
CONFIG_GENERIC_ISA_DMA=y

CONFIG_EXPERIMENTAL=y
CONFIG_CLEAN_COMPILE=y
CONFIG_ACPI=y
CONFIG_ACPI_BOOT=y
CONFIG_ACPI_INTERPRETER=y
CONFIG_ACPI_BUTTON=y
CONFIG_ACPI_FAN=y

```

B.2 Nodes Kernel

CONFIG_ACPI_PROCESSOR=y	CONFIG_IDEDMA_AUTO=y
CONFIG_ACPI_THERMAL=y	
CONFIG_ACPI_BUS=y	CONFIG_NET=y
CONFIG_ACPI_EC=y	
CONFIG_ACPI_POWER=y	CONFIG_PACKET=y
CONFIG_ACPI_PCI=y	CONFIG_UNIX=y
CONFIG_ACPI_SYSTEM=y	CONFIG_INET=y
	CONFIG_IP_MULTICAST=y
CONFIG_PCI=y	CONFIG_IP_PNP=y
CONFIG_PCI_GOANY=y	CONFIG_IP_PNP_DHCP=y
CONFIG_PCI_BIOS=y	CONFIG_IP_PNP_BOOTP=y
CONFIG_PCI_DIRECT=y	CONFIG_IP_PNP_RARP=y
CONFIG_PCI_MMCONFIG=y	
CONFIG_PCI_LEGACY_PROC=y	CONFIG_NETDEVICES=y
CONFIG_PCI_NAMES=y	
	CONFIG_DUMMY=y
CONFIG_BINFMT_ELF=y	
CONFIG_BINFMT_AOUT=y	CONFIG_E1000=y
CONFIG_BINFMT_MISC=y	
	CONFIG_NETCONSOLE=y
CONFIG_PARPORT=y	
CONFIG_PARPORT_PC=y	CONFIG_NETPOLL=y
CONFIG_PARPORT_PC_CML1=y	CONFIG_NET_POLL_CONTROLLER=y
CONFIG_BLK_DEV_LOOP=y	CONFIG_INPUT=y
CONFIG_IDE=y	CONFIG_INPUT_MOUSEDEV=y
CONFIG_BLK_DEV_IDE=y	CONFIG_INPUT_MOUSEDEV_PSAUX=y
	CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_BLK_DEV_IDEDISK=y	CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768
CONFIG_IDEDISK_MULTI_MODE=y	
CONFIG_BLK_DEV_IDECD=y	CONFIG_SOUND_GAMEPORT=y
CONFIG_IDE_TASKFILE_IO=y	CONFIG_SERIO=y
	CONFIG_SERIO_I8042=y
CONFIG_IDE_GENERIC=y	
CONFIG_BLK_DEV_CMD640=y	CONFIG_INPUT_KEYBOARD=y
CONFIG_BLK_DEV_IDEPCI=y	CONFIG_KEYBOARD_ATKBD=y
CONFIG_IDEPCI_SHARE_IRQ=y	
CONFIG_BLK_DEV_GENERIC=y	CONFIG_VT=y
CONFIG_BLK_DEV_IDEDMA_PCI=y	CONFIG_VT_CONSOLE=y
CONFIG_IDEDMA_PCI_AUTO=y	CONFIG_HW_CONSOLE=y
CONFIG_BLK_DEV_ADMA=y	
CONFIG_BLK_DEV_PIIIX=y	CONFIG_SERIAL_8250=y
CONFIG_BLK_DEV_IDEDMA=y	CONFIG_SERIAL_8250_NR_UARTS=4

```
CONFIG_PC=y
CONFIG_SERIAL_CORE=y
CONFIG_UNIX98_PTYS=y
CONFIG_LEGACY_PTYS=y
CONFIG_LEGACY_PTY_COUNT=256

CONFIG_RTC=y

CONFIG_AGP=y
CONFIG_AGP_INTEL=y

CONFIG_VGA_CONSOLE=y
CONFIG_DUMMY_CONSOLE=y

CONFIG_SPEAKUP_DEFAULT="none"

CONFIG_REISERFS_FS=y

CONFIG_PROC_FS=y
CONFIG_PROC_KCORE=y
CONFIG_DEVFS_FS=y
CONFIG_DEVFS_MOUNT=y
CONFIG_DEVPTS_FS_XATTR=y
CONFIG_TMPFS=y
CONFIG_RAMFS=y

CONFIG_NFS_FS=y
CONFIG_NFS_V3=y
CONFIG_ROOT_NFS=y
CONFIG_LOCKD=y
CONFIG_LOCKD_V4=y
CONFIG_SUNRPC=y

CONFIG_MSDOS_PARTITION=y

CONFIG_NLS=y
CONFIG_NLS_DEFAULT="iso8859-1"
CONFIG_NLS_CODEPAGE_437=y
CONFIG_NLS_ISO8859_1=y

CONFIG_EARLY_PRINTK=y

CONFIG_CRC32=y
CONFIG_X86_BIOS_REBOOT=y
```

Appendix C

Metalog

This is file /etc/metalog/metalog.conf:

```
maxsize = 100000
maxtime = 86400
maxfiles = 5
```

Kernel messages :

```
facility = "kern"
neg_regex = "(Shorewall)"
logdir = "/var/log/kernel"
```

Crond :

```
facility = "cron"
logdir = "/var/log/crond"
```

Dudes firewalled by IPTrap :

```
program = "iptrap"
logdir = "/var/log/iptrap"
```

Shorewall:

```
facility = "kern"
regex = "(Shorewall)"
logdir = "/var/log/shorewall"
```

Password failures :

```
regex = "(password|login|authentication)\s+(fail|invalid) "  
regex = "(failed|invalid)\s+(password|login|authentication) "  
regex = "ILLEGAL ROOT LOGIN"  
logdir = "/var/log/pwdfail"
```

FTP Server :

```
program = "pure-ftpd"  
logdir = "/var/log/ftpd"
```

SSH Server :

```
program = "sshd"  
logdir = "/var/log/sshd"
```

Telnet :

```
program = "login"  
logdir = "/var/log/telnet"
```

Imap :

```
program = "/usr/sbin/imapd"  
logdir = "/var/log/imap"
```

POP Toaster :

```
program = "/usr/sbin/ipop3d"  
logdir = "/var/log/pop"
```

DHCP :

```
program = "dhcpd"  
logdir = "/var/log/dhcp"
```

Mail :

```
facility = "mail"  
logdir = "/var/log/mail"
```

Everything important :

```
facility = "*"   
neg_regex= "(Shorewall:)"
```

```
minimum = 5
logdir   = "/var/log/everything"
```

Everything very important :

```
facility = "*"
minimum = 1
logdir   = "/var/log/critical"
```

#console logging :

#

```
facility = "*"
command = "/usr/sbin/consolelog.sh"
```

Appendix D

DHCP

This is the contents of the file `/etc/dhcp/dhcpd.conf`:

```
option domain-name "cs.bgsu.edu";
option domain-name-servers 129.1.2.2, 129.1.2.3;
option subnet-mask 255.255.255.0;
option routers 192.168.1.1;

ddns-update-style none;
allow booting;
allow bootp;

default-lease-time 600;
max-lease-time 7200;
subnet 192.168.1.0 netmask 255.255.255.0 {
}

option option-150 code 150 = text;

group {
    option option-150 "/grub.conf";
    filename "/pxegrub";

    host Node01 {
        hardware ethernet 00:0D:56:95:E7:5A;
        fixed-address 192.168.1.101;
        option host-name "Node01";
    }
    host Node02 {
        hardware ethernet 00:0D:56:95:E6:B4;
```

```
fixed-address 192.168.1.102;
option host-name "Node02";
}
host Node03 {
    hardware ethernet 00:0D:56:95:E8:97;
fixed-address 192.168.1.103;
option host-name "Node03";
}
host Node04 {
    hardware ethernet 00:0D:56:95:9A:67;
fixed-address 192.168.1.104;
option host-name "Node04";
}
host Node05 {
    hardware ethernet 00:0D:56:95:E7:76;
fixed-address 192.168.1.105;
option host-name "Node05";
}
host Node06 {
    hardware ethernet 00:0D:56:95:E8:CA;
fixed-address 192.168.1.106;
option host-name "Node06";
}
host Node07 {
    hardware ethernet 00:0D:56:95:EB:FF;
fixed-address 192.168.1.107;
option host-name "Node07";
}
host Node08 {
    hardware ethernet 00:0D:56:96:15:1A;
fixed-address 192.168.1.108;
option host-name "Node08";
}
host Node09 {
    hardware ethernet 00:0D:56:95:E6:59;
fixed-address 192.168.1.109;
option host-name "Node09";
}
host Node10 {
    hardware ethernet 00:0D:56:96:1B:13;
fixed-address 192.168.1.110;
option host-name "Node10";
}
host Node11 {
```

```
    hardware ethernet 00:0D:56:96:1A:C6;
fixed-address 192.168.1.111;
option host-name  "Node11";
}
host Node12 {
    hardware ethernet 00:0D:56:95:E6:E6;
fixed-address 192.168.1.112;
option host-name  "Node12";
}
host Node13 {
    hardware ethernet 00:0D:56:96:1A:5E;
fixed-address 192.168.1.113;
option host-name  "Node13";
}
host Node14 {
    hardware ethernet 00:0D:56:95:E9:E2;
fixed-address 192.168.1.114;
option host-name  "Node14";
}
host Node15 {
    hardware ethernet 00:0D:56:96:1A:A0;
fixed-address 192.168.1.115;
option host-name  "Node15";
}
host Node16 {
    hardware ethernet 00:0D:56:96:14:37;
fixed-address 192.168.1.116;
option host-name  "Node16";
}
}
```

Appendix E

PBS/Torque

This is the output of `qmgr -c "print server"`:

```
#
# Create queues and set their attributes.
#
#
# Create and define queue batch
#
create queue batch
set queue batch queue_type = Route
set queue batch max_running = 8
set queue batch route_destinations = serial
set queue batch route_destinations += parallel
set queue batch enabled = True
set queue batch started = True
#
# Create and define queue parallel
#
create queue parallel
set queue parallel queue_type = Execution
set queue parallel Priority = 50
set queue parallel resources_max.nodect = 16
set queue parallel resources_max.nodes = 16
set queue parallel resources_max.walltime = 72:00:00
set queue parallel resources_min.nodect = 2
set queue parallel resources_default.nodect = 2
set queue parallel resources_default.nodes = 2
set queue parallel resources_default.walltime = 01:00:00
set queue parallel enabled = True
```

```
set queue parallel started = True
#
# Create and define queue test
#
create queue test
set queue test queue_type = Execution
set queue test Priority = 80
set queue test acl_user_enable = True
set queue test acl_users = ecmx
set queue test acl_users += hroeck
set queue test enabled = True
set queue test started = True
#
# Create and define queue serial
#
create queue serial
set queue serial queue_type = Execution
set queue serial Priority = 50
set queue serial resources_max.nodect = 1
set queue serial resources_max.nodes = 1
set queue serial resources_max.walltime = 72:00:00
set queue serial resources_default.nodect = 1
set queue serial resources_default.nodes = 1
set queue serial resources_default.walltime = 01:00:00
set queue serial enabled = True
set queue serial started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_host_enable = True
set server acl_hosts = *.cs.bgsu.edu
set server acl_user_enable = False
set server managers = ecmx@*.cs.bgsu.edu
set server managers += hroeck@*.cs.bgsu.edu
set server managers += lweihl@*.cs.bgsu.edu
set server managers += rajaei@*.cs.bgsu.edu
set server default_queue = batch
set server log_events = 127
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
set server node_ping_rate = 300
set server node_check_rate = 600
```

```
set server default_node = 1
```

Configuration of the scheduler:

```
round_robin: false all
by_queue: True prime
by_queue: True non_prime
strict_fifo: false ALL
fair_share: false ALL
help_starving_jobs true ALL
sort_queues true ALL
load_balancing: true ALL
sort_by: shortest_job_first ALL
log_filter: 256
dedicated_prefix: ded
max_starve: 12:00:00
half_life: 24:00:00
unknown_shares: 10
sync_time: 1:00:00
```