# Informe

*Aguiar Luis, Figueira Miguel y Padrino David*

## *Multiplicación Matriz – Vector:*

Tomaremos los parámetros que necesitamos para ejecutar la función (la ubicación de los datos de la matriz y el vector).

```
library(“parallel”)
library("foreach")
library("doParallel")

#parametros

matriz <- "datos/tblAkv10x10.csv"
vector <- "datos/tblxkv10.csv"
n <- 10
limite = -1
```

Creamos una función aparte que haga el producto de Vector por Vector para poder usarla posteriormente cuando sea necesaria.

```
# funcion que multiplica 2 vectores de igual forma
# como se necesita en la multiplicacion matriz- vector
vxv <- function (vector1,vector2){
  #vector 1 es el vector de la matriz
  #vector 2 es el vector
  sum <- 0
  for (i in 1:length(vector1)){
    sum <- vector1[i] * vector2[i] + sum
  }
  return (sum)
}
```

Realizamos la lectura para determinar la cantidad máxima de valores que podemos tener en memoria

```
#open conection
con.matriz <- file(matriz, "r")
con.vector <- file(vector, "r")
```

```
# read just 1 value to know the size of the string and so
# know the max of values that can read in memory
tam.matriz <- object.size(readLines(con.matriz,1))
tam.vector <- object.size(readLines(con.vector,1))

#close conection to reboot the pointer of the file
close(con.matriz,type="r")
close(con.vector,type="r")
```

Aquí se realiza tanto el “map” como el “reduce”.

```
#function to know the limit of the memory
memoria.limite <- memlimit(limite)

#cantidad de N valores que puedo leer o almacenar en memoria
maximo.leer.matriz <-  as.integer(memoria.limite / tam.matriz) %/% 2
maximo.leer.matriz <- 2
max <- 2
if (maximo.leer.matriz > n){
  # then i can read all vector and all first row of the matrix
  # to calculate the first row of the new vector

  # open the conection with files
  con.matriz <- file(matriz, "r")
  con.vector <- file(vector, "r")

  #its the same vector for all the rows of the matrix
  valores.vector <- as.numeric(unlist(strsplit(readLines(con.vector,n,warn =
FALSE),",")))
  valores.vector <- valores.vector[seq(from = 2 , to = n*2 , by = 2)]
  for (i in 1:n){
    valores.matriz <- as.numeric(unlist(strsplit(readLines(con.matriz,n,warn =
FALSE),",")))
    valores.matriz <- valores.matriz[seq(from = 3, to = n*3,by = 3 )]
    c <- c(paste(i,as.character(vxv(valores.matriz,valores.vector)),sep=","))
    #print(c)
    writeLines(c,con.resultado)
  }
  close(con.matriz,type="r")
  close(con.vector,type="r")
  close(con.resultado,type="w")

}else{
  #n-(i*maximo.leer.matriz)
  #vector not fit in memory
  con.matriz <- file(matriz, "r")
```

```
  con.resultado <- file("resultado_vector.csv", "w")
  for(j in 1:n){
    con.vector <- file(vector, "r")
    con.intermedio <- file("intermedio.csv", "w")
    for(i in 1:ceiling(n/maximo.leer.matriz)){
      valores.vector <- as.numeric(unlist(strsplit(readLines(con.vector,max,
                                                             warn = FALSE),",")))
      valores.vector <- valores.vector[seq(from = 2 , to = max*2 , by = 2)]
      if (n < i*maximo.leer.matriz){
        valores.matriz <- as.numeric(unlist(strsplit(readLines(
                          con.matriz, n = (i*maximo.leer.matriz) - n,warn =
FALSE),",")))

        valores.matriz <- valores.matriz[seq(from = 3, to = max*3,by = 3 )]
        valores.matriz <- valores.matriz[!is.na(valores.matriz)]

      }else{
        valores.matriz <- as.numeric(unlist(strsplit(readLines(con.matriz,max,warn =
FALSE),",")))
        valores.matriz <- valores.matriz[seq(from = 3, to = max*3,by = 3 )]
      }
      writeLines(as.character(vxv(valores.matriz,valores.vector)),con.intermedio)
    }
    rm(valores.vector)
    rm(valores.matriz)
    close(con.vector,type="r")
    close(con.intermedio,type="w")
    con.intermedio <- file("intermedio.csv", "r")
    suma <- 0
    for(k in 1:ceiling(n/max)){
      suma <- suma + sum(as.numeric(readLines(con.intermedio,n = maximo.leer.matriz)))
      #print(paste(j,suma,sep=","))
    }
    close(con.intermedio,type="r")
    c <- c(paste(j,suma,sep=","))

    writeLines(c,con.resultado)
    #the result it is in the file resultado.csv

    rm(c)
    rm(suma)
  }
  close(con.matriz,type="r")
  close(con.resultado,type="w")
}
```

## *Multiplicación Matriz – Matriz:*

Consideramos el tamaño de la memoria a usar

```
memlimit <- function(size){
  if (size == -1 ){
    return(memory.limit() - 100)
  }else{
    return (size)
  }
```

Tomamos los parámetros a usar, los cuales serán las direcciones de los archivos que contienen las 2 matrices

```
#parametros
matriz  <- "tblAkv3x3.csv"
matriz2 <- "tblAkv3x3.csv"
n <- 3
limite = -1
```

Obtenemos los valores de cada columna que hay en la segunda matriz.

```
c <- c()
columna <- 3
if(columna == n){
    for(k in seq(from=columna,to=((n*n)-1),by=3)){
      a <- read.csv(matriz2,header = F,skip = k,nrows=1)
      print(a)
      c <- c(c,a$V3)
    }
}else{
    for(k in seq(from=columna,to=((n*n)-(n-columna)),by=3)){
      a <- read.csv(matriz2,header = F,skip = k,nrows=1)
      print(a)
      c <- c(c,a$V3)
    }
}
```

Cálculo de cantidad de elementos a tomar, basado en la memoria disponible.

```
con.matriz <- file(matriz, "r")
# read just 1 value to know the size of the string and so
# know the max of values that can read in memory
tam.matriz <- object.size(readLines(con.matriz,1))
#close conection to reboot the pointer of the file
```

```
close(con.matriz,type="r")
```

Producto de fila de la primera matriz por la columna de la matriz B

```
# funcion que multiplica 2 vectores de igual forma
# como se necesita en la multiplicacion matriz- vector
vxv <- function (vector1,vector2){
  #vector 1 es el vector de la matriz
  #vector 2 es el vector
  sum <- 0
  for (i in 1:length(vector1)){
    sum <- vector1[i] * vector2[i] + sum
  }
  return (sum)
}
```

Realización del MapReduce

```
con.resultado <- file("resultado_matriz.csv", "w")
con.matriz <- file(matriz, "r")
con.matriz2 <- file(matriz2,"r")
for(i in 1:n){
  valores.matriz <- as.numeric(unlist(strsplit(readLines(con.matriz,n,warn =
FALSE),",")))
  valores.matriz <- valores.matriz[seq(from = 3, to = n*3,by = 3 )]
  for (j in 1:n){
      c <- c()
      columna <- j-1
      for(k in seq(from=columna,to=((n*n)-1),by=3)){
        a <- read.csv(matriz2,header = F,skip = k,nrows=1)
        c <- c(c,a$V3)
      }
      writeLines(paste(i,j,vxv(valores.matriz,c)),con.resultado)
  }

}


close(con.resultado,type="w")
close(con.matriz,type="r")
close(con.matriz2,type="r")
```

Para el caso de cuando alguno de los lementos no cabe en memoria es muy similar al código ya presente en este informe.