Práctica 3

Teoría de Colas

Introducción

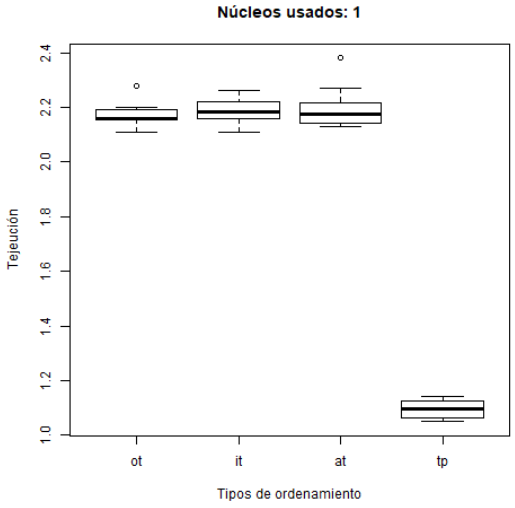
La teoría de colas es un área de las matemáticas que estudia el comportamiento de líneas de espera. Los trabajos que están esperando ejecución en un clúster esencialmente forman una línea de espera. Medidas de interés que ayudan caracterizar el comportamiento de una línea de espera incluyen, por ejemplo, el tiempo total de ejecución. En esta práctica vamos a estudiar el efecto del orden de ejecución de trabajos y el número de núcleos utilizados en esta medida.

En está practica se determinará cómo el número de núcleos afecta el tiempo de ejecución para realizar una tarea, específicamente el tiempo de un programa en calcular los números primos de una muestra de números.

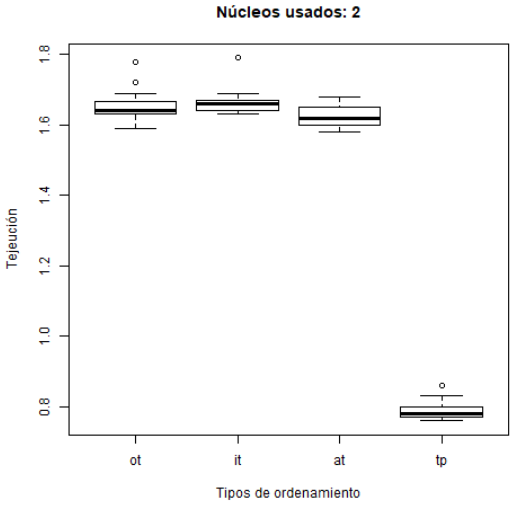
Simulación y Resultados

El orden para determinar si un número es primo o no, será por medio de cuatro órdenes el primero calculará los números primos entre un rango de valores de entre 1000-10000 en orden ascendente (ot), en orden descendente (it), en orden aleatorio (at) y por último se implementó un orden donde el programa recibe únicamente los números pares (tp) esto con el fin de facilitar las cosas para el programa y ver cómo influyen en los tiempos de ejecución. El número de réplicas para cada ordenamiento será de 20 réplicas. Se agregó un for en el programa para variar el número de clúster y realizar las réplicas para cada uno de los ordenamientos.

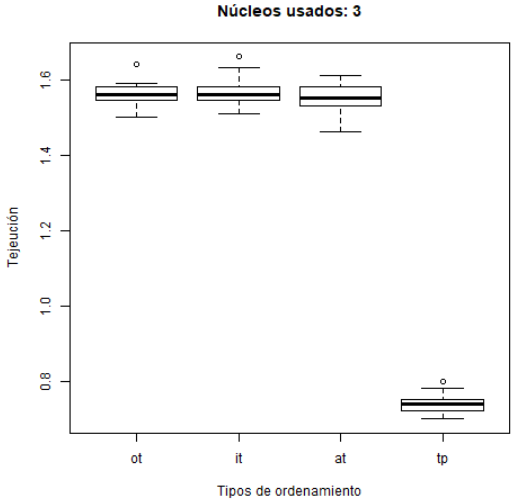
Los resultados obtenidos para los tiempos de ejecución en función del número de núcleos se muestran en la figura 1. Se observa que los tiempos de ejecución disminuyen de manera drástica al aumentar la cantidad de núcleos en donde más se puede apreciar este cambio de cuando pasamos de un solo núcleo a 2 núcleos para los siguientes incrementos en la cantidad de núcleos el cambio en el tiempo de ejecución es menos notable. En cambio para cada uno de los ordenamientos los tiempos de ejecución son muy similares a excepción del ordenamiento de números pares (tp) ya que el programa de manera casi inmediata determina que los pares no son primos ya que con ser divisible entre 2 es suficiente para comprobar que no es primo y no hay que dividir todos los enteros posibles antes del número en cuestión.


Núcleos usados: 1
Tejeución
Tipos de ordenamiento
ot
it
at
tp

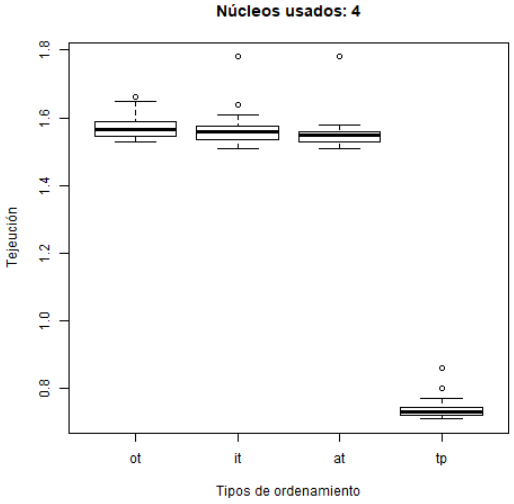
a)


Núcleos usados: 2
Tejeución
Tipos de ordenamiento
ot
it
at
tp

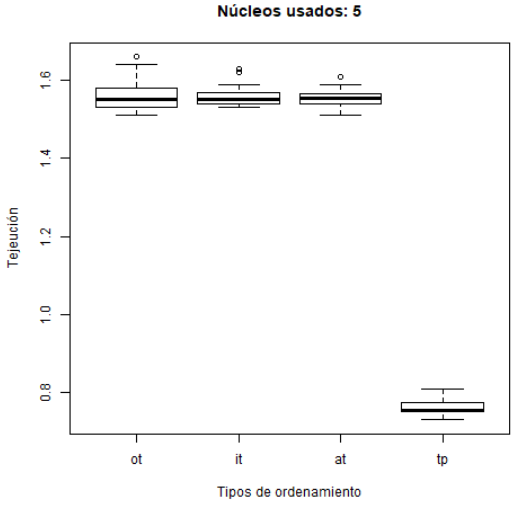
b)


Núcleos usados: 3
Tejeución
Tipos de ordenamiento
ot
it
at
tp

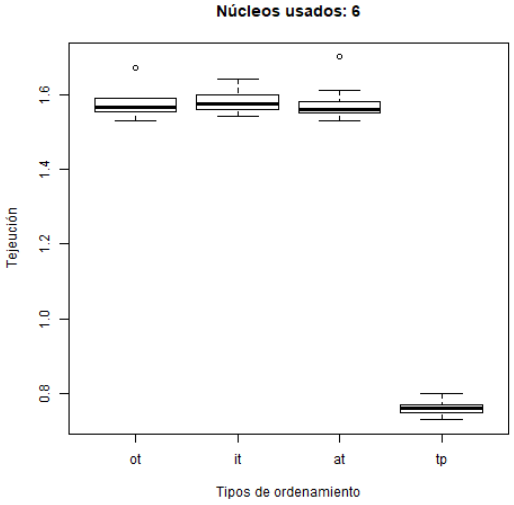
c)


Núcleos usados: 4
Tejeución
Tipos de ordenamiento
ot
it
at
tp

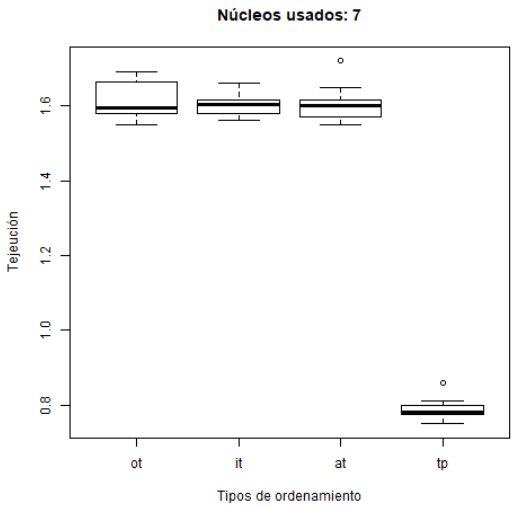
d)


Núcleos usados: 5
Tejeución
Tipos de ordenamiento
ot
it
at
tp

e)


Núcleos usados: 6
Tejeución
Tipos de ordenamiento
ot
it
at
tp

f)



g)

Figura 1. Tiempo de ejecución con distinta cantidad de núcleos para cada uno de los ordenamientos: a) 1 núcleo, b) 2 núcleos) 3 núcleos, d) 4 núcleos, e) 5 núcleos, f) 6 núcleos) 7 núcleos.

Conclusión

En conclusión, podemos observar que al aumentar el número de clúster para realizar las tareas disminuye considerablemente los tiempos de ejecución y esto es gracias a que al tener más clúster es como si se tuviera una avenida con más carriles por los cuales transitar y así evitar el tráfico en este caso de datos y así el acomodo se pueda realizar de mejor manera, restándole importancia a como vengan ordenamos los valores a acomodar aunque en algunas corridas el acomodo más eficiente parece ser el aleatorio. Sin embargo, debido a su naturaleza es muy variable el resultado.