Grid 101

Josh Hegie
grid@unr.edu
http://hpc.unr.edu

Outline

Logging In

- Hostname: hnode.research.unr.edu
- Username: NetID
- Password: NetID Password

- If you dont have these talk to the help desk at the @One lab in the KC for help.

hnode

- Submit host and login server.
- Feel free to write and compile your code here.
  - Or consider compiling your code somewhere else. More on this later.
- Do NOT[1][2] run your jobs here.
  - Everyone has to log in and work on this node, so be sure to play nice.

[1] To clarify: This means under no circumstances, never, not just once, not because you're in a hurry, not because people are hogging the nodes.

[2] No. Really. Don't do it.

So What's hnode for?

- That not up there isn't completely inclusive. Here's the kind of things that are allowed on hnode:
  - Small, < 5 second, compiles.
  - X forwarding to use gedit, gvim, XEmacs or whatever other graphical editor you like[3].
  - Submit jobs.

[3] This will tend to be incredibly slow if you're not on campus, and sometimes even if you are.

Now What?

- I see "[user@hnode]$" so now what?

Now What?

- I see "[user@hnode]$" so now what?
  - I don't know, that's up to you. I'd recommend running some code.

Outline

Getting Stuff Done

- So I'm logged in and I want to compile OpenOffice.org on hnode.

Getting Stuff Done

- ~~So I'm logged in and I want to compile OpenOffice.org on hnode.~~
- So I'm logged in and I want to compile OpenOffice.org on a compute node.

Getting Stuff Done

- ~~So I'm logged in and I want to compile OpenOffice.org on hnode.~~
- ~~So I'm logged in and I want to compile OpenOffice.org on a compute node.~~
- So I'm logged in and want to compile and run a project for my research/class.

- How much storage do I get?
  - 1 GB of home.
  - 10 GB of scratch space.
  - Home is backed up, scratch is not.

Getting to a Node

- So I just ssh to any node I want?
  - No.
    - In fact a complete no.
    - If you're wondering how complete: it's the same no as mentioned for running jobs on hnode, but without the exceptions.
- But if I can't use SSH how do I get to a node?
  - qrsh

Getting to a Node the Right Way

- Issuing the command "qrsh" on hnode will give you a shell with 1 CPU on any available host you have access to.
  - Only 1, what if I want more?
    - More on that later.
  - That I have access to?
    - Nodes are added to the grid by research groups that purchase them. As a "tax" for IT to maintain the hardware, software, cooling, etc... 20% of each groups' nodes are added to the common pool, and the rest they get to have exclusive access to.

Outline

Running Jobs

- The Sun Grid Engine (SGE) is installed on the grid.
  - It provides the commands:
    - qsub
    - qrsh
    - qstat
    - qdel
  - It also handles all of the queuing and scheduling.

qsub

- Submits jobs to the queue.
- Takes a script file as an input[4]
  - In the scripts qsub flags begin with #$ and are generally optional.

[4] Unless you plan on being really crafty.

qsub Options

- -cwd
  - SGE assumes all scripts run from / or $HOME.
  - This tells it to use your current directory instead.
- -N <Name>
  - Normally SGE names your job after your submit script. This changes it to what ever you put for "Name" ("-N foo" would name the job foo).
- -pe <PE Name> <n>
  - Determines how slots are allocated.
  - More on this later.
- -M email@place.tld
  - Where to send emails to.

qsub Options

- -m bes
  - When to send mail. In this case it's when the job (b)egins, (e)nds or is (s)uspended.
- -q Lab.owner
  - Submit to the Lab queue instead of the common one.
  - Note: You have to have permission from the person in charge of the lab to get access to their private queues, and that request will have to go from them to the Grid administrator before you can get to those machines.
- -l mem_total=200G
  - Requests a node with at LEAST 200 GB of memory. If you don't have access to such a node, your job will sit in the queue until you kill it.
  - -l can take many more flags, if you care to find out more look them up in the man pages.
- There are many more options to qsub that are not covered here. If you want to find out more reference the man pages.

qrsh

- As mentioned before, qrsh gives you a shell on one of the compute nodes.
- qrsh takes many of the same flags as qsub, most notably "-pe".

qstat

- qstat is used to get information about jobs and whether they're queued, running or errored.
- Just calling "qstat" will show your jobs that you own.
  - If you use qstat -u '*' instead it will show you all running jobs.

qstat

- Code tends to not be perfect, and jobs will sometimes fail or error out.
  - qstat -j <job-id> gives information about a specific job, including scheduling errors.
  - qstat -explain E -j <job-id> will give a better explanation of some error states.
  - qstat -t provides information about which nodes are in use in an MPI/MPICH job, as well as which are the masters and slaves.

qdel

- If you need to remove a job for some reason use the qdel command.
  - qdel <job-id>
  - All resources will get cleaned up and all output pipes will be flushed and the job will be taken out of the queue.

Sample Submit Script

```
#!/bin/bash

#$ -cwd
#$ -pe SharedMem 4
#$ -N Shared_Memory_Sample_Job
#$ -M jhegie
#$ -m bes

echo "Hello World!"
echo -n "I've allocated $NSLOTS CPU cores on "
echo -n $(cat $TMP/machines)
echo " "
```

Sample Submit Script

- This creates a job with 4 slots on one machine
  - If there are no individual machines with 4 slots available it will simply wait for one to open up

Sample Submit Script

- You'll notice a few files in you directory now.
  - pe_run1.o<job-id> is your program's STDOUT.
  - pe_run1.e<job-id> is your program's STDERR.
  - pe_run1.po<job-id> is output from the parallel environment.
  - pe_run1.pe<job-id> is errors from the parallel environment.
    - The po and pe files should be empty.

Outline

What are the Compiler Options?

- GCC
  - 4.1.2: gcc, gfortran (F95), g++
  - 4.4.4: gcc44, gfortran44 (F95), g++44
- MPICH
  - /opt/mpich/gnu/bin/{mpiCC, mpicxx}
  - /opt/mpich2/gnu/bin/{mpicc, mpicxx, mpif77, mpif90}
- OpenMPI
  - /usr/lib64/openmpi/1.4-gcc/bin/{mpicc, mpiCC, mpic++, mpif77, mpif90}
    - Built from gcc 4.1.2
  - /usr/lib64/openmpi-1.5.3/bin/{mpicc, mpiCC, mpic++, mpif77, mpif90}
    - Built from gcc 4.4.4

Building on a Node

- What's the fastest way to build code?

Building on a Node

- What's the fastest way to build code?
- Build it using make's parallelization!
- qrsh into a node with more than 1 slot (I'll assume 4).
- cd into your project directory and call 'make -j4'.
  - This tells make to try and build using 4 threads.
  - This only works as intended in a shared memory environment (no getting 16 threads across 4 boxes), and doesn't guarantee a faster build.
  - Also be aware that finding the cause of a build failure is more difficult as all 4 threads share STDOUT/STDERR.

Outline

Compiling with OpenMPI

- Use /usr/lib64/openmpi/1.4-gcc/bin/{mpicc, mpic++} or /usr/lib64/openmpi-1.5.3/bin/{mpicc, mpic++, mpif77, mpif90}
  - For any FORTRAN code use the 1.5.3 build as there's a bug in the older version that may incorrectly cause your code to not compile.
    - And if you're still using FORTRAN...
- Replace the compiler variable or all calls to gcc/g++ with the appropriate MPI call.
- For the lazy: In your .bashrc, and at the top of your submit scripts, add "export PATH=/usr/lib64/openmpi-1.5.3/bin/" to not need fully qualified paths.

A Sample OpenMPI Script

```
#!/bin/bash
#$ -cwd
#$ -pe orte 16
#$ -N MPI_Sample_Job
#$ -M jhegie
#$ -m bes
#

export PATH=/usr/lib64/openmpi/1.4-gcc/bin:$PATH

mpirun -np $NSLOTS ./mpijob
```

A Sample OpenMPI Script

- Most of that should be familiar.
- What does "-pe orte" do differently from "-pe SharedMem"?
  - orte fills up all the available slots on a machine and then spills over onto the next one until all requested slots are provided.
- mpirun actually starts the job.
  - -np tells it the number of MPI processes to spawn.
  - $NSLOTS is an environment variable set be SGE.
  - The last argument is the program to run followed by it's own arguments.

Parallel Environments

- SharedMem
  - Used to allocate a shared memory job on one, and only one, node.
- orte
  - Fills all available slots on one node before spilling over to the next one.
- orte-rr
  - Like orte (in that it's for MPI), but fills slots round robin (one per machine until all machines have a slot taken, and then starts over)
  - If you're using this one use it on a limited queue and not the public queue.
- mpich
  - Similar to orte, but sets the environment up for MPICH and MPICH2 jobs.

Outline