AMS machines

http://ams.soe.ucsc.edu/resources/computing

AMS has some decent sized machines!

- **NEW!** 3 interactive analysis servers, 32 cores each (Dell PE R820: 4 x Intel Xeon Sandy Bridge E5-4640 processor, each of which has 8 cores per cpu, 2.7 GHz, 16GB RAM, 1TB SATA hard drive). Machines are called “jerez”, “muscat” and “mencia”. **Use for regular (nonparallel) analysis**.

- **NEW!** New section of the cluster described below: 8 new nodes, where each node has 16 cores (Dell PowerEdge R420: 2 x Intel Xeon Sandy Bridge E5-2470 processor, each of which has 8 cores per cpu, 2.3 Ghz, 8GB RAM, 500 GB SATA hard drive) => total of 128 new cores. (All connected by Infiniband switch now). So the machine now in total has 128+104 = 232 cores.

**AMS cluster: GRAPE**

- **Use for parallel code**

- AMS owns a cluster of machines called **grape**

- The cluster consists of a master node and 12 compute nodes. Each node actually has 8 operational CPUs (known as cores), so there are a total of 104 cores (not too shabby). The oldest nodes are 6 Dell PowerEdge 1950 (Intel Xeon 2.33 GHz 16GB RAM), and the newer nodes are 6 Dell PowerEdge R610 (Intel Xeon 2.40Ghz 16GB RAM)

- To log in to the cluster, use ssh your_username@grape.soe.ucsc.edu using your SOE login. When you login as above, you login to the master/login node.

- Operating system is Rocks (a cluster management system related to RedHat Linux)

- Fortran, C compilers, MPI, and all the usual licensed software of SOE (Matlab, IDL, LAPack etc) are available

- Your home directory is your usual home directory.

- All usual directories (e.g. /projects, /cse/faculty) are cross-mounted
There are two sources of disk space: 12 Tbytes attached to the master node as /scratch, accessed from the compute nodes as /share/Arbeit and 1.4 Tbytes attached to the master node as /data and accessed from the compute nodes as /share/work. **None of the disk space is backed up! Use at your own risk.**

To use the machine for any serious computing, you should login in to the master node and use the batch system (described below) to submit jobs to the compute nodes. Logging in to the individual nodes disrupts the load balance of the machine and is NOT recommended.

The machine is basically split into three now. The first third of the nodes (compute-0-0 to compute-0-4) is made up of the 6 oldest nodes with batch queue called “orig”. The second third (compute-0-5 to compute-0-11) is made up of the 6 medium age nodes and is access through batch queue “new”. The last third (compute-0-12 through compute-0-19) is the newest nodes and is associated with the batch queue “newest”. Note that the oldest nodes do not allow hyperthreading (i.e. virtual threading: you cannot run more threads than there are cpus). There is also a queue called “default”. This uses all nodes across the Infiniband switch.

**Compiling parallel programs with MPI:**

If you want to run Fortran/MPI on grape and multitask, you need to do the following steps:

1. Decide which compiler you are going to use and set the environment using “modules”. To see what compilers are available, do “modules avail”. To see what compiler is loaded by default, do “modules list”. To switch to a particular compiler, do “modules switch <compiler>”

2. Compile your program using the related MPI compiler:

   mpi90
   (mpif77, mpicc)

   This produces an <executable>

**Running parallel programs:**

For the convenience of all users, you should use the Portable Batch System (PBS) job scheduler (open source version: Torque/Maui) to run parallel programs.

**Running batch jobs using PBS:**
Make a run script text file called "jobfile" (for example) which contains the following (for example):

```
<start of file>
#PBS -S /bin/bash ← or whatever Linux shell you wish to use
#PBS -u username ← your username
#PBS -N name ← some name to identify the job
#PBS -l nodes=2:ppn=8 ← number of nodes and processors per node
#PBS -l walltime=02:00:00 ← amount of time requested (2 hours)
#PBS -V ← passes all environment vars to processors (sometimes!)
#PBS -q debug ← queue: "orig", "new", "newest", "default"

cd $PBS_O_WORKDIR ←-- changes directory to current submission dir

### insert here any Linux shell commands you need to set up and run
### e.g.
cp /home/brummell/Code/nics_exec .
cp /home/brummell/Test/inputfile .

### Then run the job
mpirun -hostfile $PBS_NODEFILE -np no_of_total_processors nics_exec
@end of file>
```

To submit the job, type "qsub jobfile"

To examine the status of the job, type "qstat"

To kill the job, do a "qstat -u <your_username>" and get the job ID number, and then type "qdel <job_ID_no>"

**Running interactive jobs using PBS:**

You can run interactive jobs too: e.g.
```
qsub -l nodes=2:ppn=8 -I
```

This starts an interactive session on 2 nodes using 8 processors per node and you can then type in interactive commands in the parallel environment. If you are not running a parallel program, you should use `qsub -l nodes=1:ppn=1 -I`

PBS allows you to use the least busy nodes automatically, and to demand exclusive use of nodes (not the default).

For more information on using PBS, look at the man pages, search Google or here is a useful short summary of PBS concepts:
http://rcc.its.psu.edu/user_guides/system_utilities/pbs/
SOE machines

For interactive servers (e.g. Matlab use etc):
http://support.soe.ucsc.edu/servers

For another general cluster:
http://support.soe.ucsc.edu/craigslist