Parallel Computing with High Performance Computing (HPC)

Parallel in HPC

- Using Niagara cluster (Compute Canada) as an example, it contains 2024 nodes, each with 40 cores, for a total of 80,640 cores.
- Say if you want to request 20 cores, there are two ways to request it
  - 1 node and all 20 cores on the node
  - different nodes

One node Prallel

```
cl <- makeCluster(20)
registerDoParallel(cl)
clusterExport(cl, varlist = c("n", "mu", "sigma", "trimmean"))
t <- parLapply(cl = cl, X = 1:S, function(s) {
  # generate data
  dat <- rnorm(n, mu, sigma)
  # calculate T1
  t1 <- mean(dat)
  # calculate T2
  t2 <- trimmean(dat)
  # calculate T3
  t3 <- median(dat)
  c(t1, t2, t3)
})

# convert t to a data frame with column t1, t2, t3
t_final <- do.call(rbind, t)

# save the results
saveRDS(t_final, "t_final.rds")
```

save the R script as `example.R`

One node Prallel

Use `module spider r` to check the requirement for loading R

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=20
#SBATCH --time=0-01:30            # time (DD-HH:MM)
module load gcc/9.3.0 r/4.0.2

Rscript example.R
```

Save it as `submit.sh`

Submit the job by `sbatch submit.sh`

Multiple Nodes

- things are much more complicated
- sometimes cannot be avoided, say if you want to request 800 cores
- need to use `OpenMPI`

Multiple Nodes

```
cl <- makeCluster(800, type = "MPI")
registerDoParallel(cl)
t <- parLapply(cl = cl, X = 1:S, function(s) {
  # generate data
  dat <- rnorm(n, mu, sigma)
  # calculate T1
  t1 <- mean(dat)
  # calculate T2
  t2 <- trimmean(dat)
  # calculate T3
  t3 <- median(dat)
  c(t1, t2, t3)
})

# convert t to a data frame with column t1, t2, t3
t_final <- do.call(rbind, t)

# save the results
saveRDS(t_final, "t_final.rds")
```

save the R script as `example.R`

Multiple Nodes

```
#!/bin/bash
#SBATCH --nodes=20
#SBATCH --ntasks-per-node=40
#SBATCH --time=0-01:30            # time (DD-HH:MM)
module load gcc/9.3.0 openmpi/4.0.3 r/4.0.2

R_PROFILE=${HOME}/R/x86_64-pc-linux-gnu-library/4.0/snow/
RMPISNOWprofile;
export R_PROFILE
mpirun -np 800 -bind-to core:overload-allowed R CMD BATCH
--no-save example.R
```

Save it as `submit.sh`

Submit the job by `sbatch submit.sh`

Passing argument

- sometimes you may want to run for a set of arguments
- e.g. n = c(100, 200, 300, 400)

```
args <- commandArgs(TRUE)
n <- args[1]

cl <- makeCluster(800, type = "MPI")
registerDoParallel(cl)
clusterExport(cl, varlist = c("n", "mu", "sigma", "trimmean"))

t <- parLapply(cl = cl, X = 1:S, function(s) {
  # generate data
  dat <- rnorm(n, mu, sigma)
  # calculate T1
  t1 <- mean(dat)
  # calculate T2
  t2 <- trimmean(dat)
  # calculate T3
  t3 <- median(dat)
  c(t1, t2, t3)
})

# convert t to a data frame with column t1, t2, t3
t_final <- do.call(rbind, t)
# save the results
saveRDS(t_final, "t_final.rds")
```

Passing argument

```
#!/bin/bash
#SBATCH --nodes=20
#SBATCH --ntasks-per-node=40
#SBATCH --time=0-01:30            # time (DD-HH:MM)
module load gcc/9.3.0 openmpi/4.0.3 r/4.0.2

R_PROFILE=${HOME}/R/x86_64-pc-linux-gnu-library/4.0/snow/RMPISNOWprofile; export R_PROFILE
mpirun -np 800 -bind-to core:overload-allowed R CMD BATCH --no-save "--args $n" example.R
```

Save it as `submit.sh`

Submit the job by

```
for n in 100 200 300 400
do
  sbatch --export=n=$n submit.sh
done
```

Resources

- "A Simple SLURM Guide for Beginners" by Parice Brandies [link]
- "Submitting Parallel Blogs" by UC Berkley [link]