Introduction to Using R on ARC's Resources
STAT 6984: Advanced Statistical Computing
Justin Krometis, Advanced Research Computing, 9 November 2017

1 Hardware

Cascades is a 196-node (computer) cluster built in 2016. Almost all of the nodes on Cascades have 32 cores and 128 GB memory. Instructions and examples for using Cascades are available at `http://www.arc.vt.edu/cascades`

Descriptions and instructions for other ARC clusters are available at: `http://www.arc.vt.edu/computing`

2 Login

1. Log into the Cascades login node with your PID and password:
   (a) Mac/Linux: Open a terminal and type
   ```
   ssh YOURPID@cascades2.arc.vt.edu
   ```
   (b) Windows: Download PuTTY and in the Host Name field type: `YOURPID@cascades2.arc.vt.edu`
2. ARC's clusters are subject to two-factor authentication. One option is to simply type your password into the password field and then be *very* quick in approving the two-factor request when it comes in. However, the following is typically a better approach:
   (a) Open the Duo app on your phone
   (b) Click the key icon on the Virginia Tech line. The app will produce a six-digit code.
   (c) Log into ARC clusters
   (d) When prompted for your password, type `password,XXXXXX` where `XXXXXX` is the six-digit code produced by the Duo app

3 Software Stack

Software on ARC clusters are managed with a module environment. There are three layers of modules:

1. Base: Require neither a compiler or MPI stack.
2. Compiler: Require that a compiler (e.g. `gcc` or `intel`) be loaded.
3. MPI: Require that an MPI stack (e.g. `mvapich2`, `openmpi`, or `impi`) be loaded.

Some key commands include:

To load R on Cascades, you might use:

```
module purge
module load intel
module load mkl R/3.4.1
```

Command	Example	Meaning
`module list`	`module list`	List the modules currently loaded
`module avail`	`module avail`	List modules that are available to be loaded
`module show`	`module show cuda`	Print information about a module
`module load`	`module load cuda`	Load a module
`module unload`	`module unload cuda`	Unload a module
`module spider`	`module spider cuda`	Search for a module
`module swap`	`module swap intel gcc`	Replace one module with another & reload dependencies
`module purge`	`module purge`	Unload all modules

4 Adding R Packages

See also ARC's documentation on this topic: http://www.arc.vt.edu/userguide/r/#packages

ARC's R installations come with a variety of packages installed. However, we cannot provide everything that every user might need. For example, `spam_mc.R` requires the `mda` and `randomForest` packages. To use packages that we do not offer centrally, you can install them in your Home directory and then tell R where to find them, as follows:

1. Create a directory where the libraries will be installed, e.g.:

   ```
   mkdir $HOME/R/lib
   ```

   Note that the library install is tied to the compiler and R version you are using. If you plan to switch between clusters, compilers, or R versions, you may want to use a more complicated directory structure, e.g.

   ```
   mkdir $HOME/cascades/R/3.4.1/intel/lib
   ```

2. Add the directory that you created to the `R_LIBS` environment variable so R knows to look there when you try to load the library:

   ```
   export R_LIBS="$HOME/R/lib:$R_LIBS"
   ```

3. Open R (make sure the module is loaded - see above):

   ```
   R
   ```

4. Tell R to install the `mda` package in the directory that you just created:

   ```
   install.packages("mda", lib="~/R/lib")
   ```

   Select a mirror (e.g., `USA (TN)`) and the package should install.

5. Load the library:

   ```
   library("mda")
   ```

Repeat the last two steps to install and load the `randomForest` package.

(Note that you can put the `module load` command and/or definition of `R_LIBS` in your `.bashrc` file so that you do not have to type them every time you log in or start a job.)

5 Getting Files

To do computations on ARC's resources, you need to migrate the files that you need to it:

- You can download files from the internet with `wget`, e.g.

  ```
  wget http://bobby.gramacy.com/teaching/asc/spam.csv
  ```

- You can clone git repositories, e.g.

  ```
  git clone https://username@bitbucket.org/username/\dots
  ```

- You can copy files from your computer to ARC clusters (or back), with `scp` or `rsync`, e.g.

  ```
  scp spam.csv username@cascades2.arc.vt.edu:
  ```

Some key files for this class have been placed in a central location and can be copied to your home directory as follows:

```
cp -r /home/TRAINING/stat6984 .
```

6 Submitting a Job

Before you can do computationally-intensive work on an ARC cluster, you need to move from the login node (a computer where lots of people may be doing basic tasks) to a compute node (a computer dedicated to computing). Cascades is a shared resource, so this is done by submitting a job to a schedule, a piece of software that tries to balance many users' needs. Note: This sometimes means that you will have to wait until the resources that you need are available (i.e. until other users are done)!

However, I have reserved resources for this hands on session so you should not have to wait today.

There are two main types of jobs: interactive and batch.

6.1 Interactive Job

You can use an interactive job to get a short(-ish) interactive session on a compute node. To do this, use the development queue (`dev_q`) as follows (here we request one node for one hour with the allocation `ascclass` created for this class):

```
interact -l nodes=1:ppn=32 -l walltime=1:00:00 -Aascclass
```

(Actually, the walltime and job size are the default, so simply `interact -Aascclass` would also suffice.)

Once the job starts, you will get a session on a new computer, e.g. `ca007`. Now you can do computationally-intensive work without interfering with other users. Try running `spam_mc.R`:

```
module purge; module load intel mkl R/3.4.1
R CMD BATCH spam_mc.R &
```

Try running `top` to see what the computer is doing while it's running (hit `q` to exit). Note that R should be using more than 100% CPU. This is because R on ARC's systems is threaded - Intel builds use Intel's Math Kernel Library (MKL) and gcc builds use OpenBLAS. This threading can be controlled by setting environment variables before running, e.g.

```
export MKL_NUM_THREADS=8
export OPENBLAS_NUM_THREADS=8
```

As noted above, however, each Cascades node has 32 cores. So we can also start multiple R processes at a time, as in Homework #4:

```
reps=2
nth=32                  #One process for each core
export MKL_NUM_THREADS=1 #No multithreading inside R
for i in `seq 1 $nth`; do
    echo "running nohup R CMD BATCH '--args seed=$i reps=$reps' spam_mc.R spam_mc_$i.Rout"
    R CMD BATCH "--args seed=$i reps=$reps" spam_mc.R spam_mc_$i.Rout &
done
```

(There is no reason to use `nohup` on ARC's systems. The processes will keep running until you exit the job; they will be killed if still running after the job completes.)

Another option is to use GNU Parallel, a software package designed to easily split up embarrassingly parallel tasks:

```
module load parallel
nth=100    #Number of total processes to run
nj=$PBS_NP #Number of proccesses to run at a time
seq 1 $nth | parallel -j $nj "R CMD BATCH \"--args seed={} reps=$reps\" spam_mc.R spam_mc_{}.Rout"
```

The {} syntax is what GNU Parallel uses to substitute in a value associated with the task. We pass GNU Parallel a sequence of numbers, it creates a task for each, and substitutes the number in for {}. The variable `$PBS_NP` holds the number of cores assigned to a job. So in the interactive job requested above, `$PBS_NP=32`.

GNU Parallel has a particularly excellent set of examples:
https://www.gnu.org/software/parallel/parallel_tutorial.html

Once you are done with the interactive job, you can simply type

```
exit
```

to finish the job and be returned to the login node.

6.2 Batch Job

Batch jobs are used to launch non-interactive sessions to run larger and longer computationally-intensive programs. These jobs are typically submitted via a *submission script*. The submission script has two main parts:

1. The header, which describes the job, e.g. how many resources it needs and for how long. These lines begin with `#PBS`.
2. The body, which describes what to do once those resources have been obtained.

Here is an example submission script for submitting a one-node job for five hours to the production queue (`normal_q`), and to run `spam_mc.R` in parallel with GNU Parallel on those resources:

```
#!/bin/bash

#PBS -l nodes=1:ppn=32
#PBS -l walltime=10:00:00
#PBS -q normal_q
#PBS -W group_list=cascades
#PBS -A ascclass

cd $PBS_O_WORKDIR

module purge
module load intel mkl R/3.4.1

export R_LIBS="$HOME/cascades/R/3.4.1/intel/lib:$R_LIBS"

module load parallel
```

```
nth=100    #Number of total processes to run
nj=$PBS_NP #Number of proccesses to run at a time
reps=5

export MKL_NUM_THREADS=1

echo "$(␣date␣):␣Starting␣spam_mc"
seq 1 $nth | parallel -j $nj "R␣CMD␣BATCH␣\"--args seed={} reps=$reps\"␣spam_mc.R␣spam_mc_{}.Rout"
echo "$(␣date␣):␣Finished␣spam_mc"

echo "$(␣date␣):␣Starting␣spam_mc_collect"
R CMD BATCH spam_mc_collect.R
echo "$(␣date␣):␣Finished␣spam_mc_collect"
```

Note that we also do postprocessing with `spam_mc_collect` as part of the job.

6.2.1 Submitting and Checking a Batch Job

This script can be submitted to the scheduler as follows:

```
qsub spam_mc.qsub
```

This will return your job name of the form

```
181556.master.cluster
```

Here `181556` is the job number. Once a job is submitted to a queue, it will wait until requested resources are available within that queue, and will then run if eligible. Eligibility to run is influenced by the resource policies in effect for the queue.

To check a job's status, use the `checkjob` command:

```
checkjob -v 181556
```

You can also view all of your jobs using the `showq` or `qstat` commands:

```
showq -u $USER
qstat -u $USER
```

To check resource usage on the nodes available to a running job, use:

```
jobload 181556
```

6.2.2 Results

When your job has finished running, any outputs to stdout or stderr will be placed in the files .o and .e. These two files will be in the directory that you submitted the job from. For example, for a job submitted from `spam_mc.qsub` and with job ID `181556`, the output would be in:

```
spam_mc.qsub.o181556 #Output will be here
spam_mc.qsub.e181556 #Any errors will be here
```

Of course, for `spam_mc` we would expect most, if not all, output to be in the separate `*.Rout` files.

7 References

- http://www.arc.vt.edu/r describes how to use R on ARC's systems
- http://www.arc.vt.edu/computing describes ARC's hardware, usage policies, and software, and provides some step-by-step examples

- http://www.arc.vt.edu/storage describes ARC's storage systems
- http://www.arc.vt.edu/software describes software installed on ARC's systems
- http://www.arc.vt.edu/modules describes how to use ARC's software modules
- http://www.arc.vt.edu/scheduler describes how to interact with the scheduler - submit and check jobs, view output, etc.
- http://www.arc.vt.edu/faq provides answers to some frequently asked questions