How to Build a Linux Cluster for Scientific Computing

Benjamin Auffarth

Institute for Bioengineering of Catalonia (IBEC),

Barcelona Science Park,

C/Josep Samitier 1-5,

08028 BCN, Spain

January 15, 2009

Abstract

A beowulf cluster is a cluster of Linux computers designed to run computing jobs in parallel. This article is going to give an up-to date example of assembled hardware, and installed programs (plus configuration). Finally there will be explanations of how to run processes in parallel in computing languages such as matlab and R. After the computers are setup we may want to transfer the configuration to other computers, a process which is called cloning. In the last section, we will deal with profiling. keywords: scientific computation, cluster,

beowulf cluster, linux, matlab, R

1 Introduction

The improvement of a single processor performance seems to have reached its limits. As a solution the emergence of multi-core and many-core processors can give a better performance and reduce problems like power comsumption or heat dissipation. Multiprocessing in scientific computing however still seems to present a challenge for many research groups, which buy expensive mainframe or servers instead of investing their money in parallel architectures. This is even more astonishing, given that many computations in scientific research are *embarrassingly parallel*, called so, because each run or iteration is independent from others and in principle all could run in parallel. Examples are bootstrapping or genetic algorithms. Other computations can be parallelized by rewriting algorithms.

The idea is instead of buying one huge expensive computer, many cheap computers and have them compute in parallel. If these computers are connected in a network and run Linux[1], the architecture is called a beowulf cluster (Sterling et al., 1995; Ridge, Becker, Merkey, & Sterling, 1-8 Feb 1997). The advantage of a beowulf with respect to server workstations or mainframes is in the price/performance relationship. While this article may

[1] An important consideration in running an open–source operating system is the control and transparency it gives the advanced user. Some beowulfs run BSD, however for hardware and software compatibility, it can be expected that Linux is more straightforward.

serve as a short introduction, for broader overviews, references (Brown, 2004) and (Sloan, 2005) can be recommended.

A basic "recipe" for a cluster might be: (Brown, 2004)

1. Buy a pile of MMCOTS PCs ("pile of PCs").
2. Add a network interface card (NIC) to each.
3. Add GNU/Linux (or BSD) as operating system and middle-ware to support distributed parallel computing.
4. Parallelize your code.

We will come to each step in turn. In the next section we will see an example of some hardware with actual prices (in euros). Then we weal deal with issues of installation and configuration, before we come to running processes in parallel using numeric computing environments MATLAB and R. When everything is set up for two machines, we want to extend our beowulf to more machines (cloning of machines). In the end, we will look at profiling.

2 Hardware

A lot of attention should be focused on the hardware assembly. There are many trade-offs and considerations of speed bottlenecks. If you are a newbie in the beowulf business, you should be careful to settle on a vendor with experience, where they can give you good recommendations.

Which parameters of hardware are important depends on the computing tasks you want to dedicate the beowulf to. However you probably need above

all fast and lots of memory, a fast bus system, fast and lots of processor cache. As a general rule, in your computations you should try avoid accessing the hard disk very much, however, if you need a lot of access to the hard disk, you should look for one with high revolutions. Hard disks are one of the components that fail most, so do not try to make a bargain but rely on your experience with hard disk vendors. Bye a fast ethernet card and a good switch.

Here is a list of hardware, which we assembled at the artificial olfaction lab at the Institute for Bioengineering of Catalonia. Prices are as of May 2008 and do not include taxes.

We bought 8 computers with these parameters:

- Intel Core 2 Duo Quad Q6600 2.4 Ghz FSB1066 8MB
- Chipset Intel X38 / Intel ICH9R
- 4 GB RAM DDR3 1066 (in 2x2Gb).
- 2 x PCI Express x16, 1 x PCI Express x1, 2 x PCI-X y 1 PCI
- Marvell88E8056 Dual Gigabit LAN controller
- Realtek ALC882M 7.1 chanels (sound
- 6 USB 2.0 ports y 1 IEEE1394
- VGA 512MB Gforce8400GS PCI-e
- 160 GB de Disco Duro SATA II 3 Gb.
- DVD R/W, Multicard reader/writer
- 19“ rack computer case, 4U, with frontal lock.
- 550W Server Guru

These machines cost us 923 euros each, a reasonable price for a computer with these characteristics.

In order to operate 8 computers efficiently you need a monitor that you can connect to all machines. Better even a KVM (keyboard video mouse), which allows you to switch keyboard, monitor, and mouse between different computers by button press. A KVM with 8 ports cost us 850 euros. The KVM, as the name says, serves as keyboard, monitor, and mouse, and did great service during configuration.

We also need to connect all the computers among themselves: A switch with 16 ports, 10/100/1000, comes at 162 euros.

We want to put all the hardware somewhere, where it is save and where it has good conditions of cooling and safety: a 19 inch rack. A rack which can take up 42 units cost us 500 euros.

Don't be surprised to be charged extra for cables, screws, and multi-outlet power strips. Rails allow you to stack in and take out your computers like drawers. Additional cost: about 700 euros.

Also on top, the VAT, in Spain 18%, which makes about 1500 euros.

Total cost of the beowulf: about 11,090 euros.

In order to connect your computers, you need power lines that can support them. The configuration above needs about 4.5kWatt and we had to wait about 2 months for technicians to fix the capacity (that's Spain).

3 Installation, Configuration, and Administration

In this section, you will find basics of installation, configuration, and administration. Most important here: 1. network setup with DHCP, 2. sharing files over the network with the network file system (NFS)(Shepler et al., 2003). After we cover this, we will come to some more general ideas of administration.

Most beowulfs use a network structure of master (also network head) and slaves, so that computing jobs are going to be distributed from the master to the slaves. All machines of the cluster are connected to the switch. The head will additionally interface with an external network. Master and slaves will share user data over the network.

Practically this means, that the master (or head node) has two network interfaces (say eth0, eth1), one connected to the outside world and one connected to the cluster intranet over a network switch. All other computers computers (the slaves or nodes) are connected to the switch. In order to start processes on the cluster a user logs in on the master and spawns the processes from there to the slaves.

It is easiest, to only install the master and one slave (called *golden slave*) in a first step. Later, after setting up master and golden slave, we will dedicate a section to cloning, which is the process of creating machines that are identical to the golden slave.

3.1 Operating System and Software

We could run our beowulf on any Linux distribution. Some Linux distributions are specialized on beowulf clusters, such as Rocks Clusters(Papadopoulos, Katz, & Bruno, 2003), which can facilitate the installation and administration of clusters. However, while Rocks helps you as a beginner, you will have to learn the fundamentals anyways some time or later. Rocks clusters (v. 5) didn't recognize the network cards of our computers. We also tried the Ubuntu Server Edition (Hardy)(*Ubuntu Server Edition*, 2008), which didn't recognize the CD-ROM. Fedora 9(*Fedora project*, 2008) recognized all hardware at once, so this was our distribution of choice.

The Fedora distribution is available on the internet free of charge. All programs we need are included in the standard Fedora installation or can be installed using the Fedora package manager[2]. For master and slaves, we started from vanilla installations and added then some more software from the Fedora repositories.

Here is a selected list of software you might want to consider: the ssh server(Campbell et al., 2008), vim(Moolenaar, 2008), emacs(Stallman & Foundation, 1993), the gnu toolchain with compilers for C/C++ and Fortran(Stallman & Foundation, 1992), gnu screen(Laumann et al., 2008), subversion(Collins-Sussman, Fitzpatrick, & Pilato, 2004), git(Torvals & Hamano, 2008), some texlive(*TeX Live – TeX User Group*, 2008) latex(Lamport, 1994) packages,

[2]Except for mpich2 which needs to be compiled and Matlab, which is a commercial product.

GNU R(Ihaka & Gentleman, 1996), GNU octave(Eaton, 1997), python(Van Rossum & Drake, 2003), perl(Wall, Christiansen, & Orwant, 2000), java runtime and development kit(Gosling, Joy, Steele, & Bracha, 2000), maxima(Schelter et al., 2008), gnu scientific library(Galassi et al., 2002), blas(Dongarra, Du Croz, Hammarling, & Duff, 1990), graphviz(Ellson, Gansner, Koutsofios, North, & Woodhull, 2002), gnuplot(Williams, Kelley, et al., 1998).

All commands and instructions refer to Fedora 9, however should have close equivalents in other Linux distributions.

3.2 Networking

We want the computers to work in a local network. The easiest way to setup the network so it can be extended fast in the clustering step (as described in the last section) is by dynamic addresses (DHCP) on the basis of physical addresses of slaves' network interfaces. DHCP simplifies the installation of new nodes, because the mac address and hostname is the only thing that is different among the nodes and the DHCP server on the master can manage a new node by a new entry into the configuration file.

In this example we will set up the network IPs to 192.168.1.1 until 192.168.1.8, where 8 is the master.

slave(s):/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

master:/etc/dhcpd.conf

```
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 10.5.170.254;
# define individual nodes
subnet 192.168.1.0 netmask 255.255.255.0 {
group {
# define all slaves, the master (head node) is has a static ip
    address
host node0{
hardware ethernet 00:1E:8C:30:AC:2A;
fixed-address 192.168.1.250;
}
host node1{
hardware ethernet 00:1E:8C:30:B0:A1;
fixed-address 192.168.1.1;
}
#... Here you can put more nodes. Make a list of the mac
    addresses of all your machines and enter them in the list.
}
}
# ignore petitions from second network interface
subnet 10.5.170.0 netmask 255.255.255.0 { not authoritative; }
```

The idea is to give to slaves the names nodei corresponding to their ip address 192.168.1.i. Note that the DHCP server provides IP addresses for the other machines not for itself. The master you give a static ip address (192.168.1.250 here). In red hat based distributions this is configured in /etc/sysconfig/network-scripts/ifcfg-eth0 or /etc/sysconfig/network-

scripts/ifcfg-eth1. I configured eth0 for the organization network and eth1 for the cluster intranet.

Example files:

/etc/sysconfig/network-scripts/ifcfg-eth0 corresponds to your organization network settings.

```
DEVICE=eth0
ONBOOT=yes
...
```

/etc/sysconfig/network-scripts/ifcfg-eth1

```
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=static
NETWORK=192.168.1.0
IPADDR=192.168.1.250
TYPE=Ethernet
```

For the slaves, the interface to the cluster intranet is as follows:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
DNS1=...
DNS2=...
```

If you don't use a DNS service on your head you use the DNS service of the network of your organization. /etc/hosts

```
127.0.0.1 localhost.localdomain localhost
::1 localhost6.localdomain6 localhost6
192.168.1.250 node0
192.168.1.1 node1
# ... add more names of machines here
```

Note in /etc/hosts that in the loopback line (first line) the hostname is not given in order to avoid problems with message protocols (PVM, MPI). You need to activate ip forwarding on the head in order to have internet access on all machines. You enable the firewall and include masquerading on the network interface to you cluster. This you do by changing the /etc/sysconfig/iptables file or using some user interface, e.g. system-config-firewall on red hat based systems. Be careful not to make your firewall too restrictive as this can cause problems. In the /etc/sysconfig/network you need to have:

```
NETWORKING=YES
IPFORWARD=YES
HOSTNAME=nodei
```

... where i is your node number.

You have to reinitiate the network services and startup the dhcp server daemon (dhcpd). To have dhcpd startup at boot, in fedora the ntsysv program allows you to search a list and mark the corresponding entry. You may want to setup your printers on master and slave (you can copy an existing printer configuration recursively from /etc/cups e.g. from your local office desktop computer).

3.3 Shared Directories and SSH

We want to share data among computers. Network File System (NFS) setup is remarkably simple. You basically have to install the package, start the nfs services, and change two files. Let's see the example files:

on the master:/etc/export

```
/home/ 192.168.1.0/255.255.255.0(rw,sync,no_root_squash)
/root 192.168.1.0/255.255.255.0(rw,sync,no_root_squash)
```

on the slave(s):/etc/fstab

```
192.168.1.250:/home /home nfs rw,hard,intr 0 0
192.168.1.250:/root /root nfs rw,hard,intr 0 0
```

Sharing /home and /root directories password-less ssh login is simplified. For every user that need access to slaves the following achieves the goal:

```
> ssh-keygen -t rsa
```

confirm choices and leave password empty.

```
> cat ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

The `~/.ssh/known_host` file needs entries for each computer (e.g. by logging in to each machine once). I used option -v with the ssh command to speed up ssh changing parameters in `/etc/ssh/ssh_config` and `/etc/ssh/sshd_config`. Options that you might want to specify include protocol 2, port 22, PreferredAuthentications PublicKey, IdentityFile `~/.ssh/id_rsa`. You might want to turn off checking of `.ssh/known_hosts`, and on the slaves you might want to ignore the host ip (CheckHostIP no), and turn off any

authentication method except for public key and password. Blowfish is one of the fasted cipher methods (Allen, 2003).

These two functions can help in setting up other user accounts or installing additional software. I put them into /etc/bashrc, so I have them available.

```
# execute command on all slave nodes
function slave {
  if [ $# -ne 1 ]; then
    echo "Forma de uso: slave comando"
  else
    for i in $(seq 1 7) ;
    do
      ssh 192.168.1.$i $1 ;
    done ;
  fi
}

# copy to all slave nodes
function slavecopy {
        if [ $# -ne 2 ]; then
                echo "Forma de uso: slavecopy <origen> <
                    destino_en_los_nodos>"
        else
                for i in $(seq 1 7) ;
                do
                        scp $1 192.168.1.$i:$2 ;
                done ;
```

```
        fi
}
```

On a side note, using GNU screen the above two scripts could be already used for parallelization of computing tasks, however there are dedicated protocols for more advanced usage, you might want to have resource balancing, for example.

To create more users, create them on node0 and node1. Then from node1 you copy `/etc/passwd` and `/etc/group` to the other slaves.

4 Parallelization of computation

As the principal goal of having the cluster is to run programs in parallel on different machines, I installed protocols for message-passing for distributed-memory applications. There are two common ones: the Parallel Virtual Machine (PVM) and Message Parsing Interface (MPI).

For scientific computing we can use high-level computing platforms or languages such as C/C++ and fortran. Here we will see GNU R and matlab. R can spawn parallel jobs using either PVM or MPI. Matlab comes with an implementation of MPI (more precisely mpich2).

Note that for PVM you need to enable password-less ssh access (see previous section) from the server to all clients. Also, for PVM, MPI (includes matlab's mdce), the network configuration you have to remove the host names from the loop-back line (where it says 127.0.0.1) of the `/etc/hosts` file. Just put localhost instead. Then you need a text file with a list of all machines

you wish to use for computing and call it pvmhosts and mpihosts.

In C/C++ you can use MPI or PVM to spawn processes.

4.1 MPI

As for MPI, you can install different implementations.Compiling and installing Mpich2(Gropp, 2002) was very straightforward and the install guide(Gropp et al., 2008) is a great help in setting it up.

On the master (node0) you type:

```
mpd &
mpdtrace -l
And you should get back the node name (say node0) and port
    number (say 51227). Then you connect each node to by typing:
mpd -h node0 -p 51227
The test could be spawning of the hostname command. You can
    spawn processes with mpiexec:
mpiexec -n 10 /bin/hostname.
```

4.2 PVM

The Parallel Virtual Machine (PVM)(Beguelin, Dongarra, Jiang, Manchek, & Sunderam, 1995) allows computers which are connected over a network to be used as a single distributed parallel computer. Installation of PVM is straightforward from packages, for configuration environment variables need to be set up:

add to /etc/profile:

```
export PVM_ROOT=/usr/share/pvm3
export PVM_TMP=/tmp
export PVM_RSH=/usr/bin/ssh
export PVM_ARCH=LINUXX86_64
export PATH=/usr/local/bin/:$PATH
```

In the pvm environment (which you enter typing 'pvm') after adding your machines you can then try the following and see if you can get a list of your machines.

```
spawn -> -10 /bin/hostname
```

4.3 GNU R

In this section we see an examples of parallelization from within the R computing platform(Ihaka & Gentleman, 1996) (statistical computing similar to Matlab) using the snow package(Rossini, Tierney, & Li, 2007), which can distribute jobs using either MPI or PVM. In this subsection we will only see usage of R+PVM.

Note, that for installation of packages within R the development packages of R are needed. I just installed all R packages that were available in the R repository.

You need to install R libraries snow and rpvm on master and all your slave(s). If you don't, R lets you create the PVM cluster object but then freezes when you try to execute a job.

In R:

```
> library('snow')
```

```
> library('rpvm')
> cl<-makePVMcluster(count=2,names=c('node0','node1'))
> clusterCall(cl, function() Sys.info()[c("nodename","machine")
    ])
[[1]]
nodename machine
"node1" "x86_64"

[[2]]
nodename machine
"node0" "x86_64"
```

See reference (Rossini et al., 2007) for more explanations the possibilities of the snow package.

4.4 MATLAB

As for matlab, note that in order to run it on a 64bit system you need shared libraries (on Fedora the package is called libXp, on ubuntu ia32) and some java packages.

Matlab comes with the parallel computing toolbox, the distributed computing server, and an implementation of mpich2. You can find manuals in pdf format on the corresponding mathworks site(Inc., 2008). You start the mpich2 server as root:

```
$MATLAB/toolbox/distcomp/bin/mdce start
```

where `$MATLAB` is the directory of your matlab installation.

(Note: Here holds the same as for PVM. For mdce to work, you need

to remove from /etc/hosts the loopback line with your host name in it, i.e. 127.0.1.1 node0 becomes `your_network_ip` node0.)

We start the job manager:

```
$MATLAB/toolbox/distcomp/bin/startjobmanager -name MyJobManager
```

Connect one worker:

```
$MATLAB/toolbox/distcomp/bin/startworker -jobmanager
    MyJobManager -jobmanagerhost node0
```

where node0 is the machine where your jobmanager is running (obviously).

... and a second worker on a different machine. From node0:

```
$MATLAB/toolbox/distcomp/bin/startworker -jobmanager
    MyJobManager -jobmanagerhost node0 -name worker2 -remotehost
    node1
```

Use the option remotehost to start a worker on a different machine.

Make sure that job manager and workers are running:

```
$MATLAB/toolbox/distcomp/bin/nodestatus
```

We start matlab in desktop mode (we need the Java Virtual Machine in matlab in order to configure the *matlabpool*). In the menu under `Parallel->configure` and `parallel->administrate`, we choose MyJobManager basically and start the matlabpool:

```
>> matlabpool open
```

You should see the confirmation "Connected to a matlabpool session with 2 labs (or more)."

The simple proof of concept using the powerful parfor construct:

```
>> parfor i=1:5
unix('hostname');
end
```

In matlab don't use too much load/save, close, figure, etc. You can start matlab with the -noawt option to start it within a screen session (see next section).

5 Profiling

It is important to know how long applications take, whether they paralellize at all (or just run locally). Also if computers heat up too much, they are gone, so we see how to control the temperature.

As a general rule for paralellization: avoid network trafic and access to physical devices. You can use GNU screen to have any programs running without the need of staying logged in. If you detach from it by pressing control-A d, you can logout, having all your running programs available on next login when you attach to your screen by typing screen -d -r.

Useful commands in linux for profiling are w, ps, top. W shows you how many users are logged in and what they are doing.

If we want to know how good our matlab programs are paralellizing we can do:

```
for i in `seq 0 7` ; do echo node$i; ssh node$i "top -bn 1 |
    grep -i matlab | grep -v helper" ; done
```

5.0.1 CPU temperature

```
> cat /proc/acpi/thermal_zone/THRM/temperature
```

(you might have different THRM directories for your cores)

If this doesn't work or you want more information, install sensors (in Fedora: yum install sensors). You install it by sensors-detect and sensors then prints you information.

```
> sensors
...
Adapter: ISA adapter
Core 0: +67.0 C (high = +82.0 C, crit = +100.0 C)

coretemp-isa-0001
Adapter: ISA adapter
Core 1: +66.0 C (high = +82.0 C, crit = +100.0 C)

coretemp-isa-0002
Adapter: ISA adapter
Core 2: +61.0 C (high = +82.0 C, crit = +100.0 C)

coretemp-isa-0003
Adapter: ISA adapter
Core 3: +61.0 C (high = +82.0 C, crit = +100.0 C)
```

6 Cloning of Slaves

When master and one slave (the golden slave) are installed and configured and now we want to scale up this configuration to more slaves by replicating

the exact configuration of the golden slave.

Installing and configuring the OS on each machine manually is cumbersome and prone to error. However, nodes are identical, so why not just copy everything we need? This process is called cloning. We first setup a so-called golden node or model node and then transfer the system to other slave machines. Each new node will come with one new entry in the head node's DHCP server file (/etc/dhcpd.conf) and /etc/hosts file.

For preparation, make sure that in /etc/fstab and in the /boot/grub/-menu.lst, there are no physical addresses of hardware (e.g. a hard disk), as they will be different among the nodes. All hardware should be addressed by their subdirectory in /dev which you can see in the output when you type mount.

I used low level R/W with dd and piping to and from netcat, respectively on machine to clone from and machine to clone to. We clone using convert and copy (dd) and netcat (nc).

On node1 you run:

```
node1# dd if=/dev/hda conv=sync,noerror bs=64k | nc -l 5000
```

On node2 you run:

```
node2# nc 192.168.1.1 5000 | dd of=/dev/hda bs=64k
```

where 192.168.1.1 is the ip of node1. This presupposes the disk of node2 is at least as big as node1's.

7 Conclusions

This article gave instructions of how to install a beowulf cluster. We gave an example of hardware acquisition, went through principal steps of installation and network setup, and showed how to parallelize programs in R and matlab. We hoped to show that it is not that difficult to get lots of computing power for comparatively little money.

References

Allen, D. (2003). Eleven SSH tricks. *Linux Journal*, *2003*(112).

Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., & Sunderam, V. (1995). *PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing*. MIT Press Cambridge, MA, USA.

Brown, R. G. (2004, May). *Engineering a beowulf-style compute cluster.* `http://www.phy.duke.edu/~rgb/Beowulf/beowulf_book/beowulf_book/index.html`.

Campbell, A., Beck, B., Friedl, M., Provos, N., Raadt, T. de, & Song, D. (2008). *Openssh.* `http://www.openssh.com/`.

Collins-Sussman, B., Fitzpatrick, B., & Pilato, C. (2004). *Version Control with Subversion*. O'Reilly Media, Inc.

Dongarra, J., Du Croz, J., Hammarling, S., & Duff, I. (1990). A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software (TOMS)*, *16*(1), 1–17.

Eaton, J. (1997). *GNU Octave Manual*. Network Theory.

Ellson, J., Gansner, E., Koutsofios, L., North, S., & Woodhull, G. (2002). Graphviz-Open Source Graph Drawing Tools. *LECTURE NOTES IN COMPUTER SCIENCE*, 483–484.

Fedora project. (2008). `http://fedoraproject.org/`.

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M., et al. (2002). *GNU scientific library*. Network Theory Ltd.

Gosling, J., Joy, B., Steele, G., & Bracha, G. (2000). Java Language Specification: The Java Series.

Gropp, W. (2002). MPICH2: A New Start for MPI Implementations. *LECTURE NOTES IN COMPUTER SCIENCE*, 7–7.

Gropp, W., et al. (2008). *Mpich2 : High-performance and widely portable mpi.* `http://www.mcs.anl.gov/research/projects/mpich2/`.

Ihaka, R., & Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS*, *5*, 299–314.

Inc., M. (2008). *Parallel computing toolbox.* `http://www.mathworks.com/access/helpdesk/help/toolbox/distcomp/`.

Lamport, L. (1994). LaTeX: A Document Preparation System Users Guide and Reference Manual. *Reading, Mass*.

Laumann, O., et al. (2008). *Gnu screen.* `http://www.gnu.org/software/screen/`.

Moolenaar, B. (2008). *Vim – the editor.* `http://www.vim.org/`.

Papadopoulos, P., Katz, M., & Bruno, G. (2003). NPACI Rocks: tools and

techniques for easily deploying manageable Linux clusters. *Concurrency and Computation: Practice & Experience*, *15*(7), 707–725.

Ridge, D., Becker, D., Merkey, P., & Sterling, T. (1-8 Feb 1997). Beowulf: harnessing the power of parallelism in a pile-of-pcs. *Aerospace Conference, 1997. Proceedings., IEEE*, *2*, 79-91 vol.2.

Rossini, A., Tierney, L., & Li, N. (2007). Simple Parallel Statistical Computing in R. *JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS*, *16*(2), 399.

Schelter, W., et al. (2008). *Maxima, a computer algebra system.* `http://maxima.sourceforge.net/`.

Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., et al. (2003). RFC3530: Network File System (NFS) version 4 Protocol. *Internet RFCs*.

Sloan, J. (2005). *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI*. O'Reilly.

Stallman, R., & Foundation, F. S. (1992). *Using and Porting GNU CC*. Free Software Foundation.

Stallman, R., & Foundation, F. S. (1993). *GNU Emacs Manual*. Free Software Foundation.

Sterling, T., Savarese, D., Becker, D. J., Dorband, J. E., Ranawake, U. A., & Packer, C. V. (1995). BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th international conference on parallel processing* (pp. I:11–14). Oconomowoc, WI.

Tex live – tex user group. (2008). `http://tug.org/texlive/`.

Torvals, L., & Hamano, J. (2008). *Git–fast version control system.* `http://git.or.cz/`.

Ubuntu server edition. (2008). `http://www.ubuntu.com/products/whatisubuntu/serveredition`.

Van Rossum, G., & Drake, F. (2003). *Python Language Reference Manual*. Network Theory.

Wall, L., Christiansen, T., & Orwant, J. (2000). *Programming Perl*. O'Reilly.

Williams, T., Kelley, C., et al. (1998). GNUplot: an interactive plotting program. *Manual, version*, *3*.