Tarea 8
Simulación de Sistemas

Beatriz Alejandra García Ramos

A 3 de Octubre de 2017

1. Modelo de urnas

En esta práctica se tienen partículas que se unen para formar cúmulos y estos después pueden fragmentarse en cúmulos menores. Ésta práctica es utilizada en cuestiones químicas donde los cúmulos se pueden ver como residuos que se quieren aumentar para que al momento de filtrarse la sustancia que los contiene estos residuos no puedan pasar por ese filtro.
Para simularlo se tienen `n` partículas y `k` cúmulos que en el inicio del proceso tienen un tamaño que sigue una distribución normal. Para lograr tener los cúmulos con un mayor tamaño primero se fragmentan y luego éstas forman pares al azar para poder unirse.

2. Tareas

- La tarea base es sobre paralelizar el código de manera eficiente para lograr medir tiempos de ejecución al tener el proceso secuencial y el paralelo y lograr medir el tiempo que se ahorra al compararlas.
- Hacer variaciones en los `k` cúmulos, cambiar la manera en que las partículas se toman dependientes de la cantidad de cúmulos y verificar el ahorro significativo para estos distintos valores.
- Suponer que se tienen cúmulos suficientemente grandes para filtrarse y buscar que estos alcanzan un máximo de partículas. Analizar qué sucede haciendo al menos treinta réplicas para determinar si el número de pasos es similar cada vez que se hace la réplica.

3. Solución

Para hacer la paralelización del proceso existen dos librerías que pueden ayudar, una de ellas es donde existe la función `foreach` y otra en donde existe la función `parSapply`, dado que en las prácticas anteriores se ha trabajado con ellas y se ha descubierto que `parSapply` es más efectiva al disminuir tiempos se hizo la modificación en el código utilizando esta función.

3.1. Tarea base

Para poder realizar la tarea base se implementó la función `parSapply` paralelizando las funciones que se utilizan para fragmentar y unir los cúmulos, las cuales tienen los nombres de `rompimiento` y `uniones`, respectivamente. Éstas funciones fueron las únicas que se paralelizaron ya que haciendo el análisis del proceso no se encontró que otras tuvieran sentido paralelizarlas por el trabajo que realizaban dentro del programa.

Teniendo en cuenta que ya no es conveniente hacer otras paralelizaciones se realizaron variaciones en la duración del proceso, es decir los pasos que se dan, se realizaron las pruebas en ambos códigos,

el que está paralelizado y el que no. Las variaciones que se hicieron junto con los valores del tiempo de ejecución que se obtuvieron se encuentran en el cuadro 3.1 de la sección y visualmente en la figura 3.1, donde también se ve claramente que el proceso paralelizado, marcado en rojo, tiene una mayor duración (aunque no es un tiempo distante).

Teniendo en cuenta los tiempos que se obtuvieron se puede decir que al momento de hacer la paralelización estos no mejoran, así que el proceso secuencial es mejor que el paralelizado. Los tiempos no son muy variantes, sin embargo, podríamos hablar de que el proceso puede llegar a tener una cantidad mucho mayor de duración, la diferencia entre cada proceso podría llegar a ser variante en minutos o hasta en horas si se tiene una cantidad muy grande y esto puede afectar en el rendimiento del programa.

Tiempos de ejecución		
Duración	Sin paralelizar	Paralelizado
5	3.286006 seg	4.954754 seg
30	15.54022 seg	18.37861 seg
100	48.90014 seg	56.07408 seg
500	4.253682 min	4.527512 min
1000	7.781149 min	8.729919 min

Cuadro 1: Tiempos de ejecución comparando secuencial con paralelización haciendo variaciones en la duración.

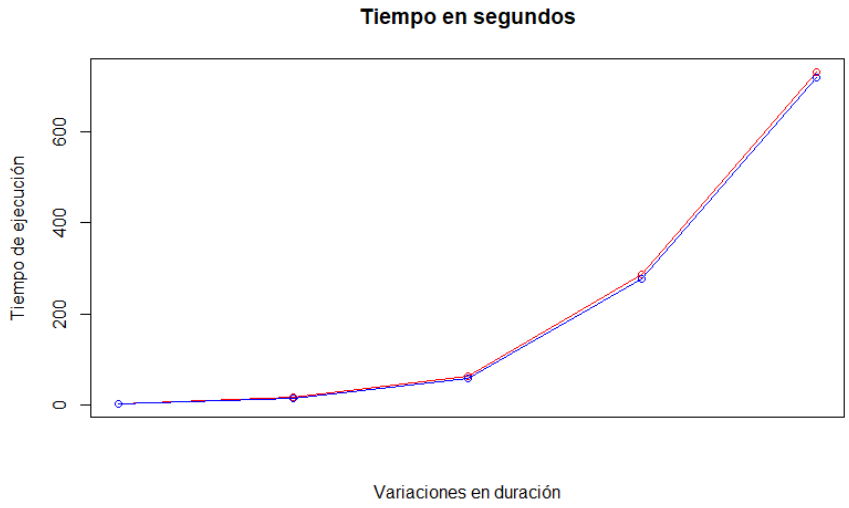

Figura 1: Tiempos de ejecución con variaciones en duración.

3.2. Reto 1

Para lograr el primer reto haciendo cambios en la cantidad de cúmulos iniciales se creó un `for` que nos ayuda a hacer las diferentes cantidades en una sola corrida del código, además se tiene la modificación de la cantidad de partículas iniciales dependiente de los `k` cúmulos al tener `30k` como la cantidad de partículas. Los tiempos estimados se guardan en un `dataframe` que sirve de apoyo para crear un gráfico que nos muestra el comportamiento de los tiempos de ejecución de las distintas corridas al variar las cantidades.

Teniendo en cuenta el cambio en la cantidad de cúmulos se muestra la figura 3.2 como referencia, la cual nos muestra el crecimiento que va teniendo el tiempo en segundos, como se puede observar, el crecimiento toma una forma parecida a una cuadrática lo cual es de esperarse ya que los aumentos que se hacen en los `k` cúmulos van creciendo linealmente.

Además se realizó una comparación al variar los cúmulos también en el proceso no paralelizado y los gráficos obtenidos se muestran en la figura 3.2, la cual tiene dos funciones, una marcada con color azul que representa el proceso paralelo y otra en color rojo que representa el proceso sin paralelizar. Claramente se tiene que el proceso paralelo tarda un poco más conforme se va aumentando la cantidad de cúmulos pero no hay una diferencia con respecto al tiempo que se tiene con el proceso sin paralelizar.

Figura 2: Tiempos de ejecución con variaciones en k de 10000, 30000, 50000, 70000, 90000 y 110000.

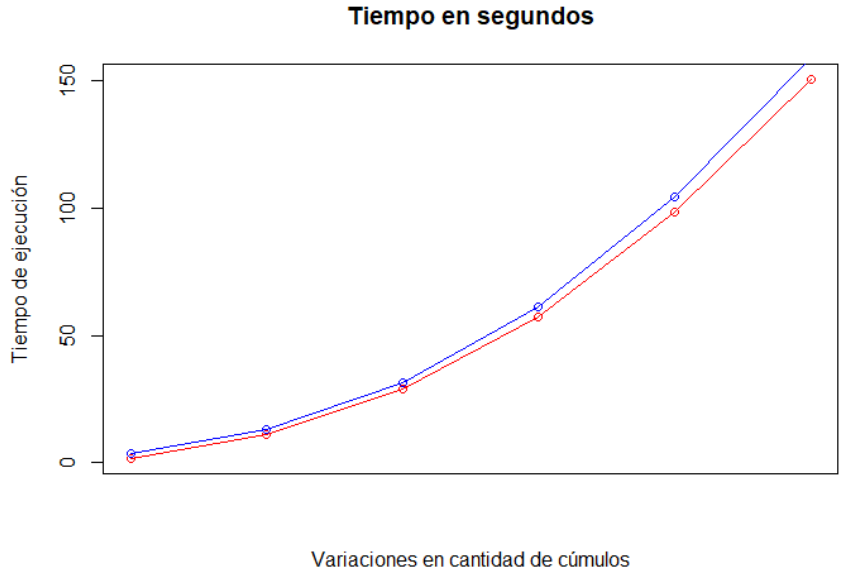

Figura 3: Tiempos de ejecución con variaciones en k de 10000, 30000, 50000, 70000, 90000 y 110000.