UA HPC Cheat Sheet for Stats Students

Zhaoying Lu
`zhaoyinglu@arizona.edu`
Zihan Zhu
`zihanzhu@arizona.edu`

1 Available Resources

UArizona HPC systems has three supercomputers, Puma, Ocelote, and ElGato. Puma is the latest supercomputer and a detailed user guide for Puma can be found on UA HPC website. For a beginner, we recommend taking the self-guided training course, Intro to HPC Workshop, to get started on using HPC resources. Check the slides and recording for HPC Workshops if you cannot attend.

2 Create Account

Sponsored HPC Account Instructions for creating your HPC account in two steps can be found in Account Creation section in User Guide.

All graduate students in Statistics & Data Science have been added to Professor Hao Zhang's group, but you need to create your own HPC Account first. Browse to `https://portal.hpc.arizona.edu/` and log in using your NetID+. This will automatically create an HPC account for you.

If you want to request sponsorship from a UArizona faculty member, you may visit `https://portal.hpc.arizona.edu/portal/sendlink.php`. On the right-hand side, enter your sponsor's email address and click send. Your sponsor will then receive an email to authorize your account. Once your request has been authorized, you will receive an email with instructions for accessing the HPC systems.

3 Logging In

We can log in HPC systems using our own terminal/SSH client or through Open OnDemand web interface.

- Terminal/SSH client
  - Mac
    - Open a terminal: Applications → Utilities → Terminal
    - On the command line, enter the following, replacing netid with your own NetID:
      ```
      $ ssh your_netid@hpc.arizona.edu
      ```
    - Enter your password and Duo-authenticating.
    - Type shell to connect to the login nodes. The default cluster for job submission is Puma. You can also type ocelote to connect to Ocelote.
  - Windows users need an ssh client such as PuTTY and MobaXterm. The instructions for downloading and using the softwares are listed in Windows Access. Then we open PuTTY, enter hpc.arizona.edu under Host Name and press Open.
- Open OnDemand. Log into `https://ood.hpc.arizona.edu/`, then select the Clusters dropdown tab and click Shell Access. The default cluster for job submission is Puma, but you can type ocelote to connect to Ocelote.

4 Transferring Data

The methods of transferring large files to/from HPC can be found in Transferring Data section in User Guide. Small files can be uploaded through Open OnDemand: Files → Home Directory → Upload.

5 Running Jobs

- Job Partition Requests. Statistics & Data Science department purchased nodes for graduate students. To view the remaining allocation of your group, use the command

  ```
  (puma) [netid@junonia ~]$ va
  ```

  in a terminal. It's recommended to use high_priority allocation first, which has 17,520 CPU Hours per month, by using command

  ```
  #SBATCH --account=haozhang
  #SBATCH --partition=high_priority
  ```

  When high-priority is depleted, use standard queue with 100,000 CPU Hours per month via command

  ```
  #SBATCH --account=haozhang
  #SBATCH --partition=standard
  ```

  If standard allocation is exhausted, we can use windfall queue with unlimited CPU hours by command

  ```
  #SBATCH --partition=windfall
  ```

  But jobs in this queue can be stopped and restarted by standard jobs.

- Batch Job. Detailed instructions with sample script are provided in section 6.

- Interactive Jobs. To access R:

  ```
  (puma) [netid@junonia ~]$ interactive
  (puma) [netid@r2u03n1 ~]$ module load R
  (puma) [netid@r2u03n1 ~]$ R
  ```

  If no options are supplied to the command interactive, your job will automatically run using the windfall partition for one hour using one CPU. To use the standard partition, include the flag "-a" followed by your group's name, e.g., interactive -a haozhang. See all the customization options in User Guide.

6 Example

6.1 MLE of Unif$(0, \theta)$

As we learned in STAT 566, MLE of Unif$(0, \theta)$ is $X_{(n)}$. Now we want to calculate its MLE via simulation (assume true theta is 2). To use HPC, we should do following steps:

- Prepare slurm script, i.e. run.slurm

```
#!/bin/bash

#SBATCH --job-name=Unif
#SBATCH --output=%x-%A_%a.out
#SBATCH --error=%x-%A_%a.err
#SBATCH --account=group_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=email@address.xyz
#SBATCH --partition=standard
#SBATCH --ntasks=1
#SBATCH --mem=2gb
#SBATCH --time=00:05:00

module load R

cd /home/u18/your_account/Unif/

Rscript MLE.R 1000000 2
```

- Prepare R script, i.e. MLE.R

```
rm(list=ls())
args <- commandArgs(trailingOnly = T)
n <- as.integer(args[1])
true.theta <- as.integer(args[2])

###########################################

X <- runif(n,0,true.theta)
theta.MLE <- max(X)

###########################################

Results = theta.MLE
save(Results, file="bias.Rdata")
```

- Upload the two scripts to path /home/u18/your_account/Unif/. The results will also be stored here.
- Login to HPC and submit the slurm script.

```
#sbatch run.slurm
```

Here are some explanations:

- The first three lines in slurm script defines the job name and name format of output files and error files. %x is job name and %A is job ID, while %a is array index.
- 1000000 and 2 in slurm script are values propagated into R script. We assign 1000000 and 2 as a list to variable args. And let n be 1000000, true theta be 2. You can use similar ways to specify hyper-parameters of your R script.

6.2 MSE of MLE

Although we know $X_{(n)}/\theta$ follows some beta distribution, here we try bootstrap method to get the MSE of MLE (Assume we find MLE is 2.04 in previous subsection). Set the number of bootstraps as $B = 100$. We can try parallel computing.

- Prepare slurm script, i.e. run.slurm

```
#!/bin/bash

#SBATCH --job-name=Unif
#SBATCH --output=%x-%A_%a.out
#SBATCH --error=%x-%A_%a.err
#SBATCH --account=group_name
#SBATCH --mail-type=ALL
#SBATCH --mail-user=email@address.xyz
#SBATCH --partition=standard
#SBATCH --ntasks=1
#SBATCH --mem=2gb
#SBATCH --time=00:05:00
#SBATCH --array=1-100

module load R

cd /home/u18/your_account/unifboot/

Rscript MLECI.R 1000000 2 2.04 ${SLURM_ARRAY_TASK_ID}
```

- Prepare R script, i.e. MLECI.R

```
rm(list=ls())
args <- commandArgs(trailingOnly = T)
n <- as.integer(args[1])
true.theta = as.integer(args[2])
theta.MLE = as.integer(args[3])
index_rep <- as.integer(args[4])

###########################################

set.seeds(index_rep)
X.boot <- runif(n,0,theta.MLE)
MLE.MSE <- mean((X.boot-true.theta)^2)

###########################################

Results = MLE.MSE
save(Results, file=paste("MSE_",index_rep,".Rdata",sep = ""))
```

Here are some explanations:

- Array=1-100 means we want to run 100 array task at the same time. $\$SLURM\_ARRAY\_TASK\_ID$ is a system variable indicates the current array task ID. It takes integer value from 1 to 100. We can use it to name our array task output, i.e. paste("MSE_",index_rep,".Rdata",sep = "") or use it to set random seed which guarantees our stochastic algorithm is reproducible.

7 Questions

Frequently Asked Questions might help you get answers. Can't find instructions that you need? Request a consultation with HPC experienced staff!