Building a Beowulf cluster

Åsmund Ødegård

April 4, 2001

1 Introduction

The main part of the introduction is only contained in the slides for this session. Some of the acronyms and names in this paper may be unknown. In Appendix B we includ short descriptions for some of them. Most of this is taken from "whatis" [6]

2 Outline of the installation

- Install linux on a PC
- Configure the PC to act as a install–server for the cluster
- Wire up the network if that isn't done already
- Install linux on the rest of the nodes
- Configure one PC, e.g the install–server, to be a server for your cluster.

These are the main steps required to build a linux cluster, but each step can be done in many different ways. How you prefer to do it, depends mainly on personal taste, though. Therefor, I will translate the given outline into this list:

- Install Debian GNU/Linux on a PC
- Install and configure "FAI" on the PC
- Build the "FAI" boot–floppy
- Assemble hardware information, and finalize the "FAI" configuration
- Boot each node with the boot–floppy
- Install and configure a queue system and software for running parallel jobs on your cluster

3 Debian

The choice of Linux distribution is most of all a matter of personal taste. I prefer the Debian distribution for various reasons. So, the first step in the cluster–building process is to pick one of the PCs as a install–server, and install Debian onto it, as follows:

- Make sure that the computer can boot from cdrom. You may have to enter the BIOS configuration-system and add cdrom before harddisk in the boot–sequence.
- Insert the Debian cdrom, and start the computer. Press «Enter» at the boot–prompt, and the installation starts right away.
- Configure the keyboard according to the present system.

- Partition the harddisk. We will use a disk–partition program called `cfdisk` for this task, the program starts when you select this menuentry. Using the up and down–arrows you select the disk–regions and the action–commands at the bottom of the screen are selected using the left and right arrows. The disk may already have been partitioned by an earlier installation of either windows or linux, you will see this as a list of partitions and no free space. If this is the case, we start by deleting all partitions by using the "[delete]" action. Then we will create a new set of partitions according to the following scheme:

Primary partitions:	Logical partitions:
hda1 = 512MB, Type Swap (82)	hda5 = 250MB, for /tmp
hda2 = 100MB, for root (/)	hda6 = 2000MB, for /usr
hda3 = 500MB, for /var	hda7 = the rest, for /home

  Each partition is created by selecting the free space, use the "[new]" action, and set the size. Place each partition at the beginning of the disk. For the first partition we need to use the "[type]" action to set the type to swap. Remark that some of the partitions are primary and some are logical. The reason for this is that a system can only have 4 primary partitions. If you need more than that, you have to create an extended partition, which is a special primary partition, with the extra capability to hold several logical partitions.

- Choose "[write]" when the information is entered, answer "Yes" and the choose "[quit]". If no partition is made bootable (with the "[bootable]" action), the program will give a warning about Dos MBR[1] being unable to boot the system. Since we are going to use a Linux MBR, this is ok.

- Initialize all partitions according to the scheme above. Choose to not retain 2.0 compatibility, accept other defaults.

- Install OS Kernel and Modules. Choose cdrom and accept defaults, leave the prompt for a path blank.

- Enter the "Configure Device Driver Modules."
  - Skip the question about loading modules from a floppy by answering "yes".
  - Browser to the "net" section and locate the "3c59c" module. This corresponds to the 3com network card which is installed in our PC.
  - Press <Enter> to install the module. Accept all default values by hitting <Enter> a few more times

- Next we configure the network
  - The hostname should be set to "server"
  - We do not want to use any DHCP service
  - Set the IP number: 10.0.0.1
  - and netmask: 255.255.255.0
  - Leave all other fields blank
  - For more info on configuration of network on Linux, we refer the reader to [3]

- Install base system from cdrom, accept defaults, and leave the path input blank. Configure the base system.

- Make linux bootable from harddisk, install lilo in MBR, install boot–record/boot–manager

- Remove the cdrom and reboot. We skip the "make boot floppy" entry for now.

[1]MBR: Master Boot Record

After the reboot, there is some additional configuration steps to perform. We don't enable MD5 and Shadow passwords. For the root password, we use "fai." The next step is to create a user account. It's a good habit to never use the root account unless it is strictly necessary, so we create a user "ws" with password "ws" for the winterschool.

Choose to remove "pcmcia" and answer "no" to the question about using PPP. Insert the cdrom again and choose the "cdrom" method for the base–installation. There is no need for adding other apt sources. Choose the "simple" method for installation. A "task selection" dialog will show some packages–set we may choose to install. We will just mark all tasks, using the spacebar. A regular Debian installation will have many more choices in this dialog. After this, the system will install a base Debian system. You have to hit <Enter> a few times. During this information you will be prompted for configuration parameters for a few packages. In these cases, consult section 3.1 below.

When the installation is finished, log in as root with the password you provided. Then we edit the file /etc/apt/sources.list to use the mounted cdrom instead of just the cdrom device as the source for packages. To do this, replace the `deb cdrom:` line with a `deb file:/cdrom` line. The list of sources is ok, but we remove the label of the cdrom in brackets, that is everything after "cdrom" up too and including "]\". The resulting line should look like this:

```
deb file:/cdrom stable main contrib local non-US/ma...
```

We are now ready to install the rest of the needed packages. For simplicity we provide a list of necessary packages on the cdrom, such that we easily install them. First, make sure that the cdrom is mounted:

```
server:~# mount cdrom
server:~# apt-get update
server:~# for i in $(cat /cdrom/clusterfiles/packages) ; do
>apt-get -y install $i
>done
```

During this installation, the system will prompt you for some configuration–parameters for a few packages. We provide the necessary information below.

This process install a new version of the Linux kernel on the PC, so reboot to run the new kernel:

```
server:~# umount /cdrom  #(and remove it)
server:~# shutdown -r now
```

After the reboot, login as root again, insert and mount the cdrom.

3.1 Configuration data

- *exim:* We configure as NON–NETWorked system. Answer "4" at the first question, and give a reasonable username when asked for a sys.admin username.
- *libpaperg:* Answer a4 for the papersize.
- *debconf:* Choose "Text" as frontend, and "Critical" as priority level.
- *kernel-image:* This ask whether you want to use existing lilo configuration. Just accept the defaults.
- *NIS:* You will be asked to give a NIS domainname. We will use the name "winterschool".
- *ntp:* Since we don't have any external servers, we use "localhost." In production you want something else!
- *Xservers:* We only want the SVGA server, so "N" to the others and "Y" to the SVGA server. This doesn't matter, we will probably not use them anyway.

4 FAI

The next step is to install and configure "FAI" (Fully Automatic Installation) on the server. This is a system which simplifies the installation of Debian on many computers. Note that this is only one option for cluster installation, others include Scyld [4], Extreme Linux, SCore and others. These are mostly based on the RedHat distribution, so pick the system which suits your taste. Here we list the basic steps for configuration of the system, for more complete information, we refer the reader to Appendix A.

We need to assign hostnames and IP–numbers to all nodes in our cluster. For simplicity we use this scheme:

Hostname	IP–number
node01	10.0.0.101
node02	10.0.0.102
...	

- Install the fai debian-package on the server with apt-get: `apt-get install fai`. This is probably already done, you can check with

```
dpkg -get-selections | grep fai
```

- Review the configuration file /etc/fai.conf

- Make the directory /usr/lib/fai/kernel and copy the kernels from /cdrom/clusterfiles/ into this directory.

```
server:~# mkdir -p /usr/lib/fai/kernel
server:~# cp -a /cdrom/clusterfiles/kernel* /usr/lib/fai/kernel
```

- Run `/usr/sbin/fai-setup`. This will create the directory `/usr/lib/fai/nfsroot` which will be used during installation of the cluster–nodes.

- Copy all files from /usr/share/doc/fai/templates/ to `$FAI_CONFIGDIR`, which is set in /etc/fai.conf (default value is /usr/local/share/fai/). These files defines the configuration of the nodes. We don't need to do any changes for the winterschool, but a brief explanation of the system is given in Appendix A

```
server:~# cp -pR /usr/.../templates/* /usr/.../fai/
```

- Add all nodes in the cluster to /root/.rhosts and /etc/hosts.equiv, each hostname on a separate line.

- Edit /usr/lib/fai/nfsroot/etc/hosts to list the server and all nodes. The resulting file should look like this:

```
127.0.0.1    localhost
10.0.0.1     server
10.0.0.101   node01
10.0.0.102   node02
10.0.0.103   node03
```

- After the configuration is finished, insert a blank floppy in the floppy–drive and run `/usr/sbin/make-fai-bootfloppy`.

5 NIS & NFS

Before we can finish the installation of linux onto the nodes, we have to configure NIS and NFS on the server. NIS is a system for sharing information about users and groups between computers and NFS is the Network FileSystem, which will be used to share diskpartitions on the cluster. We will describe a simple configuration which is sufficient for our needs. For a more thorough description of these tools, consult e.g the NIS–HOWTO and NFS–HOWTO, see [2].

5.1 NIS

To configure NIS, follows these steps.

- Make sure that the NIS domainname is set in /etc/defaultdomain. This is probably already set to "winterschool" during installation.
- Make sure that the server and all nodes are listed in /etc/hosts.
- Edit /etc/init.d/nis and set `NISSERVER=master`. This makes this host the master NIS–server for the NIS–domain. After this, restart NIS with

  ```
  server:~# /etc/init.d/nis restart
  ```

- Make sure that you have a /etc/networks file. Just use `touch` to create one if you don't already have one – the NIS initiating procedure will not build network "NIS maps" if this file is not present.
- Edit the file /etc/netgroup and make sure that all nodes and the server is added to the *faiclients* netgroup. The syntax is like this:

  ```
  faiclients (server,,) (node01,,) (node02,,) ...
  ```

- Run `/usr/lib/yp/ypinit -m` to create all "NIS maps", based on information in /etc/passwd, /etc/group and /etc/netgroup among others. The command prompts for some input. Make sure that the host *"server"* is listed as NIS server, then type «CTRL-d» and «Enter» at the next question to accept the default.

Remark that you have to update the "NIS maps" each time a user is added to your system. You also have to update these maps when a user change password using the standard `passwd` tool. The maps are updated by running `make` in the /var/yp directory. There exit tools like `yppasswd` the user could use to change password in such a way that this information is automatically inserted in the NIS maps.

5.2 NFS

The configuration of NFS is very simple, and is placed in /etc/exports. In our case, we will only export to the hosts in the *faiclients* NIS netgroup. The syntax for this is

```
/dir/to/export @nis-netgroup(options)
```

In out case this become e.g

```
/cdrom @faiclients(ro,no_root_squash)
```

The options we need is `rw`, `ro` and `no_root_squash`. The file should look like this:

```
/usr/local/share/fai @faiclients(ro)
/usr/lib/fai/nfsroot @faiclients(ro,no_root_squash)
/usr @faiclients(ro,no_root_squash)
/home @faiclients(rw,no_root_squash)
/cdrom @faiclients(ro,no_root_squash)
```

Restart the NFS service after the configfile is edited:

```
server:~# /etc/init.d/nfs-server restart
```

5.3 Remark:

Due to an error on some of the cd-rom images, we may have to issue the command `/usr/sbin/fai-setup` again now. The result of the error was that the cdrom could not be mounted in the setup process.

6 Bootp

Before we can start the installation of nodes using the FAI bootfloopy, we have to configure and start the bootp service. Bootp is a protocol for obtaining network information for a host during bootup, and it is also possible to mount a root filesystem using NFS (Network FileSystem). On the server–side, this service is usually run from *inetd*. On the client–side, that is the nodes, a bootp–request could be sent either from the kernelprocess or the network interface card could be configured to send the request. We will let the kernel send the request, so we must remember to configure the kernel for this. The linux–kernel has this support, but it is usually not turned. For convenience we have a prebuilt kernel with this support on our cluster–cdrom.

To enable *inetd* to run the bootp service, edit the file /etc/inetd.conf, and find the line containing `bootps`. Make sure that this line is uncommented, that is the line should not start with a hash (#) character. If the inetd file is changed, restart inetd with `/etc/init.d/inetd restart`.

The configuration of bootp is done in the file /etc/bootptab. There is a template file in the fai packages, so do:

```
server:~# cp /usr/share/doc/fai/examples/etc/bootptab /etc/
```

The configuration in this file need some hardware information which may be unknown — the mac address (also known as the hardware address) of the network interface card (NIC) in each node. Until we know this information, we configure bootp with a special FAI tag, T171 set to “sysinfo.” On a console, we run

```
%>tcpdump -qte broadcast and port bootpc
```

while we boot all the nodes with the created bootfloopy. This will scan the network interface broadcast messages for bootp–service. The hardware number is the numbers separated by colons on the start of each lines.

Insert the hardware–information in the bootp configuration and set the special FAI tag T171 to “install”. Remark that the hardware numbers reported by tcpdump starts with a single zero (0) before the first colon, you must use two zeros in the bootptab file. All the colons should also be removed when the numbers are inserted in the bootptab file. Below is an example bootptab file.

```
# /etc/bootptab example for FAI

.faiglobal:\
:ms=1024:\
:hd=/boot/fai:\
:hn:bs=auto:\
:rp=/usr/lib/fai/nfsroot:

# rp: $NFSROOT
# your local values
#
# sa: your tftp server (install server)
# ts: your timeserver (time enabled in inetd.conf)
# T170: location of FAI configuration directory
# T171: FAI_ACTION
# T172: FAI_FLAGS, e.g. verbose, debug, reboot, sshd
# T173: reserved for future use
# T174: reserved for backup devices and backup options; NOT YET USED
# sm: subnet mask
# gw: router/gateway address
# dn: domainname
# ds: list of nameservers

# these are optional
# ys: NIS server
# yd: NIS domainname
# nt: list of NTP servers

.failocal:\
:tc=.faiglobal:\
:sa=server:\
:ts=server:\
:T170="server:/usr/local/share/fai":\
:T171="sysinfo":\
:sm=255.255.255.0:\
:gw=10.0.0.1:\
:dn=:\
:ds=10.0.0.1:\
:ys=server:yd=winterschool:\
:nt=server:

# now one entry for each install client
faiserver:ha=0x00103B240012:bf=faiserver:tc=.failocal:T171="sysinfo":T172="sshd verbose debug ":
node01:ha=0x0050DA754D74:bf=node01:tc=.failocal:T172="sshd":
node02:ha=0x0050DA754D70:bf=node02:tc=.failocal:T172="sshd":
```

7 Install cluster

We are now ready to finish the installation of linux onto the cluster. This is now simply a matter of inserting the bootfloopy in each node, and power up. Remove the floppy as soon as it is read (the light turns off), since the pc will reboot when the installation is finished.

It may be on time to create some normal useraccounts. Use `adduser username` to do this. Then we have to run `make` in /var/yp to update the NIS maps with the new userinformation.

8 MPI

We will use the MPI system mpich on our cluster. This system exist as a debian package, and is probably already installed at the previous installation step described above. You can check this with

```
%>dpkg -get-selections | grep mpich
```

If this list mpich as “install”, everything is fine. If that is not the case, you will install it with the apt-get utility.

The only adjustment of the configuration we need to do is to add all the nodes of our cluster to the file /etc/mpich/machines.LINUX. Dual–cpu nodes are added twice. Only add hostnames, do not use “localhost” for the current host.

Some example MPI programs are included on the cdrom. Log in as a normal user, and copy:

```
server:~# cp /cdrom/clusterfiles/mpiprogs/* .
```

Then unpack and compiles some of the programs:

```
server:~# tar zxf simpleMpiProgs.tgz
server:~# mpicc -o hw hw_comm.c
server:~# mpirun -np 2 ./hw
```

9 Queue system

The queue system on our cluster will be PBS. This is a large system usable on nearly every supercomputer and cluster in the world. This fact mean that the configuration could be a bit overwhelming. But you will also be satisfied with the flexibility, even on small and simple clusters. And the footprint of the running programs is not large in terms of either memory or cpu. So, in large, its the best free package available. Here we will explain the basic steps for setting up and configure PBS. For more complete information, we refer the reader to [5].

The source code for the latest version is included on the cdrom in /cdrom/clusterfiles and is named OpenPBS_2_3_12.tar.gz. Copy this file into /usr/local/src, and unpack, cd into new directory:

```
server:~# cp /cdrom/cluseterfiles/OpenPBS* /usr/local/src
server:~# cd /usr/local/src
server:~# tar zxf OpenPBS_2_3_12.tar.gz
server:~# cd OpenPBS_2_3_12
```

The build process is just to run `configure, make and make install`, so it is very easy to build. If you want to see all the configuration–options, run

```
%> ./configure -help
```

This lists about 50 options, but we will only use one:

```
%> ./configure -set-server-home=/var/spool/pbs
```

You may use any directory which is local to each node in the cluster, hence we have to avoid the default /usr/spool/pbs. The directory you choose will be named `$PBSHOME`. The further build and configuration process is

- Run `make`, then `make install`
- Edit $PBSHOME/server_priv/nodes, add the hostnames of all computers in the cluster, one for each line. Atso, set `np=2` for dual–cpu computers (like `server np=2`).
- Edit $PBSHOME/mom_priv/config and insert the lines:

  ```
  $logevent 0x1ff
  $clienthost server
  ```

  This basic cnofiguration specify the verbosity level of logging and who will be in charge for the system. So clienthost should be the name of the host where the pbs–server will be run.
- Start the job–executor, also named mom. The program is located in /usr/local/sbin/pbs_mom.
- Start the job–server: /usr/local/sbin/pbs_server -t create. The -t create argument should only be used the first time the server is run.
- Run the scheduler: /usr/local/sbin/pbs_sched.

The next step is to configure the system. The main thing we have to do, is to start the server and the scheduler, and create a queue for jobs. The program `qmgr` is used for this, so start qmgr an do the following:

- Define yourself as manager: `> set server managers=you@host`
- Deinfe an execution queue: `> create queue ws queue_type=e`
- Enable and start queue: `> set queue ws enabled=true, started=true`
- Start scheduling: `s s scheduling=true`[1]
- Set “ws” as default queue: `s s default_queue=ws`

End the qmgr–session with “quit”. There are only a few things left before the configuration of PBS is finished: The mom (job–executor) must be installad, configured and started on each node in the cluster. Since the usr–partition is nfs–mounted from server on all host, the mom is already present. All we have to do is to build the necessary catalog structure in $PBSHOME on each node, and distribute the configuration file. PBS include a tool called `pbs_mkdirs` for building the directory structure, this tool is located in /usr/local/src/OpenPBS_2_3_12/buildutils. Make sure that the program is executable, and do the following steps for all nodes:

[1] `s s` is “set server”, “set queue” could be written as `s q`

```
%> rsh node?? .../buildutils/pbs_mkdirs mom
%> rsh node?? .../buildutils/pbs_mkdirs aux
%> rsh node?? .../buildutils/pbs_mkdirs default
%> rcp $PBSHOME/mom_priv/config node??:$PBSHOME/mom_priv/
%> rsh node?? /usr/local/sbin/pbs_mom
```

The last entry starts mom, and complete the installation of PBS. You may now use `qsub` to run jobs on your cluster.

9.1 A PBS test

Jobs are commited to the queue system using `qsub`. As a simple test you may try (with the "hw" mpi program):

```
server:~# qsub -l nodes=4 -o out -j oe -m abe
mpirun.mpich -machinefile $PBS_NODEFILE -np 4 ./hw
CTRl-D
```

A Fai configuration

Lets go through the configuration files for FAI in some more detail. The base configuration of fai on the server is done in /etc/fai.conf. Only the most essential parts are included here, check the file for more info:

```
# /etc/fai.conf -- configuration for FAI (Fully Automatic Installation)

FAI_ARCH=‘dpkg --print-installation-architecture‘

# mount point where the debian mirror is mounted
MNTPOINT=/cdrom

# location of Debian software packages (Debian mirror)
# FAI_SOURCES_LIST could be more than one line
FAI_PACKAGEDIR=server:/cdrom
FAI_BASETGZ=$MNTPOINT/dists/stable/main/disks-$FAI_ARCH/current/base?_?.tgz
FAI_SOURCES_LIST="deb file:$MNTPOINT/debian stable main contrib local
deb file:$MNTPOINT/debian stable/non-US main contrib"

# the root password on all install clients during installation process
# pw is: fai
FAI_ROOTPW="56hNVqht51tzc"

# additional packages, that will be installed into nfsroot
NFSROOT_PACKAGES="ssh netbase nfs-common"

# Set UTC=yes if your system clock is set to UTC (GMT), and UTC=no if not.
UTC=yes

# this kernel package will be installed to nfsroot, default kernel for
# booting install clients, then its version-number
KERNELPACKAGE=/usr/lib/fai/kernel/kernel-image-2.4.2_wsbootp2_i386.deb
KERNELVERSION=2.4.2

# FAI_LOGUSER: an account on install server, which saves all log-files
# and which can change the kernel that is booted via network.
LOGUSER=root

# a constant
LIBFAI=/usr/lib/fai

# directory, where the nfsroot for FAI is created
NFSROOT=$LIBFAI/nfsroot

# the configuration space
FAI_CONFIGDIR=/usr/local/share/fai
```

The rest of the configuration is done in the files in /usr/local/share/fai. The most essential files are listed in Figure 1. If you browse that directory you will find more files, but those are not necessary for our installation, but since many of those files shows the versatility of the FAI package, I have left them there. The configuration of FAI is heavily based on the use of *cfengine*. The concept

```
class/ATA33.source                          files/etc/nsswitch.conf/NIS
class/DEFAULT.source                        files/etc/nsswitch.conf/NONIS
class/S00hostname.sh                        files/root/.bash_profile
class/S01alias.sh                           files/root/.bashrc
class/S05modules.sh                         files/root/.rhosts
class/S07disk.pl                            files/root/.rhosts/NET_9
class/S24nis.sh                             package_config/BOOTP_SERVER
class/S90partitions.sh                      package_config/COMPILE
class/S99last.sh                            package_config/DEBIAN_DEVEL
class/faiclient                             package_config/GERMAN
class/faiserver                             package_config/KERNEL_SOFT
class/faisimple                             package_config/MINI_SOFT
class/faisimple.source                      package_config/NFS_SERVER
class/global.mod                            package_config/NIS
disk_config/4GB                             package_config/NTP
disk_config/faiserver                       package_config/REMOVE
disk_config/15GB                            package_config/SERVER
fai_config/global.conf                      scripts/BASE
files/etc/alternatives/FAICLIENT            scripts/BOOT
files/etc/hosts/NET_9                       scripts/DEFAULT
files/etc/hosts.allow                       scripts/LAST
files/etc/hosts.allow/NET_9                 scripts/NETWORK
files/etc/hosts.deny                        scripts/NIS
files/etc/hosts.deny/NET_9                  scripts/NONIS
files/etc/hosts.equiv                       scripts/RESOLV
files/etc/hosts.equiv/NET_9                 scripts/X11
files/etc/nsswitch.conf
```

Figure 1: The most essential configuration files for FAI

of this tool is to assign a set of classes to each host, according to hostname, disksize, memorysize or whatever you want. Then you make a set of scripts which describe the actions you want to do for each class. When cfengine is run on a specific host with a certain configuration, only the actions which apply to the set of classes defined for the currunt host, are done.

All the classes are defined in scripts in the class directory. You can either use bourne shell scripts, with an ".sh" suffix, or perl–scripts with ".pl" suffix. The output of these scripts are the class–tags, and each script could define multiple tags, separated by whitespace. An example of such a script is included in Figure 2. Remark that already defined classes could be used in the scripts defining classes.

In the same directory, you can also set variables in files with the ".source" suffix. These files are read by the install–script after all classes are defined, and only files matching "<CLASS>.source" for the classes defined for the current host are read.

The two other central part of the configuration is the scripts defining the actions for each class, and the debian–packages we want to associate with the classes. The scripts are placed in the scripts subdirectory and the package–definition in the package_config subdirectory. All these file should have the appropriate class–tag as name. The scripts could be a mix of either shell scripts, perl scripts, expect scripts or cfenfine scripts. In the files which describe the package selections, we can define two actions: install or remove, and each file must define at least one of these. The action is set by lines like "PACKAGES <action>". The word "PACKAGES" must start at the first colomn on the line. The following lines up to end of file or the next "PACKAGES" line simply list the names of the packages, separated by whitespace.

The scripts and the package files in the templates will do just fine for our purposes, so we leave them alone. But some of the scripts will copy files from the files subdirectory onto the nodes. The organization in this catalog is that what will be the filename on the node, is a directory here, and inside the directory we have a file named according to the class it belongs to.

B Acronyms and explanations

BIOS: Basic Input Output System. A set of low level service routines incorporated into read only memory on the mainboard. This system serves as a logical interface to the hardware.

boot: To boot (as a verb; also "to boot up") a computer is to load an operating system into the computer's main memory or random access memory (RAM). Once the operating system is loaded (and, for example, on a PC, you see the initial Windows or Mac desktop screen),

```
#! /bin/sh
for c in $classes
do
 if [ -r /fai/disk_config/$c ]
 then
  grep -v "^#" /fai/disk_config/$c | \
  grep -q '[[:space:]]/scratch[[:space:]]' && echo "NFS_SERVER SCRATCH"

  grep -v "^#" /fai/disk_config/$c | \
  grep -q '[[:space:]]/files/scratch[[:space:]]' && echo "NFS_SERVER FILES_SCRATCH"

  grep -v "^#" /fai/disk_config/$c | \
  grep -q '[[:space:]]/tmp[[:space:]]'  && echo "TMP_PARTITION"

  exit
 fi
done
```

Figure 2: Example class definition script: S90partitions

it's ready for users to run application . Sometimes you'll see an instruction to "reboot" the operating system. This simply means to reload the operating system (the most familiar way to do this on PCs is pressing the Ctrl, Alt, and Delete keys at the same time).

BOOTP: Bootstrap Protocol. A system for providing a mapping between hardware address of the network interface card (NIC) and the IP address. Other possibilities of the system is to provide a root filesystem exported via NFS to either diskless clients or for installation purposes. The system could also provide a boot kernel for clients.

cfengine: cfengine is a language based system specifically designed for testing and configuring unix-like systems attached to a TCP/IP network. You can think of cfengine as a very high level language – much higher level than Perl or shell. A single statement can result in many hundreds of links being created, or the permissions of many hundreds of files being set. The idea of cfengine is to create a sin- gle file or set of configuration files which will describe the setup of every host on your network.

GNU: Gnu is Not Unix. GNU is a UNIX-like operating system that comes with source code that can be copied, modified, and redistributed. The GNU project was started in 1983 by Richard Stallman and others, who formed the Free Software Foundation . Stallman believes that users should be free to do whatever they want with software they acquire, including making copies for friends and modifying the source code and repackaging it with a distribution charge. The FSF uses a stipulation that it calls copyleft . Copyleft stipulates that anyone redistributing free software must also pass along the freedom to further copy and change the program, thereby ensuring that no one can claim ownership of future versions and place restrictions on users.

The "free" means "freedom," but not necessarily "no charge." The Free Software Foundation does charge an initial distribution price for GNU. Redistributors can also charge for copies either for cost recovery or for profit. The essential idea of "free software" is to give users freedom in how they modify or repackage the software along with a restriction that they in turn do not restrict user freedom when they pass copies or modified versions along.

MBR: Master Boot Record. The Master Boot Record (MBR) is the information in the first sector of any hard disk or diskette that identifies how and where an operating system is located so that it can be boot (loaded) into the computer's main storage or random access memory. The Master Boot Record is also sometimes called the "partition sector" or the "master partition table" because it includes a table that locates each partition that the hard disk has been formatted into. In addition to this table, the MBR also includes a program that reads the boot sector record of the partition containing the operating system to be booted into RAM. In turn, that record contains a program that loads the rest of the operating system into RAM.

NFS: Network File System. A client/server application that lets a computer user view and optionally store and update file on a remote computer as though they were on the user's own computer. The user's system needs to have an NFS client and the other computer needs the NFS server. Both of them require that you also have TCP/IP installed since the NFS server and client use TCP/IP as the program that sends the files and updates back and forth. (However, the User Datagram Protocol, UDP, which comes with TCP/IP, is used instead of TCP with earlier versions of NFS.)

NIS: Network Information System. A network naming and administration system for smaller networks that was developed by Sun Microsystems. NIS+ is a later version that provides additional security and other facilities. Using NIS, each host client or server computer in the system has knowledge about the entire system. A user at any host can get access to files or applications on any host in the network with a single user identification and password. NIS is similar to the Internet's domain name system (DNS) but somewhat simpler and designed for a smaller network. It's intended for use on local area networks.

References

[1] The Internet Engineering Task Force website. http://www.ietf.org/.

[2] Linux documentation project - howto's. http://www.linuxdoc.org/HOWTO.

[3] Terry Dawson Olaf Kirch. The network administrator's guide. http://www.linuxdoc.org/LDP/nag2/nag2.pdf, 2000.

[4] The Scyld Website. http://www.scyld.com/.

[5] Veridian Systems. *Portable Batch System Administrator Guide*, release 2.3 edition, August 2000.

[6] The whatis website. http://www.whatis.com.