# 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) (Fig. 1). I did this once and it resulted in a quick email from IT.


Figure 1: Don't give yourself super user privileges while driving Beach.

# 3 Getting set up, logged in, and oriented

## 3.1 Getting an account

## 3.2 Logging in

## 3.3 Getting oriented

# 4 Submitting and monitoring single jobs

## 4.1 A test "model"

## 4.2 Writing a submission script

## 4.3 Submitting and monitoring jobs with Torque

# 5 Submitting and monitoring batch jobs

## 5.1 Submission scripts for batch processing

## 5.2 Submitting and monitoring batches of jobs

# 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.