Iowa High Performance Computing Summer School 2012

Getting Online

Welcome to the fourth annual Iowa High Performance Computing Summer School, June 6–8, 2012. I have compiled these notes as a reference to help you get online on Helium, the 3508-core shared cluster here at the University of Iowa, and on Moffett, the 4536-core SiCortex 5832 at the Rosen Center for Advanced Computing at Purdue University.

1 ITS Computer Training Room Workstations

For this course, we will be using the Macs in the ITS Computer Training Room, 2523 UCC. We use the Macs, rather than Windows machines, because the Mac OS X operating system is based on Unix, and therefore is a good platform for running Xwindows to connect remotely to the computer cluster. Here I outline how to get set up on the local workstation for easy use during this course.

1. If the computer at your workstation displays a Windows username and password input, tap the "Scroll Lock" key twice rapidly to toggle to the Mac.

2. Log onto the workstation using your HawkID and password.

   (a) If this does not work, log in once using username: `hpcuser` and password `hpciowa05312012`. Then, immediately log out (under the Apple menu) and try again to log in with your HawkID.

3. Open a Finder window by clicking on the Finder (face icon) at the left hand side of the Dock at the bottom of the screen.

4. Navigate to the Applications/Utilities/ directory and double click on X11 to start Xwindows. This will bring up an `xterm` terminal window which we will use to connect to the remote machines.

5. Useful software on the Mac workstations:

   (a) emacs: This is a very useful editor for writing code, among other things. It can be found in the Applications directory.
   NOTE: We have had problems with this application hanging up. If, after starting `Emacs` from the Applications directory, a window does not appear, follow these steps: (1) In the `xterm` window type
   `ps aux | grep Emacs`
   The second number after your username is the process ID, *pid*. (2) Issue the following command to kill `Emacs`:
   `kill` *pid*
   where *pid* is the integer number from the previous command. (3) Then launch `Emacs` again from the Applications directory, and it should work.

   (b) gnuplot: Handy plotting program. Can be launched from the xterm window by typing `gnuplot`.

   (c) Spaces: Virtual desktop for the Mac operating system. Can be found in the Applications/Utilities/ directory.

2 Logging onto and Running Parallel Programs on Helium

The directions in this section will get you logged onto and running on Helium.

1. Helium is a 3508-core shared cluster computer on the University of Iowa campus. The 359 compute nodes consist of the following configuration:
   (A) 200 compute nodes of:
   Dual Quad Core Intel(R) Xeon(R) CPU X5550 @ 2.67GHz Processors
   24 Gb DDR3 1333MHz Memory
   1Tb Storage
   Most of the interconnect is Infiniband, although some of the nodes are connected only by Gigabit Ethernet.
   (B) 159 compute nodes of:
   Dual Hex-Core Intel(R) Xeon(R) Processors
   24 Gb DDR3 Memory
   1Tb Storage

2. Online documentation for Helium can be found at
   `https://www.icts.uiowa.edu/confluence/display/ICTSit/User+Documentation`

3. Access to Helium uses Active Directory and requires your current HawkID and password. Accounts on Helium have been set up for you for use during this course.

4. To get connected to Helium, be sure you have Xwindows running on your machine and pull up an xterm window. Connect to helium using ssh,
   `ssh -X -4` *`username`*`@helium.hpc.uiowa.edu`
   where *`username`* is your HawkID, and enter your *password* at the prompt. This will put you into your home directory on Helium. Note that the `-X` option enables Xforwarding so that you can pull up an Xwindow of an application running on the remote machine (Helium) on the monitor of your local machine. The `-4` option forces the network to use the IPv4 address instead of the IPv6 address (using the IPv6 address can make the login process very slow).

5. Module Set Up: The first time, and only the first time, that we log onto Helium, we need to set up our `.bashrc` file so that the software modules that we need are loaded automatically.

   (a) To view the modules that are already loaded, use the command `module list`, and to see available modules, use `module avail`.

   (b) To add the necessary lines to the `.bashrc` file, in your home directory, open the `.bashrc` file using `emacs` in the background using
   `emacs .bashrc &`
   A window will open showing the contents of your `.bashrc` file. At the end of the file, after then line
   `# User specific aliases and functions`
   add the following two lines
   `module load intel_11.1.072`
   `module load openmpi_intel_1.4.3`
   Next hit `CTRL-x CTRL-s` to save the changes, and `CTRL-x CTRL-c` to quit `emacs`.

   (c) Now you may exit your login shell and follow the previous instruction to log in via `ssh` again. Once you have logged on again, you may use `module list` to verify that the necessary modules are loaded.

6. We are going to copy an example parallel code, `HYDRO`, to Helium using `scp`. Follow the instructions below:

   (a) Create a directory named `hydro` in your home directory on Helium
   `mkdir hydro`

   (b) Go to the IHPC 2012 website at `http://www.physics.uiowa.edu/~ghowes/teach/ihpc12/index.html` and follow the Examples link. Download the tar file of `HYDRO`, `hyd120605.tar`, to a directory on your local machine.

   (c) Open a new xterm window on your machine and navigate to the directory in which you just put `hyd120605.tar`.

(d) Now, we will copy this tar file over to your home directory on Helium using `scp`
`scp -4 hyd120605.tar` *username*`@helium.hpc.uiowa.edu:~/hydro/`
and enter your *password* at the prompt.

(e) In the window on Helium, go into the `hydro` directory and unpack the tar file
`tar -xvf hyd120605.tar`

7. Compiling the parallel code `HYDRO`:

(a) The tar file for `HYDRO` contains a `Makefile` that allows for easy compilation of the code on different platforms (different computers). `HYDRO` is written in Fortran90 using MPI for parallelization.

(b) Compile the code by typing
`make`
This will produce an executable `hydro.e`

8. Running parallel programs on Helium

(a) Helium uses the Sun Grid Engine (SGE) as a scheduler for parallel jobs. More information can be found online at `https://www.icts.uiowa.edu/confluence/display/ICTSit/Basic+Job+Submission`

(b) The examples below will use `HYDRO` with the sample input file `sample1.in`. The code requires the first argument after the executable to be the input file, thus the command will be
`hydro.e sample1.in`

(c) We can choose to run either interactively or in batch mode. Interactive runs generally run immediately (if resources are available), whereas running in batch mode puts the job into a queue to be run when resources become available. As we write our first parallel codes today, we will generally run in interactive mode, since we want the results right away (and since we have resources reserved for this course). Typically, when running your codes at a national supercomputing center, you will almost exclusively run in batch mode.

(d) BATCH MODE: The usual method for running on a shared cluster or at a national supercomputing center is to run in batch mode, submitting your jobs using a script.

i. An example shell script for running on Helium is included in the tar archive, `sample1_helium.sh`, for submitting a batch job to SGE to run `HYDRO` (for the input file `sample1.in`) follows:

```
#!/bin/sh
# Job Submission script

#$ -q IHPC
#$ -pe 12cpn 24
#$ -l h_rt=00:10:00
#$ -N sample1
#$ -o sample1.log
#$ -j y
#$ -V
#$ -cwd
#$ -l ib=1

echo "Job begin:"`date`
echo "Run sample1 on Helium, 24 proc"
mpirun -n 24 hydro.e sample1.in
echo "Job end:"`date`
```

ii. Each of the options on the lines above specifies a different aspect of the run:
`-q IHPC` specifies the IHPC training queue for the job (another option is `all.q`)

`-l h_rt=00:10:00` specifies the time limit in HH:MM:SS format
`-pe 12cpn 24` requests 24 cores using 12 cores per node
`-N sample1` specifies the name name of the job
`-o sample1.log` specifies the name of the file to send the standard output
`-j y` merges the stdout and stderr output,
`-V` imports environmental variables to the parallel SGE environment
`-cwd` places the output files in the current working directory.
`-l ib=1` requests Infiniband nodes only
The echo lines above simply write to the log file a few useful comments, but are not necessary.

iii. To submit the job, use
`qsub sample1_helium.sh`

iv. You may then check to see that the job is in the queue or running using `qstat -u` *username*. This will produce output that looks like
```
job-ID prior name user state submit/start at queue slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
736050 0.50944 sample1 ghowes r 05/31/2012 21:59:58 IHPC@compute-6-177.local 16
```
The `job-ID` in this example is `736050`.

v. If you are having to wait a long time for your job to start, you can change the processor request line in the script above to
`-pe orte 24` (requests any 24 cores using OpenMPI)
This command will take any 24 available cores, but since those cores may be on many different nodes, performance will likely suffer.

vi. To delete a job, submit a `qdel` command followed by the job-ID
`qdel 736050`

(e) INTERACTIVE MODE: The `qlogin` command is used to create an interactive session via SGE. Unlike many schedulers, using `qlogin` with SGE is unusually complicated, requiring the user to perform a number of steps that are normally handled by the scheduler for batch jobs. More details about this can be found in the online documentation at
`https://www.icts.uiowa.edu/confluence/display/ICTSit/qlogin`
The most important issue here is that it is up to the user to ensure that any MPI job started using `qlogin` employs *only* the hosts allocated by SGE. Therefore, it is very important to follow the directions below carefully to set up an interactive session using `qlogin`:

i. Begin an interactive session using 16 cores using the command
`qlogin -l ib=1 -pe orte 16 -q IHPC -l h_rt=02:00:00`
The options for this commmand specify the following:
`-l ib=1` requests Infiniband nodes
`-pe orte 16` requests 16 cores using OpenMPI
`-q IHPC` employs nodes in the IHPC training queue (`all.q` is another option)
`-l h_rt=01:00:00` specifies an interactive session time in hh:mm:ss

ii. The `qlogin` command will produce a result something like
```
Your job 734954 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 734954 has been successfully scheduled.
Establishing builtin session to host compute-3-63.local ...
```
As long as cores are available, you should not have to wait very long. The two important pieces of information here are the job number (`734954`) on line 2 and the masterq host (`compute-3-63.local`) on the last line.

iii. You may query the schedule to see only your own running jobs using the `qstat` command, `qstat` which will produce output something like the following,
```
qstat
job-ID prior name user state submit/start at queue slots ja-task-ID
-----------------------------------------------------------------------------------------
734954 0.50948 QLOGIN ghowes r 05/31/2012 14:19:15 all.q@compute-3-63.local 16
```
Here you can see that your interactive session is running on the queue `IHPC` using `compute-3-63.local` as the masterq host.

iv. To see the hosts that have been assigned to your interactive job using SGE, you can use the following command, `cat /opt/gridengine/default/spool/compute-3-63/active_jobs/734954.1/pe_hostfile` where you replace `734954` with your job number from line 2 and `compute-3-63` with your masterq host. A sample of the output follows:
```
compute-3-63.local 8 all.q@compute-3-63.local UNDEFINED
compute-3-64.local 8 all.q@compute-3-64.local UNDEFINED
```

v. The tricky part about using `qlogin` for an interactive session is that you need to specify the hosts in a way that OpenMPI will understand. To do so, you can use the command
```
cat /opt/gridengine/default/spool/compute-3-63/active_jobs/734954.1/pe_hostfile | awk '{print $1,"slots="$2}' > hostfile
```
to create the file `hostfile` which consists of the following lines:
```
compute-3-63.local slots=8
compute-3-64.local slots=8
```

vi. Now that you have everything set up, you are ready to perform and interactive run using the `mpirun` command. First, however, you will need to navigate to the directory that contains the parallel code that you want to run (note that launching `qlogin` will have landed you back in your home directory, which is probably not the directory in which you want to work). Thus, change directory to `/hydro`
```
cd hydro
```

vii. Run the test case `sample1.in` using the following command:
```
mpirun -n 16 hydro.e sample1.in
```
In this case, because you are running interactively, the output of `HYDRO` for this run will be sent to standard output (the screen).

viii. IMPORTANT: When you are finished running your interactive session, you should exit so that you free up the reserved cores for other jobs
```
exit
```

3 Logging onto and Running Parallel Programs on Moffett

The directions in this section will get you logged onto and running on Moffett.

1. Moffett is a 4536-core SiCortex 5832 located at the Rosen Center for Advanced Computing at Purdue University. You will be set up with a guest account on this machine during your participation in this course. Your guest account *username* and *password* will be given to you on paper. Detailed instructions on the use of Moffett can be found online at
`http://www.rcac.purdue.edu/userinfo/resources/moffett/newuser.cfm`

2. To get connected to Moffett, be sure you have Xwindows running on your machine and pull up an xterm window. Connect to moffett using ssh,
`ssh -X` *username*`@moffett.rcac.purdue.edu`

and enter your *password* at the prompt. This will put you into your home directory on Moffett. Note that the `-X` enables Xforwarding so that you can pull up an Xwindow of an application running on the remote machine (Moffett) on the monitor of your local machine.

3. We are going to copy and example parallel code, `HYDRO`, to Moffett using `scp`. Follow the instructions below:

   (a) Create a directory named `hydro` in your home directory on Moffett
   `mkdir hydro`

   (b) Go to the IHPC 2010 website at `http://www.physics.uiowa.edu/~ghowes/teach/ihpc09/index.html` and follow the Examples link. Download the tar file of `HYDRO`, `hyd100524.tar`, to a directory on your local machine.

   (c) Open a new xterm window on your machine and navigate to the directory in which you just put `hyd100524.tar`.

   (d) Now, we will copy this tar file over to Moffett using `scp`
   `scp hyd100524.tar` *username*`@moffett.rcac.purdue.edu:~/hydro/`
   and enter your *password* at the prompt.

   (e) In the window on Moffett, go into the `hydro` directory and unpack the tar file
   `tar -xvf hyd100524.tar`

4. Compiling the parallel code `HYDRO`:

   (a) The tar file for `HYDRO` contains a `Makefile` that will allows for easy compilation of the code on different platforms (different computers). This code is written in Fortran90 using MPI for parallelization.

   (b) First, open up the `Makefile` using emacs running in the background
   `emacs Makefile &`
   Edit the Makefile by changing the SYSTEM options variable
   `SYSTEM=SiCortex`
   and save the changes (using `CTRL-x CTRL-s`).

   (c) Compile the code by typing
   `make`
   This will produce an executable `hydro.e`

5. Running parallel programs on Moffett

   (a) Unlike many parallel computers which use Portable Batch System, or PBS, for job scheduling, Moffett uses the Simple Linux Utility for Resource Management, or SLURM. Although the syntax of the commands differs between these two systems, in general the operation is similar. Here we will give you a few of the basics on running parallel codes using SLURM on Moffett. Much more detail can be found online at `http://www.rcac.purdue.edu/userinfo/resources/moffett/newuser.cfm`

   (b) The examples below will use `HYDRO` with the sample input file `sample1.in`. The code requires the first argument after the executable to be the input file, thus the command will be
   `hydro.e sample1.in`

   (c) We can choose to run either interactively or in batch mode. Interactive runs generally run immediately (if resources are available), whereas running in batch mode puts the job into a queue to be run when resources become available. As we write our first parallel codes today, we will generally run in interactive mode, since we want the results right away (and since we have resources reserved for this course). Typically, when running your codes at a national supercomputing center, you will almost exclusively run in batch mode.

   (d) INTERACTIVE MODE: To run interactively, the following syntax applies
   `srun -p <partition> -n <tasks> <executable> [args]`
   Here, we select the default partition on Moffett, `scx-comp`. The number of tasks is the number of MPI processes, or equivalently the number of cores you choose to use. In the case of running `HYDRO` on 16 cores,

we would use
`srun -p scx-comp -n 16 ./hydro.e sample1.in`
In this case, because you are running interactively, the output of `HYDRO` for this run will be sent to standard output (the screen).

(e) To check the queue, use the command
`squeue`

(f) To cancel a job that has been submitted or kill a job that is running, use
`scancel <jobid>`

(g) BATCH MODE: The usual method for running on a shared cluster or at a national supercomputing center is to run in batch mode, submitting your jobs using a script. The scripts for SLURM are different from those using PBS, so here we will provide an example for using SLURM on Moffett. Here is a shell script named `sample1_moffett.sh` for running the `sample1.in` run with `HYDRO` using 16 cores

```
#!/bin/sh
#SBATCH -n 16
#SBATCH -t 00:10:00
#SBATCH -o sample1.log
#SBATCH -e sample1.err
#SBATCH -J sample1
#SBATCH -p scx-comp
echo Running hydro on 16 processors
srun ./hydro.e sample1.in
echo hydro run complete
```

Each of the options on the lines above specifies a different aspect of the run:
`-n` specifies the number of cores,
`-t` specifies the time limit in HH:MM:SS format,
`-o` specifies the name of the file to send the standard output,
`-e` specifies the name of the file to send the error output,
`-J` specifies the name of the submitted job,
`-p` specifies the partition on which to submit the job.
The echo lines above simply write to the log file a few useful comments, but are not necessary.

To submit the job, use
`sbatch sample1_moffett.sh`

You may then check to see that the job is in the queue or running using `squeue`.

4 Compiling Your Parallel Code

To compile an MPI code on either Helium or Moffett, you use `mpif90` for Fortran 90 or `mpicc` for C. Examples are below.

1. FORTRAN 90: If your code is name `program.f90`, you can compile using
   `mpif90 -o program.e program.f90`
   where the option `-o program.e` names the resulting executable `program.e` (the default behavior is to name the resulting executable `a.out`).

2. C: If your code is name `program.c`, you can compile using
   `mpicc -o program.e program.c`
   where the option `-o program.e` names the resulting executable `program.e` (the default behavior is to name the resulting executable `a.out`).

3. Note that on Helium, you must be sure that the following modules are loaded
   `intel_11.1.072`
   `openmpi_intel_1.4.3`
   You should have this set up in your `.bashrc` file, but if not you can load the modules yourself by issuing the `module load` commands yourself, for example, `module load intel_11.1.072`.