Reporte práctica tres: Teoría de colas

José Anastacio Hernández Saldaña
Posgrado de Ingeniería de Sistemas
1186622
jose.hernandezsld@uanl.edu.mx

27 de agosto de 2017

Resumen

Este es un reporte sobre la práctica tres sobre el tema teoría de colas que se realizó en la clase de Simulación de Sistemas, cómputo paralelo en R.

1. Tarea: Diferencia en tiempos de ejecución de los diferentes ordenamientos al variar el numero de núcleos que realizan una tarea.

La teoría de colas es el estudio de matemático de las líneas de espera dentro de un sistema. En esta práctica se estudió la fila de espera generada por las actividades que se mandaron a ejecutar y fueron atendidas por el procesador. Al utilizar el paralelismo, se aprovecha la característica incorporada a los procesadores desde hace algunos años de contar con más de un núcleo de procesamiento, asi que la fila de espera se divide entre la cantidad de servidores (núcleos del procesador) asignados y el conjunto de actividades tendría que completarse en un tiempo menor que el tiempo realizado si se utilizara solo un núcleo del procesador.

1.1. Diseño del Experimento

Para este experimento se utilizó una función para identificar si dado un número n, este es primo o no, esto lo realiza revisando el modulo generado al dividir el número n entre el número dos, y los número impares desde tres hasta $\sqrt{n}$, si el modulo es cero con algún valor, entonces el número no es primo y si para ningún valor del rango el modulo es 0, entonces el número n es primo. Este algoritmo en el peor de los casos, hace aproximadamente $\sqrt{\frac{n}{2}}$ operaciones para saber si un número es primo.

Ahora que ya se tiene la función que sera efectuada por el procesador, se generó una cola de trabajo con una cantidad de números a los cuales había que identificar si son primos o no.

Tomando como base el ejercicio en la página del curso, donde 3 series de números secuenciales fueron utilizados: ordenados de manera ascendente, ordenados de manera descendente y ordenados de manera aleatoria. Se comenzó con este ejemplo para comenzar la practica, y las series de números utilizados eran los números del 50,000 al 100,000.

Para comenzar la experimentación, se utilizó una computadora con las siguientes especificaciones, Procesador Intel Core i7-4790 CPU @ 3.6Ghz × 8 y Memoria RAM de 24 Gb. Con lo que se contaban con 7 núcleos de procesador para realizar los experimentos. Se dejó 1 núcleo para procesos del Sistema operativo y que no fuera a afectar los resultados al utilizar los 8 núcleos del procesador.

Para el experimento se utilizaron 30 iteraciones para cada experimento y el código implementado para la paralelizacion fue el de la librería parallel de R, iterando desde 1 núcleo hasta 7, y también se usó un código que se implementaba sin llamar a la librería y se usara solo 1 núcleo, para comparar el tiempo entre el código que llamaba y no llamaba a la librería.

1.2. Resultados

Los resultados obtenidos de esa experimentación son los mostrados en la figura 1. De donde se pudieron observar los siguiente puntos.

1. El tiempo requerido para terminar la tarea para un procesador que llamaba y que no llamaba a la librería fue mayor para el código que mandaba llamar a la librera parallel, esto debido al tiempo de procesamiento que toma el intentar paralelizar el código y que no es necesario porque

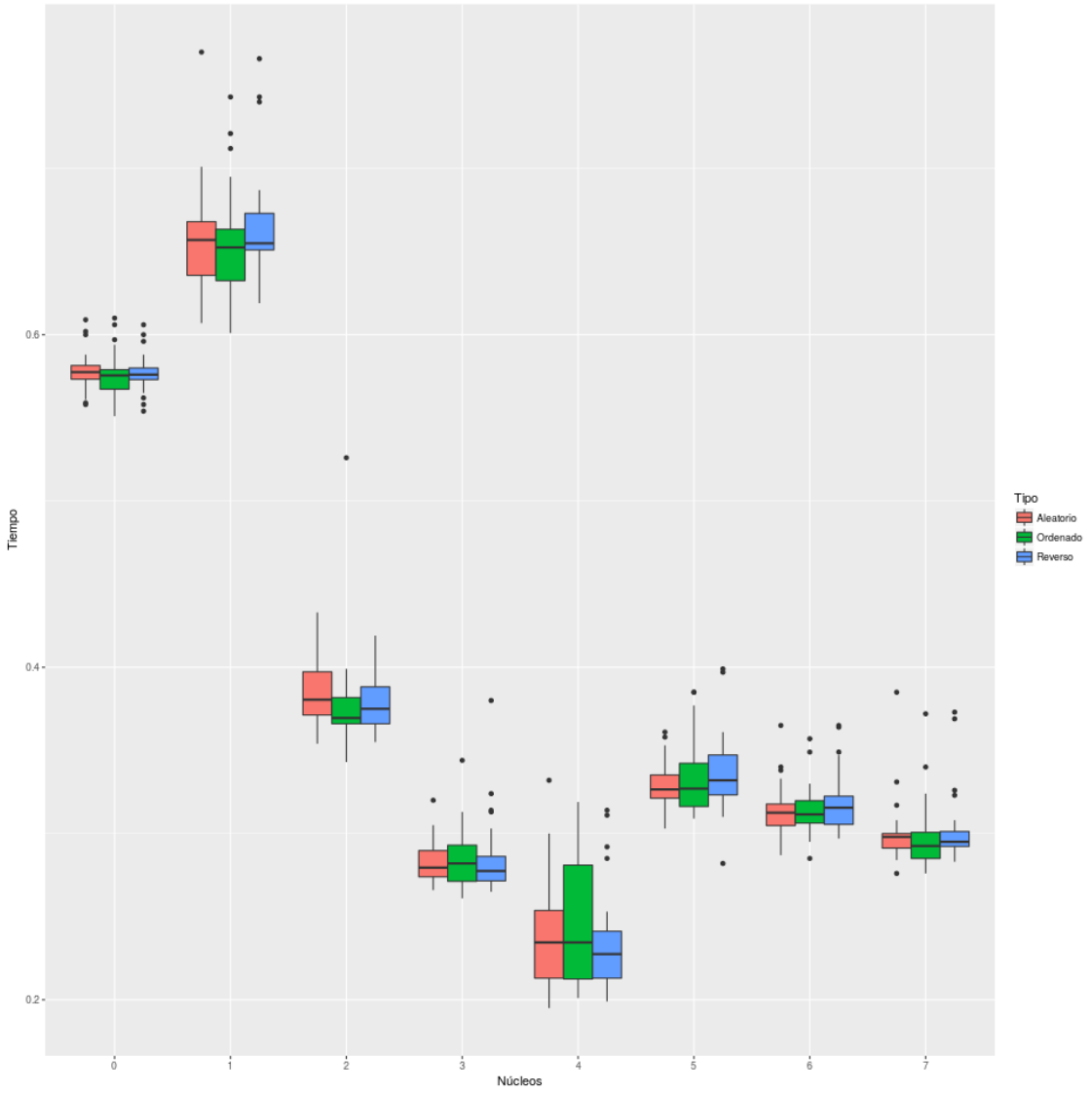

Figura 1: Tiempos alcanzados por cada tarea y cada núcleos en los primeros experimentos

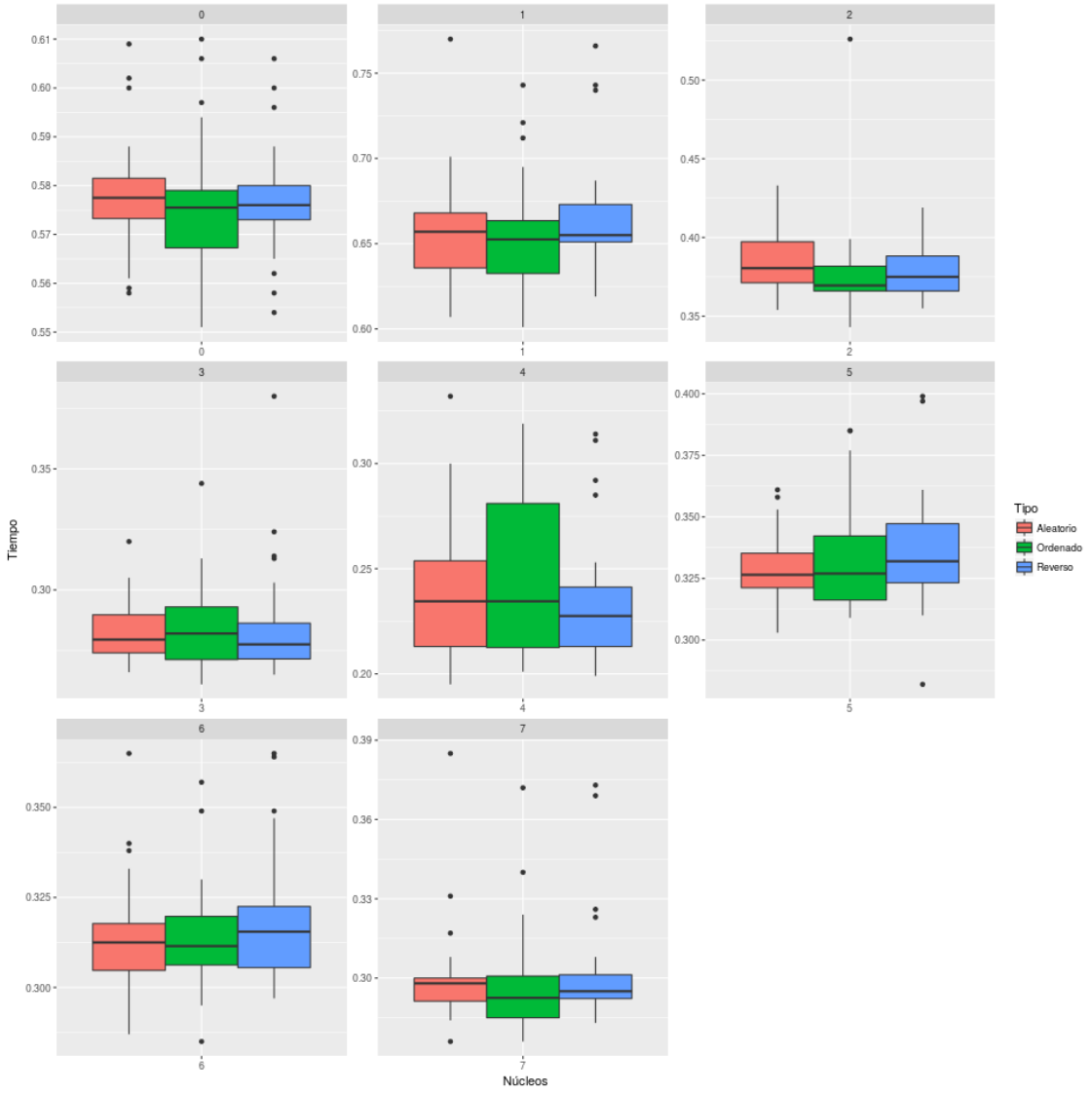

Figura 2: Tiempos alcanzados por cada tarea dividido por núcleo

solo hay 1 núcleo, mientras el código que no mandaba llamar a la librería comenzaba a laborar de inmediato, por lo que tenemos esa diferencia en los tiempos. Esta medición, sirvió también para tener como cota superior de lo que podríamos aceptar para utilizar la paralelización. Aquí cabe mencionar que se probó el código de la librería doPararllel para R, que también se utiliza para paralelizar tareas, pero por alguna razón esta implementación daba mayores tiempos que la opción iterativa de un núcleo, por lo que se descarto para ser utilizada en esta practica.

2. Para los implementaciones paralelas de más de dos núcleos, tenemos que sí hay una mejora en el tiempo de procesamiento de las tareas.
3. Se alcanzo el menor tiempo de procesamiento con cuatro núcleos utilizados, después de esto, los tiempos eran mayores que el tiempo obtenido por cuatro núcleos.
4. Con respecto al tiempo entre los tipos de tareas, tenemos que no se ven diferencias significativas entre los tipos esto lo podemos observar en la figura 2, es decir, independientemente de la tarea asignada, en la mayoría de los casos, terminaban al mismo tiempo, esto hace pensar que las tareas, es decir, las series de números, tenían la misma estructura y al dividirse entre los núcleos cada núcleo tenia carga similar.

2. Extra Uno: Argumentar causas en los tiempos de ejecución debido a los núcleos asignados

Como se pudo apreciar en el experimento anterior, hay una diferencia significativa al paralelizar con dos o más núcleos, esto debido a que al haber más núcleos disponibles las actividades se reparten entre los núcleos y se reduce el tiempo de procesamiento, sin embargo entre las tareas no hubo diferencias ¿por qué sucedió esto? pareciera que la estructura de la serie es las misma para las tres.

2.1. Diseño del Experimento

La primer hipótesis que se plantea es que la estructura de las tres series es la misma, y que al dividirse las tarea se distribuye de la misma manera entre los núcleos. en las tres series, la mitad de los valores eran divisibles entre dos y tardaban una comparación en saber que no eran primos, de la mitad restante, cerca de la tercera parte es divisible entre 3, tardando solo 2 comparaciones, etc. Esto hace que la tarea se realice en menos tiempo, pero también hace que las diferencias entre los núcleos sea homogéneas.

Lo que se planteo fue cambiar la serie para cambiar la estructura, en este caso se pensaron los siguientes casos.

1. Pares: Se utilizó un conjunto de n números pares, esperando que sea el que menos tiempo toma ya que todos los elementos los procesará con una sola operación.
2. Impares: Se utilizo un conjunto con n números impares, esperando que demore más que los casos secuenciales
3. Máximo primo: se utilizó un conjunto donde se incluye el mayor número primo menor a el mayor numero utilizado M. este conjunto debería ser el que tome más tiempo de todos, ya que cada operación realiza tarda $\sqrt{\frac{M}{2}}$ operaciones.

   Los siguientes casos se utilizaron para indagar el funcionamiento de la paralelizacion, es decir, saber si se asignan las tareas en función del primer procesador que termine una tarea o si se divide la cola en partes iguales entre los procesadores y cada procesador trabaja con su propia linea de espera.
4. Medio ascendente: este conjunto consta de la mitad de los n elementos como números pares, y la mitad otra son repeticiones del máximo primo.
5. Medio descendente: este conjunto es similar al anterior, solo que los primeros elementos son los máximo primo y la segunda mitad, números pares.
6. Medio intercalado: este conjunto contiene de igual manera, mitad de números pares y mitad de números máximo primo, pero de manera intercalada.
7. Medio intercalado 2: este conjunto contiene de igual manera, una mitad de números pares y la otra mitad de repeticiones del número máximo primo, pero en cuatro partes, primera cuarta parte de números máximo primo, la segunda cuarta parte de números pares, la tercera cuarta parte de máximo primo y la ultima parte de números pares.

Para este experimento también se hicieron 30 replicas para cada núcleo, el intervalo usado fue el mismo de 50,000 a 100,000, solo que para las series de pares y impares, se necesitan 25,000 números pares y otros tantos impares, por lo que se tomaron del intervalo de 100,000 a 150,000, y el Máximo Primo utilizado fue el más cercano a 150,000, así que todos las series eran de 50,000 elementos.

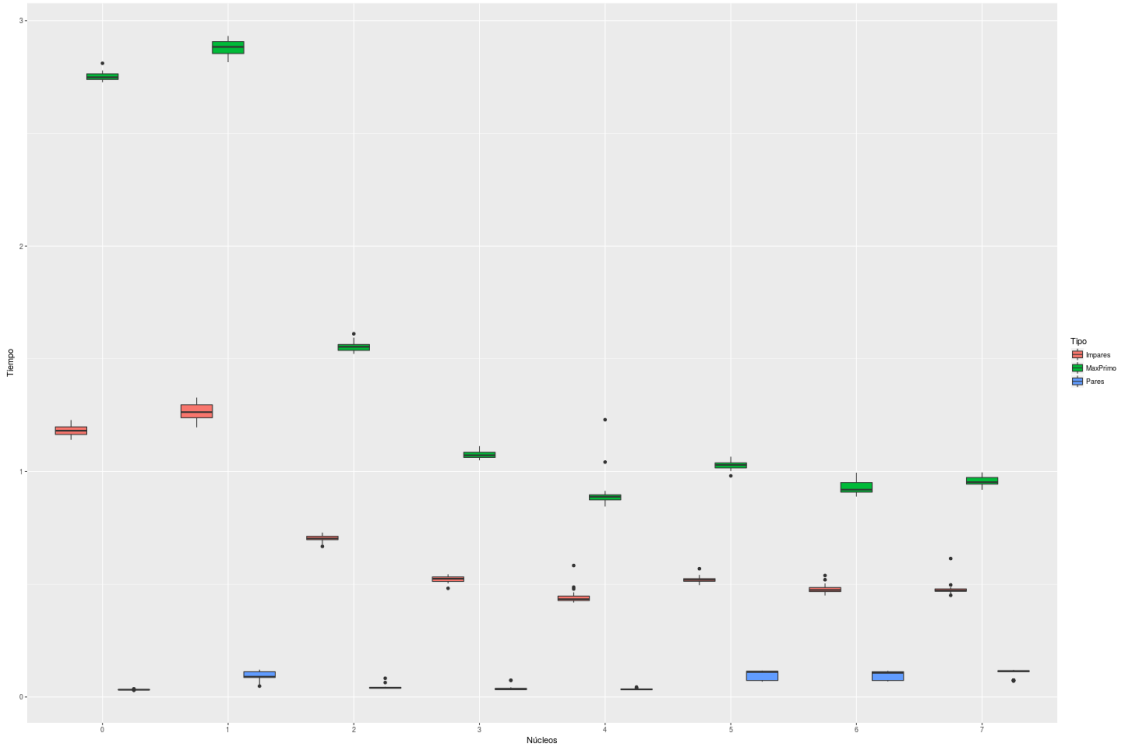

Figura 3: Tiempos alcanzados para las tareas de pares, impares y máximo primo

2.2. Resultados

En estos resultados se mantuvo lo que se pudo apreciar en la primera experimentación, la paralelizacion de cuatro núcleos dio tiempos menores, y las series de números sucesivos ordenados ascendente, descendente y aleatorios se comportaron de manera similar al primer experimento.

Las estructura añadida de números pares, dio como se esperaba, una cota inferior, aunque hubo un resultado que fue interesante, como todos los números de la serie requieren de una cantidad constante de pasos para ser clasificados y esta cantidad de pasos es muy pequeña, se vieron mejores resultados con un solo núcleo que con la paralelizacion de cinco o mas núcleos, seguramente debido al tiempo que se toma en paralelizar las tareas afecta al resultado final.

Las series de impares y de máximo primo, como también se esperaba dieron una cota media y alta de lo que se podía esperar. La serie de máximo primo que contaba con un numero constante de operaciones pero para cada actividad, pero mucho mayor a las demás, se puedó apreciar el efecto del paralelismo en esta tarea, donde también, al ser asignados más de cuatro núcleos, no se mostraba una mejora en el tiempo, estos resultados se pueden apreciar en la figura 3.

Los resultados más interesantes de este experimento fueron los que se encontraron para las series que sirvieron para ver el funcionamiento de la paralelizacion y la cola de espera que se genera para cada núcleo del procesador, estos resultados se aprecian en la figura 4.

Aquí se puede apreciar que el tiempo que demora en realizar una tarea que toma la misma cantidad de operaciones, sí se ve afecta por el número de núcleos, los casos medio ascendente y descendente, siempre tomaron tiempos similares al ser trabajados en cada uno de los núcleos, pero los dos casos intercalados mostraron siempre un tiempo de ejecución menor en comparación con los 2 anteriores, el caso mas evidente fue para dos núcleos donde el tiempo que demoró en completar los medios ascendentes y descendente son muy similares a su tiempo de un solo núcleo, es decir, si la tarea se dividió en dos mitades, una mitad con los números primos y otra con los pares, el tiempo que demoraba era el tiempo que tarda en procesar $\frac{n}{32}$ Máximo primos y este tiempo es muy similar al de un núcleo ya que tiempo de procesar pares es muy pequeño, como se vio con el ejemplo de pares, por lo que seguramente la librería parallel parte el arreglo entre el número de núcleos asignados a la tarea. De no se asi, se esperaría que los cuatro dieran resultados muy similares, ya que asignaria a un núcleo libre el siguiente de la fila, intercalando los valores entre los núcleos y reduciendo el tiempo de ejecución de la tarea para dos núcleos.

Para los cuatro casos, se vio también que se dio su mejor comportamiento con cuatro núcleos asignados, y para los casos siguientes no se vio mejora, al igual que en los experimentos anteriores.

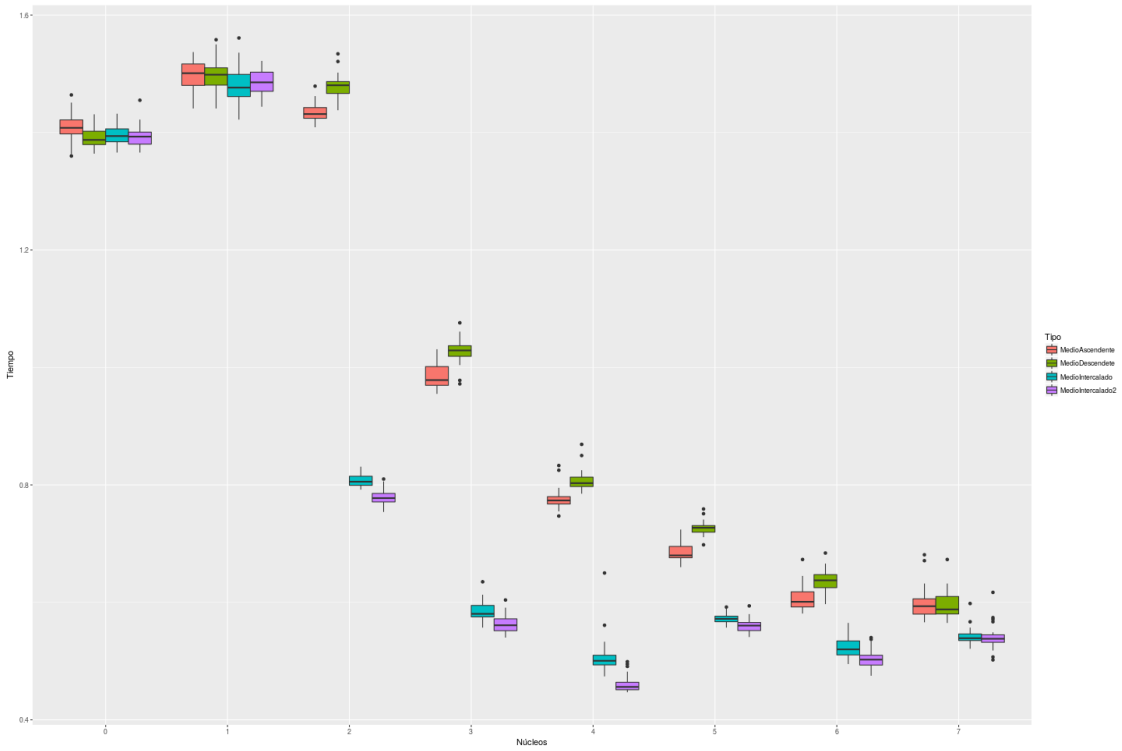

Figura 4: Tiempos alcanzados para las tareas de mitades

3. Extra Dos: Pruebas estadísticas a las series sucesivas

Aquí se hizo una prueba de hipótesis para los resultados del experimento de la tarea, donde se establece que las varianzas de los resultados obtenidos por las series de números sucesivos ordenados de manera ascendente, descendente y aleatoria son iguales en cada núcleo para más de dos núcleos.

Para poder realizar el análisis de varianza ANOVA, se necesita cumplir los supuestos de la independencia de las observaciones, tener distribución de residuales normal y tener homogeneidad de las varianzas.

Se cuenta con la independencia de los observaciones, dado la forma en que se realizo el experimento, es decir ningún experimento dependía de la ejecución u orden del anterior, ya que no hay memoria de los resultados ni tampoco entre núcleos, ya que cada uno genera una cola diferente.

Para comprobar que la distribución de residuales viene de una distribución normal se realizo una prueba de Shapiro-Wilk para cada núcleo, teniendo los resultados en la tabla 1

Núcleos	Valor p
2	6.504e-14
3	6.736e-13
4	1.968e-11
5	3.277e-09
6	2.523e-06
7	4.778e-08

Cuadro 1: Tabla de valores p de la prueba Shapiro-Wilk para cada núcleo con las 3 serie de ordenamientos

Como se puede apreciar para un valor de $\alpha = 0{,}0005$, debemos desechar la hipótesis nula de que los datos vienen de una distribución normal. Por lo tanto se procedió a la prueba paramétrica de Kruskal-Wallis para comprobar que no había varianzas entre los ordenamientos en cada nivel, los resultados se pueden ver en la siguiente tabla 2

Con los resultados de la prueba, podemos ver con un valor de $\alpha = 0{,}0005$, que solo para dos núcleos, hay diferencias en las varianzas y de tres núcleos en adelante, se puede aceptar que las varianzas son iguales para cada ordenamiento, que era lo que se observó en los resultados de la tarea.

Núcleos	Valor χ^2	Grados de libertad	Valor p
2	62.663	2	2.471e-14
3	13.079	2	0.001445
4	1.0875	2	0.5806
5	3.4505	2	0.1781
6	3.7433	2	0.1539
7	1.959	2	0.3755

Cuadro 2: Tabla de valores de la prueba Kruskal-Wallis para cada núcleo con las 3 serie de ordenamientos

4. Conclusiones

En esta práctica se trabajo el efecto del paralelismo de tareas desde la teoría de colas, en como su uso reduce el tiempo de procesamiento de las taras. Por otra parte se pudo observar que hay estructuras en las tareas que pueden afectar el tiempo de ejecución de la Tarea en función de su estructura y el número de núcleos del procesador asignados para la tarea y se puedo conocer la manera en como funciona la librería parallel para poder utilizarla de manera que se pueda aprovechar su paralelizacion.