Computación de alto rendimiento

Milagros Camila Briceño Francia

## Computación de alto rendimiento (HCP)

* Agiliza la investigación.
* La mayoría de las librerías de R utilizan un core o núcleo.
* Optimizar el rendimiento informático.
* Procesamiento de grandes volúmenes de datos.

## Librerías de R

Se utilizaron las siguientes librerías:

library(foreach)  
library(parallel)  
  
library(doParallel)

## Loading required package: iterators

library(doRNG)

## Loading required package: rngtools

## Paralelización

Algunas funciones de R para la paralelización:

#Número de núcleos o cores de la CPU  
detectCores() #8  
  
#Crea un conjunto de copias de R para   
#la ejecución en paralelo.  
cl = makeCluster(num)  
  
#Registra o conecta al worker cl  
registerDoParallel(cl)  
  
#Desconecta al worker   
stopCluster(cl)  
  
#Número de workers actuales  
getDoParWorkers()

## Ejemplos de Aplicación

#foreach y %do%  
x = foreach(a=1:10, b=rep(10, 5), .combine=c) %do% {a + b}; x

## [1] 11 12 13 14 15

m = matrix(rnorm(9), 3,3)  
foreach(i =1:ncol(m), .combine=c) %do% mean(m[,i])

## [1] -0.9952769 -0.2426145 0.1249707

# %dopar%  
cl = makeCluster(3); registerDoParallel(cl)  
  
a = matrix(1:16, 4, 4)  
b = t(a)  
  
foreach(b=iter(b, by='col'), .combine= cbind) %dopar%(a %\*%b)

## [,1] [,2] [,3] [,4]  
## [1,] 276 304 332 360  
## [2,] 304 336 368 400  
## [3,] 332 368 404 440  
## [4,] 360 400 440 480

stopCluster(cl)

## Tarea 2

### Función original

#Número de simulaciones  
n = 100000  
  
#Función con Pvalor como umbral   
pvalor1 = function(x){  
 mc = rnorm(x)   
 pvalor = t.test(mc)$p.val  
 return(ifelse(pvalor<0.05,1,0))  
}

## Tarea 2

### Función original

#Creamos un vector con el tamaño de las muestras  
data = rep(100,n)  
  
#Utilizamos sapply para aplicar la función creada  
set.seed(123)  
system.time({res1 = mean(sapply(data,pvalor1))})

## user system elapsed   
## 7.75 0.04 7.90

res1

## [1] 0.05039

## Tarea 2

### Desarrollo con Paralelización

#Solución 1  
cl = makeCluster(5); registerDoParallel(cl)  
registerDoRNG(123)  
system.time({res1.1 = foreach(a=rep(100,n),  
 .combine = c) %dopar%{   
 pvalor1(a)};res1.1=mean(res1.1)})

## user system elapsed   
## 64.47 5.95 72.06

stopCluster(cl)  
res1.1

## [1] 0.05121

## Tarea 2

### Desarrollo con Paralelización

#Solución 2  
cl = makeCluster(5)  
registerDoParallel(cl)  
clusterSetRNGStream(cl, iseed = 123)  
system.time({res1.2 = mean(parSapply(  
 cl,rep(100,n),pvalor1))})

## user system elapsed   
## 0.01 0.04 3.08

stopCluster(cl)  
res1.2

## [1] 0.04842

## Tarea 3

### Función Original

Rprof('profilling.out')

#Varianza muestral sin corrección  
varsc = function(x) var(x)\*(length(x)-1)/length(x)  
#Varianza muestral con corrección  
#var  
  
#Simulación   
#Tamaño de las muestras  
N = c(10,100,1000)  
  
#Varianza poblacional  
sigma = c(1,10,100)  
  
#Repeticiones  
n=100000

## Tarea 3

### Función Original

#Función  
simu.n = function(m,n,sigma,mu=0,seed=2022){  
 set.seed(seed)  
 m.s2 = replicate(m,varsc( #Sin corregir  
 rnorm(n,mean=mu,sd=sqrt(sigma))))  
 set.seed(seed)  
 m.s2c = replicate(m,var(  
 rnorm(n,mean=mu,sd=sqrt(sigma)))) #Corregida  
   
 sesgo.s2 = mean(m.s2)-sigma  
 sesgo.s2c = mean(m.s2c)-sigma  
   
 #Evaluamos cual presenta el menor sesgo  
 cat("Simulación sigma: ",sigma," n: ",n,  
 "\n Varianza ",mean(m.s2),  
 "(s/corregir) ", mean(m.s2c), "(corregida)",  
 "\n Sesgo: ",mean(sesgo.s2),  
 "(sin corregir) ", mean(sesgo.s2c), "(corregida) \n")  
 cat('Corregida: ',ifelse(sesgo.s2c < sesgo.s2,T,F),"\n ")  
}

## Tarea 3

### Función Original

#Ejecución de la función  
for (s in sigma) {  
 for (n1 in N) {  
 simu.n(n,n1,s)  
 }  
}

## Simulación sigma: 1 n: 10   
## Varianza 0.8985382 (s/corregir) 0.9983758 (corregida)   
## Sesgo: -0.1014618 (sin corregir) -0.001624242 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 1 n: 100   
## Varianza 0.9898972 (s/corregir) 0.9998961 (corregida)   
## Sesgo: -0.01010283 (sin corregir) -0.0001038697 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 1 n: 1000   
## Varianza 0.9991066 (s/corregir) 1.000107 (corregida)   
## Sesgo: -0.0008934051 (sin corregir) 0.0001067016 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 10 n: 10   
## Varianza 8.985382 (s/corregir) 9.983758 (corregida)   
## Sesgo: -1.014618 (sin corregir) -0.01624242 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 10 n: 100   
## Varianza 9.898972 (s/corregir) 9.998961 (corregida)   
## Sesgo: -0.1010283 (sin corregir) -0.001038697 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 10 n: 1000   
## Varianza 9.991066 (s/corregir) 10.00107 (corregida)   
## Sesgo: -0.008934051 (sin corregir) 0.001067016 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 100 n: 10   
## Varianza 89.85382 (s/corregir) 99.83758 (corregida)   
## Sesgo: -10.14618 (sin corregir) -0.1624242 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 100 n: 100   
## Varianza 98.98972 (s/corregir) 99.98961 (corregida)   
## Sesgo: -1.010283 (sin corregir) -0.01038697 (corregida)   
## Corregida: FALSE   
## Simulación sigma: 100 n: 1000   
## Varianza 99.91066 (s/corregir) 100.0107 (corregida)   
## Sesgo: -0.08934051 (sin corregir) 0.01067016 (corregida)   
## Corregida: FALSE   
##

Rprof()

## Tiempos de ejecución de la Tarea 3

#Funciones con mayor tiempo de ejecución  
head(summaryRprof("profilling.out")$by.self)

## self.time self.pct total.time total.pct  
## "rnorm" 85.18 60.12 85.96 60.67  
## "var" 16.52 11.66 134.46 94.90  
## "stopifnot" 14.00 9.88 19.86 14.02  
## "is.data.frame" 7.78 5.49 93.74 66.16  
## "pmatch" 3.78 2.67 3.98 2.81  
## "lapply" 3.06 2.16 141.34 99.76

## Tarea 3

### Función con Paralelización

tarea3= function(m,n,sigma,mu=0){  
 cl = makeCluster(5); registerDoParallel(cl)  
 clusterSetRNGStream(cl, iseed = 123)  
 m1 = parApply(cl,parSapply(cl,rep(n,m),rnorm,  
 mean=mu, sd=sqrt(sigma)),2,varsc)  
 stopCluster(cl)  
   
 cl = makeCluster(5); registerDoParallel(cl)  
 clusterSetRNGStream(cl, iseed = 123)  
 m2 = parApply(cl,parSapply(cl,rep(n,m),rnorm,  
 mean=mu, sd=sqrt(sigma)),2,var)  
 stopCluster(cl)  
   
 #Sesgo  
 sesgo.s2 = mean(m1)-b  
 sesgo.s2c = mean(m2)-b  
 res = c(a,b,mean(m1),sesgo.s2,mean(m2),sesgo.s2c)  
 return(res)  
}

## Tarea 3

### Función con Paralelización

options(digits = 3)

#Combinaciones tamaño de muestra  
cl = makeCluster(5)  
registerDoParallel(cl)  
  
system.time({  
 res2 = foreach(a=rep(N,3),b=sort(rep(sigma,3)),.combine = rbind) %do% {  
 tarea3(n,a,b)} })

## user system elapsed   
## 96.9 61.4 347.1

res2

## [,1] [,2] [,3] [,4] [,5] [,6]  
## result.1 10 1 0.899 -0.10086 0.999 -0.000956  
## result.2 100 1 0.990 -0.00981 1.000 0.000191  
## result.3 1000 1 0.999 -0.00125 1.000 -0.000254  
## result.4 10 10 8.991 -1.00861 9.990 -0.009563  
## result.5 100 10 9.902 -0.09811 10.002 0.001908  
## result.6 1000 10 9.987 -0.01253 9.997 -0.002536  
## result.7 10 100 89.914 -10.08606 99.904 -0.095627  
## result.8 100 100 99.019 -0.98111 100.019 0.019084  
## result.9 1000 100 99.875 -0.12534 99.975 -0.025361