PBS/PBSPro and R

PBS/ PBS Pro and R

R is a free software environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

Submitting an R job via PBS

R jobs can be submitted to any HPC queue using the same basic style of PBS command files that are used by other jobs.

Below is an example of a PBS command file used to run the program a basic R program on the HPC, that only uses one cpu.

```
#!/bin/bash -l
#### job name & output files
#PBS -N R_Test
#
#### select resources
#PBS -l nodes=1:ppn=1
#PBS -l walltime=0:10:00
#PBS -l mem=1g
#
#### redirect output
#PBS -j oe
#
#### mail options
#PBS -m ae
#PBS -M ruser@usq.edu.au
#
#### queue name
#PBS -q default

#### change directory to working directory
cd $PBS_O_WORKDIR

#### load module
module load r/3.1.2-gnu

#### run compute
R --vanilla --quiet < PlotCos.r
```

Below is the R script used above.

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

```
# PlotCos.r    R program to plot cosine values
seq(-1000, 1000, by=.1) -> x
plot (x, cos(x), type="l")
```

Below is an example of a PBS command file used to run the program a parallel R program on the HPC.

```
#!/bin/bash -l
#### job name & output files
#PBS -N Rmpi_Test
#
#### select resources
#PBS -l nodes=2:ppn=2
#PBS -l walltime=0:10:00
#PBS -l mem=1g
#
#### redirect output
#PBS -j oe
#
#### mail options
#PBS -m ae
#PBS -M ruser@usq.edu.au
#
#### queue name
#PBS -q default

#### change directory to working directory
cd $PBS_O_WORKDIR

#### load module
module load r/3.1.2-gnu
module load openmpi/1.8.8-gnu

#### run compute with no output from R
mpirun --quiet -hostfile $PBS_NODEFILE -np 1 R --vanilla --quiet CMD BATCH
gibbs-bivar-task-par.r

#### run same compute with limited output from R
####mpirun --quiet -hostfile $PBS_NODEFILE -np 1 R --vanilla --quiet <
gibbs-bivar-task-par.r
```

Below is the R script used above.

```
# gibbs-bivar-task-par.r    Simple 10-fold cross-validation
mpi.master <- function(y, rho, Niter) {

  # Load the R MPI package if it is not already loaded.
  if (!is.loaded("mpi_initialize")) {
   library("Rmpi")
  }
  # 4 slaves will start based on "nprocs" option to qsub command
  mpi.spawn.Rslaves()

  mpi.bcast.Robj2slave(mpi.slave)

  mpi.bcast.Robj2slave(y)
  mpi.bcast.Robj2slave(rho)
  mpi.bcast.Robj2slave(Niter)

  theta.startmat <- matrix(c(5,5, -5,5, 5,-5, -5,-5),4,2,byrow=T)
  mpi.bcast.Robj2slave(theta.startmat)

  mpi.remote.exec(mpi.slave())

}

mpi.slave <- function() {

  # Each slave gets its own copy of ind and chain based on mpi process rank
  ind <- mpi.comm.rank()

  thetamat <- matrix(0, Niter,2)
  thetamat[1,] <- theta.startmat[ind,]

  for(i in 2: Niter){
    thetamat[i, 2] <- rnorm(1, y[2] + rho *(thetamat[i-1, 1] - y[1]),
sqrt(1 - rho^2))
    thetamat[i, 1] <- rnorm(1, y[1] + rho *(thetamat[i, 2] - y[2]), sqrt(1
- rho^2))
  }

  write.table(thetamat, file=paste("output.",ind, sep=""),
              quote=F, sep="\t", row.names=F, col.names=F)

}

y <- c(0,0)
rho <- 0.8
Niter <- 1000000

start <- Sys.time()
mpi.master(y, rho, Niter)
Sys.time() - start

# Tell all slaves to close down, and exit the program
mpi.close.Rslaves(dellog=FALSE)
mpi.quit()
```

References:

1. Acadia University

2. R Project
3. R User Group - Brisbane