Práctica 8

Modelo de Urnas

17 de Abril del 2018

Modelo de Urnas

Un modelo de urnas esta pasado en la distribución de playa, se trata de un modelo discreto, también conocido como distribución del contagio Su utilidad radica precisamente en que es capaz de medir la aleatoriedad de los modelos de contagio de la extensión–propagación de la información sobre cualidades de los productos o servicios.

Los modelos de contagio tienen especial importancia debido a que la ocurrencia de un suceso tiene el efecto de cambiar la probabilidad de las posteriores ocurrencias de ese mismo suceso.

En la realización de la práctica 8 se simula un proceso de fragmentación y coalescencia de cúmulos de partículas ***n***, formando un número de cúmulos ***k***, mediante un modelo de urnas.

Primeramente, se generan ***k*** números enteros distribuidos normalmente, ***k*** representa el número de cúmulos iniciales y la suma de todos ellos es igual al número de partículas ***n***.

Se genera un parámetro ***c*** como el tamaño critico para que los cúmulos más pequeños se unan y los más grandes se fragmenten de acuerdo con la distribución de probabilidad especifica.
Los cúmulos pequeños se unen de acuerdo con una distribución de la forma:

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

Los cúmulos grandes se fragmentan de acuerdo con una distribución sigmoide:

$$Coalescencia\ (x) = e^{-x/C} \qquad (2)$$

Tarea

Paralelizar eficientemente el código proporcionado en clase y medir el tiempo optimizado.

Para la paralelización del experimento se utilizó la librería doParallel además se crearon dos funciones, una función para repetir la fase de unión y otra la de fragmentación.

```r
#Función Rompiendo
rompiendo <- function(){
  cumulos <- integer()
  urna <- freq[i,]
  if (urna$tam > 1) { # no tiene caso romper si no se puede
    cumulos <- c(cumulos, romperse(urna$tam, urna$num))
  } else {
    cumulos <- c(cumulos, rep(1, urna$num))
  }
  return(cumulos)
}
```

Figura 1 Función para Romper

Para esta simulación se tomaron en cuenta las siguientes consideraciones:

k = 1000, 5000, 10 000

n = 20k

Para cada experimento se realizaron 20 replicas y 10 iteraciones, el código completo de la práctica se encuentra disponible en el repositorio GitHub

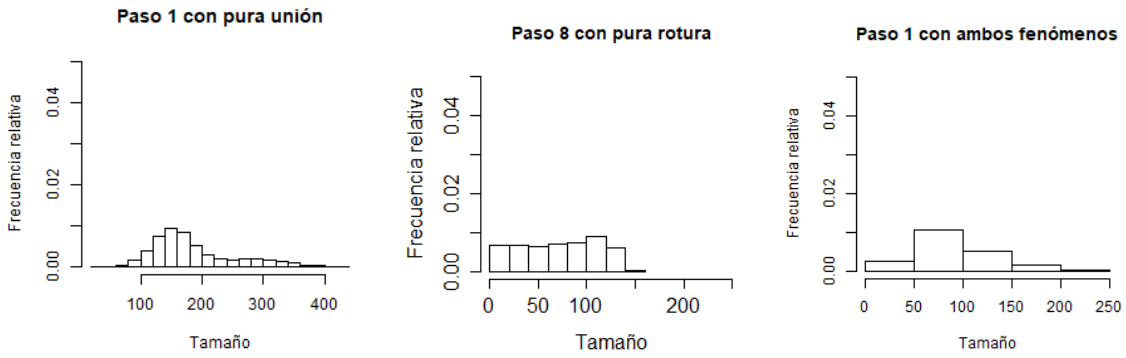
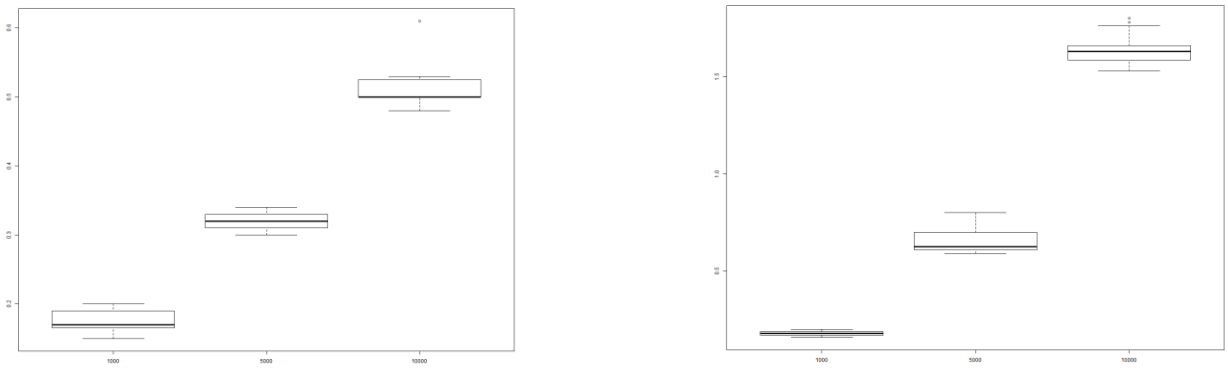


Figura 2 Ejemplos de distribución de tamaños por proceso

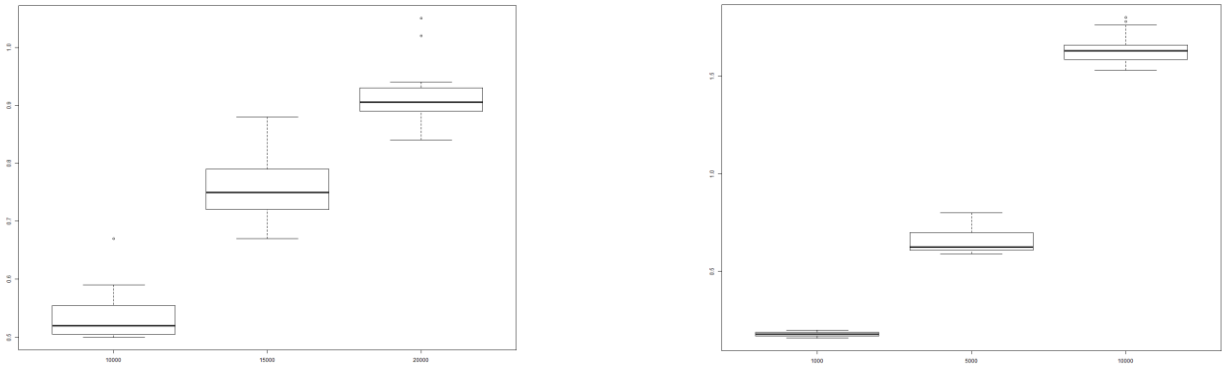
Resultados

Los Resultados obtenidos muestran una diferencia de tiempo de ejecución entre una simulación paralelizada y una sin paralelizar (Código original).



A) Paralelo *B) Secuencial*

Figura 3.- Resultados de experimentación graficando Tiempo contra tamaño ≤ 1000



A) Paralelo *B) Secuencial*

Figura 4.- Resultados de experimentación graficando Tiempo contra tamaño en un rango de 1000 a 20000

Los resultados de la simulación los vemos en las figuras 3 y 4, donde el eje y nos indica el tiempo de ejecución (segundos) y el eje x el tamaño k.

En los gráficos podemos ver que para valores inferiores (k) a 10,000 no se aprecia una eficiencia significativa al paralelizar la simulación, se logra apreciar un poco mas en los valores de 15,000 y 20,000.

Conclusiones

En los gráficos obtenidos se observo diferencias entre los procedimientos:

Secuencial: para valores pequeños resulta mas eficiente esta rutina que la paralela, mientras que para grandes valores es más rápida la paralela, por lo que no siempre el uso de paralelizar es influyente en el tiempo de ejecución, finalmente realice una simulación más, cambiando el valor de k esto me permitió observar una gran diferencia entre una simulación paralelizada y una no.

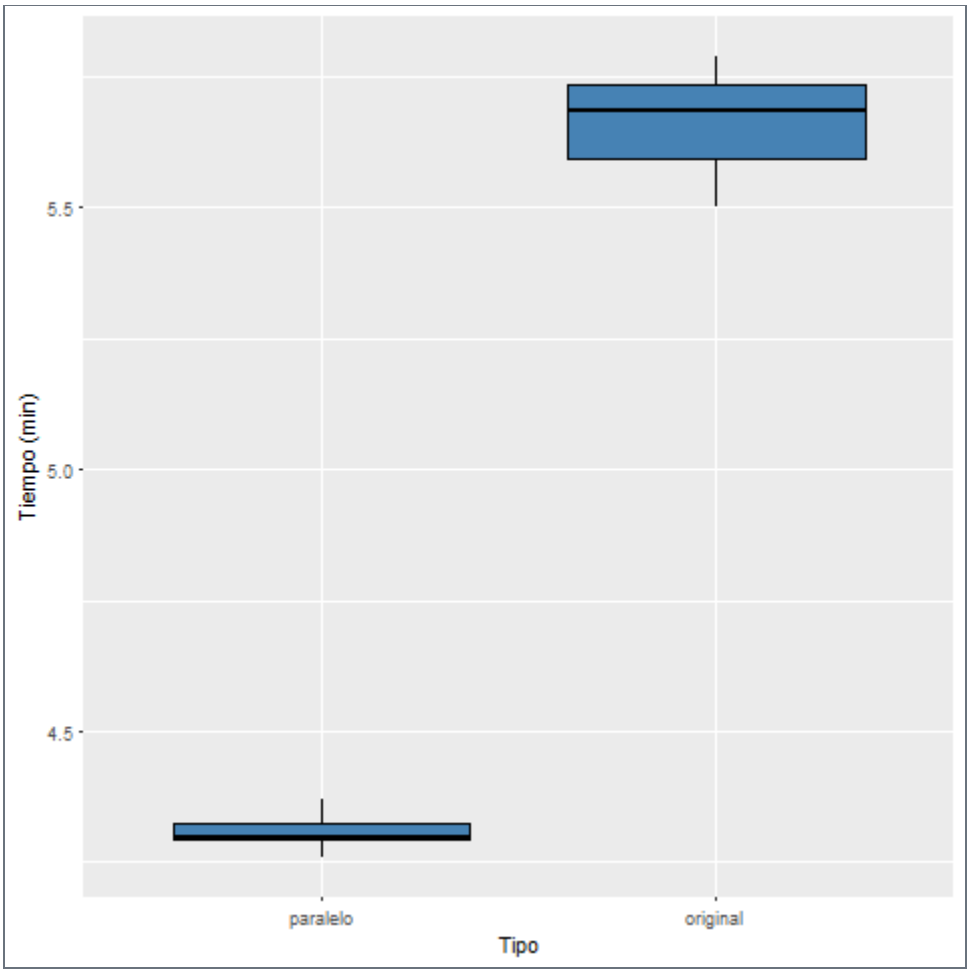


Figura Experimento con k 150000, donde se aprecia que el tiempo paralelizado y secuencial tiene un gran cambio.