RENDIMIENTO DE LA MULTIPLICACIÓN DE MATRICES (SECUENCIAL VS HILOS)

Kevin Alexander Moreno, Cristian Camilo Holguin, Edward Narvaez
kevin_utp@utp.edu.co, cris.635@utp.edu.co, edwardnarvaez@utp.edu.co

High Performance Computing
Ingeniería de Sistemas y Computación
Universidad Tecnológica de Pereira
2019

RENDIMIENTO DE LA MULTIPLICACIÓN DE MATRICES

(Secuencial VS Hilos)

Índice **Pág.**

RENDIMIENTO DE LA MULTIPLICACIÓN DE MATRICES

(Secuencial VS Hilos)

1. Introducción

El presente informe pretende evaluar el rendimiento en tiempo de ejecución del algoritmo tradicional de multiplicación de matrices tanto de forma secuencial (serial) cómo utilizando hilos (threads), con el fin de determinar con cual se obtiene un mejor desempeño y analizar los resultados para conocer a qué se debe el comportamiento obtenido.

En la ejecución en serie o secuencial cada una de las operaciones (instrucciones) realizadas se ejecutan en un único hilo principal, en un solo procesador, una a una, por tanto es posible que no se aprovechen toda la capacidad de procesamiento de la CPU si este cuenta con varios núcleos. Por su parte, en la ejecución concurrente, el programa tiene un hilo principal que se divide en múltiples hilos, que se ejecutan a su vez en los hilos del procesador, de esta forma se mejora el rendimiento de los procesos. Los resultados se analizaran comparando el rendimiento en tiempo de CPU y de reloj de la multiplicación de matrices en serie con la multiplicación de matrices utilizando hilos.

2. Algoritmo utilizado

El algoritmo utilizado en el kernel del programa es el tradicionalmente utilizado para realizar la multiplicación de matrices. Aunque no es el algoritmo más rápido garantiza estabilidad numérica en los resultados.

Para N = 2

Matriz A			**Matriz B**			**Matriz C**	
2	2	x	3	4	=	2*3+2*3	2*4+2*4
1	1		3	4		1*3+1*3	1*4+1*4

El número de multiplicaciones es 2^3 = 8
El número de sumas es 2^2 * (2-1) = 4

Para N = 3

Matriz A				**Matriz B**				**Matriz C**		
1	1	1	x	4	5	6	=	1*4+1*4+1*4	1*5+1*5+1*5	1*6+1*6+1*6
2	2	2		4	5	6		2*4+2*4+2*4	2*5+2*5+2*5	2*6+2*6+2*6
3	3	3		4	5	6		3*4+3*4+3*4	3*5+3*5+3*5	3*6+3*6+3*6

El número de multiplicaciones es 3^3 = 27
El número de sumas es 3^2 * (3-1) = 18

Para N = n
Las multiplicaciones totales son n^3
Las sumas totales son n^2 * (n-1)
Por tanto, el total de operaciones del algoritmo es: (n^3) + (n^2 * (n-1))

Código del kernel del programa en C++ para multiplicar matrices

```
for( i = 0; i < N; i ++ )
    for( j = 0; j < N; j ++)
        for( k = 0; k < N; k ++ )
            C[i][j] += A[i][k] * B[k][j];
```

3. Granularidad

Para la ejecución concurrente de la multiplicación de matrices A x B utilizando hilos, se crearon tantos hilos como filas contiene la matriz resultante C. De esta forma cada hilo se encarga de realizar las operaciones entre una fila **i** de la matriz A y todas las columna **j** de la matriz B.

Para el caso de la matriz de N = 3

Matriz A				**Matriz B**				**Matriz C**		
1	1	1		4	5	6		1*4+1*4+1*4	1*5+1*5+1*5	1*6+1*6+1*6
2	2	2	x	4	5	6	=	2*4+2*4+2*4	2*5+2*5+2*5	2*6+2*6+2*6
3	3	3		4	5	6		3*4+3*4+3*4	3*5+3*5+3*5	3*6+3*6+3*6

El primer hilo del programa se encarga de las operaciones resaltadas en amarillo, entre la fila 1 de la matriz A y todas las columnas de la matriz B, para obtener la fila 1 de la matriz C.

Matriz A				**Matriz B**				**Matriz C**		
1	1	1		4	5	6		1*4+1*4+1*4	1*5+1*5+1*5	1*6+1*6+1*6
2	2	2	x	4	5	6	=	2*4+2*4+2*4	2*5+2*5+2*5	2*6+2*6+2*6
3	3	3		4	5	6		3*4+3*4+3*4	3*5+3*5+3*5	3*6+3*6+3*6

Así también, el segundo hilo del programa se encarga de las operaciones resaltadas en verde, para obtener la fila 2 de la matriz C.

La granularidad utilizada permite que cada hilo pueda ejecutarse de forma independiente, sin alterar los resultados de las operaciones realizadas concurrentemente por los demás hilos. Sin embargo, si el tamaño de las matrices es muy grande se crea una cantidad de

hilos que serán detenidos temporalmente, debido a que el quantum del proceso principal debe dividirse entre un mayor número de hilos y dependiendo de la prioridad dinámica del hilo asignada por el sistema operativo puede tardar más tiempo en completar las instrucciones. Se analizará si utilizando múltiples hilos se obtiene un mejor rendimiento que únicamente uno solo.

4. Complejidad

Se calculó que el número de operaciones realizadas para multiplicar dos matrices de tamaño $n_x n$ de forma secuencial es:

$(n^3) + (n^2 * (n-1))$

Por lo tanto la complejidad del algoritmo de multiplicación de matrices es $O(n^3)$ y esto se debe a que cada uno de los n^2 elementos de la matriz C lleva tiempo O(n) en computarse.

5. Especificaciones del equipo

El equipo en el que se ejecutó el programa tiene las siguientes especificaciones:

Procesador: Intel Core i5 6200U 2.80 GHz (2 Núcleos)
Memoria: 8 GB DDR3 1600 MHz
GPU Integrada: Intel HD Graphics 520
GPU: NVIDIA GeForce 940M 2GB DDR3
Disco: 1 TB HDD 5400 RPM
OS: Debian GNU/Linux 10 (buster) x86_64

La ejecución se realizó con el sistema operativo en modo consola para evitar que los procesos del entorno de escritorio (Gnome) y del sistema consumieran recursos adicionales.

6. Resultados y análisis

Se tomaron los tiempos para valores de N = 100, 250, 500, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 7000, que se repitieron 10 veces, tanto para el kernel de ejecución secuencial cómo el kernel de ejecución por hilos. Para cada ejecución se almacenó el tiempo de CPU y el tiempo de reloj para calcular el promedio de los tiempos para cada valor de N. Los promedios se registraron en las tablas 1 y 2.

Promedio Tiempos - Kernel del programa Secuencial											
N	**100**	**250**	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**4000**	**5000**	**7000**
Tiempo reloj(s)	0.00525946	0.08014456	0.7702397	7.532103	28.52905	88.53015	180.0475	312.7728	719.3652	1309.141	3684.005
Tiempo CPU (s)	0.0052537	0.0801262	0.7701306	7.530666	28.52503	88.52241	180.0261	312.7358	719.204	1308.983	3683.842

Tabla 1. Promedio Tiempos - Secuencial

En la ejecución secuencial el tiempo de reloj es prácticamente el mismo tiempo de CPU (tabla 1), por su parte en la ejecución por hilos el tiempo de CPU es en promedio 3,910937 veces mayor que el tiempo de reloj (ver tabla 3). Esto indica que el kernel de multiplicación de matrices secuencial utiliza únicamente un hilo de un solo núcleo del procesador para realizar las instrucciones y el kernel que utiliza hilos mediante la librería `pthread` de C/C++ distribuye las instrucciones entre los dos núcleos del procesador (núcleos disponibles), y a su vez en los dos hilos de cada núcleo; debido a esto el tiempo de CPU es casi 4 veces mayor que el tiempo de reloj.

Promedio Tiempos - Kernel del programa por Hilos											
N	**100**	**250**	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**4000**	**5000**	**7000**
Tiempo reloj(s)	0.005083755	0.04887562	0.3896952	3.958441	15.44685	39.64069	80.94272	141.8395	371.8211	803.006	3619.247
Tiempo CPU (s)	0.0173282	0.1815802	1.537078	15.78128	61.63785	158.3602	323.0149	566.8045	1486.3766	3210.643	14468.895

Tabla 2. Promedio Tiempos - Hilos

Diferencia Tiempo CPU/ Tiempo de Reloj - Kernel del programa por Hilos											
N	**100**	**250**	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**4000**	**5000**	**7000**
Diferencia	3,408544	3,715149	3,944308	3,986741	3,990318	3,994890	3,990660	3,996098	3,997558	3,998280	3,997764
Promedio	**3,910937**										

Tabla 3. Diferencia Tiempo CPU / Tiempo de Reloj - Hilos

Respecto a la diferencia entre el tiempo de CPU del kernel secuencial y el tiempo de CPU del kernel por hilos, no se evidencia un patrón entre la diferencia entre ambos, sin embargo el tiempo de CPU por hilos crece mucho más rápido a medida que crece el valor N de las matrices (ver gráfico 2), debido a que utiliza todos los núcleos y subprocesos del procesador y esto se ve reflejado en la disminución del tiempo de reloj del kernel por hilos (ver gráfico 1).

6.1 Rendimiento con la granularidad utilizada:

Cabe resaltar que los tiempos de CPU para el para el tamaño de las matrices de N = 7000, fue de 14468.895 s para el kernel por hilos y de 3683.842 s para el kernel secuencial, pero la diferencia entre el tiempo de reloj entre ambos fue tan sólo 65 segundos aproximadamente. Creemos que este comportamiento se debe a que se crearon 7000 hilos entonces el consumo de recursos del kernel por hilos fue mucho mayor y no se obtuvo

mejores resultados que el kernel secuencial, como sí ocurrió con valores más pequeños del tamaño N de las matrices.

Entonces es recomendable utilizar una estrategia diferente a crear el mismo número de hilos como filas tiene la matriz resultante, para obtener un mejor rendimiento con matrices de gran tamaño.

6.2 Gráficas de Tiempos:

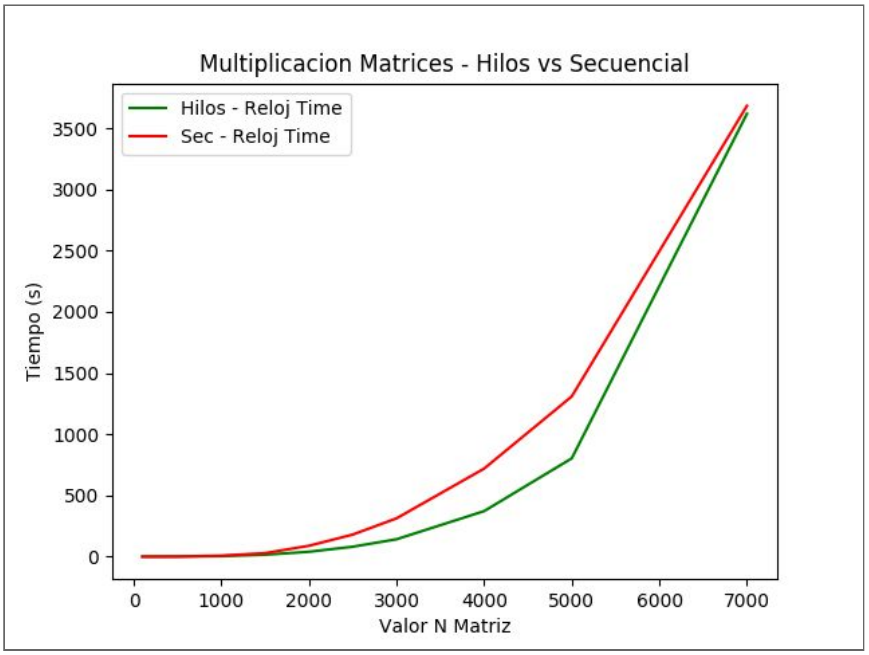


Gráfico 1. Tiempos de Reloj de multiplicación de matrices.

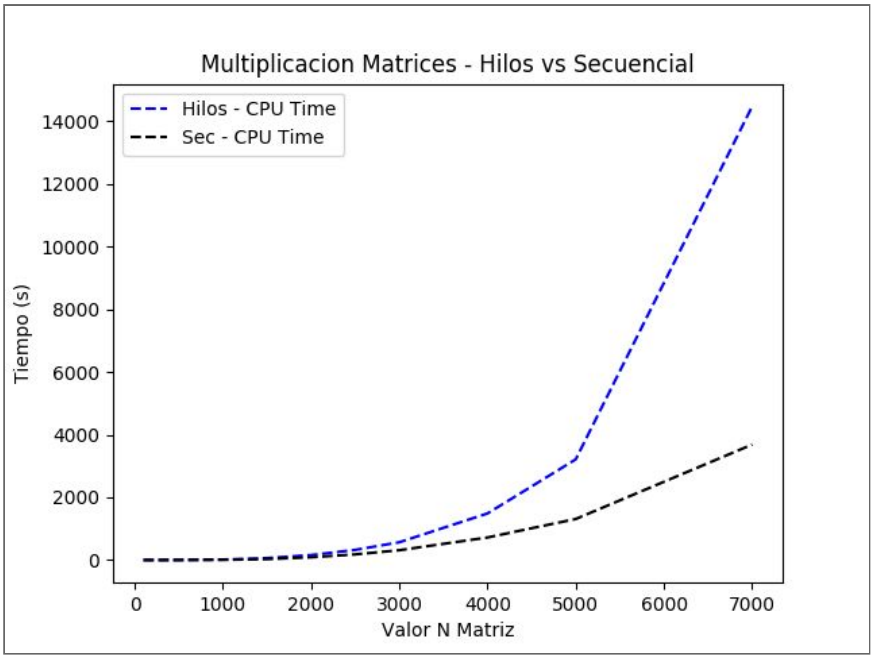


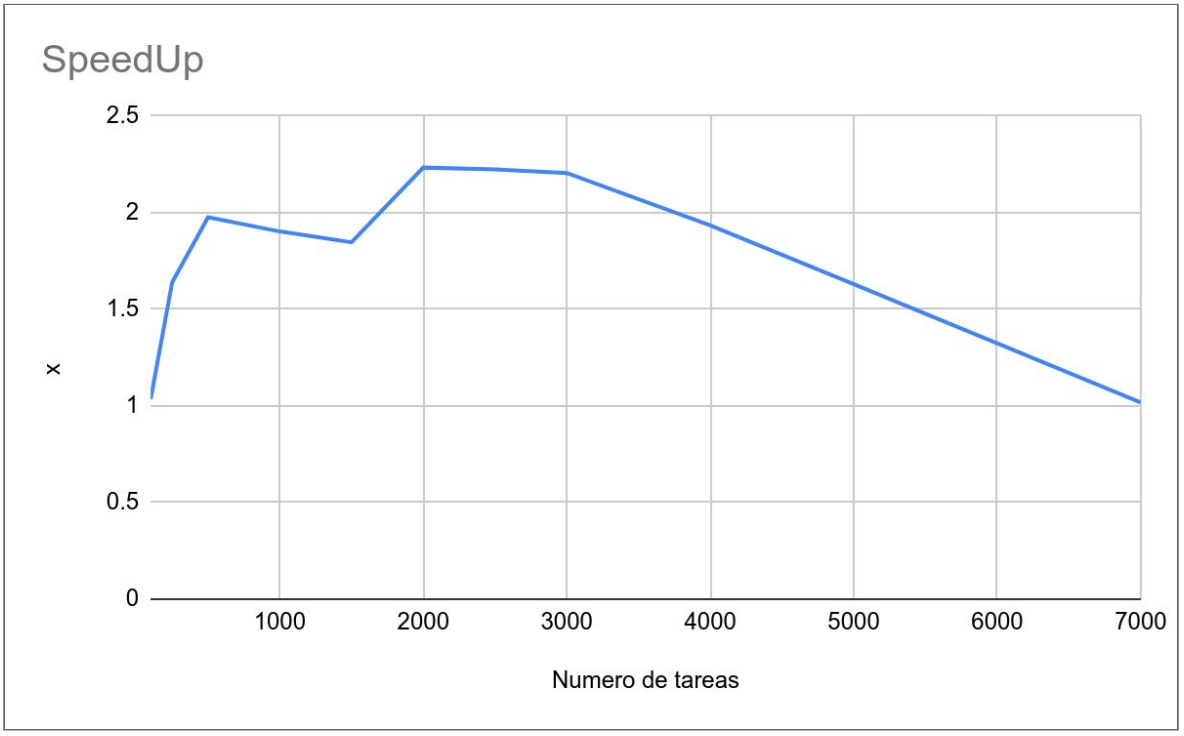
Gráfico 2. Tiempos de CPU de multiplicación de matrices.

Los programas para realizar las multiplicaciones de matrices de forma secuencial y por hilos se realizaron en C++ utilizando la librería `pthreads`, y las gráficas se realizaron en python utilizando la librería `matplotlib`.

El código fuente se puede consultar en:
https://github.com/kevin4lexander/HighPerformanceComputing/tree/master/Taller1

7. SpeedUp

Diferencia Tiempo de Reloj Secuencial / Tiempo de Reloj por Hilos											
N	**100**	**250**	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**4000**	**5000**	**7000**
Secuencial	0.00525946	0.08014456	0.7702397	7.532103	28.52905	88.53015	180.0475	312.7728	719.3652	1309.141	3684.005
Hilos	0.005083755	0.04887562	0.3896952	3.958441	15.44685	39.64069	80.94272	141.8395	371.8211	803.006	3619.247
Diferencia	**1.034562051**	**1.639765593**	**1.976518315**	**1.902795318**	**1.846917009**	**2.233315061**	**2.224381637**	**2.205117756**	**1.934707847**	**1.630300396**	**1.017892672**



Gráfica 3. SpeedUp: Tiempo de Reloj Secuencial / Tiempo de Reloj por Hilos

8. Conclusiones

- La granularidad utilizada para dividir las operaciones de las matrices entre los diferentes hilos influye en el rendimiento, por lo tanto, se recomienda evitar la creación de una gran cantidad de hilos ya que se puede generar un deadlock

(bloqueo mutuo) en el sistema, es decir que el programa se queda congelado y es necesario matar el proceso. Y también es necesario utilizar la granularidad correcta porque puede suceder el problema de condición de carrera, debido a que los hilos comparten el mismo espacio de memoria, entonces la ejecución de una gran cantidad de hilos que pueden intentar acceder y modificar un recurso, genera corrupción en los datos y disminuye la confiabilidad de los resultados.

- El rendimiento medido en tiempo de CPU y tiempo de reloj es mejor utilizando hilos que realizando las instrucciones de forma serial, sin embargo limitar la cantidad de hilos es importante para que el rendimiento se mantenga a medida que la cantidad de datos a procesar es mucho mayor.

- Para el caso de la multiplicación de matrices utilizando hilos, el tiempo de CPU equivale aproximadamente al tiempo de reloj multiplicado por el número de núcleos del procesador y el número de hilos de cada uno de los núcleos (siempre y cuando los threads creados utilicen todos los núcleos e hilos del procesador). Para el caso de la multiplicación de matrices secuencial, el tiempo de CPU corresponde al tiempo de reloj.