Reporte práctica 3
Teoría de colas

1. INTRODUCCIÓN

Un sistema de colas se puede describir como sigue. Un conjunto de "clientes" llega a un sistema buscando un servicio, esperan si este no es inmediato, y abandonan el sistema una vez han sido atendidos. En algunos casos se puede admitir que los clientes abandonan el sistema si se cansan de esperar. El término "cliente" se usa con un sentido general y no implica que sea un ser humano, puede significar piezas esperando su turno para ser procesadas o una lista de trabajo esperando para imprimir en una impresora en red.

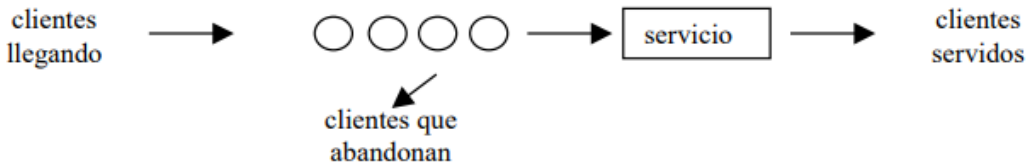

Figura 1. Sistema de colas básico en dónde se ejemplifica un solo parámetro y función.(Sabater G, 2015)

No todos los sistemas se representan como en la figura 1 en algunos casos es necesario definir mayor número de parámetros y funciones (Sabater G, 2015).

En la presente práctica se pretendió estudiar el efecto del orden de los trabajos asignados, así como el número de "trabajadores" (núcleos) asignados a dichos trabajos.

2. SIMULACIÓN

El trabajo asignado en la simulación es el ordenamiento de una cantidad de números primos, que se determinaron mediante una función en dónde se comparaba que un número no fuera divisible entre ningún entero mayor a uno o menor a si mismo. Por tanto, *n* (el número) se dividía entre los enteros desde 2, si el residuo de la división no vale cero, entonces *n* es primo. Al inicio también se coloca la condicionante que, si *n* es igual a uno o dos regresa un VERDADERO, afirmando que es primo.

Para el trabajo en paralelo se utiliza la librería *doParallel*. Se fija un rango para comparar los números, en este caso va desde mil hasta tres mil y se comparan diferente cantidad de réplicas (veinte y cien); se crean variables para el ordenamiento de los números primos; original (*ot*), que va en orden ascendente. Invertido (*it*), y como su nombre lo indica va en orden descendente. Y por último un orden aleatorio (*at*).

Posteriormente se crea un *data.frame* en dónde se guardarán los vectores de los datos. Utilizando un *for* se varía el número de núcleos trabajando desde 3 hasta 1 núcleo.

Para guardar los valores obtenidos, se crea un vector vacío para cada orden (*ot, ir, at*), y se repite de uno hasta las réplicas establecidas la determinación de tiempo para cada ordenamiento.

Para finalizar se combinan los vectores creados (*ot, it, at*), se crea y nombra un archivo e imagen en dónde se visualiza el diagrama caja-bigote de cada corrida con diferente número de núcleos. Se repitió lo anterior para 100 réplicas.

3. RESULTADOS

En la figura 1 se muestra cómo varía la distribución de los tiempos dependiendo de el orden que se elija y la cantidad de núcleos que estén trabajando. Para dos casos diferentes de núcleos trabajando (tres y dos) se observa que para el ordenamiento al azar demora menos tiempo que los otros ordenamientos. Para todos los ordenamientos cuando se tienen 3 núcleos trabajando la cantidad de tiempo disminuye considerablemente.

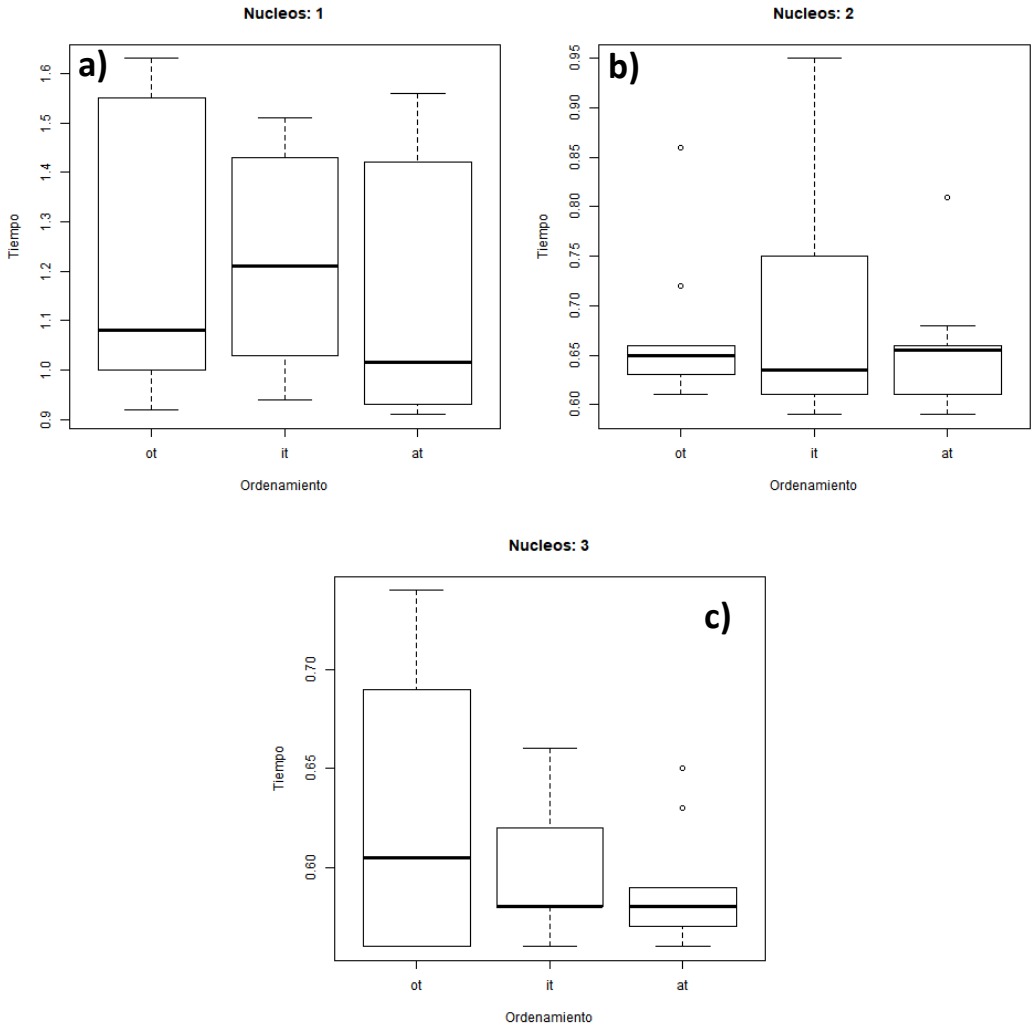

Figura 2. Gráficas caja-bigote con veinte réplicas mostrando los tres diferentes ordenamientos con diferente cantidad de núcleos, un núcleo (a), dos núcleos (b) y tres núcleos (c).

Para cien repeticiones la disminución más marcada en tiempos se observa al utilizar dos y tres núcleos en dónde además los datos tienen menor variación y se distribuyen en un rango más corto de tiempo.

Cuando se tiene solamente un núcleo trabajando los tiempos de ejecución para el ordenamiento al azar son menores que los otros dos ordenamientos, pero al estar trabajando dos y tres núcleos se tienen tiempos de ejecución similares entre los diferentes ordenamientos.

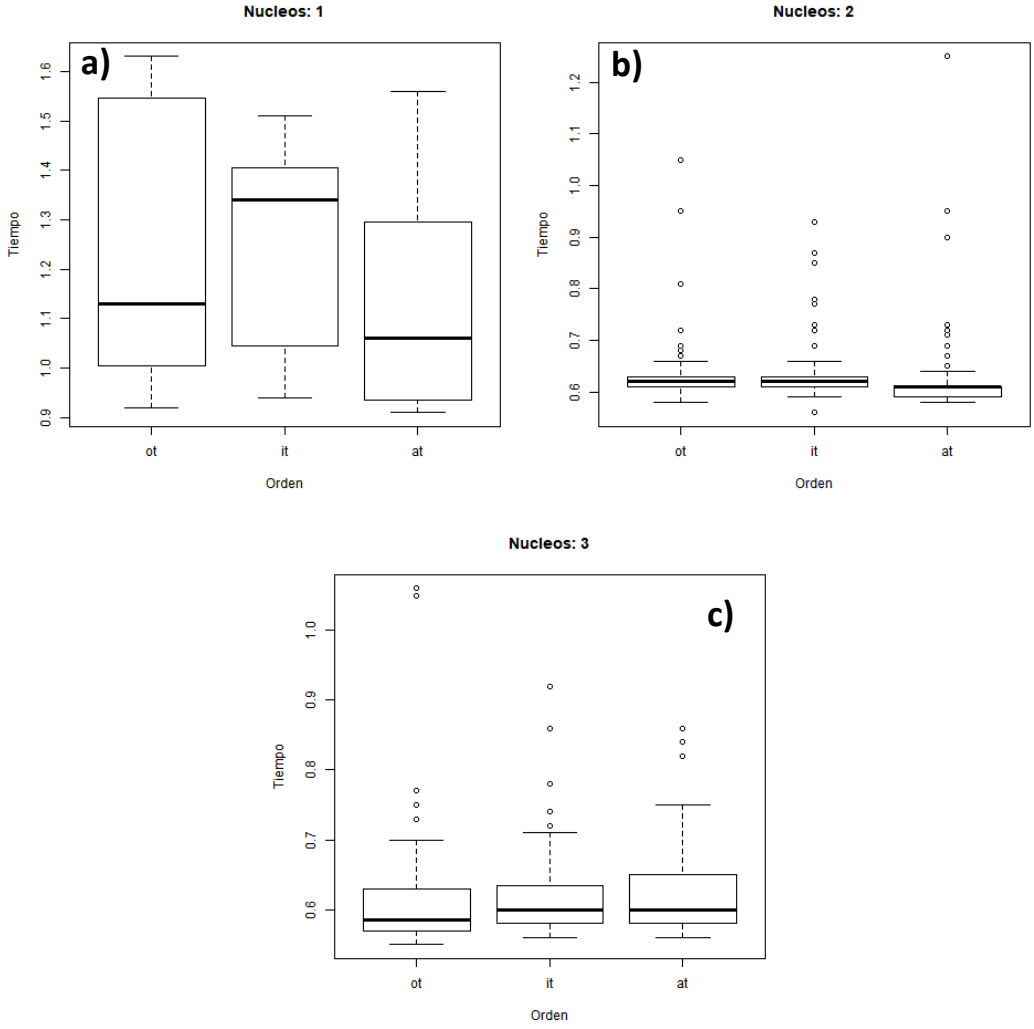

Figura 3. Gráficas caja-bigote con cien réplicas dónde se observan los tres diferentes ordenamientos con diferente cantidad de núcleos; un núcleo (a), dos núcleos (b) y tres núcleos (c).

Si se comparan las figuras 2 y 3, se puede observar fácilmente que para dos y tres núcleos hay menos variación en los valores de tiempo de ejecución, mientras que cuando sólo un núcleo trabaja los tiempos de ejecución para ambas repeticiones son similares.

4. CONCLUSIONES

Con base en lo observado en la práctica se puede concluir que los tiempos de ejecución disminuyen al aumentar el número de núcleos trabajando para realizar las tareas.

Al aumentar el número de repeticiones de las tareas el tiempo disminuye esto debido a que se priorizan las actividades o tareas más sencillas de realizar y las tareas más complejas se reparten entre los trabajadores (núcleos).

En la mayoría de los casos el menor tiempo dentro de cada cantidad de núcleos es para el “ordenamiento” al azar, esto debido a que no es en sí un orden y genera números puramente al azar, en contraste con los otros ordenamientos en los que es necesario saber cuál es mayor, cuál menor e ir comparando con los siguientes para obtener el orden deseado.

Si se requiere un gran número de repeticiones es recomendable optar por la mayor cantidad de núcleos disponibles para realizar las tareas, así se disminuye el tiempo de ejecución y se obtienen datos menos variables.