High Performance Computing for BU Economists

Marc Rysman

Boston University

July 16, 2020

Introduction

- We have a fabulous super-computer facility at BU.
- It is free for you and easy to gain access.
- It can help you with your research and make you happy.

Why use a super-computer?

- Access many processors simultaneously:
  - You can complete you computer jobs faster.
  - You can complete many jobs at once.
- Access from any computer (or phone?)
  - Check on your jobs.
  - Start them with new parameters.
- Super-computers don't get restarted.
- Look cool – impress friends (and potential employers)!

Outline

1. When is parallel processing helpful in economics?
2. What is the super-computer at BU?
3. How do I get an account?
4. All the Linux you need to know.
5. An example of parallel processing.

Parallel Processing

- Using multiple processors simultaneously.
  - or multiple nodes of a processor simultaneously.
- Can mean that your program splits into different threads and then comes back together.
  - Also called *data parallel*
- Or it can mean that you submit pieces of your job simultaneously to the cluster.
  - Also called *task parallel* or *embarrassingly parallel*.

Matrix multiplication

- Suppose you wish to do matrix multiplication $Z = XY$.
- Break up X two parts, so:

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

- Now separately compute:

$$Z_1 = X_1 Y \quad Z_2 = X_2 Y \quad \text{and let} \quad Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

- Compute Z_1 and Z_2 simultaneously to compute Z in half the time.
- This is what Gauss and Stata do automatically when you do matrix multiplication.

Simulation

- Suppose you are computing the probability of some event as a function of a parameter $P(\theta)$:

$$P(\theta) = \int_{\nu} \mathbb{1}\{f(\theta, \nu) = 1\} g(\nu) d\nu.$$

  - Example: f returns whether a worker is employed, or whether a firm enters.
- You wish to simulate. That is, you draw ns values of ν from $g(\nu)$ and compute:

$$\widehat{P(\theta)} = \frac{1}{ns} \sum_{s=1}^{ns} \mathbb{1}\left\{f(\theta, \nu^s) = 1\right\}.$$

Parallel simulation

- We can break up draws into two sets of draws of size $ns/2$, and compute simultaneously:

$$\widehat{P_1(\theta)} = \frac{1}{ns/2} \sum_{s=1}^{ns/2} \mathbb{1}\{f(\theta, \nu^s) = 1\}$$

$$\widehat{P_2(\theta)} = \frac{1}{ns/2} \sum_{s=ns/2+1}^{ns} \mathbb{1}\{f(\theta, \nu^s) = 1\}$$

- Then:

$$\widehat{P(\theta)} = \frac{\widehat{P_1(\theta)} + \widehat{P_2(\theta)}}{2}$$

Bootstrap

- We have a statistic $T(Z)$ computed as a function of data set Z.
- We draw ns samples from Z, creating samples Z_s.
- We compute statistic $T_s = T(Z_s)$ from each sample, and use the distribution of T_s to make inference on $T(Z)$.
- With parallel processing, we compute values of T_s simultaneously.
- Amenable to embarrassingly parallel approach.

Dynamic Programming

Consider the following investment problem:

- Capital k is discrete from 1 to $\overline{k}$ (set $\overline{k} = 40$).
- k depreciates each period.
  - Drops 0 states with prob p_1, 1 state p_2 and 2 states p_3.
  - $p_1 + p_2 + p_3 = 1$.
- Firm makes a binary choice whether to invest or not: $a \in \{0, 1\}$.
- $a = 1$ raises k by 5 next period, and then depreciation applies.
- Flow profit is $\pi(k)$ and cost of investment is c.
- Firms draws logit ε_a for each choice each period.

Bellman Equation

$$V(k, \overrightarrow{\varepsilon}) = \max_{a \in \{0,1\}} \quad \left\{ \pi(k) + \varepsilon_0 + \beta E\left[V(k', \overrightarrow{\varepsilon}') | k, a = 0, \overrightarrow{\varepsilon} \right], \right.$$
$$\left. \pi(k) - c + \varepsilon_1 + \beta E\left[V(k', \overrightarrow{\varepsilon}') | k, a = 1, \overrightarrow{\varepsilon} \right] \right\}$$

$$k' = \begin{cases} k + 5a & \text{with prob } p_1 \\ k - 1 + 5a & \text{with prob } p_2 \\ k - 2 + 5a & \text{with prob } p_3 \end{cases}$$

Impose logit ε and integrate:

$$V(k) = \ln \left(\sum_{a \in \{0,1\}} \exp\left(\pi(k) - c\mathbb{1}\{a = 1\} + \beta E\left[V(k') | k, a \right] \right) \right)$$

Empirical implementation

Define transition matrices:

$$T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ p2+p3 & p1 & 0 & 0 & 0 & 0 & \dots \\ p3 & p2 & p1 & 0 & 0 & 0 & \dots \\ 0 & p3 & p2 & p1 & 0 & 0 & \dots \\ 0 & 0 & p3 & p2 & p1 & 0 & \dots \\ \vdots & & & & & & \end{bmatrix}$$

$$T_1 = \begin{bmatrix} 0 & 0 & 0 & p3 & p2 & p1 & 0 & \dots \\ 0 & 0 & 0 & 0 & p3 & p2 & p1 & \dots \\ 0 & 0 & 0 & 0 & 0 & p3 & p2 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & p3 & \dots \\ \vdots & & & & & & & \end{bmatrix}$$

Bellman again

$$\underset{40\times 1}{V(k)} = \ln\left(\sum_{a\in\{0,1\}} \exp\left(\underset{40\times 1}{\pi(k)} - c\mathbb{1}\{a=1\} + \beta \underset{40\times 40}{T_a} \quad \underset{40\times 1}{V(k)}\right)\right)$$

- Break up elements of bellman equation:

$$V(k) = \begin{bmatrix} V_1(k_1) \\ V_2(k_2) \end{bmatrix} \quad \pi(k) = \begin{bmatrix} \pi(k_1) \\ \pi(k_2) \end{bmatrix} \quad T_a = \begin{bmatrix} T_{1a} \\ T_{2a} \end{bmatrix}$$

- Then compute simultaneously:

$$V_i(k_i) = \ln\left(\sum_{a\in\{0,1\}} \exp\left(\pi(k_i) - c\mathbb{1}\{a=1\} + \beta T_{ia} V(k)\right)\right)$$

What is at BU?

- Within the IS&T, there is the Research Computing Services (RCS) Group.
- They manage the Scientific Computing Facility (SCF).
- The SCF includes the Shared Computing Cluster (SCC).
- The SCC runs Linux and has more than 7000 processors.
- It has many programs such Stata, SAS, Matlab, Gauss, R, C, Fortran.

Where are the computers?

- The facilities are at the the Massachusetts Green High Performance Computing Center (MGHPCC) in Holyoke, MA.
- MGHPCC was opened in 2013.
- Building is joint with Harvard, MIT, UMass and Northeastern, with substantial state support.

How do I get access?

- Graduate students cannot have their own account.
- You can be added to a faculty account, or to the department account.

Faculty or department account?

- Use faculty accounts for *joint work or research assistance* with a faculty member.
- Use the department account for *your own work*.

- To get access: send e-mail to the RCS Liason with your full name, your login name (Kerberos), and the e-mail address that you want to use.

Using the SSC

- The next several slides describe software for accessing the SSC and some basic Linux commands.
- However, RCS has recently introduce a new browser-based graphical system for accessing and using the SCC that is arguably superior.
- I leave the following slides here for now, but the new method is:

  `http://www.bu.edu/tech/support/research/system-usage/scc-ondemand/`

What software do I use?

You need:

- A telnet-type software to log into the SCC cluster and issue commands.
- An FTP type software to move files from your computer to the SCC.
- BU gives you this for free.
  - X-Win32
    - `http://www.bu.edu/tech/desktop/site-licensed-software/xwindows/`
  - FileZilla
    - `http://www.bu.edu/tech/desktop/support/software/windows/filezilla/`
  - MobaXterm.
    - Does both telnet and FTP. More powerful but a little more complicated.
    - Free at `http://mobaxterm.mobatek.net/`

Setting up X-Win

- Use the wizard to create a new connection.
- Name: SCC
- Type: ssh
- Host: scc1.bu.edu (or scc2)
- login: <userID>
- password: <password>
- command: Linux XTERM
- Accept the host server public key – first time only.
- If you need to enter a port, try 22.

Linux commands
Directories

- Create a directory: `mkdir dirname`
- Switch to a sub-directory of the current directory:
  `cd dirname`
- Switch up a directory: `cd ..`
- Switch to a sub-directory of a different directory:

  ```
  cd ~\dir1\dir2\dirname
  ```
- Remove a directory: `rmdir dirname`

Linux commands
copy and remove files

- See the contents of a directory: `ls`
- See the contents plus more information: `ls -l`
- Copy a file from the current directory to a different directory:

  `cp filename ~\dir1\dir2\`
- Remove a file: `rm filename`
- Create or edit a file: `pico filename`
  - RCS has developed its own Linux cheat sheet:

    `http://scv.bu.edu/documents/Linux_SCC_CheatSheet.pdf`

The Batch System

- You can use software in interactive mode or batch mode.
- Interactive mode is how you use Stata or Matlab on your PC.
  - The program stops if you shut your computer, or close X-Win32.
- In batch mode, you submit the code to the batch processor.
- The cluster will process jobs based on priority.
  - Shutting your computer does not affect the job. Great for long jobs!
- The slides describe some commands to use, but RCS provides its own cheat sheet for these commands: `http://scv.bu.edu/documents/SCC_CheatSheet.pdf`

Batch commands

- Submit a job: `qsub` plus a lot of other stuff.
  - See next few slides.
- See the queue: `qstat`
- See the jobs submitted under your username: `qstat -u <username>`
- Delete a job: `qdel jobNumber`.

Interactive mode

- Gauss: type: `gauss`
- Matlab: type: `matlab`
- SAS: type: `sas`
- Stata: type: `xstata`

Multiprocessor Interactive

- By default, interactive mode uses a single processor.
- To use multiprocessor interactive mode, type: `qsh -pe omp 4`
- This brings up a new interactive window that uses 4 processors.
- Then type: `xstata-mp`
- Getting to large memory nodes is similar, and depends on how many processors you ask for. See: `http://www.bu.edu/tech/support/research/system-usage/running-jobs/batch-script-examples/#MEMORY`

Submit a job in Gauss

- From the directory of your file `prog.g`, type:

```
qsub  -b y tgauss -b prog.g
```

  - `qsub` is the submission command
  - `-b y` tells that `tgauss` is a binary file.
  - `tgauss` is the batch version of gauss.

Submit a multi-processor job in Gauss:

- Include `-pe omp 4`
- Tells how many processors to use.
  - Can be up to 16 in Gauss, or 64 more generally.
- Sometimes Gauss uses too many processors. This can be controlled with

  `OMP_NUM_THREADS=4`
- as in:

  ```
  qsub -v OMP_NUM_THREADS=4 -pe omp 4 -b y tgauss

   -b prog.g
  ```

`-v` controls environment variables

Wall clock time

- The job will automatically be killed after 12 hours of “wall clock time."
- Pretty annoying!
- You can control this – the max is 5 days (120 hours) for some multiprocessor jobs and 30 days for single-processor jobs, and some multiprocessor jobs (called OMP instead of MPI jobs). Type:

```
-l h_rt=120:00:00
```

- as in:

```
qsub -l h_rt=120:00:00  -pe omp 4 -b y

tgauss -b prog.g
```

Submit a job in Stata

- Create a file called `stataBatch.csh` with 2 lines to it:

stataBatch.csh:

```
#!/bin/csh

stata-mp -b do program.do
```

- Submit the file: `qsub -pe omp 4 stataBatch.csh`

Submit a job in Matlab

- To run `program.m`, create a file called `matlabBatch.csh` with 2 lines to it:

matlabBatch.csh:

```
#!/bin/csh

matlab -nodisplay -r program
```

- Submit the file: `qsub matlabBatch.csh`
- Note: no `.m` to `program` in `matlab` command.

Processors in Matlab

- If Matlab is using too many processors:

```
matlab -nodisplay -singleCompThread -r program
```

- Or ask for multiple processors:

```
qsub -pe omp 4 matlabBatch.csh
```

- If Matlab asks for too many processors, get help!

Hints for batch jobs.

- The first time you use the batch system, your job will probably start right away.
- As you use it more, the *fair use* policy will lead to your priority being degraded and you will queue longer.
- Using less processors or asking for less duration improves your job's priority rating.
- Asking for 12 hours or less is great because then you can run on other peoples' cluster computers that they are not using.

The Economics Group at the SCC

- In addition, the Economics department holds a set of computers at the SCC.
- Members of our department, including graduate students and faculty, have priority on these computers.
- If your job is queuing and these computers become available, you will be put ahead of SCC users outside of our department that submitted before you or otherwise had higher priority.
- We have dedicated two of the computers to be "whole-node only." You can get on them only if you are asking for 28 cores.
- If you are running jobs with many cores, you will see an improvement in performance if you ask for 28 cores rather than some other number.

Parallel processing in Gauss

Gauss calls it *multithreading*

- type `threadstat` in front of commands to be executed in parallel.
- type `threadjoin` to tell Gauss that parallel part is over.
- Example: Suppose matrix *B* has 20 rows and you wanted to parallelize `A = B*C` across 2 processors.
- Single-processor:

```
A=B*C;
```

- Multi-processor:

```
threadstat A[1:10,.]=B[1:10,.]*C;
threadstat A[11:20,.]=B[11:20,.]*C;
threadjoin;
```

- Gauss help says to use multi-threading only if the action takes more than 0.01 seconds.

Parallel processing commands in Matlab

- I know less about this, and it seems more complicated.
- There are two approaches in Matlab:
- `parfor` processes loops in parallel.
- `SPMD` is for everything else.
- Both approaches require:
  - `matlabpool open 4` says that we will use 4 processors.
  - `matlabpool close` says that parallel section of code is over.
- `help@scc.bu.edu` will help with your code.
- Also, there are tutorials on the SCV web page for parallel processing in Matlab.

An example with `parfor`

- Do matrix multiplication row-by-row:
- Single processor:

```
for i=1:20
    A[i,.] = B[i,.]*C;
end;
```

- Multiple processor:

```
matlabpool open 4
parfor i=1:20
    A[i,.] = B[i,.]*C;
end;
matlabpool close
```

- `parfor` will break up job into 20 pieces that will be handled as processors become available. `matlabpool` asks for 4 processors.

More in Matlab parallel processing

- More complex parallel processing can be handled with the `SPMD` command.
- But I can't teach that, although I do provide an `SPMD` example on the web site.
- In general, my understanding is:
  - If you were going to loop anyway, use `parfor`.
    - Example: Simulating agents.
  - If you are breaking up a complex problem into pieces, use `SPMD`.
    - Example: Value function iteration.
  - Note: I can often avoid looping over simulated agents by using multi-dimensional matrices (which is very fast!).

Code-tuning example

- Based on real experience with a former grad student, now professor in Beijing.
- Suppose we wish to simulate a model:
  - Many independent markets.
  - In each market, K firms choose enter or not in an exogenous order. in which firms choose enter in an exogenous order.
  - Firms enter if:

$$\pi_{im} = x_{im}\beta + \delta N_{i-1m} + \varepsilon_{im}$$

  - N_{i-1m} is number of entrants before i.
  - $\varepsilon_{im} \sim \mathcal{N}(0, 1)$.
- This was nested in an estimation algorithm so we needed to do this many times.

First round of code

1. Loop $s = 1, \ldots, S$ (draws)
    2. Loop $m = 1, \ldots, M$ (markets)
        3. Get firms in $\{m, s\}$.
        4. Sort firms in $\{m\}$ by i (entry order).
        5. Loop $i = 1, \ldots, K$ (firms)
            6. Compute $\pi_{ims} = x_{im}\beta + \delta N_{i-1ms} + \varepsilon_{ims}$
            7. Set $N_{ims} = N_{i-1ms} + \mathbb{1}\{\pi_{ims} > 0\}$
        8. Go to 5.
    9. Go to 2.
10. Go to 1.

Problems

- We should sort firms only once, and we are doing it S times.
- Clever use of matrices could eliminate a loop (but not the i loop).
- Parallelize outermost loop.

Improved code

1. Sort firms by i in every market m.
2. Parallel Loop $m = 1, \ldots, M$ (markets)
    3. Loop $i = 1, \ldots, K$ (firms)
        4. Compute $\pi_{im.} = x_{im}\beta + \delta N_{i-1m.} + \varepsilon_{im.}$
            - for all draws at once.
        5. Set $N_{im.} = N_{i-1m.} + \mathbb{1}\{\pi_{im.} > 0\}$
    6. Go to 3.
7. Go to 2.

20 times faster.

Tutorials

- Self-teaching slide-sets: `http://www.bu.edu/tech/research/training/tutorials/list/`
- Classes (free): `http://www.bu.edu/tech/training/classroom/scv-tutorials/`
- Examples:
  - Introduction to Linux
  - Introduction to R
  - Introduction to SAS
  - Introduction to MATLAB
  - Tuning MATLAB code for better performance
  - MATLAB Parallel Computing Toolbox
  - Graphics and Images for Publication and Presentation

Conclusion

Happy Dissertating!