Pontificia Universidad Javeriana

Facultad de Ingeniería de Sistemas

Informe Taller de Paralelismo

Autor: Juan Santiago Saavedra Holguin

28 de agosto de 2025

Índice

1. Introducción

El procesamiento paralelo se ha convertido en un componente esencial dentro de la computación de alto rendimiento (HPC, por sus siglas en inglés). Aplicaciones científicas, de ingeniería y de análisis de datos requieren ejecutar operaciones matriciales de gran tamaño de manera eficiente, lo cual motiva el uso de librerías y entornos de programación paralela.

Entre los distintos modelos de programación, OpenMP se ha consolidado como una de las herramientas más utilizadas debido a su simplicidad y portabilidad para explotar el paralelismo en arquitecturas de memoria compartida [3]. Sin embargo, la ganancia en rendimiento no es lineal con respecto al número de hilos debido a factores como la sobrecarga de sincronización, la competencia por recursos de hardware y la variabilidad inducida por el sistema operativo.

El objetivo de este informe es evaluar el desempeño de un algoritmo de multiplicación de matrices (MM) bajo distintas dimensiones (valores menores a 12 000) y diferentes configuraciones de paralelismo en OpenMP (1, 4, 8, 16 y 20 hilos). Para ello se diseñaron y ejecutaron experimentos controlados, cuyo análisis se centra en el cálculo de métricas como *speedup*, eficiencia y estabilidad estadística de los tiempos de ejecución.

2. Marco Teórico

2.1. Multiplicación de Matrices

La multiplicación de matrices es una de las operaciones fundamentales en la computación científica y en la inteligencia artificial. Su complejidad computacional clásica es $O(n^3)$ para matrices cuadradas de dimensión n, lo cual genera tiempos de ejecución crecientes a medida que aumenta el tamaño de los datos. Existen algoritmos optimizados (por ejemplo, Strassen), pero el método clásico sigue siendo relevante por su estabilidad numérica y facilidad de paralelización.

2.2. OpenMP

OpenMP (Open Multi-Processing) es una API estandarizada para la programación de sistemas de memoria compartida mediante el uso de hilos [1]. Su modelo está basado en directivas que permiten al programador paralelizar bucles y secciones de código con un mínimo esfuerzo de implementación. Aunque OpenMP puede mejorar significativamente el rendimiento, el escalamiento depende de factores como la arquitectura del procesador, el número de núcleos disponibles y el costo de coordinación entre hilos.

2.3. Speedup y Eficiencia

El *speedup* (S) mide la mejora relativa al comparar el tiempo secuencial T_1 con el tiempo paralelo T_p utilizando p hilos:

$$S(p) = \frac{T_1}{T_p}$$

La eficiencia (E) cuantifica qué tan bien se aprovechan los recursos de procesamiento:

$$E(p) = \frac{S(p)}{p}$$

En la práctica, debido a la Ley de Amdahl [2], el speedup se ve limitado por la fracción secuencial del programa.

3. Diseño del Experimento

3.1. Configuración de Pruebas

Se hicieron experimentos de multiplicación de matrices cuadradas con tamaños que iban desde 800 hasta 13 600, aumentando poco a poco de manera regular. Para cada tamaño se probaron distintas configuraciones con 1, 2, 4, 8, 16 y 20 hilos usando OpenMP.

3.2. Metodología

El proceso seguido para realizar las pruebas fue el siguiente:

1. Se seleccionaron distintos tamaños de matrices cuadradas (N), desde 200 hasta 12 000. En la práctica, las ejecuciones se realizaron entre 800 y 8 800 (para el último tamaño solo se probaron las configuraciones con 1 y 2 hilos).

2. Para cada tamaño se evaluaron configuraciones con 1, 2, 4, 8, 16 y 20 hilos utilizando OpenMP.

3. Cada combinación de parámetros (N, p) se ejecutó 30 veces para disminuir el efecto de otros procesos del sistema operativo sobre los tiempos de ejecución.

4. Con los resultados obtenidos se calcularon valores estadísticos: promedio, desviación estándar e intervalos de confianza al 95 %.

5. A partir de estas mediciones se derivaron las métricas de *speedup* y eficiencia.

6. Finalmente, se elaboraron gráficas comparativas para analizar la escalabilidad del algoritmo según el número de hilos y el tamaño de la matriz.

4. Resultados

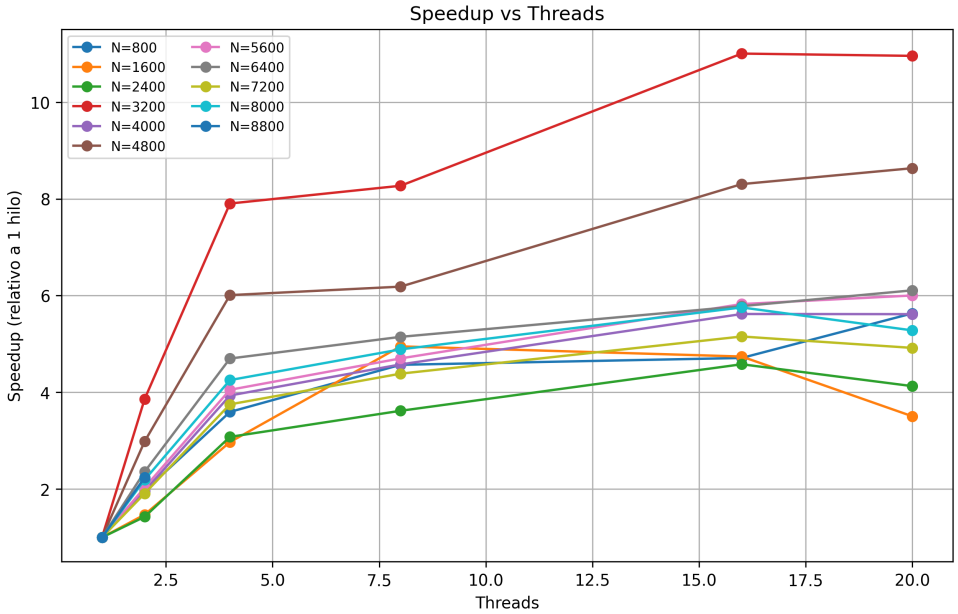

Figura 1: Speedup medio obtenido en función del número de hilos, tomando como referencia la ejecución secuencial. Se muestran los resultados para distintas dimensiones N, promediados sobre 30 repeticiones.

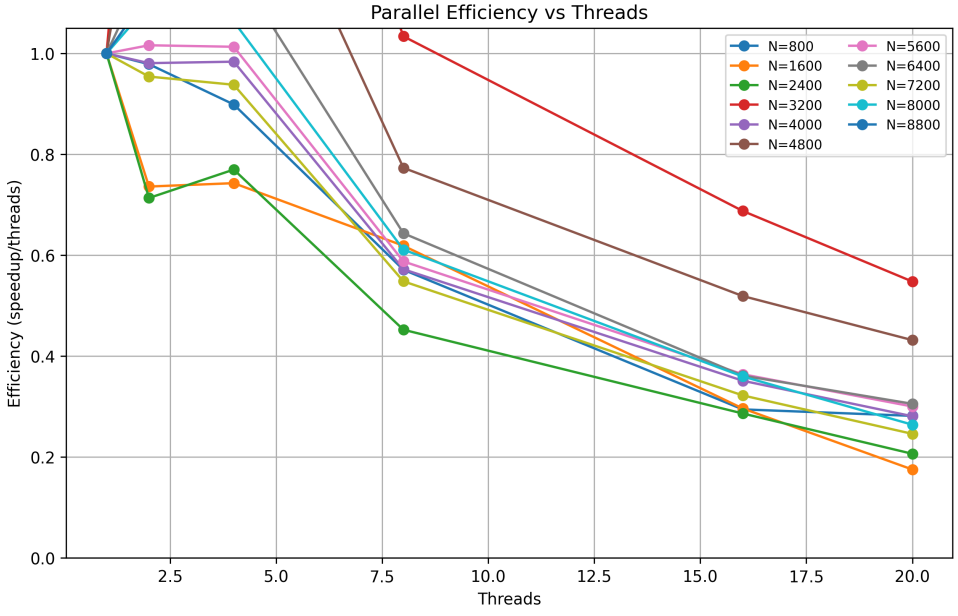

Figura 2: Eficiencia paralela en función del número de hilos para distintas dimensiones N. La eficiencia se define como $E = \frac{\text{Speedup}}{\#\text{hilos}}$.

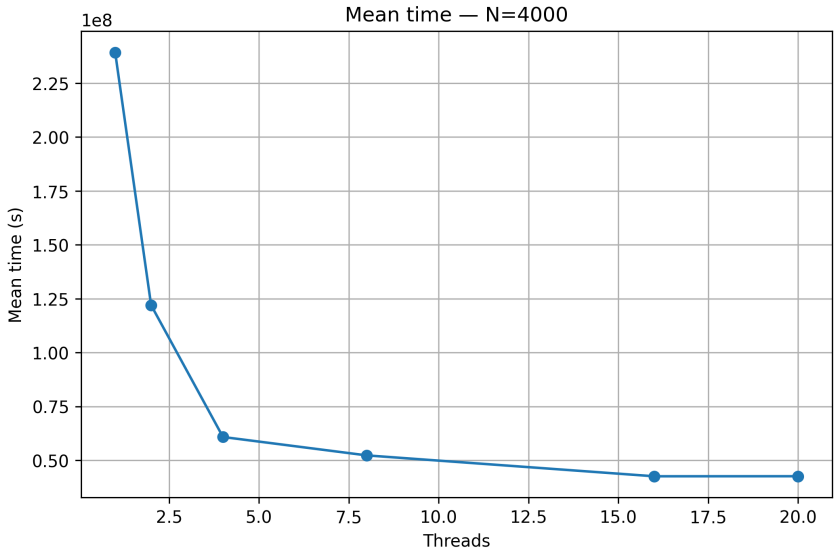

Figura 3: Tiempo medio de ejecución (con intervalo de confianza al 95 %) en función del número de hilos para el caso $N = 4000$. Cada punto corresponde al promedio de 30 repeticiones.

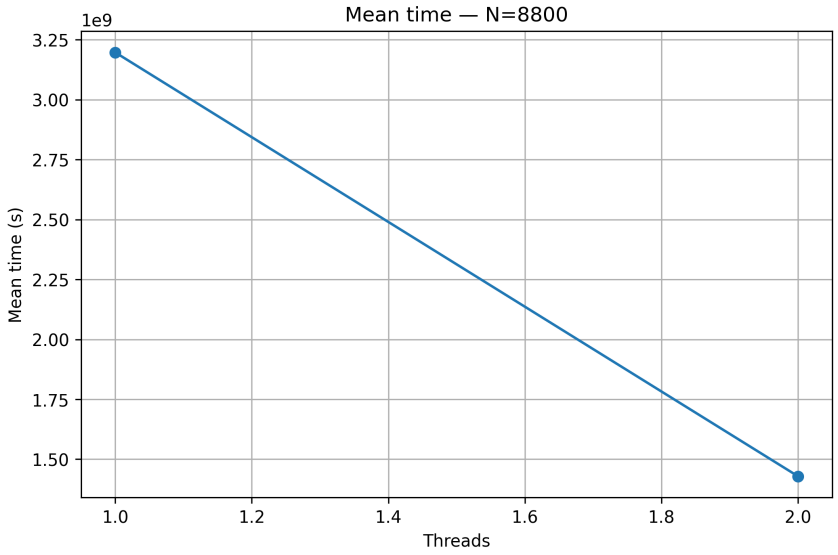

Figura 4: Tiempo medio de ejecución en función del número de hilos para el caso $N = 8800$. Por limitaciones de tiempo en la experimentación, solo se realizaron ejecuciones con 1 y 2 hilos. Cada punto corresponde al promedio de 30 repeticiones.

5. Conclusiones

1. A través de las pruebas realizadas se comprobó que el paralelismo con OpenMP sí logra reducir los tiempos de ejecución en la multiplicación de matrices, sobre todo en los primeros incrementos de hilos. Esto demuestra que la estrategia es efectiva cuando el problema tiene un tamaño considerable.

2. El *speedup* aumentó al crecer el número de hilos, pero no de forma lineal. En la práctica, se evidenció que duplicar los hilos no implica duplicar el rendimiento, lo cual confirma las limitaciones planteadas por la Ley de Amdahl y los costos asociados a la sincronización entre hilos.

3. La eficiencia fue buena en configuraciones con pocos hilos, pero decayó a medida que se incrementaron. Esto indica que, después de cierto punto, la sobrecarga de coordinación empieza a pesar más que la ganancia de paralelizar.

4. El tamaño de la matriz influyó bastante en los resultados: con tamaños grandes el paralelismo se aprovechó mejor, mientras que en tamaños pequeños la gestión de hilos restó rendimiento. En otras palabras, no siempre "más hilos = más rápido", sino que depende de la magnitud del problema.

5. Finalmente, cabe resaltar que este experimento requirió un tiempo de ejecución prolongado (el programa estuvo corriendo cerca de una semana). Esto refleja no solo la magnitud del análisis, sino también la importancia de planear bien los recursos computacionales. Como ingeniero en formación, esta experiencia me permitió comprender de manera práctica que los conceptos de *speedup* y eficiencia no son solo teoría, sino que se reflejan directamente en la realidad de la programación paralela.

Referencias

Referencias

[1] Chandra, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J., Menon, R. *Parallel programming in OpenMP*. Morgan Kaufmann, 2001.

[2] Amdahl, G. M. Validity of the single processor approach to achieving large scale computing capabilities. *AFIPS spring joint computer conference*, pp. 483–485, ACM, 1967.

[3] OpenMP Architecture Review Board. *OpenMP Application Program Interface*, versión 5.2, 2023. Disponible en: `https://www.openmp.org/specifications/`