Práctica 8: Modelo de urnas

En la práctica 8 se modela el fenómeno de fragmentación y coalescencia de una cantidad de partículas `n`, que forman un número de cúmulos `k`, los cuales pueden unirse o fragmentarse para formar cúmulos más grandes o más pequeños siguiendo la probabilidad `rotura` (Ecuación 1) y `unión` (Ecuación 2), las cuales se encuentra en función del tamaño del cúmulo, es decir, los cúmulos más pequeños tienen mayor probabilidad a unirse a otros para formar cúmulos más grandes mientras que cúmulos más grandes tienen una mayor probabilidad de fragmentarse en cúmulos más pequeños.

$$rotura(x) = \frac{1}{1+e^{\frac{c-x}{d}}} \qquad (1)$$

$$union(x) = e^{-x/c} \qquad (2)$$

Meta

Paralelizar de manera eficiente el código proporcionado en clase que simula la fragmentación y la coalescencia de partículas.

Desarrollo del código

En primer lugar, se discutirán las modificaciones realizadas al código ejemplo [1] proporcionado en clase. Con el objetivo de paralelizar este código, fueron creadas tres funciones: `para.romperse`, `para.unirse` y `para.nt`, las cuales realizan los cálculos incluidos en `for (i)` de las líneas `65`,`80` y `93` del código secuencial. Se decidió paralelizar estas líneas de código ya que realizan una gran cantidad cálculos independientes durante varios ciclos, los cuales pueden ser realizados de manera eficiente utilizando el paquete `do.Parallel` ahorrando así, un poco de tiempo de ejecución de esta simulación.

```
>para.romperse<-function (){
+  cumulos <- integer()
+  urna <- freq[i,]
+  if (urna$tam > 1) {
+    cumulos <- c(cumulos, romperse(urna$tam, urna$num))
+  } else {
+    cumulos <- c(cumulos, rep(1, urna$num))}
+
+  return(cumulos)
+  }
```

```
>para.unirse<-function(){
+  cumulos <- integer()
+  urna <- freq[i,]
+  cumulos <- c(cumulos, unirse(urna$tam, urna$num))
+  return(cumulos)
+}


>para.nt<-function(){
+  suma <- juntarse[2*i-1] + juntarse[2*i]
+  return(suma)
+}

> cumulos<-foreach(i=1:dim(freq)[1],.combine=c) %dopar% para.romperse()
> cumulos<-foreach(i=1:dim(freq)[1],.combine=c) %dopar% para.unirse()
```

Durante la ejecución del código original, es decir, el código secuencial, nos encontramos con funciones `assert` del paquete `testit`, las cuales son útiles asegurarse que la cantidad de partículas `n`, no se vea modificada durante este trabajo. Con el objetivo de cumplir con este requerimiento, fue necesario remplazar algunas asignaciones del vector `cumulos` y remplazarlas por los nombres `mayores` y `pares`.

```
> juntarse <- -cumulos[cumulos < 0]
> mayores <- cumulos[cumulos > 0]
> assert(sum(mayores) + sum(juntarse) == n)
> nt <- length(juntarse)
+
> if (nt > 0) {
+    if (nt > 1) {
+      juntarse <- sample(juntarse)
+     pares<-foreach(i=1:floor(nt / 2),.combine=c) %dopar% para.nt()
+            cumulos<-c(mayores,pares)
+           }
+
+    if (nt %% 2 == 1) {
+    cumulos<-c(pares,mayores,juntarse[nt])
+    }
+  }
>
> assert(sum(cumulos) == n)
```

Para obtener el tiempo de ejecución de ambos códigos se realizó una diferencia entre un valor inicial de tiempo `ti` y uno final `tf` utilizando el comando `Sys.time()`. Se creyó necesario desarrollar un nuevo código en el cual se observa la comparación entre ambos códigos .Éste, al igual que los códigos necesarios para el desarrollo de esta práctica se encuentra dentro de este repositorio. Las figuras obtenidas para esta práctica fueron creadas utilizando la librería `ggplot2`.

```
>ciclos<-6
>k<-125000
>resultados<-data.frame()

>for (replicas in 1:ciclos){
+
+  source('~/GitHub/Simulacion/Simulacion/P8/Cod_P8_T8.R')
+  t.paralel<-cbind("paralelo",replicas, k,tiempo)
+
+  source('~/GitHub/Simulacion/Simulacion/P8/Original/Cod_P8.R')
+  t.original<-cbind("original",replicas, k,tiempo)
+
+  resultados<-rbind(resultados,t.paralel,t.original)
+
+}
```

Resultados

Con el fin de verificar que la paralelización fue llevada a cabo de manera exitosa sin modificar la aplicación inicial del código secuencial, se prosiguió a obtener un histograma del primer y del último paso de ambos códigos (paralelizado y secuencial), los gráficos resultantes se muestran en la figura 1. Los histogramas de lado izquierdo de la figura corresponden a aquellos obtenidos durante la ejecución del programa paralelo. La figura 1 visualiza muy pocos cambios entre ambas distribuciones, por lo que se puede afirmar que ambos códigos realizan la misma función.

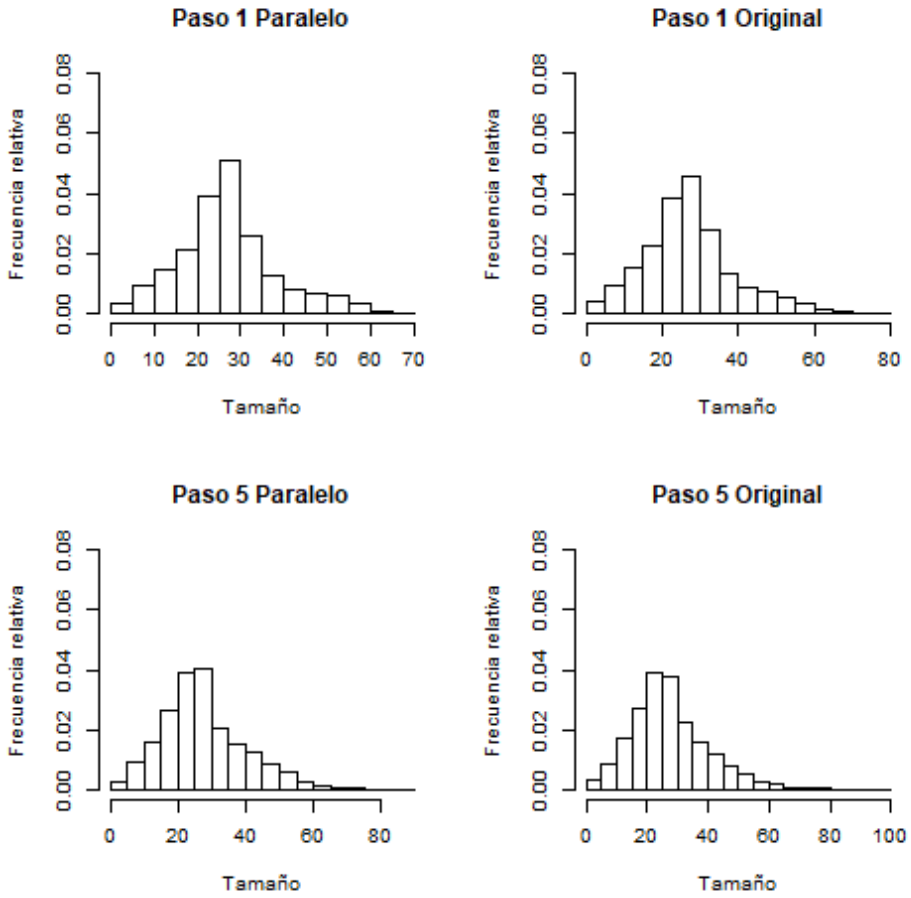

Figura 1 Comparación visual de la distribución entre los resultados de ambos códigos.

En la figura 2 se observa que el programa secuencial toma menor tiempo que el programa paralelizado, esto se debe a la baja cantidad de valores, en este caso valores de k, puesto que se destina una mayor cantidad de recursos (tres núcleos) al programa paralelizado los cuales no son aprovechados del todo. Por lo tanto, se procedió a aumentar el valor de k a 125000, el resultado se muestra en la figura 3. En este caso se puede observar que el programa paralelizado tiene un menor tiempo de ejecución cuando se compara por el programa secuencial. Esto nos permite afirmar que la paralelización cumplió su objetivo de trabajar de manera más eficiente ahorrando así tiempo de ejecución.

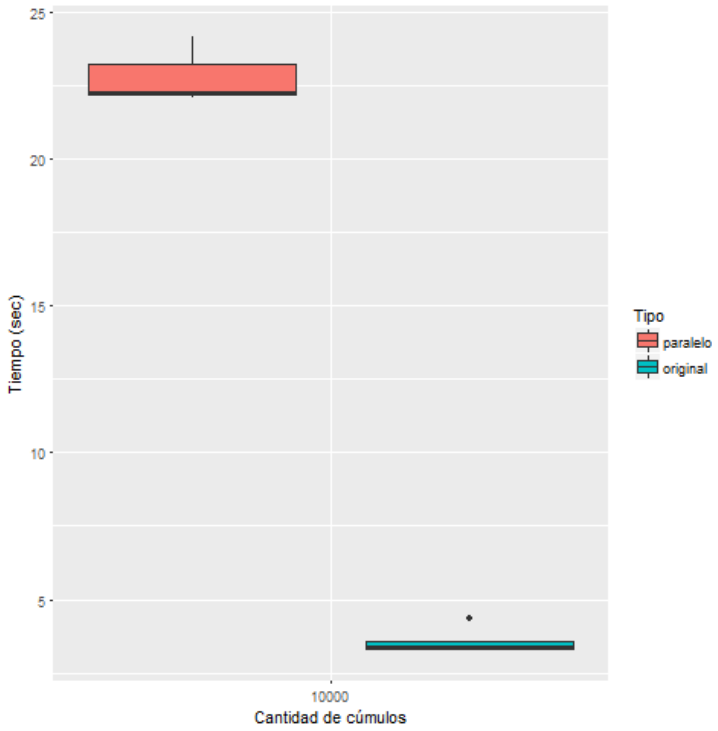

Figura 2 Medición del tiempo de programa original y paralelizado para k con valor 10000

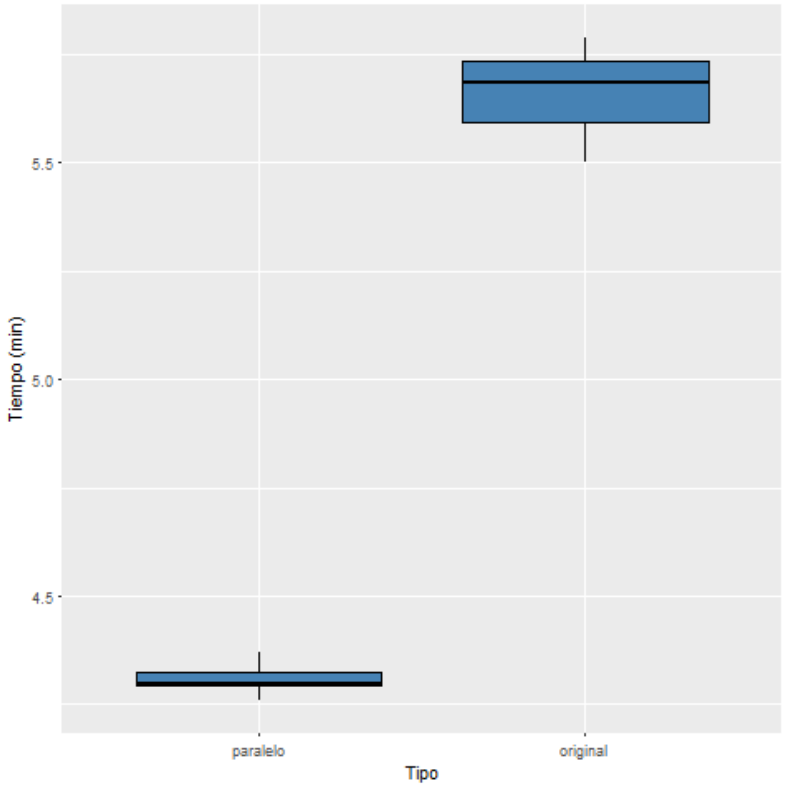

Figura 3 Medición del tiempo de programa original y paralelizado para k con valor 125000

Práctica 8: Reto 1

Meta

Comparar el tiempo de ejecución entre la simulación desarrollada secuencialmente y la simulación paralelizada variando la cantidad de cúmulos k y manteniendo la relación $n = 30k$.

Desarrollo del código

El código utilizado para el reto 1 es muy parecido al código de comparación de la tarea, con la excepción de la variación en el valor de k.

```
>ciclos<-5
>resultados<-data.frame()
>
>for (replicas in 1:ciclos){
+  for (k in seq(50000,200000,50000)){
+
+  source('~/GitHub/Simulacion/Simulacion/P8/Cod_P8_T8.R')
+  t.paralel<-cbind("paralelo",replica, k,tiempo)
+
+  source('~/GitHub/Simulacion/Simulacion/P8/Original/Cod_P8.R')
+  t.original<-cbind("original",replica, k,tiempo)
+
+  resultados<-rbind(resultados,t.paralel,t.original)
+  }
+}
```

Para corroborar que la diferencia en tiempo de ejecución resulta en una diferencia significativa, se procedió a utilizar el comando t.test que realiza la prueba tStudent en la cual se comparan los valores de dos vectores

```
>colnames(resultados)<-c("Tipo","Replica","k","Tiempo")
>resultados$Tiempo<-as.numeric(levels(resultados$Tiempo))[resultados$Tiempo]
>resultados[resultados$Tiempo>30,4]<->resultados[resultados$Tiempo>30,4]/60
>
>for (k in seq(50000,200000,50000)){
+prueba.par<-resultados[resultados$k==k& resultados$Tipo=="paralelo",]
+prueba.org<-resultados[resultados$k==k & resultados$Tipo=="original",]
+
+dat.par<-prueba.par$Tiempo
+dat.org<-prueba.org$Tiempo
+
```

```
+test<-t.test(dat.org,dat.par)
+print(test)
+}
```

Resultados

En la tabla 1 utilizó el comando

Tabla 1 Valores resultantes de la prueba tStudent para el valor de tiempo del programa original y el paralelizado

Cantidad de cúmulos *k*	p-valor	Diferencia
50000	1.409×10^{-6}	Altamente significativa
100000	0.02613	Significativa
150000	0.005121	Altamente significativa
200000	1.977×10^{-5}	Altamente significativa

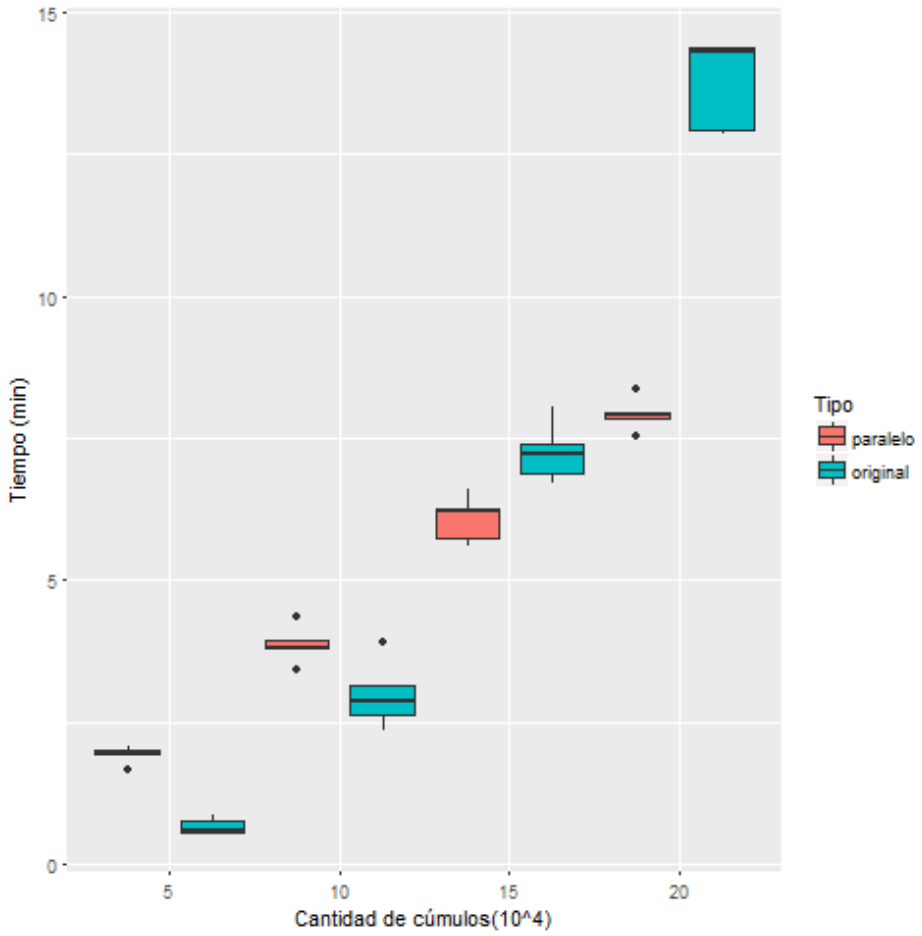

Figura 4 Medición del tiempo del programa original y paralelizado con diferentes números de k

Referencias

[1] Elisa.dyndns-web.com. (2017). P8 — R paralelo — Schaeffer. [online] Available at: http://elisa.dyndns-web.com/teaching/comp/par/p8.html [Accessed 2 Oct. 2017].