Concurrencia y Distribución

Práctica 2 - Preguntas P3 Yeray Lage Freitas

c) Diagrama de ejecución de hilos

- ¿Puedes distinguir entre la creación del hilo, la ejecución y la terminación final?
  Para los tiempos solicitados he utilizado la función System.currentTimeMillis(), de este modo tener precisión suficiente y distinguirlos. Además, para una mayor facilidad y limpieza he creado una clase Output de la cual creo objetos con el tiempo de creación y posteriormento les pongo los de ejecución y terminación.

  Para conocer el momento de creación del hilo he decidido imprimir el valor en el momento posterior al new Thread pero anterior también al start() del mismo, ya que será justo cuando se cree pero antes de enviarlo a cola de ejecución.

  Por otro lado, para conocer cuándo éste comienza la ejecución lo hago en la primera línea del run() de la clase que implementa Runnable, ya que será la que ejecute el hilo cuando se le den los recursos.

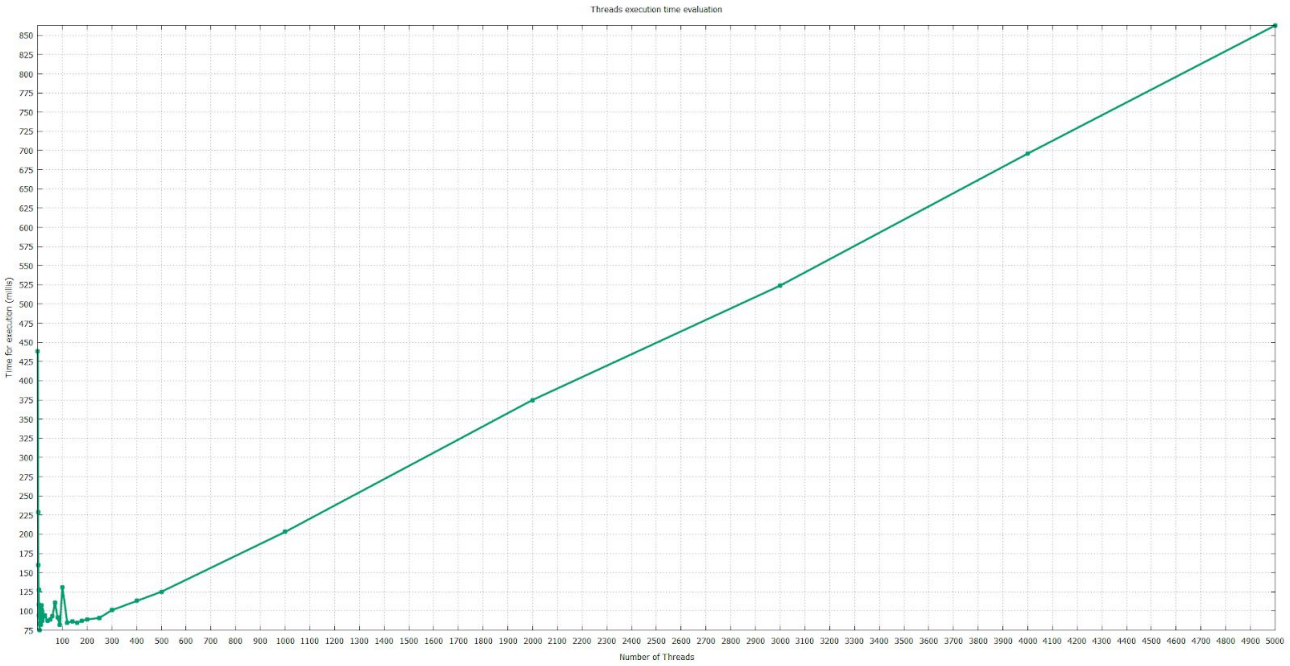
  Y finalmente, dado que utilizo join() para detectar cuándo todos los hilos finalizan, pensé que podría detectar ahí cuándo el hilo finaliza, pero si por un casual el hilo 1 termina más tarde que el 7, el join() esperaría a que acabe el 1 para preguntar a los demás y el valor de finalización del 7 sería falseado. Por lo tanto, el valor de finalización lo obtengo en la última línea del run(), ya que aproximadamente es lo que hará antes de finalizar su trabajo.

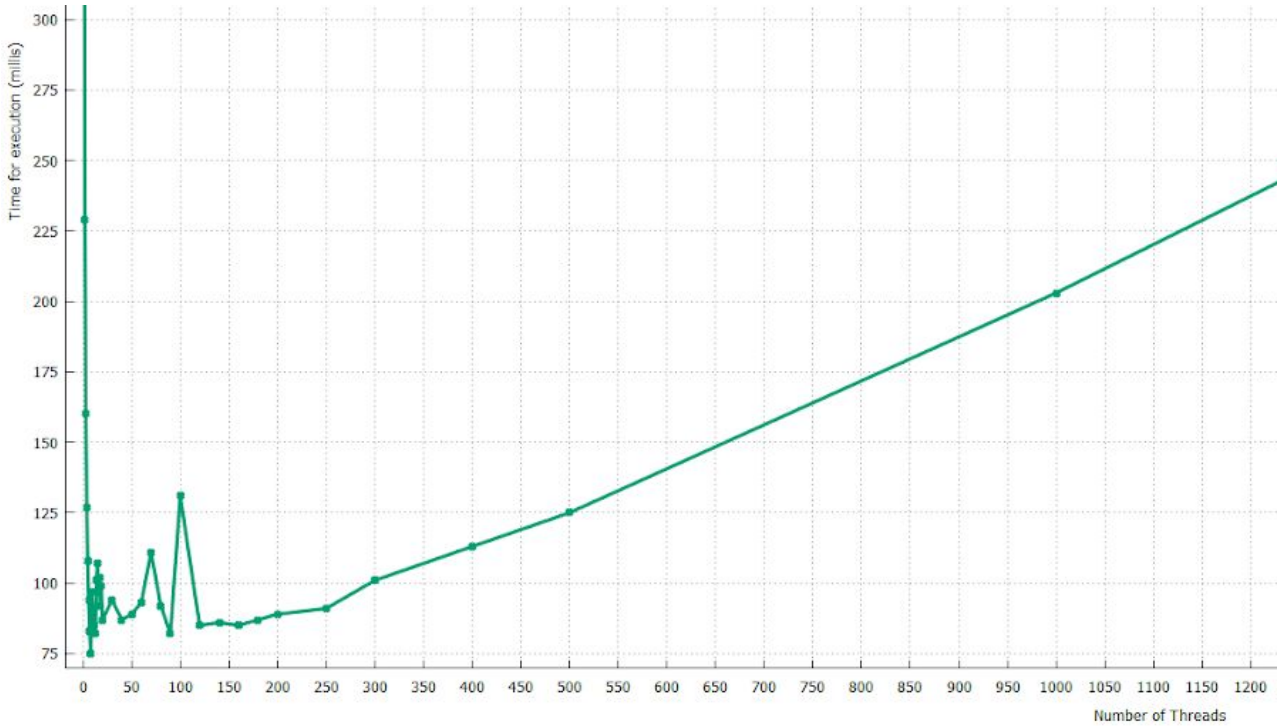
- ¿Qué problemas encuentras cuando intentas determinar las diferentes fases?
  El principal “problema” sería que aunque pueda parecer que en el momento que se llama a .start() sería cuando comienza la ejecución del hilo, esto no es correcto. El .start() envía el hilo a la cola de ejecución y dicha ejecución comenzará en un momento en el fututro desde la llamada a .start(). Por ello, lo obtento en el .run().

e) Haz un gráfico de tiempo de ejecución

Teniendo en cuenta los datos generados con el script generateStatistics.sh que ejecuta el código de Main.java con diferente número de hilos, he podido crear las gráficas solicitadas que explicaré a continuación.

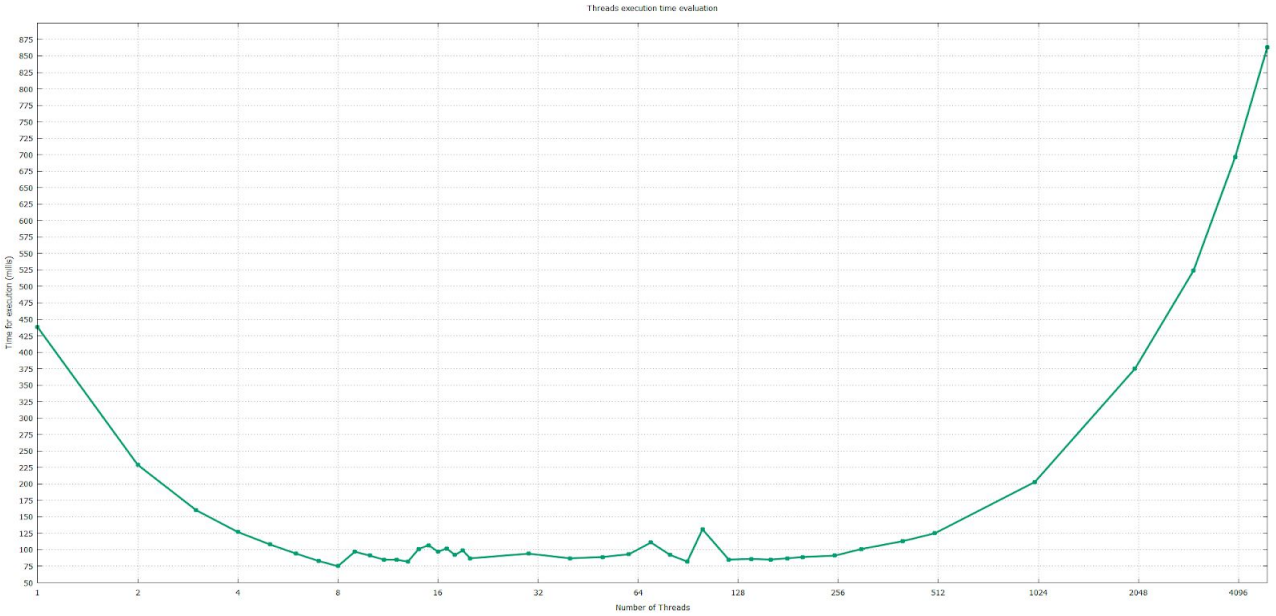
Gráfica lineal





Como podemos ver en estas dos gráficas (la primera segunda con zoom), los valores son bastante estables entre 75 y menos de 150 millis al incio, elevándose sin fin a partir de 150 threads. Con esto podemos comprobar claramente que no por mayor número de threads el resultado será mejor. Hay que buscar un equilibrio que veremos más tarde.

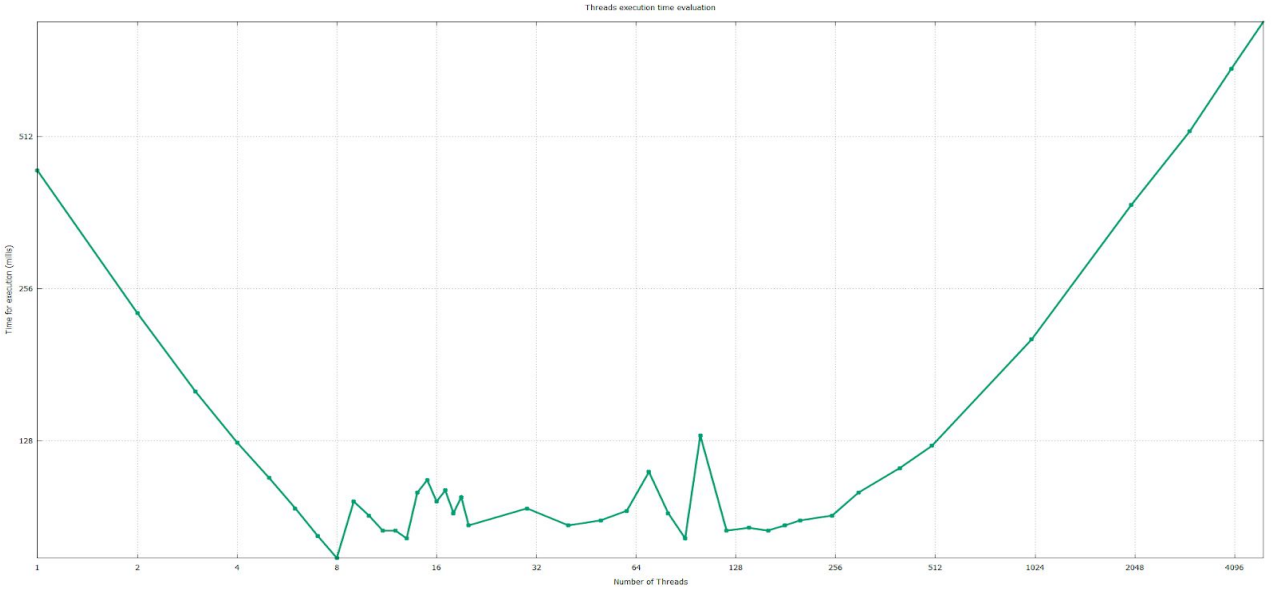
Gráfica semilogarítmica



En esta gráfica podemos ver com mayor claridad lo dicho anteriormente, se puede ver que la variación en los valres bajos de hilos es mucho mayor que posteriormente, dándonos valores óptimos con pocos hilos, entre 4 y 128, pero con un valor claramente mínimo al usar 8 hilos.

Este valor no es para nada aleatorio, ya que con 8 hilos hayamos el valor mínimo gracias a que son el número de procesadores lógicos del ordenador utilizado. También podemos ver que con valores de hilos muy bajos (1, 2, 3) el tiempo es mucho mayor, ya que no repartimos el trabajo entre los procesadores disponibles.

Gráfica log-log



Al igual que en las otras gráficas, podemos sacar las mismas conclusiones pero viendo los resultados de un modo más claro dado al cambio de escala.

Conclusiones finales

Como se dijo anteriormente, las conclusiones que se pueden sacar del experimento son primero que no por mayor número de threads conseguiremos un mayor rendimiento, y que el valor más optimo será aproximadamente con la misma cantidad de threads que de procesadores.

Pero además, podemos saber que el uso de threads para nuestro programa no ser la mejor opción, y esto dependerá del equipo en el que vayamos a utilizar el programa. Por ello, deberemos valorar individualmente en cada caso si el uso de los threads es recomendable y con cuántos como máximo deberemos hacerlo.