

# Using the PCN Cluster, “Amdahl”

Preston M. Smith  
psmith@physics.purdue.edu \*

August 29, 2003

## 1 Introduction

In Spring of 2002, the Physics Computer Network (PCN) deployed a “Beowulf” style cluster of Linux systems. The low cost of commodity x86 PC hardware, combined with the Open Source Linux operating system allows a parallel supercomputer to be created for a fraction of the cost of a commercial cluster system.

The cluster at PCN is based on the Debian GNU/Linux <sup>1</sup> system, a distribution of Linux dedicated to providing a Free operating system, in the collaborative, open spirit of GNU. Debian was created in 1993 by Ian Murdock, a graduate of Purdue’s Computer Science program.

Similarly, the cluster at PCN is operated in a collaborative way: research groups in the department who wish to apply cluster computing to their research can contribute nodes to the cluster. Contributing nodes to the cluster allows reseachers to combine their nodes, which they have priority access to, with those of other groups into a greater aggregate system.

## 2 Overview of the Cluster

Accessing the PCN cluster ‘amdahl’ is done by logging into the system *amdahl.physics.purdue.edu* via ssh. Your account on the PCN cluster will be the same as your usual one: the username and password are the same as you would use to log into *bohr* or *curie*. If you do not already have access to the cluster, you can get your account activated by sending an email message to [staff@physics.purdue.edu](mailto:staff@physics.purdue.edu).

### 2.1 The Master Node *amdahl*

The system *amdahl* is provided for interactive use of the cluster. Development software (editors, compilers, libraries, plotting tools, job submission software) are all available for use

---

\*Thanks to the following people: Olga Barannikova and Shu-Ju Tu for testing computations on the cluster, Brad Parks for readability-checking this manual, and Brian D. Haymore of the University of Utah’s Center for High Performance Computing for his help jumpstarting the scheduler configuration.

<sup>1</sup><http://www.debian.org>

on *amdahl*. Amdahl should be used to develop, compile, and test your computational jobs, but not to run them. Once you are reasonably certain that your job will run as expected, use PBS <sup>2</sup> to submit your job for execution on compute nodes. PBS will automatically find the best available system(s) to execute on. Further information on use of PBS can be found later in this document.

## 2.2 Centralized Resources, and Hardware Information

The network of the cluster is set up on a private network, separate from the “real” Internet. The only way to access the cluster is by logging into the front-end node *amdahl*. The private network is physically designed to be as flexible as possible: currently it offers a 3Com 24-port 100 MBps Ethernet switch, and a 3Com 12-port gigabit Ethernet switch serving as the backbone of the cluster network. Future network expansion is possible, should cluster participants require it. Additional gigabit Ethernet switches could be added, or even specialized network technologies like Myrinet.<sup>3</sup>

Many gigabytes of disk storage are available within the cluster: nearly a Terabyte of RAID-5<sup>4</sup> storage is provided by a pair of SnapAppliance Network Attached Storage devices. The automounter, and familiar directories like */home* and */cluster* allow transparent access to all of the disk resources in the cluster.

As it is located on a private network, a cluster user will note that they have a separate home directory while using the cluster. This is simply a performance consideration: access to a directly attached disk or local-to-the-cluster network will be many times faster than leaving the cluster network to access your usual PCN home directory. Also, a separate home directory removes the potential nuisance of disk quotas interfering with computations.

For convenience, access to your PCN */home* and */project* directories is possible on the master node *amdahl*, so that you can easily transfer program source, data and output files. Access to PCN (that is, non-cluster) */home* and */project* areas can be found under */Home* and */Project*.

## 3 Using the Cluster for Work

### 3.1 Compilers

A number of compilers are available for programming: The GNU compiler collection (*gcc*, *g77*, *g++*) are installed. For programs that need OpenMP<sup>5</sup> for parallel programming, the Portland Group’s compiler suite is available on *amdahl* as well. (*pgf77*, *pgf90*, *pgcc*, *pgCC*). Manpages for all of the installed compilers are accessible in the cluster.

---

<sup>2</sup>Portable Batch System. See <http://www.openpbs.org>

<sup>3</sup>A scalable cluster interconnect system. See <http://www.myrinet.com/myrinet/overview/index.html>

<sup>4</sup>Redundant Array of Inexpensive Disks. RAID-5 stripes data across a set of disks, with parity information to protect against disk failure.

<sup>5</sup>A portable, scalable model that gives shared-memory parallel programmers a simple and flexible interface for developing parallel applications.

### 3.1.1 Efficiency Tips

Portland Group compilers can optimize for Athlon processors with the command-line options `-tp athlon -Mvect=prefetch`. Complete documentation for the Portland compilers can be found on the WWW at <http://amdahl.physics.purdue.edu/pgi/>. Remember, if you optimize your code for Athlon processors, be sure to request Athlon nodes in your PBS script. (`#PBS -l nodes:1:athlon`)

## 3.2 Installed Software

In an effort to provide an environment for any computation, the cluster offers a wide variety of programming languages and libraries. Not only parallel applications can be run on the cluster, but batch computations as well. Following is a list of currently installed languages and libraries. If a library or piece of software that you need for your research is not found on the cluster, please let PCN know, and we'll get it installed for you.

### 3.2.1 Languages

- **Interpreted/Scripting languages:** Tcl/Tk, Perl, Java, Python
- **Compilers:** GNU Compiler Collection (`gcc`, `g++`, `g77`)
- **Compilers:** Portland Group Compiler Suite (`f77`, `f90`, `C`, `C++`, `HPF`, debugger, and profiler)
- **Math applications:** Matlab 6.1, GNU Octave 2.0, Maple 7 and 8, Mathematica 4
- **Debuggers:** `gdb`, as well as graphical frontends `xxgdb` and `ddd`

### 3.2.2 Libraries

- **Physics Applications:** ROOT, CERNlibs, GEANT 4, CLHEP
- **Numerical Libraries:** BLACS, SCALAPACK, LAPACK, LAPACK95, ARPACK, ARPACK++, ATLAS
- **Scientific Toolkits:** PETSC, Scientific Computing Toolkit (from Sandia), GNU scientific library (GSL), IMSL C and f90 libraries

### 3.2.3 Parallel Programming Toolkits

- **Internode parallelism:** MPICH, PVM (Parallel Virtual Machine)
- **Interprocessor parallelism:** `pthread`s, OpenMP (provided by Portland Group compilers)

### 3.2.4 Other applications

- **Document and Image preparation:** `xfig`, The GIMP,  $\text{\LaTeX}$
- **Editors:** XEmacs 21, Pico, NVI
- **Plotting and Fitting Tools:** Grace, Gnuplot, Supermongo, `mn_fit`

### 3.3 Notes on Installed software

Unless specified here, all software packages listed are available in default paths, without specifically setting up an environment for them. You can either run these commands when you need them, or add them to your shell init files (`.cshrc` for `csh/tcsh`, or `.profile` for `bash/sh/ksh`) so that the environment is set up correctly at each login.

#### 3.3.1 IMSL Libraries

IMSL Fortran and C libraries are installed in `/usr/local/vni`. To set your environment to access IMSL libraries, run the following commands:

- **csh/tcsh:** `source /usr/local/bin/cttsetup.csh`
- **bash/ksh:** `. /usr/local/bin/cttsetup.sh`
- To link with the IMSL libraries after running `cttsetup.(c)sh`:  
`pgf90 file1.o file2.o -o program $LINK_F90_SHARED` Or, use `$LINK_F90_STATIC` to link statically.

Remember to add a `source /usr/local/bin/cttsetup.csh|sh` line to your PBS scripts if you are using shared IMSL libraries. Otherwise, your PBS job will not be able to find the IMSL libraries when it tries to execute! Alternatively, you could link IMSL libraries statically into your program. `ld -static` will link a program statically.

Help and examples for IMSL can be found in `/usr/local/vni/CTT4.0/help` and `/usr/local/vni/CTT4.0/examples`, respectively. IMSL documentation can be found on the WWW at <http://amdahl.physics.purdue.edu/imsl>

#### 3.3.2 GEANT4

GEANT4 is installed in `/usr/local/geant4`. To set your environment to access GEANT4 libraries, run the following commands:

- **csh/tcsh:** `source /usr/local/geant4/env.csh`
- **bash/ksh:** `. /usr/local/geant4/env.sh`

Remember to source `/usr/local/geant4/env.(c)sh` in your PBS scripts if your program requires GEANT libraries.

GEANT4 documentation can be found on the WWW at <http://amdahl.physics.purdue.edu/geant>. Examples can be found in `/usr/local/geant4/examples`

#### 3.3.3 Mathematica

Due to licensing restrictions, Mathematica is nodelocked to the master node on amdahl, and cannot be run on the compute nodes.

In order to make sure that your X display has access to the fonts that Mathematica uses, you will need to make sure that the font files are available to your X server.

For example, to prepare a Unix system such as one available in rm. 139, log in to the local system with the display and run the following commands:

```
mkdir $HOME/Fonts
scp -r amdahl:/usr/local/mathematica/SystemFiles/Fonts/* $HOME/Fonts

xset fp+ $HOME/Fonts/Type1
xset fp+ $HOME/Fonts/AFM
xset fp+ $HOME/Fonts/BDF
```

And your X display will know about Mathematica's fonts. Future sessions can be made to use the Mathematica fonts simply by re-running the `xset` commands. A similar process can be used to run Mathematica on `heisenberg.physics`.

### 3.3.4 Maple 8

X11 Maple (`xmaple`) can be run on the master node, and provide a GUI, if run from an X display. Alternatively, Maple can be run from within a PBS script, in just the same manner as Matlab. See the section "Serial Computations" for an example.

### 3.3.5 Portland Group Compilers

The PGI compilers (`pgcc`, `pgCC`, `pgf77`, `pgf90`) and compiler wrappers that use the PGI compilers (`mpicc`, `mpif90`, `mpif77`) are only licensed to run on the master node of the `amdahl.physics` cluster. As a result, attempting to compile program code in a PBS script will probably fail.

## 3.4 Submitting Jobs for Execution

The cluster uses the OpenPBS resource manager for job submission. PBS enables users to simply define the parameters of what their job needs (x number of processors, with y amount of memory, and for z amount of time), and PBS considers all of these parameters and schedules the job for execution on the appropriate node(s). With PBS, there is no need to manually seek out a node that is unused to run your computations on, or worry about other users starting a job on the same node(s) as yours.

Jobs are submitted to PBS with a job script that defines a number of parameters to inform PBS about how to schedule it. We will step through a sample PBS script and explain what each directive does.

### 3.4.1 Example PBS Script

Commands that are directives to PBS will always begin with `#PBS`

These are how you tell PBS what to do, not comments!

```
#PBS -N jobname
```

Set a job name. This will allow you to identify your job in PBS.

```
#PBS -e jobname.err
```

```
#PBS -o jobname.log
```

Set filenames for PBS to log standard error and standard output.

```
#PBS -m abe
```

```
#PBS -M username@physics.purdue.edu
```

Send mail to the username following -M, when the event specified with -m happens to the job. ((a)bort, (b)egin, or (e)nd).

```
#PBS -l nodes=2:dual:athlon
```

```
#PBS -l mem=512mb
```

```
#PBS -l walltime=02:00:00
```

Request 2 nodes, with the properties “dual” and “athlon”. This job requests 512 MB of memory, and expects to run for 2 hours.

```
#PBS -W qos=1
```

**Important!** Specifies quality-of-service, so that jobs submitted go to an appropriate set of nodes. More information about QOS can be found below.

The remainder of a PBS script is essentially a shell script that runs the commands that make up your job. For example, the rest of a PBS script could be:

```
echo Running on host 'hostname'
echo Time is 'date'
echo Directory is 'pwd'
$HOME/computejob
```

### 3.4.2 Notes on PBS Scripts

One important point to be aware of is that PBS scripts will, by default, be run by the shell that is your login shell. For example, if your login shell is `tsh`, your PBS script will be run in `tsh`, and should use `cs`h syntax.

You can, however, make your PBS script run in any shell. For example, if you are a `tsh` user, but prefer writing scripts in Bourne shell (`/bin/sh`), you can make the script run in `/bin/sh` by starting the PBS script with a `#!` line, like `#!/bin/sh`. Remember what shell your script is written for, and it'll save many headaches.

Further examples of PBS environment variables, PBS directives, etc. can be found on the WWW at <http://amdahl.physics.purdue.edu/pbs/sample.pbs>

### 3.4.3 Node and Job Attributes

You can further refine exactly which nodes you want your job to run on by specifying node attributes. the `pbsnodes -a` command (see below) can help you see which attributes are defined for nodes. The table below illustrates some kinds of node properties you can choose from.

| Network | CPU Type | CPU MHz | Memory | SMP status |
|---------|----------|---------|--------|------------|
| gigabit | athlon   | p1800   | m1500  | dual       |
|         | p3       | p500    | m256   |            |

By using these node attributes, you can, for example, request that your job only be scheduled on a node that has an 'athlon' processor, and gigabit ethernet.

Node attributes are passed to PBS on a `-l` line:

```
#PBS -l nodes=1:athlon:gigabit
```

Or, if you need a dual processor system and 2 GB of RAM:

```
#PBS -l nodes=2:dual:m2048
```

#### 3.4.4 Job Walltime Estimates

A job's walltime is another attribute that is helpful to inform PBS about. (Think of walltime as the time elapsed when watching the clock on the wall.) Setting a good walltime helps the scheduler assign jobs to nodes in an optimal fashion. By default, PBS assumes that a job will take one hour, unless you tell it otherwise.

**This means that if you know that your job will take six hours, then you should certainly tell PBS, otherwise your job will be ended after an hour!**

You tell PBS your job's walltime with another `-l` directive:

```
#PBS -l walltime=4:00:00 # Set walltime to 4 hours
```

If you need to get an estimate of the walltime that your job will take, use the `time` command to get one:

```
time -p ./mycomputation
real 10:23
user 5:02
sys 3.63
```

Use the result from 'real' as an estimate for the computation's walltime. In this example, the walltime was 10:23, so requesting 12 minutes when running your production jobs is probably reasonable.

Accurately communicating your jobs' time requirements to PBS will allow PBS to allocate resources most effectively, and prevent your job from being prematurely terminated.

#### 3.4.5 Serial Computations

Non-parallel applications are equally well suited for cluster use. Simply develop and test your code as usual on the master node *amdahl*, and when it's time to run, submit your program into PBS, where it'll be scheduled and run on compute nodes.

In addition to programs developed from scratch, other computational programs can be run from PBS. Matlab, Maple, or Octave can be run from a PBS script on compute nodes. For example, to run a Matlab job on the cluster:

- Create a file containing the appropriate matlab commands that you wish to execute. For example, create a file called `cosplot.m`:

```
% MATLAB M-file example to approximate a sawtooth
% with a truncated Fourier expansion.
```

```
nterms=5;
fourbypi=4.0/pi;
np=100;
y(1:np)=pi/2.0;
x(1:np)=linspace(-2.0*pi,2*pi,np);
for k=1:nterms
twokm=2*k-1;
y=y-fourbypi*cos(twokm*x)/twokm^2;
end;
plot(x,y);
print -deps matlab_test_plot.ps;
quit;
```

- Run matlab from the command line of your PBS script:  
`/usr/local/bin/matlab < cosplot.m`
- Your job will be scheduled, run, with matlab processing the m-file and generating a plot, `matlab_test_plot.ps`.

### 3.4.6 Running MPI Parallel Programs with PBS

MPI Programs must be run from a program that acts as a launcher, `mpirun`. In order to run an MPI program from within PBS, there are a couple of points to remember:

Be sure to tell PBS how many processors your MPI job needs:

```
#PBS -l nodes=4
```

Now, run your executable on the nodes that PBS has assigned to your job. Make sure that the number of nodes you asked PBS for (`-l nodes=4`) matches the number that you tell `mpirun` to use!

You **must** use `mpirun`'s `-machinefile` option to tell it what nodes to run on. The environment variable `$PBS_NODEFILE` contains the list of nodes PBS has assigned to the job, and will ensure that your `mpirun` program runs on the nodes that PBS has allocated for it.

```
mpirun -np 4 -machinefile $PBS_NODEFILE mpi-program
```

### 3.4.7 Running PVM Programs with PBS

You can spawn PVM programs from within PBS as well. Simply submit your job with the relevant time, resource, `cpu`, etc. constraints, and use `$PBS_NODEFILE` again to tell PVM what hosts PBS has allocated for your job.

```
echo "add " 'cat $PBS_NODEFILE' | pvm
echo "conf" | pvm
pvm-program
```

### 3.4.8 Running OpenMP Programs with PBS

If you have compiled an OpenMP parallel program with `pgcc`, `pgf77`, or `pgf90`, (and specified `-mp`) you can run it with PBS as well. You will need lines similar to the following into your PBS job to run your OpenMP program:

```
$OMP_NUM_THREADS=2 # Set this to the num. processors per node
./openmp-program
```

When using SMP in a compute job, be sure to specify that you need SMP nodes when submitting the job to PBS:

```
#PBS -l nodes=1:ppn=2 # Request 1 node, with 2 processors per node
```

### 3.4.9 Combining Multiprocessor and Parallel Processing

To take advantage of nodes' SMP capabilities, as well as inter-node parallelism with MPI, programs can be written to make use of both MPI and OpenMP. Combine both steps listed above to submit the jobs to PBS.

### 3.4.10 PBS Commands

**qsub:** Once a PBS job script is created, it is submitted to PBS via the `qsub` command. In its simplest form, `qsub` takes a single parameter, the name of the script file that you wish to submit.

**qstat:** The `qstat` command will allow you to view the contents of the PBS queue.

```
node1:~/test> qstat
Job id          Name          User          Time Use S Queue
-----
147.node1      testjob       psmith        0 R default
```

**qdel:** The `qdel` command takes a single argument, a job number. You can use `qdel` to abort execution of your job: `qdel 147` would cancel execution of the job shown in the `qstat` example above.

**qalter:** The `qalter` command is helpful for altering the parameters of a job **after** it's submitted. `qalter` takes two arguments: the PBS directive that you wish to change (like `-l`), and the job number that you want to change. For example, if you forgot to set the walltime that your job requires, you can change it after it's been submitted:

```
node1:~> qalter -l walltime=4:00:00 147
```

**pbsnodes:** The `pbsnodes` command, while a useful PBS administration command, can also be informative to the PBS user. `pbsnodes -a` will list all PBS nodes, their attributes, and job status. This is a useful way to get a list of valid machine properties for use in a `#PBS -l` directive.

```
node1:~> pbsnodes -a
node2
    state = free
    np = 2
    properties = gigabit,pcn,m2048,dual,p1800,athlon
    ntype = cluster
```

### 3.4.11 Quality of Service

Quality of service is the mechanism where the owners of cluster nodes get guaranteed access to their nodes. The Quality of service mechanism works as follows:

- Node owners have a quality of service (QOS) defined for their nodes. The owner, and any other user that the node owner wishes to grant priority access to, are authorized to use a specific QOS.
- All users are able to specify a job with a QOS of 1, which allows the job to be run on **any** node in the cluster. The catch, though, is that the job will be subject to time limits, resource limits, and being out-prioritized by the node owner's jobs. Not specifying a QOS will give a job the default QOS of 1.
- Allowing all users access to all nodes allows the cluster's aggregate power to be much greater than it would be otherwise. In return, node owners gain priority access to nodes that they have contributed.

**For example:** Let's say that a specific group of researchers (let's call them the nuclear group) have purchased 5 nodes for their computations. A QOS is defined for all members of this group: "nuclear". Now, members of the nuclear group are able to specify jobs with a QOS set to "nuclear". With the QOS set to "nuclear", PBS will schedule the job to run in their nodes only, ignore any time or resource constraints, and give their jobs a priority boost of 100000, essentially keeping anybody else from being scheduled into these nodes while the owners have jobs scheduled.

Another user (we'll call him Bob), is a graduate student not authorized for any special QOS. He can submit his jobs with a QOS of 1, and the jobs will be scheduled on whatever node(s) are available. In this case, the "nuclear" nodes are available, and Bob's job is scheduled to run on one of them. Bob, though, is subject to a time limit: he cannot run a job for longer than, say, 6 hours on the "nuclear" nodes. If a "nuclear" user submits a set of jobs to the group's nodes three hours after Bob does, they have the guarantee that Bob's job will not use the "nuclear" node any longer than 3 more hours, and their job will be running when that 3 hours has passed. Node owners who expect to use their nodes very often can determine the time limit that they would like to impose on their nodes..

Additional capabilities are possible, such as reserving groups of nodes exclusively during time periods. Please contact PCN if you have special requests regarding node reservations.

### 3.4.12 Current QOS Settings

| Node Group | # CPU | Owner's QOS Name | Notes                                  |
|------------|-------|------------------|--|
| PCN        | 4     | pcn              |  |
| Star       | 10    | star             | Nodes configured for remote AFS access |
| Rockphys   | 4     | rock             |  |
| Neutron    | 8     | neutron          |  |

**All jobs submitted with the QOS of 1 are subject to the following limitations:**

- User may only have two jobs running at once, **or**
- Utilize a total of six nodes **or**
- There may only be a total of 5 QOS 1 computations running at once.
- If your QOS 1 computation is stuck in the “queued” state and not running, chances are, one of these restrictions is delaying it's running.

If these restrictions are insufficient to complete your computations, contact PCN, and a reservation can be created for special jobs. Alternatively, follow the instructions in Appendix A to contribute compute nodes, which will have no time or node number restrictions for your computations.

## 3.5 Effective use of Storage

### 3.5.1 NFS Storage

Currently, there is 50 GB of space within the cluster dedicated to home directories. There is no quota restriction on cluster home directories, so be aware of your disk usage. It's always possible for a single runaway job to fill up everybody's home directories. Read below for information on using shared and scratch space if you computations will generate a great deal of output.

For performance considerations, nodes within the cluster use a different home directory for your user account than on other PCN hosts like *curie* or *bohr*. Keeping home directories local to the cluster can greatly help I/O performance, as NFS traffic does not have to be routed off of the cluster's high speed private network. For convenience's sake, though, your regular PCN home directory is available as `/Home/username` on the master node *amdahl* only.

Nearly a Terabyte of RAID-5 storage is available within the cluster, in the location `/cluster/groupname`, much like storage is available on other PCN systems in `/project`. If you need space on the cluster RAID, please let PCN know, and a storage area will be created for your group.

### 3.5.2 Accessing Cluster Storage

If you need to upload data, files, etc. to your cluster directories, it is now possible to access it from outside the cluster! On PCN servers, (sorry, no Linux workstations) you can access everything via the automounter. For example, for user *psmith* to access his cluster home directory, he simply has to log in to *bohr* or *curie* and run

```
cd /cluster/psmith
```

. Similarly, to access the cluster files for his group, “PCN”, he can reach them in /cluster/PCN.

FTP is available to transfer files to and from the cluster as well. Just ftp to *ftp.physics.purdue.edu* and cd to /cluster/directoryname.

AppleTalk access is available now as well. Just add the proper lines to your .AppleVolumes file on *bohr* or *curie*:

```
/cluster/directory      Directory on Cluster
```

SMB (Windows Filesharing) can access the cluster via the server “FILES” in the PCN Windows Domain. The share “clusterhome” on FILES will serve the cluster directory of the authenticated user. Other directories (/cluster/groupname) will need a share created, so if you need SMB access, please let PCN know.

### 3.5.3 Local scratch and temporary storage

Each node offers two local filesystems intended to help programs which need higher-speed I/O than NFS can offer.

**/tmp:** /tmp is available on each node, purely as temporary storage. The size of /tmp space can vary on nodes, from 500 MB to 1 GB. /tmp is cleaned on nodes at each reboot, and nightly, so be sure not to leave anything in /tmp.

**/scratch:** /scratch is essentially a much larger /tmp filesystem. /scratch is not cleaned at reboots, instead, data can potentially remain on a /scratch space until it’s not been accessed for 14 days. /scratch filesystems are much larger than /tmp, to the order of 10-20 GB.

### 3.5.4 I/O tips

To most effectively make use of the disk resources within the cluster, consider the following scenario:

- A user edits, compiles, and tests a computational program interactively on *amdahl*.
- 30 GB of data sets are created (in 30 1 GB files), and uploaded into /cluster/groupname on the cluster.
- A PBS script is created, with steps like the following:

```
# Create scratch directory on the execution node
mkdir /scratch/$PBS_JOBID
# Copy data files into scratch directory
cp /project/user/datafiles/file1.dat /scratch/$PBS_JOBID
# cd into the /scratch directory
cd /scratch/$PBS_JOBID
```

```
# Run your executable, with input and output in the /scratch directory.
$HOME/bin/program < file1.dat > file1.out
# Copy files back home and cleanup
cp file1.dat /project/user/output && \
cd .. && rm -rf /scratch/$PBS_JOBID
```

- Taking steps like those above can help I/O-bound jobs, by not forcing every file read and write to do so via NFS, instead utilizing the compute nodes' locally attached storage.
- Taking steps like this will also reduce the possibility of a network hang causing your computation to terminate!

## 4 Useful Resources for Further Information

### 4.1 Further Reading

- Cluster status (Ganglia Cluster Toolkit) <http://amdahl.physics.purdue.edu/ganglia>
- MPICH Documentation: <http://amdahl.physics.purdue.edu/mpi>
- Portland Group Compilers: <http://amdahl.physics.purdue.edu/pgi>
- OpenMP Tutorial: <http://hpcf.nersc.gov/training/tutorials/openmp>
- CERNlibs documentation: <http://wwwinfo.cern.ch/asdoc/Welcome.html>
- IMSL documentation: <http://amdahl.physics.purdue.edu/imsl>
- GEANT4 documentation: <http://amdahl.physics.purdue.edu/geant>
- Beowulf Project: <http://www.beowulf.org>

### 4.2 Using the PCN RS/6000 systems for research computing

The host *heisenberg.physics.purdue.edu* is an IBM RS/6000, running AIX 4.3.3, with 4 PowerPC 604e processors, and 2 GB of memory. Heisenberg is a fully-supported research computing system, with IBM compilers, and many development languages and scientific toolkits. Contact PCN if you need access to *heisenberg*, or if you need a software package or toolkit installed for your computations.

Heisenberg is useful for long-running, parallel processing computations, which are easily migrated to Purdue's largest computing resource, the IBM SP.

Future enhancements may include clustering PCN's RS/6000 systems, adding the power of AIX to PCN's parallel computing offerings.

### 4.3 The IBM SP

Massively parallel applications that go beyond what a Linux cluster can offer can be run on Purdue's IBM SP. Any researcher can obtain access to the SP2 by obtaining a Form 6 from MATH 231.

An introduction to the SP2 can be found at <http://www-rcd.cc.purdue.edu/SP2/qs.html>

## 5 Appendix A: How to Contribute

- **Buy nodes:** You get priority access to nodes you buy, while being able to draw on the aggregate computational power of the cluster as a whole.
- **Buy networking hardware:** If your research requires high-speed networking, like gigabit ethernet or Myrinet, you can purchase the appropriate hardware, which PCN will install and maintain on your systems.
- **Buy Storage:** If your research requires large amounts of data storage, inexpensive Network Attached Storage devices can be easily added to the cluster to provide hundreds of gigabytes of storage.
- If you wish to contribute to the cluster in any form, please don't hesitate to contact PCN.

Access to the cluster is available for researchers who have not contributed hardware, but is subject to availability of compute resources.

## 6 Appendix B: Cluster Hardware Specifications

### PCN Compute Node)

- Dual Athlon MP 1800
- 1 GB Memory
- 18 GB Seagate Ultra-160 SCSI disk
- 3com 3c905b 100 Mbps and 3c996-TX gigabit Ethernet adapters

### Fileserver node

- Dual Pentium III, 1 GHz
- 384 MB Memory
- 18 GB Seagate Ultra-160 SCSI disk
- 3com 3c905b 100 Mbps Ethernet adapter

### Infrastructure

- 3com SuperStack 3300 24-port Fast Ethernet Switch
- 3com SuperStack 4900 12-port copper Gigabit Ethernet Switch
- Open Storage Solutions 220 GB Ultra-160 Omega SG2 RAID
- SnapAppliance Guardian 4400 640 GB Network Attached Storage

### STAR Nodes (5)

- Dual Athlon MP 1800
- 1.5 GB Memory
- 40 GB IBM DeskStar Ultra ATA disk
- 3com 3c996-TX Gigabit Ethernet adapter

### **Rockphys Nodes (2)**

- Dual Athlon MP 1800
- 1 GB Memory
- 40 GB IBM DeskStar Ultra ATA disk
- 3com 3c905 Fast Ethernet adapter

### **Neutron Nodes**

- (2 nodes) Dual Athlon MP 2000
- 512 MB Memory
- 40 GB IBM DeskStar Ultra ATA disk
- Intel EtherExpress Pro Fast Ethernet adapter
- Intel EtherExpress Pro 1000 Gigabit Ethernet adapted
  
- (2 nodes) Pentium 4 1.7 GHz
- 1 GB Memory
- 40 GB IBM DeskStar Ultra ATA disk
- 3com 3c905 Fast Ethernet adapter
  
- (2 nodes) AMD Athlon 1 GHz
- 1.5 GB Memory
- 40 GB IBM DeskStar Ultra ATA disk
- 3com 3c905 Fast Ethernet adapter