High performance computing in R using doSNOW package

Five minutes could make your simulation Five times faster or more

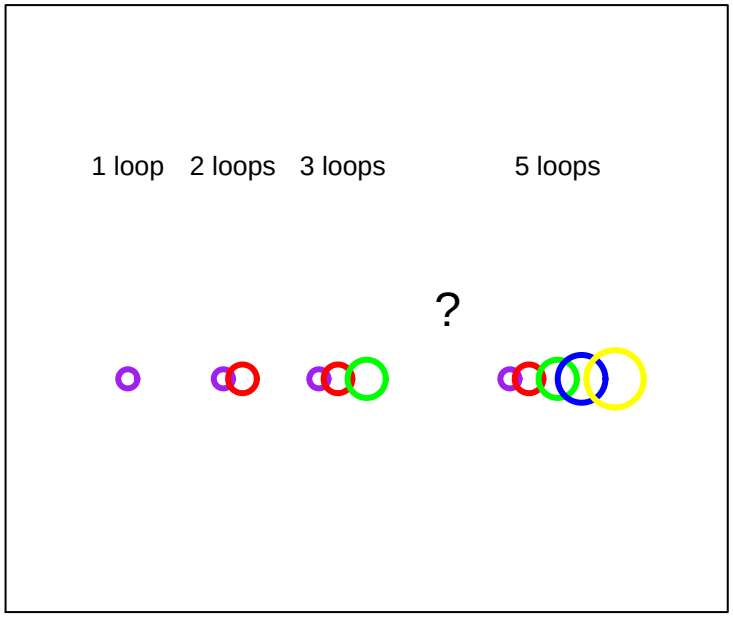
Minchun Zhou

October 10, 2014

Contents

1 Opening

What's a single criterion for a good R program?



No "For" loops!

2 Parallel programming in R

- What's parallel computing?

  1. Splitting the problem into pieces
  2. Executing the pieces in parallel
  3. Combining the results back together.

- Difference in node, processor and core

  1 Node = Multiple processors

  1 Processor = Multiple cores

  In my computer:

  Number of Processors: 1
  Total Number of Cores: 4

- Parallel programming in R :

  Packages: Snow, multicore, foreach, parallel, doMC, doSNOW

  Multicore and doMC work only on a single node (and not on Windows).doMC and doSNOW are based on foreach package. doSNOW can be used in all platforms.

3 doSNOW package

3.1 Parallel programming with Foreach

- Number of cores in your computer

```
library(parallel)
detectCores()

## [1] 8

# This number devided by 2 is the number of cores in your computer
```

- Register clusters

```
library(doSNOW)

## Loading required package:  foreach
## Loading required package:  iterators
## Loading required package:  snow
##
## Attaching package:  'snow'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, clusterSplit, makeCluster,
##    parApply, parCapply, parLapply, parRapply, parSapply,
##    splitIndices, stopCluster

NumberOfCluster <- 4
# how many jobs you want the computer to run at the same time

cl <- makeCluster(NumberOfCluster)  # Make clusters
registerDoSNOW(cl)  # use the above cluster
# your parallel programming code code code
stopCluster(cl)  # close clusters
```

- Review: For loop syntax

  for (i in vector){
  code
  return(object)
  }

- Foreach loop syntax

  foreach (i=vector, .combine='fun') %dopar% {
  code
  return(object)
  }

  By default, the results are returned in a list.

  .combine = 'fun'
  fun = c : vector
  fun = cbind: column bind
  fun = rbind: row bind
  fun = + : sum data
  fun = * : multiple data
  fun can be any function that you define.

```
x <- matrix(NA, ncol = 5, nrow = 5)
for (i in 1:5) {
    x[i, ] <- (1:5)^i
}
x
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    1    4    9   16   25
## [3,]    1    8   27   64  125
## [4,]    1   16   81  256  625
## [5,]    1   32  243 1024 3125
```

```
library(doSNOW)
NumberOfCluster <- 4
cl <- makeCluster(NumberOfCluster)
registerDoSNOW(cl)
x0 <- foreach(i = 1:5) %dopar% {
    y <- (1:5)^i
    return(y)
}
class(x0)
```

```
## [1] "list"
```

```
x <- foreach(i = 1:5, .combine = "rbind") %dopar% {
    y <- (1:5)^i
    return(y)
}
stopCluster(cl)
x
```

```
##          [,1] [,2] [,3] [,4] [,5]
## result.1    1    2    3    4    5
## result.2    1    4    9   16   25
## result.3    1    8   27   64  125
## result.4    1   16   81  256  625
## result.5    1   32  243 1024 3125
```

```
# which is equavilant to use .combine='fun' using the following function
fun <- function(...) {
    rbind(...)
}

cl <- makeCluster(NumberOfCluster)
registerDoSNOW(cl)
x <- foreach(i = 1:5, .combine = "fun") %dopar% {
    y <- (1:5)^i
    return(y)
}
stopCluster(cl)
x
```

```
##          [,1] [,2] [,3] [,4] [,5]
## result.1    1    2    3    4    5
## result.2    1    4    9   16   25
## result.3    1    8   27   64  125
## result.4    1   16   81  256  625
## result.5    1   32  243 1024 3125
```

```
fun <- function(...) {
    rbind(...) + 1
}

cl <- makeCluster(NumberOfCluster)
registerDoSNOW(cl)
x1 <- foreach(i = 1:5, .combine = "fun") %dopar% {
    y <- (1:5)^i
    return(y)
}
stopCluster(cl)
```

```
x1

##          [,1] [,2] [,3] [,4] [,5]
## result.1    5    6    7    8    9
## result.2    5    8   13   20   29
## result.3    4   11   30   67  128
## result.4    3   18   83  258  627
## result.5    2   33  244 1025 3126

x

##          [,1] [,2] [,3] [,4] [,5]
## result.1    1    2    3    4    5
## result.2    1    4    9   16   25
## result.3    1    8   27   64  125
## result.4    1   16   81  256  625
## result.5    1   32  243 1024 3125

# be careful to use your own function, you need to take care of all results
```

- When use foreach function, you need to put library(package) within the loop. It's like you open multiple new R windows. Or you can use .packages option.

```
library(rms)
NumberOfCluster <- 4
cl <- makeCluster(NumberOfCluster)
registerDoSNOW(cl)
x0 <- foreach(i = 1:4, .combine = "rbind") %dopar% {
    y <- coef(ols(rnorm(10) ~ rnorm(10)))
    return(y)
}

## Error:  task 1 failed - "could not find function "ols""

x1 <- foreach(i = 1:4, .combine = "rbind", .packages = "rms") %dopar% {
    y <- coef(ols(rnorm(10) ~ rnorm(10)))
    return(y)
}
x2 <- foreach(i = 1:4, .combine = "rbind") %dopar% {
    library(rms)
    y <- coef(ols(rnorm(10) ~ rnorm(10)))
    return(y)
}
stopCluster(cl)
cbind(x1, x2)

##          Intercept Intercept
```

```
## result.1  -0.40349   0.05660
## result.2  -0.13257   0.41074
## result.3   0.05878  -0.01726
## result.4   0.33880   0.39169
```

- You don't need to load data in each loop.

```
data <- data.frame(x1 = rnorm(10), x2 = rnorm(10), x3 = rnorm(10))
NumberOfCluster <- 4
cl <- makeCluster(NumberOfCluster)
registerDoSNOW(cl)
x <- foreach(i = 1:10, .combine = "c") %dopar% {
    y <- sum(data[i, ])
    return(y)
}
stopCluster(cl)
y

## Error:  object 'y' not found
```

- There is no interaction between loops. Forexample, you can not use the result from j^{th} loop in $j + 1^{th}$ loop.

```
x <- 0
for (i in 1:10) {
    y <- x[i] + i^i
    x <- c(x, y)
}
x

##  [1] 0.000e+00 1.000e+00 5.000e+00 3.200e+01 2.880e+02 3.413e+03 5.007e+04
##  [8] 8.736e+05 1.765e+07 4.051e+08 1.041e+10

x <- 0
cl <- makeCluster(5)
registerDoSNOW(cl)
x <- foreach(i = 1:10) %dopar% {
    y <- x[i] + i^i
    x <- c(x, y)
    return(x)
}
stopCluster(cl)
x

## [[1]]
## [1] 0 1
##
```

```
## [[2]]
## [1]  0 NA
##
## [[3]]
## [1]  0 NA
##
## [[4]]
## [1]  0 NA
##
## [[5]]
## [1]  0 NA
##
## [[6]]
## [1]  0  1 NA
##
## [[7]]
## [1]  0 NA NA
##
## [[8]]
## [1]  0  1 NA NA
##
## [[9]]
## [1]  0 NA NA
##
## [[10]]
## [1]  0 NA NA NA
```

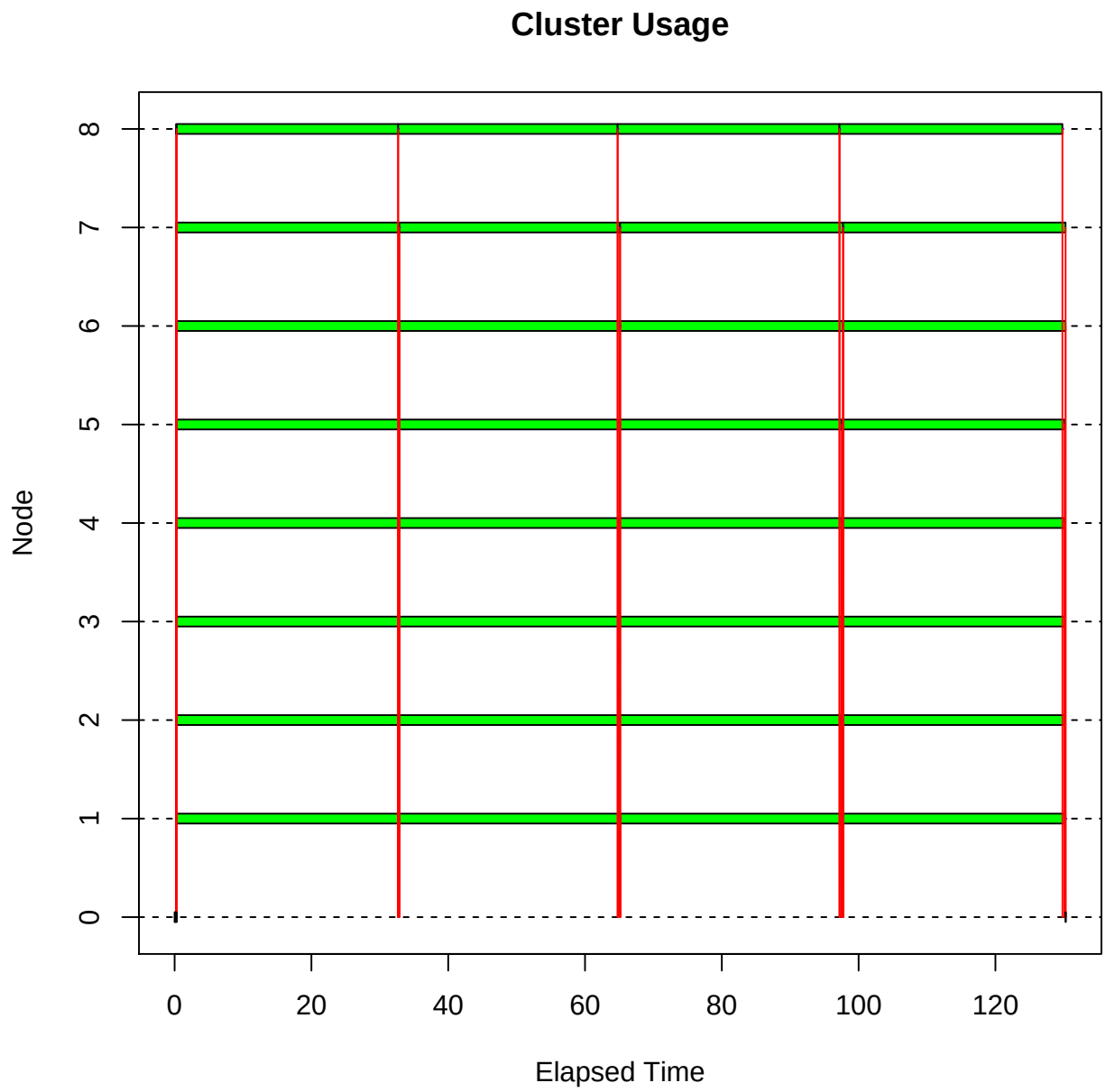
3.2 Number of clusters and how much time you can save

```
rrt.all = c(12, 12.5, 13.5, 16)
rrd = 15
tms.all = c(c(7, 8, 9, 10, 44, 45, 46)/60, c(27, 28, 29, 30)/60 + 12, 16)

rrt = rrt.all
tms = tms.all

loops = 32
err = 0.025
cv = 0.25

n <- 1
source("/Users/Michaelzmc/Documents/simuresult/snowtest/bigsimu/0608simu/design.R")

# one simulation
t0 <- system.time({
    set.seed(n)
    pk_simulate(tms = tms, rrt = rrt, rrd = rrd, err = err, cv = cv)
```

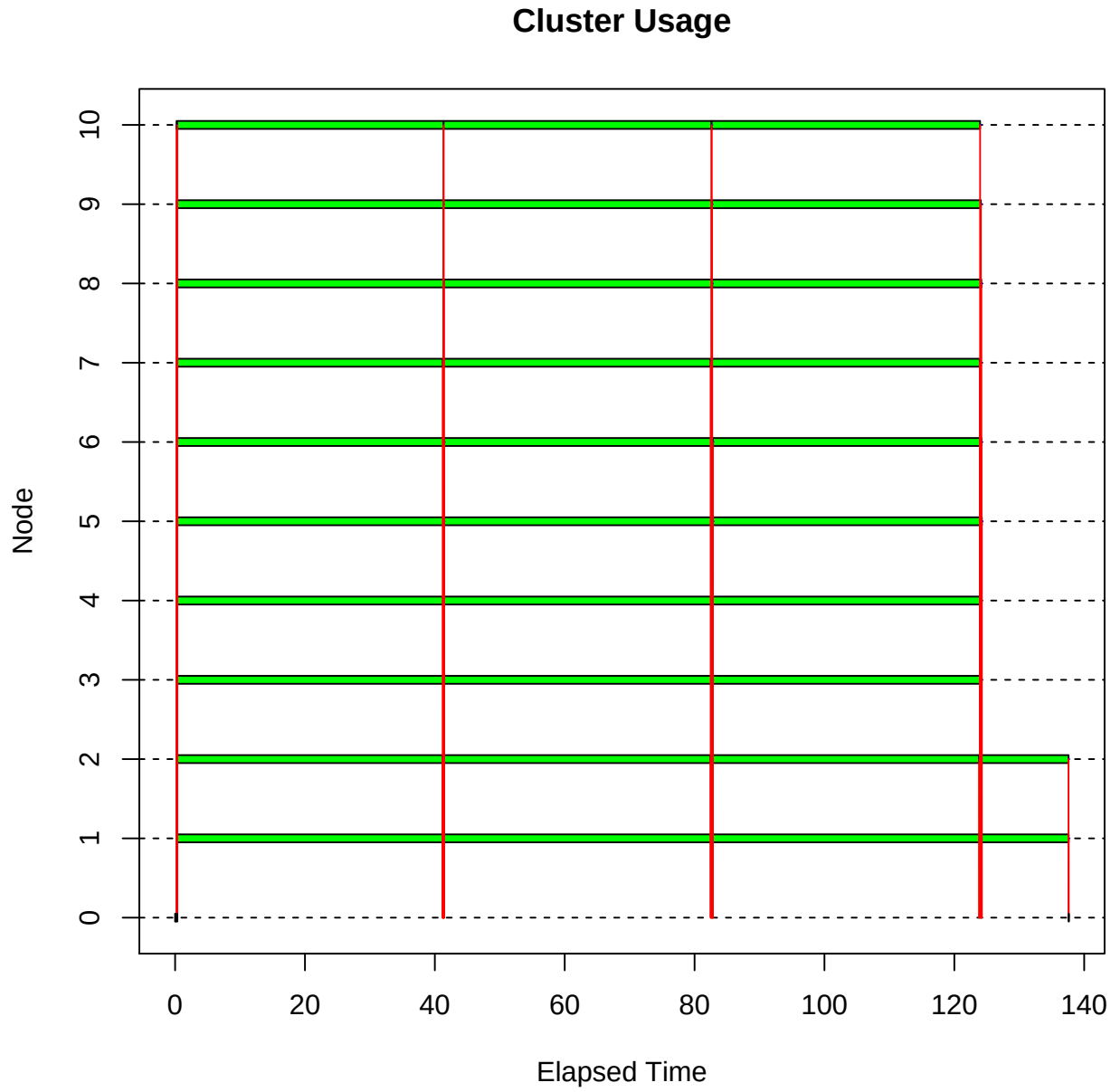
```
})

# 32 simulations
t1 <- system.time({
    y <- replicate(loops, {
        set.seed(n)
        pk_simulate(tms = tms, rrt = rrt, rrd = rrd, err = err, cv = cv)
    })
})

# number of clusters to try
n.cluster <- c(2, 4, 8, 10, 16, 32)

for (j in n.cluster) {

    t2 <- system.time({
        cl <- makeCluster(j)
        registerDoSNOW(cl)
        t.snow <- snow.time({
            x <- foreach(i = 1:loops) %dopar% {
                set.seed(n)
                return(pk_simulate(tms = tms, rrt = rrt, rrd = rrd, err = err,
                  cv = cv))
            }
        })
        stopCluster(cl)
    })

    # auto naming
    assign(paste0("t2.", j), t2)
    assign(paste0("t.snow.", j), t.snow)

}

rbind(t0, t1, t2.2, t2.4, t2.8, t2.10, t2.16, t2.32)

##       user.self sys.self elapsed user.child sys.child
## t0       15.825    0.012   15.83      0.000     0.000
## t1      508.658    0.326  508.97      0.000     0.000
## t2.2      0.722    0.033  224.15      0.002     0.002
## t2.4      0.709    0.054  165.20      0.005     0.003
## t2.8      0.409    0.062  132.28      0.009     0.007
## t2.10     0.402    0.051  140.24      0.012     0.009
## t2.16     0.414    0.053  138.65      0.018     0.013
## t2.32     0.440    0.057  142.22      0.037     0.029

# list(t0, t1, t2.2, t2.4, t2.8, t2.10, t2.16, t2.32, t.snow.2, t.snow.4,
# t.snow.8, t.snow.10, t.snow.16,t.snow.32)
```

I always use two times the number of cores in the computer. For example, There are 4 cores in my computer. I will make 8 clusters. On Accre, they have 8 cores or 12 cores. You can use 16 or 24 clusters depending on the node.

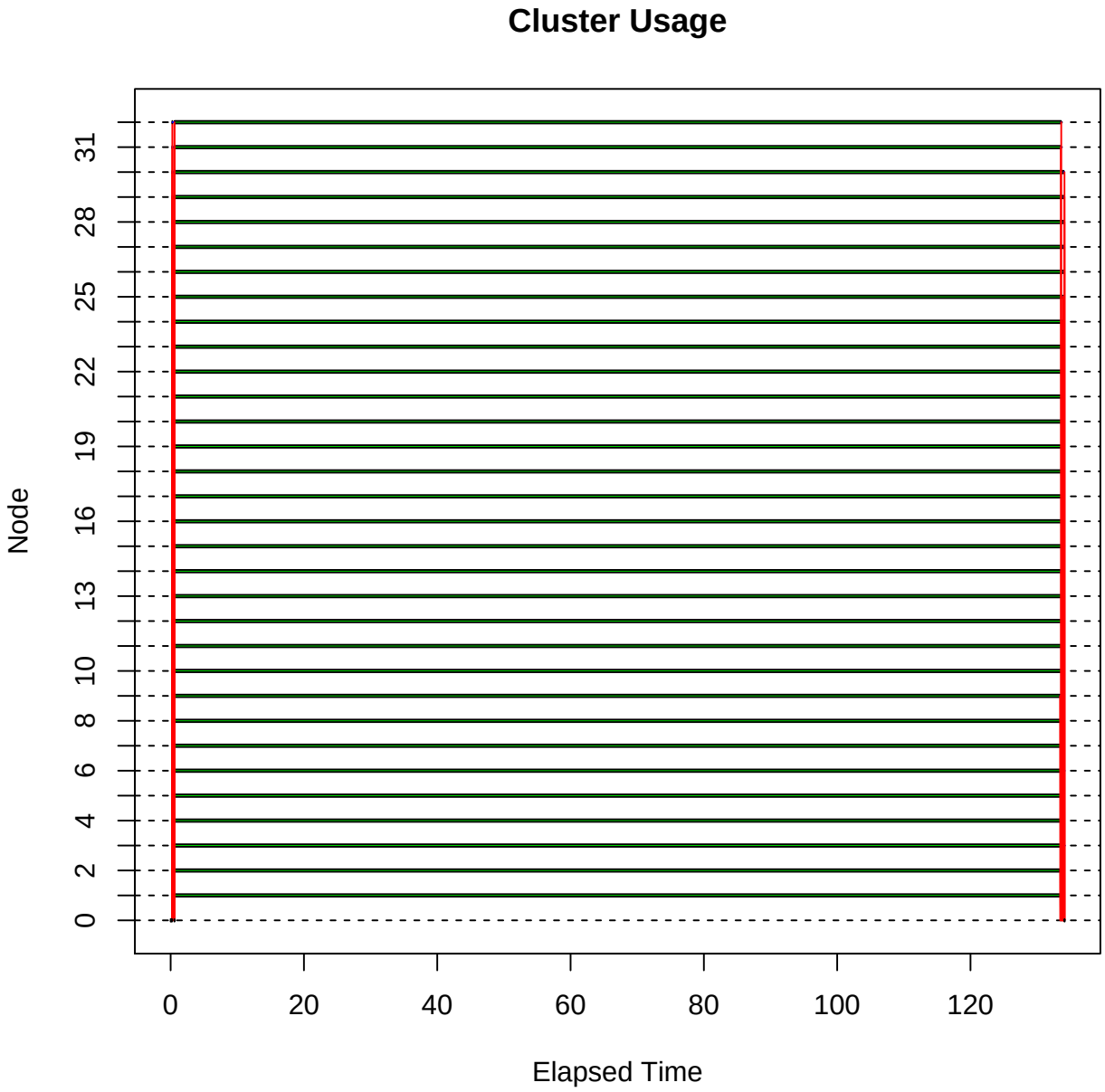
```
plot(t.snow.8)
```




```
plot(t.snow.10)
```




```
plot(t.snow.32)
```




```
time.eachjob <- function(n, t.snow) {
    mean(sapply(1:n, function(x) mean(t.snow$data[[x]][, 5]), simplify = "array"))
}
time.eachjob(2, t.snow.2)

## [1] 13.09

time.eachjob(4, t.snow.4)

## [1] 18.14

time.eachjob(16, t.snow.16)
```

```
## [1] 44.41

time.eachjob(32, t.snow.32)

## [1] 66.63
```

Another example:

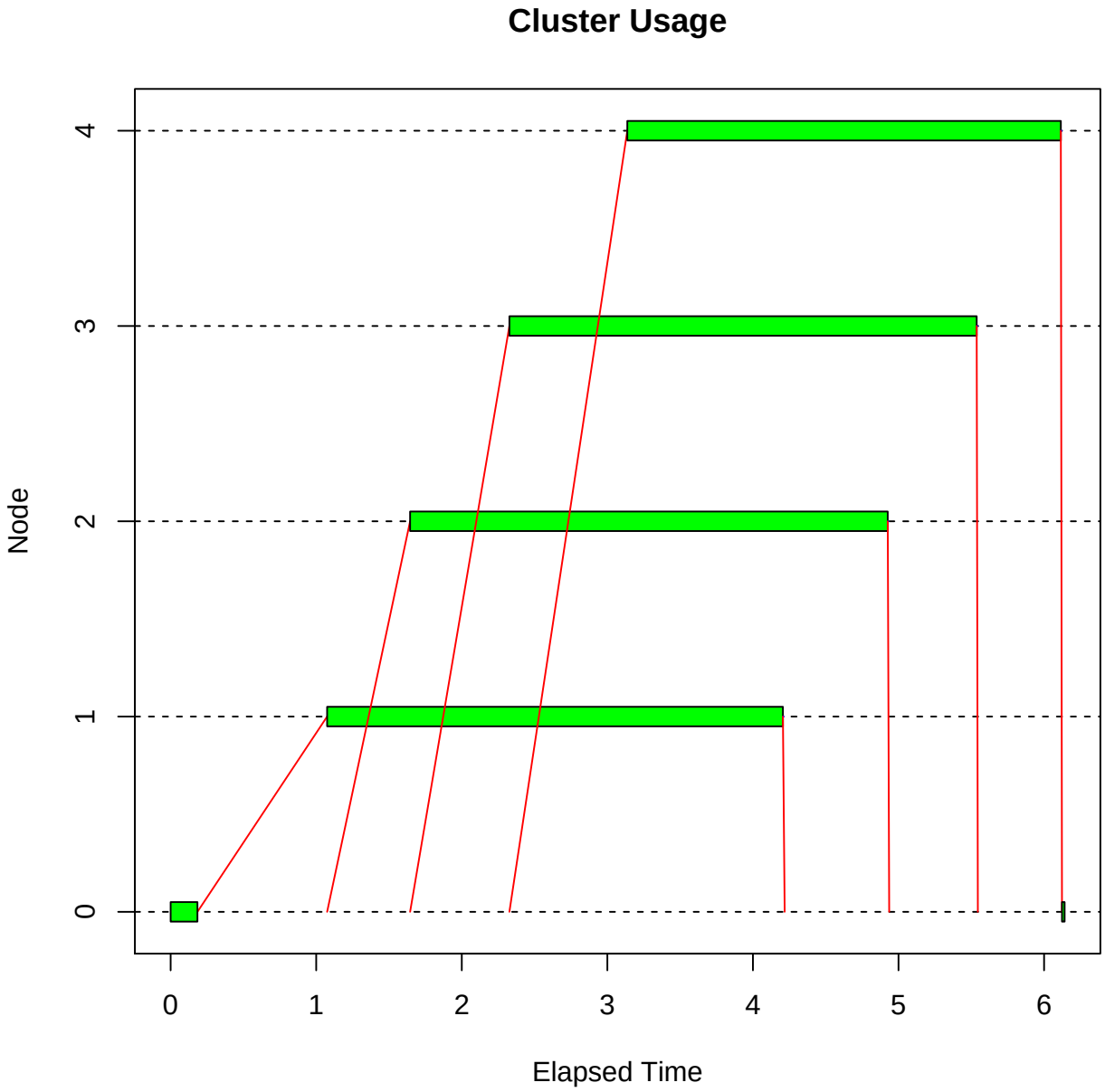
```
library(gdata)
library(XML)
library(RCurl)
library(doSNOW)
# clean the data in the xls file
clean <- function(data) {
    d <- data[, c(2, 3, 5, 6, 7)]
    colnames(d) <- c("use", "address", "city", "date", "price")
    p <- sub("\\$", "", d$price)
    p <- sub("\\,", "", p)
    d$price <- as.numeric(p)
    d <- subset(d, use %in% c("SINGLE FAMILY", "RESIDENTIAL CONDO") & price <=
        3e+05 & price >= 150000)
    d$address1 <- paste0(d$address, " ", d$city)
    # df$distance <- sapply(df$address, function(x) latlon2ft(origin=gsub(' ',
    # '_', x), destination='Nashville') , simplify='array')
    return(d)
}

m13 <- rep(c("jan", "feb", "mar", "04", "05", "06", "07", "08", "09", "10",
    "11", "12"), 8)

t1 <- system.time({
    x <- NULL
    for (i in m13) {
        x <- rbind(x, clean(read.xls(paste0("http://www.padctn.com/forms/2013/",
            i, "_z_1.xls"))))
    }
})

# start reading data
cl <- makeCluster(8)
registerDoSNOW(cl)
t2 <- system.time({
    z1.13 <- foreach(i = m13, .combine = "rbind") %dopar% {
        library(gdata)
        library(XML)
        library(RCurl)
        x <- read.xls(paste0("http://www.padctn.com/forms/2013/", i, "_z_1.xls"))
        return(clean(x))
    }
```

```
})
stopCluster(cl)

rbind(t1, t2)

##    user.self sys.self elapsed user.child sys.child
## t1      1.55    0.306   56.01      36.39     1.818
## t2      0.27    0.020   12.00       0.00     0.000
```

3.3 Higher Level SNOW Functions

$parLapply, parSapply, parApply, parRapply, parCapply, parMM(cl, A, B)$

```
A <- matrix(rnorm(1e+07), nrow = 1000)
t1 <- system.time(A %*% t(A))
cl <- makeCluster(4)
registerDoSNOW(cl)
t2 <- system.time({
    t.snow <- snow.time(parMM(cl, A, t(A)))
})
stopCluster(cl)
rbind(t1, t2)

##    user.self sys.self elapsed user.child sys.child
## t1     8.650    0.012   8.663          0         0
## t2     2.058    0.492   6.141          0         0

print(t.snow)

## elapsed    send receive  node 1  node 2  node 3  node 4
##   6.141   2.953   0.038   3.131   3.281   3.210   2.978

plot(t.snow)
```



4 Boost your simulation more than 10 times faster

- EPP
  http : *//biostat.mc.vanderbilt.edu/wiki/Main/ACCRE*
- unix script — "Sed" command

  1. Create a main R file and a main pbs file.
  2. Write a script to substitute any parameters (in R file and pbs file) and save them into different folder. Submit jobs
  3. Create a R file to read results.