Parallel Computing in R

ResBaz Dunedin Feb 8th 2017

Author: Tom Kelly, PhD Candidate, University of Otago

This lesson assumes you are familiar with repeating operations in loops e.g.:

```
for(ii in 1:10){
  print(ii^2)
}
```

Clearly for more iterations or more complex functions this could soon take a considerable amount of time:

```
max_n <- 100
vector <- rep(NA, max_n)
```

```
system.time({
  for(ii in 1:max_n){
    vector[ii] <- ii^2
  }
})
```

```
for(max_n in c(100, 1000, 10000, 100000)){
  print(max_n)
  print(system.time({
    for(ii in 1:max_n){
      vector[ii] <- ii^2
    }
  }))
}
```

Apply functions

```
max_n <- 100
vector <- lapply(1:max_n, function(x) x^2) #create list
vector <- sapply(1:max_n, function(x) x^2) #create vector
vector
```

```
system.time({
  for(ii in 1:max_n){
    vector <- lapply(1:max_n, function(x) x^2) #create list
  }
})
```

```
system.time({
  for(ii in 1:max_n){
    vector <- sapply(1:max_n, function(x) x^2) #create vector
  }
})
```

```
for(max_n in c(100, 1000, 10000, 100000)){
  print(max_n)
  print(system.time({
```

```
    vector <- sapply(1:max_n, function(x) x^2) #create vector
  }))
}
```

Vectorisation

```
max_n <- 100
vector <- 1:max_n
vector <- vector^2
vector
```

```
for(max_n in c(100, 1000, 10000, 100000)){
  print(max_n)
  print(system.time({
      vector <- 1:max_n
      vector <- vector^2
  }))
}
```

Parallelisation

If iterations independent:

```
library("snow") #simple network of workstations
cl <- makeSOCKcluster(2) # number of cores
#makeMPIcluster(2)
#makeCluster(2)
clusterExport(cl, list=ls())
```

```
max_n <- 100
vector <- parLapply(cl, 1:max_n, function(x) x^2) #create list
vector <- unlist(vector)
vector
```

```
system.time({
  vector <- parLapply(cl, 1:max_n, function(x) x^2) #create list
  })
stopCluster(cl)
```

```
cl <- makeSOCKcluster(2) # number of cores
clusterExport(cl, list=ls())
for(max_n in c(100, 1000, 10000, 100000)){
  print(max_n)
  print(system.time({
    vector <- parLapply(cl, 1:max_n, function(x) x^2) #create list
  }))
}
stopCluster(cl)
```

```
cl <- makeSOCKcluster(3) # number of cores
clusterExport(cl, list=ls())
for(max_n in c(100, 1000, 10000, 100000)){
```

```
  print(max_n)
  print(system.time({
    vector <- parLapply(cl, 1:max_n, function(x) x^2) #create list
  }))
}
stopCluster(cl)
```

Note the increase in speed is near linear

However there is an "overhead time" to set up cluster (which only worth it for larger jobs):

```
system.time({
  cl <- makeSOCKcluster(2) # number of cores
  #makeMPIcluster(2)
  #makeCluster(2)
  clusterExport(cl, list=ls())
  stopCluster(cl)
})
```

```
for(max_n in c(100, 1000, 10000, 100000)){
  print(max_n)
  print(system.time({
    cl <- makeSOCKcluster(2) # number of cores
    clusterExport(cl, list=ls())
    vector <- parLapply(cl, 1:max_n, function(x) x^2) #create list
    stopCluster(cl)
  }))
}
```

```
for(max_n in c(100, 1000, 10000, 100000)){
  print(max_n)
  print(system.time({
    cl <- makeSOCKcluster(3) # number of cores
    clusterExport(cl, list=ls())
    vector <- parLapply(cl, 1:max_n, function(x) x^2) #create list
    stopCluster(cl)
    }))
}
```

Thus there are diminishing returns in practice (when increasing the number of cores) due to more communication between them.

Question Time

```
max_n <- 100

#for loop
for(ii in 1:max_n){
  print(ii^2)
}

#lapply
vector <- lapply(1:max_n, function(x) x^2) #create list
```

```r
#parLapply on 2 cores
cl <- makeSOCKcluster(2) # number of cores
clusterExport(cl, list=ls())
vector <- parLapply(cl, 1:max_n, function(x) x^2) #create list
stopCluster(cl)
```

```r
#setup data
dataset <- matrix(rnorm(600), 60, 10)
head(dataset)
```

```r
#for loop:
mg_data <- matrix(NA, nrow(dataset)/3, ncol(dataset))
for(ii in 1:(nrow(mg_data))){
  mg_data[ii,] <- svd(dataset[(ii-1)*3+1:3,])$v[,1]
}

#make an lapply loop
mg_data <- matrix(NA, nrow(dataset)/3, ncol(dataset))
mg_data <-lapply(1:(nrow(mg_data)), function(ii){
  ##fill in function
})
mg_data <- t(as.data.frame(mg_data))

#make it parallel
cl <- makeSOCKcluster(2) # number of cores
clusterExport(cl, list=ls())
mg_data <- matrix(NA, nrow(dataset)/3, ncol(dataset))
mg_data <-parLapply(#here, #here, function(ii){
  #here
  })
mg_data <- t(as.data.frame(mg_data))
stopCluster(cl)
```

Answer Time

```r
#make an lapply loop
mg_data <- matrix(NA, nrow(dataset)/3, ncol(dataset))
mg_data <-lapply(1:(nrow(mg_data)), function(ii){
  svd(dataset[(ii-1)*3+1:3,])$v[,1]
})
mg_data <- t(as.data.frame(mg_data))

#make it parallel
cl <- makeSOCKcluster(2) # number of cores
clusterExport(cl, list=ls())
mg_data <- matrix(NA, nrow(dataset)/3, ncol(dataset))
mg_data <-parLapply(cl, 1:(nrow(mg_data)), function(ii){
  svd(dataset[(ii-1)*3+1:3,])$v[,1]
})
mg_data <- t(as.data.frame(mg_data))
stopCluster(cl)
```

Running scripts

```
source("loop_script.R")
```

```
system("Rscript loop_script.R") #run bash command
library("data.table")
mg_data <- fread("mg_data.csv", data.table = F)
head(mg_data)
dir()
```

Passing arguments to Rscripts

```
system("Rscript loop_script_Rstudio_generic.R 3 100")
library("data.table")
mg_data <- fread("mg_data.csv", data.table = F)
head(mg_data)
dim(mg_data)
dir()
```

```
for(ii in c(10, 20, 30)){
  system(paste("Rscript loop_script_Rstudio_generic.R 3", ii))
}
dir()
```

Passing arguments to bash

```
bash loop_script_generic.sh 3 80
ls
```

```
bash loop_script_generic.sh 3 {2..5} 10 15
ls
```

Running scripts in the background

```
nohup Rscript loop_script_Rstudio_generic.R 3 100 &
nohup bash loop_script_generic.sh 3 2..5 10 15 &
```

Running R on the cluster

```
sbatch loop_script_generic.sl 3 105
sbatch loop_script_generic_array.sl 3 35 45 55
```

Remote access

```
rsh pan #remote shell
rcp loop_script_generic_array.sl pan:/projects/uoo00010/test_dir #remote copy
```

"secure shell"

```
ssh pan
scp loop_script_generic_array.sl pan:/projects/uoo00010/test_dir
```

sync (move changes)

```
rsync -u loop_script_generic_array.sl pan:/projects/uoo00010/test_dir
```

ssh alias

```
cat ~/.ssh/config
ssh simon.kelly@login.uoa.nesi.org.nz
exit
ssh -XY pan
exit
```

```
ssh biochembioinfo.otago.ac.nz
exit
ssh bio
exit
```

Supplementary Resources

NeSI Set Up (Wiki) for Otago Users

https://github.com/dannybaillie/NeSI

Install R packages on NeSI

https://github.com/TomKellyGenetics/install.nesi

Notes for this guide

https://github.com/TomKellyGenetics/R-Parallel-Lesson