ECON 512: Empirical Methods **Fall 2018/Spring 2019**

Lecture 16: Intro to PSU Computing Cluster

Instructor: Paul Grieco *Date: Mache 11*

These lecture notes are for my personal use. If you are reading them, I decided to distribute them as an experiment. There are typos and probably outright errors, if you find them please accept my apologies and report them to me.

In addition, information about the computing cluster tends to change over time. For that reason, and to avoid re-inventing the wheel, I'll use links to information whenever possible so that the underlying documentation should be up to date. Of course, the downside of this is that links will inevitably break.

16.1 High Performance/High-Throughput Computing

Throughout the course, we've worked with models that could be computed relatively easily on a common laptop. However it's easy to see that research level computing would take more computing power. Today we'll look at some of those resources that are available to you through Penn State. First, let's distinguish between High-Performance and High Throughput computing:

- **High Performance Computing:** Is tasks that require a large amount of computing power in a short period of time. Example: an advertiser needs to monitor who is clicking on a page and run ad auctions in real time while serving the ads without users feeling lags in page loads.
- **High Throughput Computing:** Tasks which require large amounts of computing over larger periods of time (days) and may be easily spreadable into independent tasks. Example: analyzing collected data from physics experiments to confirm the discovery of the Higgs boson.

Why do we care about this distinction? Because HTC is a lot easier than HPC. HPC will require high level parallelization and communication between processors to work in concert. HTC will look a lot more like a large number of computers doing independent things with some post-processing at the end. Fortunately, most economics research involves need for HTC.

Where is this useful in economics:

- Multi-starting an optimization problem.
- Using multiple starts of an MCMC.
- Performing a series of counterfactual simulations over a grid of parameter values.
- Bootstrap or subsampling for inference.
- Testing performance of an estimator via monte carlo simulation.

These are basically "parallelizable" as a sequence of independent jobs that don't talk to one another. Economists sometimes do parallelize within their code, but usually it is pretty simple.

- Solving a share inversion in BLP for multiple markets simultaneously instead of sequentially.

- Solving heterogeneous agents decision problems that are integrated over to find a likelihood value.
- Solving optimization problems that are part of a dynamic programming problem (e.g., value function iteration).

These sorts of tasks require parallel computations within an optimization, so they are not easily split into distinct jobs. You could handle them with something like a "parfor" loop in MATLAB. R, python and julia have similar capabilities. You probably would not need something as fancy as MPI/C++, but it would work if you wanted. The point here is you are now using multiple cores on a single job.[1]

So basically for today, we are thinking about high throughput computing. Essential code that could run on your laptop but takes all night, and you'd like to do that basic task 200-300 times.

16.2 Penn State's Cluster: ACI-ICS

At Penn State the computing cluster is ICS-ACI (Institute for CyberScience - Advanced CyberInfrastructure). Kind of a clunky name, but so be it. It's website is here: `https://ics.psu.edu/computing-services/` and the documentation is good and improving rapidly.

It consists of two closely related systems:

- ACI-I is a set of interactive nodes, which are equipped with graphics accelerators that make them nice for working with graphical interfaces. For the most part, if you want to work with an IDE, log in here. These machines do not do batch processing.
- ACI-B are the batch computing nodes. These are pure number crunchers. You can log into them directly, but doing much more than command line operations will be painful.

You can use these as you want. For my workflow, I usually log into an ACI-I node from my desktop using open on demand to bring up a graphical interface. Then from that interface have a terminal on ACI-B open for submitting jobs. I will show you what I mean.

16.3 Getting Started

Most of the info you need is here: `https://ics.psu.edu/computing-services/getting-started/`.

Grad students using an account must list a faculty member as an advisor. Any faculty member will do, but you should probably ask them first. In addition, the system sends the administrators purge student accounts every couple of years in the summer by emailing faculty and asking if accounts still need access. So its better if your faculty advisor is someone who works with you.

Your account gives you access to the open computing queue. The Department of Economics also manages a paid allocation with more rights. If you think you need this, see your advisor (not for class work). The open queue gives you access to 100 cores with 48 hour wall-time.

[1] I should be careful here, on modern multi-core systems, even something as simple as solving linear equations in MATLAB will make use of multiple cores for a single job automatically without the user having to do anything special like a parfor loop. For this reason, you may find that requesting multiple cores speeds up what "looks" like single core code.

16.4 Logging into ACI-I with Open On Demand

Most of the documentation for a graphical login uses Exceed on Demand `https://ics.psu.edu/computing-services/ics-aci-user-guide/#05-04-connecting-aci` however the system is migrating to Open on Demand, which allows access through a standard web browser (i.e., Chrome). Here's how:

1. Go to `https://portal.aci.ics.psu.edu/`, log in with your psu credentials (you have an aci account by now).
2. Under "Interactive Apps" select "ACI Interactive Desktop".
3. The default desktop environment, is usually fine. Click "Launch"
4. It will take a couple minutes for the desktop to be set up, have coffee.
5. Once the desktop is running, you can launch a VNC connection to it from "My Interactive Sessions". Once launched, this will "look and feel" like a linux desktop (because it is). You can close it down and reconnect it from another computer. It expires after the time you set, of course any work you do will be saved to the ACI file system.

16.5 Working within ACI

16.5.1 It runs Linux...

More or less, this is like any linux system. Some basic linux commands are listed here `https://ics.psu.edu/computing-services/ics-aci-user-guide/#05-01-system-usage`

16.5.2 Modules

One mild quirk is the use of modules to keep the software stack that is available appropriate for each user. It just means that you must load specific software you want to use. You can see what modules are available with:

```
module avail
```

And you can load a module with e.g.,

```
module load matlab
```

You can also arrange to pre-load modules by adding them to your BASH profile. This just saves you from typing "module load matlab" or similar every time you log in.

Full details: `https://ics.psu.edu/computing-services/ics-aci-user-guide/#05-02-module-system`

16.5.3 Logging into the batch nodes

I just log in within my ACI-I session, I use ssh:

```
ssh aci-b.ics.aci.psu.edu
```

I do all coding/debugging work on ACI-I, this is just a window to use to submit jobs. There are other ways to do this as well.

16.5.4 Moving Files on/off ACI

Lots of options: `https://ics.psu.edu/computing-services/ics-aci-user-guide/#05-05-transferring-data-aci`, I mostly use sftp, because I am old.

There is a graphical interface to your files via open on demand under the "Files" tab.

Also, git is available if you want to store your code in a repository in the cloud (bitbucket or github). I've found they are a conveient way of going between your desktop, laptop, and the cluster.

16.6 The point of it all: Submitting Jobs

Again, this documentation is a good place to start: `https://ics.psu.edu/computing-services/ics-aci-user-guide/#07-00-running-jobs-on-aci-b`. I'll go through submitting a single job.

ACI-B compute nodes use the TORQUE resource manager and Moab scheduler. You interface with these using the Portable Batch System (PBS). PBS commands look like comments in a regular shell script. You then submit that script to the scheduler using the command `qsub`. So the workflow is:

1. Write your program in your favorite software package.
2. Write a short bash script that sets the PBS environment variables, and opens and runs your program.
3. (Optional, but a good idea) Test your script by running it directly from the command prompt.
4. Submit it to the scheduler using `qsub <myscript>`

This workflow submits a single job. Once you do this, you'll want to write a script to submit many jobs, that just involves writing a loop to automate writing the shell scripts for different parameters and submitting them.