

Cluster Computing

Author: Andrea S Foulkes, Gregory J Matthews, Nicholas G Reich

Biostatistics in Practice: High-Performance Computing with R
7 February 2014

*This material is part of the **statsTeachR** project*

Made available under the Creative Commons Attribution-ShareAlike 3.0 Unported License: http://creativecommons.org/licenses/by-sa/3.0/deed.en_US

Overview

- ▶ Parallel computing allows us (for certain problems) to split a big job up into many smaller parts and run them in parallel.
- ▶ Our laptops or desktops can split a job up into as many threads as are available.
- ▶ This is likely a relatively small number like 4, 8, or 12.

Overview

- ▶ **Cluster computing** connects many computers (nodes) together on a local network.
- ▶ This allows a pooling of resources to increase computing power.
- ▶ This allows a user to access hundreds or even *thousands* of cores (if they need them).

MGHPCC

- ▶ The Massachusetts Green High Performance Computing Center (MGHPCC) is one of these clusters.
- ▶ Each of the participating institutions has their own distinct cluster. We will be using the UMass cluster.
- ▶ In order to connect to this cluster, a user has to be on the UMass campus (or use VPN to the UMass campus.)
- ▶ We will need to VPN into the UMass network since we are currently off campus.

Hypothetical cluster computing workflow

- ▶ Transfer data/scripts to cluster
- ▶ Log in to the cluster
- ▶ Submit a job to the “scheduler”
- ▶ Transfer data/results from cluster if needed

The process we will use today

Our order of operations...

- ▶ Log in to the cluster
- ▶ Transfer data/scripts to cluster
- ▶ Submit a job to the “scheduler”
- ▶ Look at results

The process we will use today

Our order of operations...

- ▶ **Log in to the cluster**
- ▶ Transfer data/scripts to cluster
- ▶ Submit a job to the “scheduler”
- ▶ Look at results

How to pretend you are at UMass using VPN

- ▶ Mac: System Preferences > Network > VPN on left
- ▶ Windows: Cisco VPN client
 - ▶ Server address: vpn2.oit.umass.edu
 - ▶ Group name: umass
 - ▶ Password: vpn4umass
 - ▶ Account name: UMass user ID OR bip[number] (temporary for BIP2014)
 - ▶ Password: Enter your password

Accessing the MGHPCC

- ▶ Register for the MGHPCC (for UMass):
<https://www.umassrc.org/hpc/>
- ▶ A login and password will be (have been?) assigned to you.
- ▶ At the moment, this will be different than your UMass user ID or email address. (NetID integration coming soon!)

Accessing the MGHPCC

- ▶ Unix or Mac
 - ▶ Unix or Mac: Create a secure shell: `ssh username@ghpcc06.umassrc.org`
- ▶ Windows
 - ▶ PuTTY: `ghpcc06.umassrc.org` (you will be prompted for user name and password)
- ▶ Enter password when prompted. (First time you login, you will need to change your password, then login again.)
- ▶ You are now connected to the cluster!
- ▶ Remember: You can only access the MGHPCC if you are on UMass's network (physically on campus or VPN)

The process we will use today

Our order of operations...

- ▶ Log in to the cluster
- ▶ **Transfer data/scripts to cluster**
- ▶ Submit a job to the “scheduler”
- ▶ Look at results

Transferring Data to MGHPCC

- ▶ We recommend using Cyberduck, a graphical SFTP client.
- ▶ Other options include scp and sftp for Mac/Unix command line users or PSCP/PSFTP for Windows users.
- ▶ Let's try to move the BiPSandbox folder to your /home/username folder on the MGHPCC.

The process we will use today

Our order of operations...

- ▶ Log in to the cluster
- ▶ Transfer data/scripts to cluster
- ▶ **Submit a job to the “scheduler”**
- ▶ Look at results

Software available on the MGHPCC

- ▶ Once logged in, you will be placed in your home directory (/home/username).
- ▶ Full list of available software:
http://wiki.umassrc.org/wiki/index.php/Provided_Software
- ▶ There is a wide array of available software options.

Using software on the MGHPCC

- ▶ In order to use software on the server, you'll need to load modules that contain the software.
- ▶ We are interested in using R here.
- ▶ To load R we use the command: `module load R/3.0.1`
- ▶ We can also unload R with the command: `module unload R`

Software

- ▶ Loading the R module will allow us to run R interactively (by typing R at the prompt) on the MGHPCC servers.
- ▶ But you should not do this!

Batch Mode

- ▶ Rather than running R code interactively, we can also run R in BATCH mode.
- ▶ This first requires you to write an R script that you want to run.
- ▶ Then at the command prompt, you can submit the script by typing: `R CMD BATCH [options] scriptName.R`
- ▶ But you should not do this either!
- ▶ We want to submit batch jobs to the cluster, not to the machine that you login to.

Submitting jobs to the cluster

- ▶ In order to submit a job to the cluster we need:
 - ▶ A R script that we wish to run
 - ▶ A shell script that calls the R script and submits the job to the cluster.
- ▶ We will then submit the job to the LSF (Load Sharing Facility) scheduler.
- ▶ **Note:** Commands submitted to LSF are just like if they were run from a command prompt.

LSF common commands

- ▶ `bsub` - submit a job
- ▶ `bkill` - kill a job
- ▶ `bjobs` - view status of jobs
- ▶ `bpeek` - view output / error files
- ▶ `bhist` - job history
- ▶ `bqueues` - available queues

Batch Mode

- ▶ Once we have a .R file that we want to run we can submit it to the LSF job scheduler.
- ▶ What actually gets submitted to the LSF scheduler is a shell script that calls the R file (which we'll call example.R).
- ▶ We need to create a file that contains all of the code we would have run at the shell prompt.
- ▶ Example of a shell script is below. We'll call this shell script example.sh

```
$ module load R/3.0.1
```

```
$ R CMD BATCH --vanilla example.R
```

Submitting jobs to the cluster

- ▶ The command **bsub** allows us to submit the shell script to the cluster.
- ▶ The command below will submit the shell script `example.sh` to the cluster.

```
$ bsub example.sh
```

Submitting jobs to the cluster: Options

- ▶ We can also set many options in our job submission.
- ▶ `-n`: Number of cores requested
- ▶ `-W`: Wall clock time
- ▶ `-R`: Memory per job
- ▶ `-q`: Which queue to submit to

```
$ bsub -n 4 -R "rusage[mem=2048]" -W 0:10 -q long  
example.sh
```

Submitting jobs to the cluster: Local

- ▶ An example shell script is in `BiPSandbox/module3/submitLocalParallelJob.sh`.
- ▶ Here we submit to the short queue.
- ▶ Commands to submit from within a `.sh` file is the same as when submitting from the command line:
`bsub < submitLocalParallelJob.sh`

Submitting jobs to the cluster: Distributed

- ▶ We can also specify options in the shell script that we submit to the LSF scheduler.
- ▶ An example shell script is in `BiPSandbox/module3/submitDistrParallelJob.sh`.
- ▶ Commands to submit is again:
`bsub < submitDistrParallelJob.sh`
- ▶ “Proper” way to distribute your job is using something like MPI.
- ▶ In this example, output is also distributed, less convenient.