Computing using Roar Collab

Stephen Berg

Department of Statistics, Pennsylvania State University
sqb6128@psu.edu
Overview

Goals for talk

- introduce high performance computing resources available at Penn State
- show how to submit computing jobs to Roar Collab (ICDS computing cluster)
- show how to submit jobs to the Statistics Department computing allocation
Computing resources

- Institute for Computational and Data Sciences
  - manages high-performance computing resources at Penn State

- Roar
  - older ICDS computing cluster
  - prior to spring 2023
  - most access ending October 1, 2023

- Roar Collab
  - current ICDS computing cluster
  - Spring 2023 onward
  - These slides are for Roar Collab.
Roar Collab account

- contact ICDS: https://www.icds.psu.edu/roar-collab-user-guide/
- request a Roar Collab account
- in addition, you should contact me (sqb6128@psu.edu) to be added to the department computing allocation (allocation name muh10)
Accessing Roar Collab

Once you have an account, there are two ways to access Roar Collab (that I know of):

▶ graphical user interface (GUI) web portal: https://rcportal.hpc.psu.edu
  ▶ the Roar Collab link is different from the previous Roar web portal link, which was https://portal2.aci.ics.psu.edu
  ▶ use the https://rcportal.hpc.psu.edu link that goes to Roar Collab

▶ ssh, via a terminal/command line interface
  ▶ Connect to submit.hpc.psu.edu: in Mac/Linux terminal, type
    ssh sqb6128@submit.hpc.psu.edu
    (change the username to your PSU username)
  ▶ other aspects of command line interface standard: cd, rm, ls
Computing on Roar Collab

Web interface via https://rcportal.hpc.psu.edu:

Notice! When using the RC shell and you have Microsoft MFA enabled you will get a silent push to your two factor device. If you enter your password and nothing happens please check your device.
Computing on Roar Collab, from web portal

- appears on first page of web portal
- like running RStudio on your own computer
  - open, edit, save, and source scripts like on your regular computer
  - but with more available CPU’s
- some packages are pre-installed
- new packages go to a local library
- type `.libPaths()` in the RStudio Server R console to see the directories where R is looking for packages
Resource queues

ICDS maintains open computing resources for general users, as well as paid computing allocations that give certain jobs priority scheduling

▶ the general computing queue name is open
▶ the Statistics Department allocation account name is muh10
▶ to use the Statistics Department allocation for an RStudio Server session, set account to muh10 and partition to SLA Prio in the RStudio Server job setup page dropdown menus
▶ otherwise, you can use the open queue (account open and partition open)
Scheduling and batch jobs

- Job scheduling on Roar Collab is handled via a job-scheduling program called Slurm.
- Slurm is a very common scheduler and lots of info available online on customizing job descriptions.
- To submit a batch job via the command line interface, make a .sh file
  - (many examples online)
  - Submit using `sbatch fileName.sh`
  - Check status of your jobs using, eg, `squeue -u sqb6128` (you can also use rc web portal GUI).
- Jobs that request fewer resources (cpu’s, memory, time) will generally be scheduled earlier.
- Jobs that exceed the requested resources (eg memory, time) will be terminated.
- The stat allocation has 200 cpu’s.
Scheduling and batch jobs

Contents of example submit file, submitTest.sh, for running an array of 400 jobs, with a maximum of 10 jobs running at a time

```bash
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --cpus-per-task=1
#SBATCH --ntasks=1
#SBATCH --mem=1GB
#SBATCH --time=1:00:00
#SBATCH --account=muh10
#SBATCH --partition=sla-prio

#SBATCH --array=1-400%10

module load r/4.2.2
R CMD BATCH test.R test_${SLURM_ARRAY_TASK_ID}
```
Scheduling and batch jobs

Contents of an example R file, test.R:

- gets the job id (eg, for set.seed(job_id))
- saves output to a numbered RData file in the output/ directory

R code in test.R:

```r
slurm_array_id <- Sys.getenv('SLURM_ARRAY_TASK_ID')
n <- as.numeric(slurm_array_id)
df=data.frame(id=n)
save(df, file=paste0("output/test_",n,".RData"))
```
Some basics

- test your job on a tiny version of your problem first, to detect bugs
  - make sure output is saved properly
  - don’t wait 24 hours to find simple mistakes that make output useless or cause programs to crash

- you can check what is being run on the stat allocation via
  squeue --account muh10
  on the terminal

- you can check on your own jobs using
  squeue -u sqb6128
  or look at the Jobs dropdown on the rc web portal