Informal introduction to supercomputing

Charlie Shobe

August 28, 2016

Contents

1 Introduction

This is a short cheat sheet to help get readers up to speed on how to drive the CU-CSDMS HPCC, more informally called "Beach." It assumes only very basic knowledge of terminal commands and the command line. Probably the things described here will be more difficult on a Windows OS than a Mac/Linux OS.

1.1 Acknowledgements

My limited understanding of supercomputing is thanks to Mariela Perignon, Mark Piper, Eric Hutton (all CSDMS), and Lauren Wheeler (U. New Mexico). All code and workflows shown here were developed in conjunction with one or more of them.

1.2 Structure of Beach

Beach is a collection of many "nodes," or individual computers that are networked together. Most nodes are "compute nodes," meaning that they are the nodes responsible for running your slow fancy model. The whole show is run by the "head node," which is the node you are automatically working in when you log in to Beach. Anything you formally ask Beach to run is called a "job." The head node is responsible for taking user job submissions and parsing them out to the different compute nodes to be run. To control the flow of jobs from users through the head node to different compute nodes and back again, Beach uses a job management system called Torque. Because of that, driving Beach involves a combination of Linux command line commands (when you're talking directly to the computer) and specialized Torque commands (when you're talking to the job management system).

1.3 Why use a supercomputer?

Some supercomputers are blazing fast. Beach is not blazing fast. Each node is about as fast as a standard laptop. However, there are still major advantages to working on a supercomputer instead of your own machine:

1. **Run multiple jobs at once**

   If you need to run your model many times, as most of us do, you can submit many jobs to Beach concurrently such that they will run at the same time. This means you're no longer harnessing the power of one laptop, but ten or twenty laptops simultaneously. This has enabled me to do hugely expensive parameter space analysis in hours or days, as opposed to months.

2. **Keep your computer free for development and debugging**

   Once a model is stable and simply needs to be run many times to generate data, those runs don't need supervision but still take up a large amount of memory and processing power. Outsourcing runs that don't need to be closely watched to a remote supercomputer frees up your personal computing resources for further development work even while your stable models are running.

3. **It's a form of backup**

   While keeping your models and driver files on a remote supercomputer should certainly not be the only form of backup you have, it's handy to know that you can always retrieve those files (subject to admin restrictions of course). This means that if you break your model on your computer or lose an essential file, it's easy to retrieve the stable version you stored on the supercomputer.

2 Cautionary notes

2.1 Don't run jobs on the head node

As discussed above, the sole purpose of the head node is to take your requests for jobs to be run and pass those jobs to the rest of the nodes for the actual computation. Please never, ever, ever run a model on the compute node. It only has so much processing power, and if it's slowed down by a large computation it will not be able to do its job of parsing out jobs and collecting their output. This makes the CSDMS and IT folks very unhappy.

What this comes down to in a practical setting is that you should never be typing something like `python run charlie-big-slow-model.py` into the command line when you're driving Beach. Because the head node is the node you're logged into by default, this would run the model on the head node. This is why users submit jobs through the Torque system using submission scripts. When a job is formally submitted to Torque, the head node knows not to run the job itself, but to pass it on to one of the compute nodes.

2.2 Don't use super user privileges

Beach uses a fairly user-trusting file system, in which you can install software packages to your working directory and do other things that aren't often allowed on shared computing resources. The downside of this is that it isn't terribly difficult to mess things up, not only for yourself but possibly for other users as well. What this

comes down to is that it's a bad idea to try to use super user privileges (or "sudo" in Unix speak). I did this once and it resulted in a quick email from IT.

3 Getting set up, logged in, and oriented

3.1 Getting an account

To get an account, you need to become a member of CSDMS (free) then apply for an account through the CSDMS website. It's a really easy process, and it takes 3-5 days for your account to be approved.

3.2 Logging in

Logging in to Beach is really easy. From a terminal window, simply type `ssh username@beach.colorado.edu`. Assuming your user name is connected with an active account, it will then prompt you for your password. Once you put in your password, you'll be logged in to Beach.

3.3 Getting oriented

To find out where you are in Beach's file system, type `pwd`. By default, you will always start in your personal directory after logging in. The path is `/home/username`. Generally you will not need to go *higher* in Beach's file system than this, because all of your work should be contained in your named directory. Within your named directory, you are king of the castle. You can create directories, copy files to and from them, edit text files, and do any number of other normal computer things from the command line.

3.4 Copying files to and from Beach

Most of the work involved with using a supercomputer is file management. You need to be able to copy files into your named directory so that Beach can find them to run your models. To copy files, Beach users are required to use `scope` (secure copy) rather than `cp` (copy). I don't know what makes `scp` more secure; it just is what it is. To copy a file from your computer to Beach, you need to be in the relevant directory and then type `scp myfile.txt username@beach.colorado.edu:/home/username`. The general syntax, as you can probably tell, is `copy this thing from this place to that place`, and any directory path after the colon is telling your computer to

copy the file(s) to that specific location on Beach. The structure is similar when copying files from Beach to your computer. This is especially relevant for retrieving model results. From the folder on your computer where you want the files copied to, typing `scp username@beach.colorado.edu:/home/username/myresults.dat .` will bring `myresults.dat` from Beach back to your machine. The `.` means that the destination is the current folder; you could also type the full path of wherever you want the file to go. `scp` also accepts multiple files in the same command, and full directories can be copied using `scp -r` (r for "recursive").

4 Submitting and monitoring single jobs

4.1 A test "model"

For this tutorial, we will use a very simple python "model" so that we can have confidence that no errors we see are due to errors in our model. Our "model" will read input parameters from a text file, do some basic calculations, and then write some output to a text file. In this case, it will read in a set of numbers and write out the average and standard deviation. With typical academic creativity, we will name it "model.py." It is important to note that the files you are running do *not* need to be python files. Beach supports C/C++, MATLAB, FORTRAN, and probably other languages that I've never even heard of.

4.2 Writing a submission script

Let's suppose we want to do a single model run on Beach. Remember that it is not likely to be faster than on your personal machine, but the other benefits stated above still apply. The simplest and most reproducible way to talk to the Torque job management system is to write a shell script that tells it how to run your model. The simplest possible script we could write to run our model would look like this:

```
#!/bin/sh
cd /home/chsh5846/folder-holding-model-and-inputs
python model.py
```

All that this script does is 1) tell Beach this is a shell script, 2) change directory to where our model and input data lives, and 3) run the model.

4.3 Submitting and monitoring jobs with Torque

Once we write this shell script, save it (say, `script.sh`) and copy it to our working directory, "submitting" the job to Beach is easy:

`qsub script.sh` By using the `qsub` submission command instead of simply running the shell script, we've ensured that the job will be distributed to a compute node rather than run on the head node. By default, any outputs from the model will be written to the same folder that the model and input data are in. In the case of our short "model," the execution time is effectively instantaneous. However, there are a couple of good Torque commands to know if you submitted a job that takes longer to run. `qstat` will bring up the queue, which is the full list of all jobs being run on Beach and those queued up but not yet running. You'll notice that every job has an ID number in the leftmost column. This ID is useful for the other commonly used Torque command, `qdel`. As you might imagine, `qdel #####` will delete the job with the number that was provided as an argument. There are many other ways in which these commands can be used, and many other Torque commands, but `qsub`, `stat`, and `qdel` are by far the most common.

5 Submitting and monitoring batch jobs

The real advantages of supercomputing become apparent when you submit many jobs to be run concurrently. While this could be done by changing the parameters (model name, input file, whatever) in your shell script, it is far easier to submit a script that will allow for "batch processing," meaning that a single shell script submits multiple jobs that are run concurrently on different compute nodes.

5.1 Submission scripts for batch processing

Submission scripts for batch processing are significantly more complicated than for a single model realization. This is because the different model driver files and input files must be kept in separate folders so that the output of older runs is not overwritten by the output of newer runs. I'm happy to share the batch scripts that I use, and examples are also available on the CSDMS wiki pages.

6 Resources

- CSDMS HPCC Guidelines page: Discusses logging in and copying files to and from Beach.
- CSDMS Torque Cheat Sheet: Has examples of basic submission scripts and a basic guide to the most common Torque commands.
- Torque online documentation: Has detailed documentation for all torque commands and workflows.