Using the CRUG Cluster

In this document I give a brief overview of using Carleton's Computational Research Users Group (CRUG) Cluster. You can read about the cluster here, so I don't intend to provide an in-depth discussion, rather this document serves as a crash course for statistics students doing research that could benefit from running R jobs on the CRUG server.

Uploading files to the cluster

To run jobs on the CRUG cluster, you will need to use an sftp or scp client to upload your code and data to `command.dmz.carleton.edu`. The Mike Tie recommends using Cyberduck if you are a Mac user and WinSCP if you are a Windows user; however, you can use any client you're comfortable with (e.g., FileZilla).

Once you have installed an sftp/scp client, you'll need to connect to `command.dmz.carleton.edu` and upload the necessary files. To do this, open a connection and use the following information:

- Server/host: `command.dmz.carleton.edu`
- Port: this will likely be autofilled by the client (e.g., use 22 for SFTP)
- Username: this your Carleton username
- Password: this is your Carleton password

Once you establish a connection, you can easily upload/download files.

Running a job on the cluster

Once you have written all of your code and have upload the necessary files to `command.dmz.carleton.edu`, you are ready to run a job.

One option is to use an RStudio interface to the `command.dmz.carleton.edu`. Mike Tie has a tutorial on this here.

I prefer running my script directly in the command line, as I find it easier, and I can use the Slurm workload manager to allocate and distribute my job without worrying about how to use `{rslurm}` or a similar R package. (Special thanks to Josh Davis for his tutorial on doing this! The below is adapted from his tutorial.)

Prepping your R script for use with Slurm The good news is that a simple addition to your R script is all that's needed to run your job via Slurm. Once you have finalized your script, then add the following at the top of your `.R` file:

```
#!/usr/bin/env /usr/local/bin/Rscript
```

If your job requires a specific version of R (e.g., the packages don't work on all versions of R), then you may need to edit this. For example, as of this writing (July 2021) I need to add this line

```
#!/usr/bin/env /opt/R/4.0.4/bin/Rscript
```

to run scripts that load `{lme4}` and `{foreach}`.

Once you've made this addition to your R script and uploaded this file (and all the other scripts it sources or files it loads), you're ready to run your job.

Typical workflow Below is the workflow I suggest for submitting a typical job via Slurm. There are special cases, but if you hit one you should talk to Mike Tie for help.

1. SSH into `command.dmz.carleton.edu` by running the following in your terminal/shell

   ```
   ssh command.dmz.carleton.edu
   ```

You'll be prompted for your password. Note that you can open a shell in RStudio by going to `Tools -> Shell`. If you're off campus, then you can use the VPN to get behind the firewall. Once you're connected, you should see something like this:

2. To run your job in Slurm, use an `srun` command, such as `srun -n1 -p StudentSlurm -o myOutput.txt ./myScript.R &`
    - The number following the `-n` is the number of processors to use, so if you wanted to run your job on 12 cores, then use `-n12`.
    - `StudentSlurm` is the name of the partition on the cluster that you'll be using. I don't think you will need to change this, but it knowing that seems useful.
    - `myOutput.txt` is the name of the output file that will be created. Use an informative name here. This file can help pinpoint issues later, but you may never need it.
    - `myScript.R` is the name of the R script that you want to run. Be sure to run the right script!
    - Ending the command with `&` allows the job to run in the background so that you can logoff of `command.dmz.carleton.edu` without interrupting your job.

Useful Slurm commands The above workflow is the minimal set of commands that you need, but there are other commands worth knowing. This is a brief overview of the commands that I have used on the command server.

- To learn about the cluster run `sinfo` in the shell. This will show the partitions and nodes, and what resources are allocated/free.
- To see the queue of jobs, run `squeue`.
- If you need to cancel a job early (e.g., you realize you're running an infinite loop or you caught a bug) then you can use `scancel myJobID`. To find what you should enter for `myJobID`, run `squeue`. For example, you might run `squeue` and see the following:

```
JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
3051  SevenDay     one.runS    auser  PD       0:00      2      (Resources)
2886  TwoDay       script.w    buser   R 1-18:42:30      1         crug4
2887  FourHour*    aprogr.R    buser   R   03:05:42      1         crug3
2888  SevenDay   longjob.py     jdoe   R 5-18:42:23      1      compute1
2889  StudentSlurm myScript.R    aloy   R       4:00      1         crug2
```

  If your username were aloy, then you would run `scancel 2889` to cancel the job.

Useful unix commands This is a brief overview of commands that might also be useful to you if you are unfamiliar with the unix shell.

- `exit` or `logout` to log out of the command server, use either of these commands.
- `ls` will list all of the files/folders in your current directory. This can be helpful if you forget the name of your script. (Or forgot that you capitalized a single letter!)
- `cd dirname` will set the working directory to what you specify for `dirname`.