Hilos para Multiplicación de matrices y Resolución de un sistema de ecuaciones lineales

Students:

Anthony Daniel, Bautista León
Katheryn Ximena, Peralta Haro
Renzo Renato, Quispe Amao
Universidad Nacional de Ingeniería, Facultad de Ciencias,
e-mail: abautistal@uni.pe, katheryn.peralta.h@uni.pe, rrquispea@uni.pe

Curso:

CC4P1 Programación Concurrente y Distribuida
Laboratorio 01

Abstract

El objetivo de este trabajo es familiriazarnos con el uso de Hilos *(Thread)* sincronizados en la máquina virtual de java (JVM). Para ellos se plantean dos casos: Multiplicacón de Matrices y Resolución de un sistema de ecuaciones lineales. En el primer caso se logró una mejora a medida que aumentamos el número de hilos. Lo cual demuestra la ventaja de utilizar hilos para disminuir el tiempo de ejecución para problemas con complejidad O(n^2). En el segundo caso , el algoritmo iterativo Jacobi para resolver sistemas de ecuaciones lineales, no mejora cuando se tiene sistemas de menos de 1000 variables, el algoritmo tiende a aumentar el tiempo de ejecución y demuestra que no siempre trabajar con hilos significará una mejora en el rendimiento.

Keywords: Thread, JVM, Jacobi, complejidad, método iterativo.

Contents

---

1 Introducción

En la vida diaria, realizamos varias tareas al mismo tiempo. Asi mismo, en la programación también podemos ejecutar varios programas concurrente e independientemente.
Una base de los SO es compartir su conjunto finito de recursos entre los múltiples procesos. Cada proceso corre en su propio espacio de direcciones que evita que los procesos interfieran entre sí.
Esto se ha trabajado desde los 60's 70's y Java tomo varios de estos conceptos y creó los llamados *thread* o hilos.
Una aplicación Java puede constar de varios hilos de control, cada uno corriendo concurrentemente con otros. Todas las aplicaciones en Java tiene al menos un hilo de control llamado main, en el cual se ejecuta el método principal. Un hilo individual ejecuta sus instrucciones en orden secuencial una después de otras.

2 Marco Teórico

La Máquina Virtual Java (JVM) es capaz de manejar multihilos, es decir, puede crear varios flujos de ejecución simultánea, administrando los detalles como asignación de tiempos de ejecución o prioridades de los hilos.
Java proporciona soporte para hilos a través de una interfaz y un conjunto de clases. La interfaz de Java y las clases que proporciona algunas funcionalidades sobre hilos son:

- *Thread*
- *Runnable* (interfaz)
- *ThreadDeath*
- *ThreadGroup*
- *Object*

Tanto las clases como la interfaz son parte del paquete básico de java (*java.lang*).

Clase *Thread*

Es la clase en Java responsable de producir hilos funcionales para otras clases. Para añadir la funcionalidad de hilo a una clase solo se hereda de ésta.
La clase *Thread* posee el método *run*, el cual define la acción de un hilo y , por lo tanto, se conoce como el cuerpo del hilo. La clase *Thread* también define los métodos *start* y *stop*, los cuales permiten inciar y detener la ejecución del hilo.
Por lo tanto, para añadir la funcionalidad deseada a cada hilo creado es necesario redefinir el método *run*. Este método es invocado cuando se inicia el hilo. Un hilo se inicia mediante una llamada al método *start* de la clase *Thread*.

Interfaz *Runnable*

La interfaz *Runnable* permite producir hilos funcionales para otras clases. Para añadir la funcionalidad de hilo a una clase por medio de *Runnable*, solo es necesario implementar la interfaz.
La interfaz *Runnable* proporciona un método alternativo al uso de la clase *Thread*, para casos en los que no es posible hacer que la clase definida herede de la clase *Thread*.
Las clases que implementan la interfaz *Runnable* proporcionan un método *run* que es ejecutado por un objeto *Thread* creado. Ésta es una herramienta muy útil y, a menudo, es la única salida que se tiene para incorporar hilos dentro de las clases.

3 Metodología

Hay dos formas de crear hilos: crear un objeto de la clase *Thread* o bien extendiendo la interfaz *Runnable*. Todos los hilos son instancias de la clase *Thread* por lo tanto lo natural es crearlos extendiendo dicha clase. El método *run* es el punto de entrada a la ejecución del hilo, debe sobreescribirse en las subclases.
La ejecución del hilo al terminar la ejecución secuencial de las instrucciones en el método *run*.

```
publicc class miHilo extendes Thread {
    public void run () {
    // Cuerpo
    }
    // Otros métodos
}
```

Para inciar un nuevo hilo definido en miHilo, se debe crear una instancia de esta clase y llamar al método start(), el cual invoca indirectamente a run.

4 Desarrollo de la Investigación

4.1 Algoritmo de multiplicación de matrices

Este algoritmo multiplica dos matrices A y B de dimensiones $n*m$ y $m*p$ respectivamente. El resultado es una matriz $n*p$. El algoritmo multiplica una fila de la matriz A con una columna de la matriz B obteniendo una componente de una matriz C.

$$c_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} \tag{1}$$

Para utilizar hilos en el algoritmo de multiplicación de matrices se creo una clase llamada HiloMatriz que hereda de Thread. Cada HiloMatriz va a multiplicar un conjunto de filas de la matriz A con todas las columas de la matriz B.

4.2 Algoritmo iterativo Jacobi

El algoritmo iterativo es usado para hallar una solución aproximada de un sistema de ecuaciones de la forma $AX = b$. El método de Jacobi consiste en usar una fórmula como iteración desde un punto fijo dado. La formula viene de la descomposición de la matriz del sistema A en una matriz Diagonal D y una matriz R , $A = D + R$. El sistema queda expresado como:

$$x^{(k+1)} = D^{-1}(b - Rx^{(k)}) \tag{2}$$

Donde k es el contador de iteración. Finalmente podemos expresar el $x^{(k+1)}$ con la siguiente fórmula.

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}), i = 1, 2, 3... \tag{3}$$

Este algoritmo iterativo tiene una pequeña porción de instrucción que se pueden trabajar paralelamente. Así que construimos HiloJacobi que hereda de Thread, al igual que en multiplicación de matrices opera sobre un pequeño conjunto de filas de la matriz del sistema a resolver.

5 Resultados y Discusión

5.1 Resultado

5.1.1 Multiplicación de Matrices

Para las pruebas del algoritmo distribuido en hilos se trabajó con matrices cuadradas de dimensiones 500, 1000, 1500 y 2000 con pruebas con 2, 4, 6 y 8 hilos.

Figure 1: Prueba con matrices cuadradas de dimension 500.

Figure 2: Prueba con matrices cuadradas de dimension 1000.

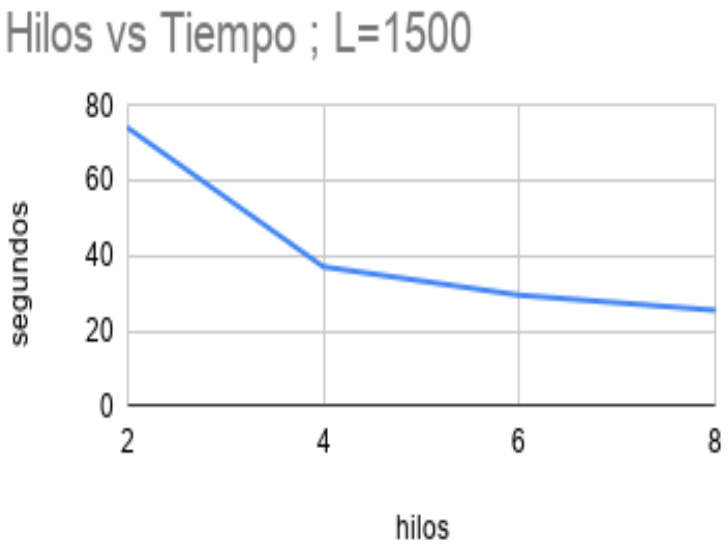

Figure 3: Prueba con matrices cuadradas de dimension 1500.

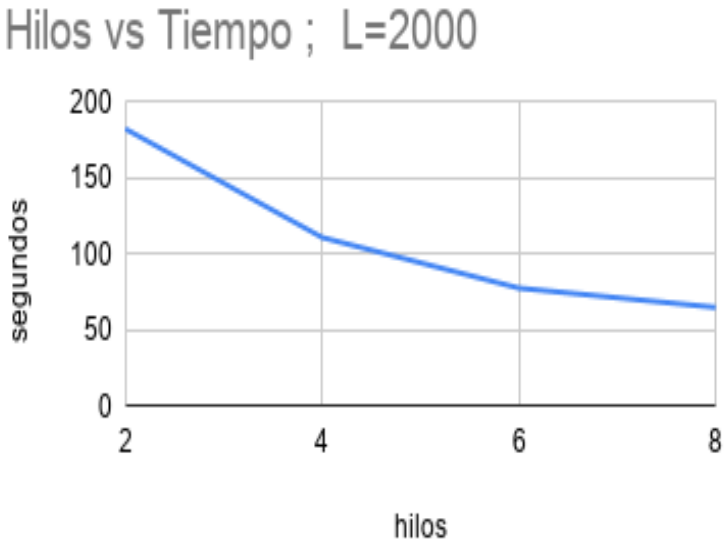

Figure 4: Prueba con matrices cuadradas de dimension 2000.

5.1.2 Método de Jacobi

Para las pruebas del algoritmo distribuido en hilos se trabajo con 500, 1000, 2000, 5000, 10000 variables y con 2, 4, 6, 8 hilos.

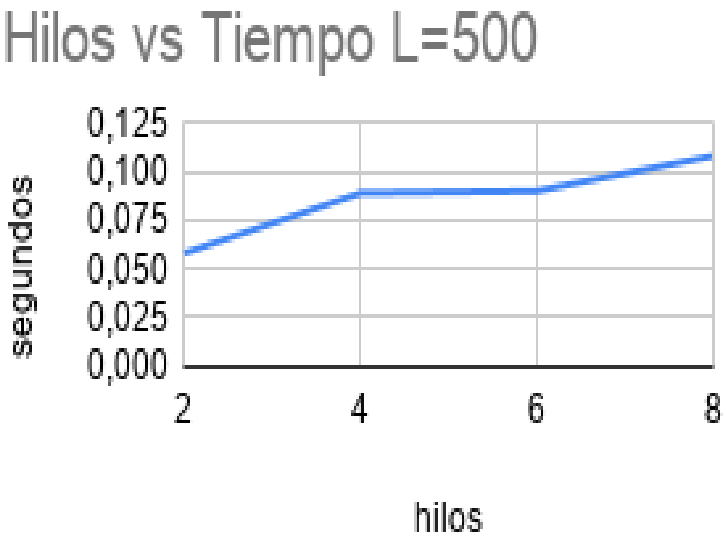

Figure 5: Prueba con un sistema de 500 variables.

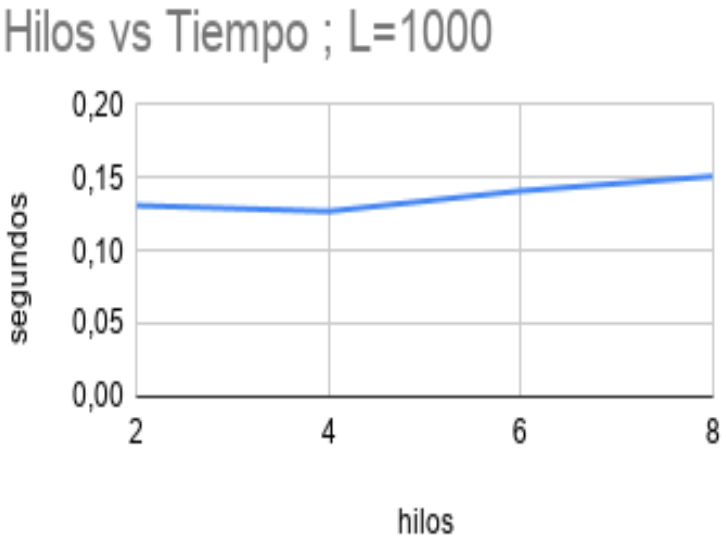

Figure 6: Prueba con un sistema de 1000 variables.

Figure 7: Prueba con un sistema de 2000 variables.

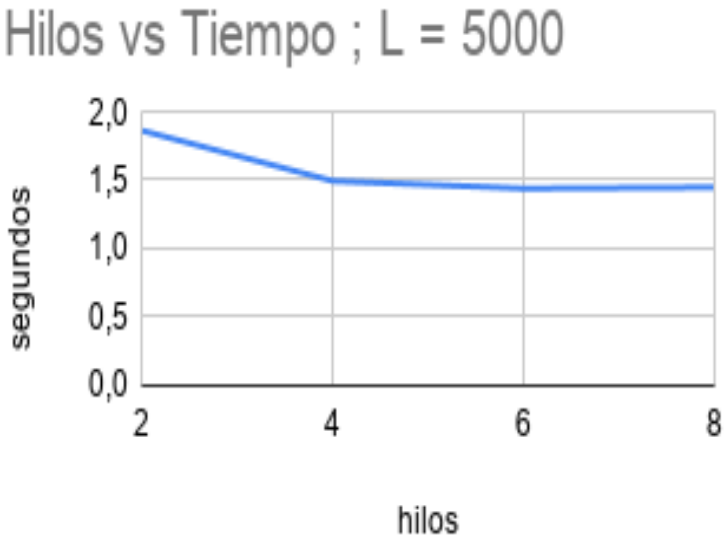

Figure 8: Prueba con un sistema de 5000 variables.

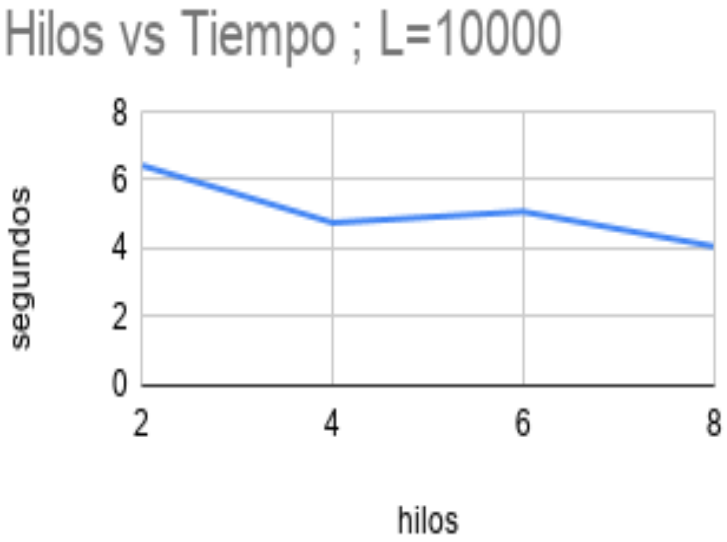

Figure 9: Prueba con un sistema de 10000 variables.

5.2 Discusiones

- La mejor relación de costo/beneficio para la multiplicación y el método iterativo es 4 y 6 hilos respectivamente.
- Para el método de Jacobi el número de incógnitas debe ser mayor a 1000 para que la paralelización sea rentable (Figura 5).

6 Conclusiones

- Es recomendable utilizar hilos para disminuir el tiempo de ejecución para algoritmos como la multiplicación de matrices.
- Utilizar hilos en porciones de código que sean independientes entre si.
- El algoritmo iterativo Jacobi para sistemas de ecuaciones lineales disminuye su tiempo de ejecución al aumentar los hilos y la cantidad de variables.
- La velocidad del programa que trabaja con memoria compartida no solo depende de la cantidad de hilos que se asigne. La mejor relación costo/beneficio de la multiplicación de matrices (6 hilos) y la resolución del sistema de ecuaciones lineales (4 hilos) se limitó a la lectura y escritura simultánea que realizaban los hilos.

7 Anexo Codigo

`https://github.com/renzoqamao/CC4P1-ProgramacionConcurrente-Distribuida`

8 Anexo Documentación

- `https://es.wikipedia.org/wiki/M%C3%A9todo_de_Jacobi`
- `http://profesores.elo.utfsm.cl/~agv/elo330/2s10/lectures/Java/threads/JavaThreads.html#:~:text=En%20Java%20un%20hilo%20es,algo%20similar%20a%20exec().`