COMPARACIÓN DEL RENDIMIENTO DE LA MULTIPLICACIÓN DE MATRICES CON OPENMP: UN ENFOQUE DE FILAS POR COLUMNAS VERSUS FILAS POR FILAS

Sergio Andrés Rojas Moreno
Pontificia Universidad Javeriana
Maestría en Inteligencia Artificial
Computación de Alto Desempeño
Bogotá, Colombia
s_rojas@javeriana.edu.co

***Resumen*—En el ámbito de la computación de alto desempeño, la multiplicación de matrices constituye una operación fundamental que se beneficia significativamente de la paralelización. Este trabajo explora la eficiencia de la multiplicación de matrices utilizando *OpenMP*, comparando dos enfoques: el método tradicional de multiplicar filas por columnas y un método alternativo de filas por filas. Para ello, se realizaron experimentos con matrices de tamaño $N \times N$, variando desde 100 hasta 1000, en tres diferentes arquitecturas de hardware con números de núcleos/hilos disponibles de 10, 20 y 40. Los resultados revelaron tendencias claras acerca de cómo el tamaño de la matriz y la cantidad de hilos afectan el tiempo de procesamiento. Adicionalmente, se analizaron algunas métricas para programas en paralelo, como el *speedup* y la eficiencia, proporcionando una comprensión más profunda de la escalabilidad y las ventajas de la paralelización en el contexto de la multiplicación de matrices.**

***Index Terms*—Multiplicación de matrices, OpenMP, paralelización, Computación de Alto Desempeño (HPC), programación en C, benchmarking de procesadores, tiempos de ejecución.**

I. INTRODUCCIÓN

La computación de alto desempeño (HPC) ha emergido como un campo fundamental en la ciencia y la ingeniería, permitiendo la realización de cálculos complejos y el procesamiento de grandes volúmenes de datos a una alta velocidad. Entre las diversas operaciones que se llevan a cabo en este campo, la multiplicación de matrices destaca por su relevancia y aplicabilidad en una amplia gama de disciplinas, desde la física y la ingeniería hasta la inteligencia artificial. Históricamente, la multiplicación de matrices ha sido un foco de estudio debido a su intensivo uso de recursos computacionales y su potencial para la optimización. En la era actual, donde el paralelismo y la escalabilidad se han vuelto necesarios para el rendimiento computacional, explorar métodos eficientes para esta operación es de cierta forma relevante. Este documento se centra en la evaluación de la eficiencia de la multiplicación de matrices utilizando *OpenMP*, un modelo de programación ampliamente reconocido por su facilidad de uso y eficacia en la paralelización de procesos.

El objetivo principal de esta investigación es comparar dos enfoques diferentes para la multiplicación de matrices: el método tradicional de filas por columnas y un enfoque alternativo de filas por filas y ver sus ventajas teniendo en cuenta el uso de apuntadores. A través de una serie de experimentos realizados en matrices de tamaño $N \times N$, que varían desde 100 hasta 1000, y en tres arquitecturas de hardware diferentes con 10, 20 y 40 núcleos/hilos máximos disponibles, se busca identificar patrones sobre cómo el tamaño de la matriz y el número de hilos influyen en el rendimiento del procesamiento. Este análisis no solo proporciona insights sobre la efectividad de los métodos de multiplicación de matrices en entornos paralelos, sino que también contribuye al entendimiento de conceptos clave en la computación de alto desempeño, tales como el *speedup* y la eficiencia. Al explorar el uso efectivo de múltiples núcleos/hilos y el manejo avanzado de apuntadores en la multiplicación de matrices, el proyecto proporciona un aprendizaje práctico para comprender mejor las técnicas de paralelización y optimización de recursos en entornos de cómputo intensivo.

Por otro lado, este trabajo inicia explicando la metodología, la cual es una descripción detallada del experimento, incluyendo la configuración del entorno y las características de las máquinas utilizadas. La sección de Implementación describe cómo se implementó el algoritmo de multiplicación de matrices para las dos aproximaciones. El Diseño Experimental explica cómo se estructuró el experimento, incluyendo los diferentes tamaños de las matrices y la cantidad de hilos utilizados. A continuación, en la sección de Resultados, se presentan los datos recolectados. Este apartado incluye tablas y gráficos que muestran cómo los tiempos de ejecución varían con el tamaño de la matriz y el número de hilos. Posteriormente, la sección de discusión permite interpretar los resultados, identificar patrones y comparar los rendimientos de las dos

estrategias de multiplicación de matrices. También, se discuten las métricas de rendimiento seleccionadas. Finalmente, la sección de Conclusiones resume los hallazgos principales y su significado.

II. Metodología

Para esta sección, se proporciona una explicación del procedimiento experimental, abarcando los siguientes aspectos: Configuración del entorno, Implementación y Diseño experimental.

II-A. Configuración del Entorno:

Esta sección detalla las especificaciones técnicas de los equipos de cómputo utilizados en los experimentos. Tal como se presenta en la Tabla I, se emplearon tres máquinas distintas: una con arquitectura ARM y las otras dos basadas en tecnología *Intel* para servidores. Específicamente, la máquina número tres está equipada con dos sockets para procesadores y el modelo del procesador en cuestión tiene 24 núcleos/hilos (por socket), por lo tanto, se optó por utilizar un total de 40 núcleos/hilos para mantener la consistencia en el incremento exponencial de la capacidad de procesamiento entre las máquinas.

No.	Modelo de Procesador	Max. Cores	Alias
1	Apple M1 Max	10	MacBook
2	Intel Xeon W-1290	20	Sistemas
3	Intel Xeon E5-2650v4	40	Cratos

Tabla I: Especificaciones de las 3 diferentes máquinas/nodos usados

A pesar de que las máquinas número dos y tres operaban bajo sistemas basados en *Linux*, la máquina número uno funcionaba con *MacOS*, un sistema de tipo *Unix*. Esto introduce variaciones en la gestión de operaciones nativas, como la compilación de código en C. En el caso de *MacOS* con procesador M1, fue necesario emplear *Xcode* junto con su conjunto de herramientas asociadas. El compilador utilizado no fue el habitual *GCC*, sino *Clang*, que forma parte del proyecto *LLVM*. Para la integración con *OpenMP* y la compatibilidad con *LLVM*, se configuraron unas rutas en el entorno de desarrollo para las bibliotecas y los *headers* de *OpenMP* y *LLVM*. Esta configuración permitió la correcta compilación y ejecución de los algoritmos de multiplicación de matrices en el sistema operativo *MacOS*, asegurando así la coherencia metodológica a través de las diferentes arquitecturas de hardware.

II-B. Implementación:

La fase de implementación se centró en dos variantes del algoritmo de multiplicación de matrices, cuyas diferencias radican en la gestión de la memoria y el acceso a los datos. El primer enfoque, filas por columnas, emplea una secuencia de acceso tradicional, mientras que el segundo enfoque, filas por filas, optimiza el recorrido de memoria. En la versión filas por columnas, la inicialización y el proceso de cálculo se alinean con la estructura de datos lineal de la memoria, lo que resulta en un patrón de acceso secuencial a los elementos de las matrices. En contraste, el enfoque filas por filas, aunque computacionalmente equivalente, modifica el patrón de acceso a través de apuntadores, permitiendo un recorrido más eficiente en términos de localidad espacial y aprovechamiento de la caché de la CPU. El código para ambos métodos se presenta a continuación para ilustrar la equivalencia funcional y las diferencias en el manejo de los apuntadores.

- *Enfoque de Filas por Columnas*

Este método implica recorrer cada fila de la primera matriz y cada columna de la segunda matriz para calcular los elementos de la matriz resultante.

```
for (i = 0; i < SZ; i++){
    for (j = 0; j < SZ; j++){
        double *pA = a + (i * SZ);
        double *pB = b + j;
        for (k = 0; k < SZ; k++, pA++, pB += SZ){
            c[i * SZ + j] += *pA * *pB;
        }
    }
}
```

- *Enfoque de Filas por Filas*

En este enfoque, se recorren las filas de ambas matrices, considerando la segunda matriz como si cada fila fuera una columna.

```
for (i = 0; i < SZ; i++){
    for (j = 0; j < SZ; j++){
        double *pA = a + j + (i * SZ);
        double *pB = b + (j * SZ);
        for (k = 0; k < SZ; k++, pB++){
            c[i * SZ + k] += *pA * *pB;
        }
    }
}
```

A pesar de sus diferencias, ambos enfoques de multiplicación de matrices, Filas por Columnas y Filas por Filas, conducen al mismo resultado, pero difieren significativamente en el manejo de los datos y la eficiencia del acceso a la memoria. En el método Filas por Columnas, el recorrido se realiza iterando sobre cada fila de la primera matriz A y cada columna de la segunda matriz B. Esto implica un cambio en el apuntador de la columna (pB) en cada iteración del *for* más interno, incrementándolo por el tamaño de la matriz. Este enfoque puede resultar en un acceso menos secuencial a la memoria, especialmente en matrices grandes, debido a que el apuntador pB salta entre diferentes posiciones de la memoria, afectando potencialmente la localidad espacial y el rendimiento del caché.

En contraste, el método Filas por Filas recorre las filas de ambas matrices, tratando las filas de B como si fueran columnas. En este enfoque, el apuntador de la segunda matriz (pB) avanza de manera secuencial, mejorando la localidad espacial y optimizando el uso de la caché de memoria.

El apuntador de la primera matriz (pA), mientras tanto, permanece constante durante el cálculo de cada elemento de una fila en C, avanzando solo cuando se cambia de fila. Esto facilita un acceso más eficiente y secuencial a los elementos de A y B.

A pesar de estas diferencias en el acceso y la iteración sobre los elementos de las matrices, ambos enfoques garantizan el mismo resultado final. Esto se debe a la naturaleza fundamental de la operación de multiplicación de matrices: cada elemento de la matriz resultante C es la suma de los productos de los elementos correspondientes de una fila de A y una columna de B. Ambos métodos, aunque difieren en el orden de acceso y en la forma en que recorren las matrices, realizan esta misma operación fundamental, asegurando que, independientemente del enfoque, el resultado de la multiplicación sea idéntico. Por tanto, gracias a estos accesos optimizados y a la paralelización implementada, ofrecen un rendimiento mejorado en comparación con implementaciones más simples que no consideren estos aspectos de localidad y uso eficiente de la memoria caché.

II-C. Diseño Experimental

El diseño experimental de este proyecto se centró en cuantificar y comparar el rendimiento de dos estrategias distintas de multiplicación de matrices bajo la paralelización con *OpenMP*. Para ello, se definieron los parámetros de prueba, que incluyen el tamaño de las matrices y la cantidad de hilos de ejecución, y se establecieron las condiciones bajo las cuales se recopilaron y analizaron los datos.

Inicialmente, se seleccionaron tamaños de matrices que comenzaban en 100x100 y aumentaban hasta 1000x1000. Este rango se eligió para proporcionar una evaluación exhaustiva de cómo la complejidad computacional afecta al rendimiento en sistemas multicore. Se estableció que para cada tamaño de matriz, se ejecutarían pruebas con un incremento progresivo en el número de hilos, comenzando con un solo hilo y aumentando hasta el máximo que cada arquitectura de hardware podría soportar de manera eficiente, concretamente 10, 20 y 40 hilos. Esta escalabilidad permitió observar el comportamiento del *speedup* en relación con el número de hilos y la eficiencia de la paralelización.

El experimento se estructuró en varias fases: preparación de datos, ejecución, recolección de resultados, presentación y análisis de resultados. Cada fase fue diseñada para asegurar la integridad y la reproducibilidad de los resultados. En la fase de preparación, se generaron matrices con valores predefinidos para garantizar los mismos datos iniciales de prueba. Durante la ejecución, se registró el tiempo de cálculo de cada operación para analizar posteriormente la eficiencia del proceso y el speedup.

Para garantizar la validez de los resultados, cada prueba se repitió 30 veces para cada configuración permitiendo obtener en promedio aceptable en términos estadísticos. La recopilación de datos se realizó de manera automatizada para reducir el riesgo de errores humanos y para facilitar el manejo de grandes volúmenes de datos. Los tiempos de ejecución se almacenaron en un formato estructurado (número de hilo: tiempo de ejecución en microsegundos) para su posterior análisis.

La configuración de los escenarios de prueba se adaptó a cada arquitectura de hardware específica. Se realizaron ajustes en el entorno de compilación y ejecución para optimizar el uso de los recursos disponibles en cada máquina. Además, se prestó especial atención a las configuraciones de compilación, como las banderas de optimización y las versiones de *OpenMP*, para asegurar la consistencia entre las pruebas. Finalmente, el análisis de los datos se centró en identificar las tendencias y correlaciones entre el tamaño de las matrices, el número de hilos y los tiempos de ejecución. Se utilizaron los resultados para evaluar el grado de mejora que la paralelización ofrecía en cada caso.

III. Resultados

A continuación se presentan los datos recolectados. Este apartado incluye tablas y gráficos que muestran cómo los tiempos de ejecución varían con el tamaño de la matriz y el número de hilos/núcleos. En las siguientes tablas se detalla como para cada tamaño de matriz siempre hubo una cantidad de núcleos/hilos que predominó, y no necesariamente la máxima cantidad de *cores* disponibles. Adicionalmente, se muestran las gráficas referentes a métricas de speedup y eficiencia. Tener en cuenta que se separan los enfoques de filas por columnas y filas por filas para cada maquina usada.

Tamaño de Matriz	Mejor Tiempo (µs)	núcleos/hilos Totales
100.0	159.683	8.0
200.0	1053.913	8.0
300.0	3855.217	8.0
400.0	9567.688	8.0
500.0	20199.042	8.0
600.0	31905.163	8.0
700.0	51521.743	10.0
800.0	76839.143	10.0
900.0	112635.063	8.0
1000.0	146873.603	10.0

Tabla II: Resultados de la multiplicación de matrices en diferentes tamaños en la maquina de 10 núcleos/hilos con el método de Filas por Columnas.

Tamaño de Matriz	Mejor Tiempo (µs)	núcleos/hilos Totales
100.0	70.450	6.0
200.0	332.714	7.0
300.0	1074.772	6.0
400.0	2214.005	7.0
500.0	4154.558	8.0
600.0	7031.057	7.0
700.0	11021.821	8.0
800.0	16047.862	7.0
900.0	22336.363	8.0
1000.0	32402.092	8.0

Tabla III: Resultados de la multiplicación de matrices en diferentes tamaños en la maquina de 10 núcleos/hilos con el método de Filas por Filas.

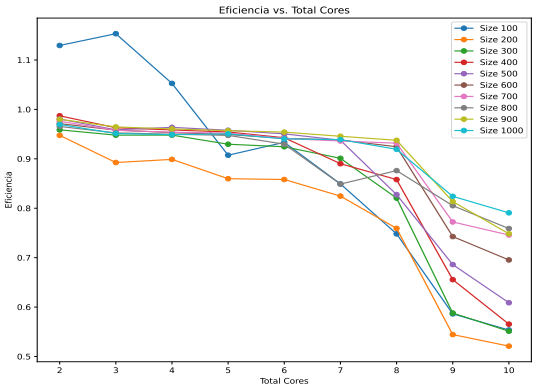

Figura 1: Eficiencia contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 10 núcleos/hilos con el método de Filas por Columnas.

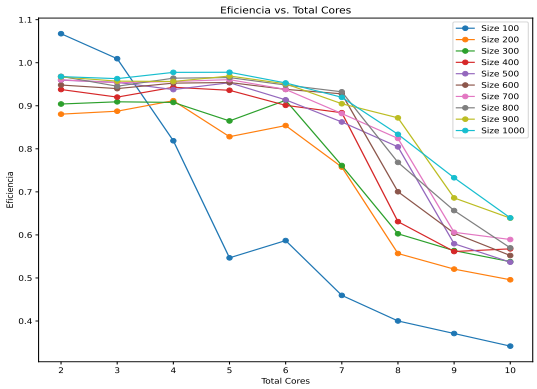

Figura 4: Eficiencia contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 10 núcleos/hilos con el método de Filas por Filas.

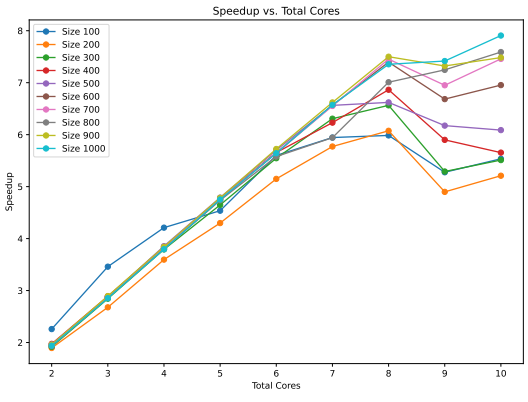

Figura 2: Aceleración o *speedup* contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 10 núcleos/hilos con el método de Filas por Columnas.

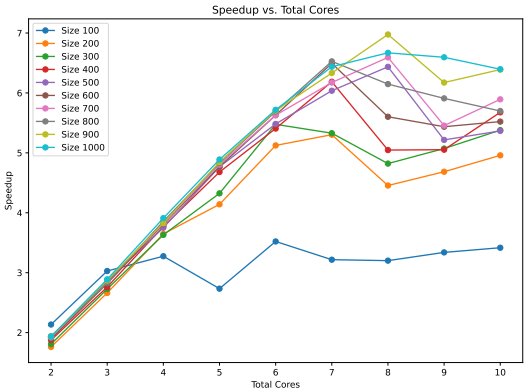

Figura 5: Aceleración o *speedup* contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 10 núcleos/hilos con el método de Filas por Filas.

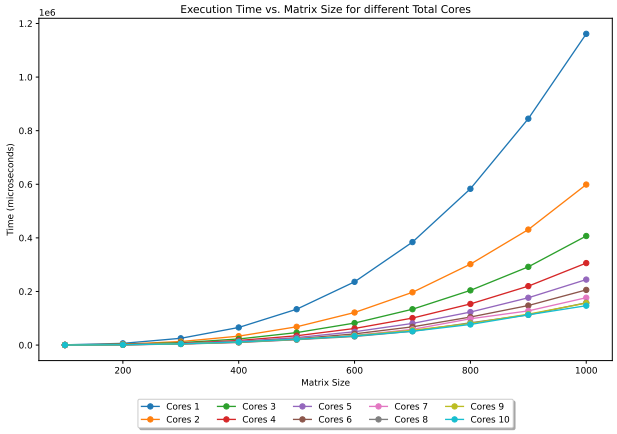

Figura 3: Tiempo contra diferentes tamaños matriz en la maquina de 10 núcleos/hilos con el método de Filas por Columnas.

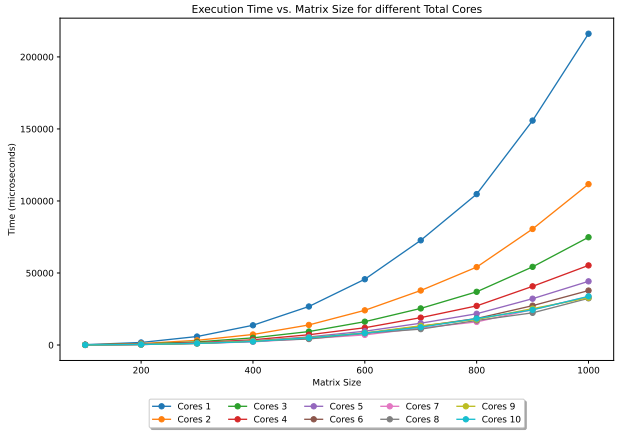

Figura 6: Tiempo contra diferentes tamaños matriz en la maquina de 10 núcleos/hilos con el método de Filas por Filas.

Tamaño de Matriz	Mejor Tiempo (µs)	núcleos/hilos Totales
100.0	60.375	17.0
200.0	483.537	17.0
300.0	1696.120	17.0
400.0	5492.282	17.0
500.0	11241.245	17.0
600.0	20175.745	17.0
700.0	36776.928	17.0
800.0	87906.052	7.0
900.0	110509.167	7.0
1000.0	200281.848	7.0

Tabla IV: Resultados de la multiplicación de matrices en diferentes tamaños en la maquina de 20 núcleos/hilos con el método de Filas por Columnas.

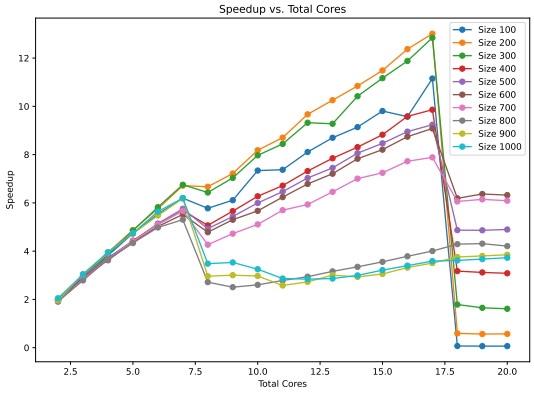

Figura 8: Aceleración o *speedup* contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 20 núcleos/hilos con el método de Filas por Columnas.

Tamaño de Matriz	Mejor Tiempo (µs)	núcleos/hilos Totales
100.0	60.890	16.0
200.0	421.410	17.0
300.0	1425.716	17.0
400.0	3979.688	16.0
500.0	8310.638	16.0
600.0	14055.092	17.0
700.0	20646.761	17.0
800.0	34362.056	16.0
900.0	46374.178	17.0
1000.0	70104.412	17.0

Tabla V: Resultados de la multiplicación de matrices en diferentes tamaños en la maquina de 20 núcleos/hilos con el método de Filas por Filas.

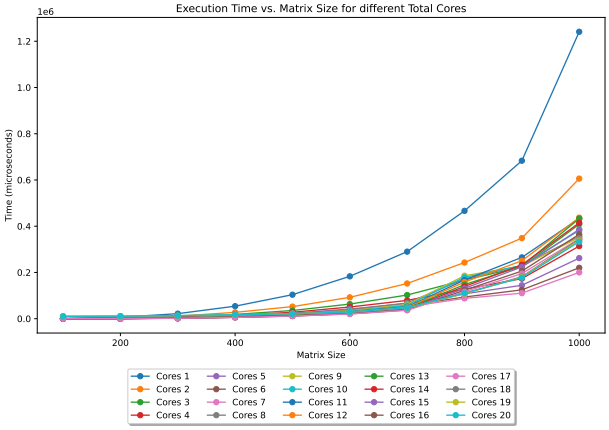

Figura 9: Tiempo contra diferentes tamaños matriz en la maquina de 20 núcleos/hilos con el método de Filas por Columnas.

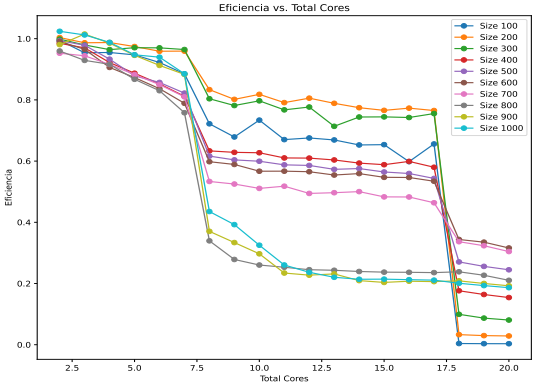

Figura 7: Eficiencia contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 20 núcleos/hilos con el método de Filas por Columnas.

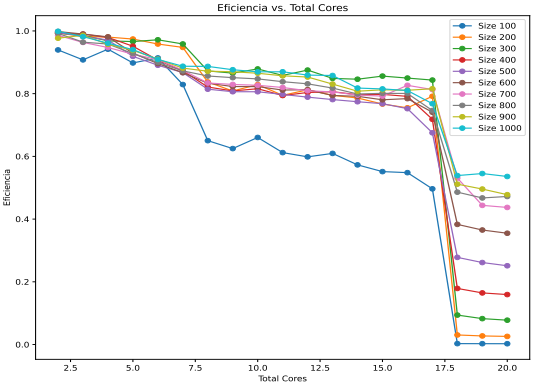

Figura 10: Eficiencia contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 20 núcleos/hilos con el método de Filas por Filas.

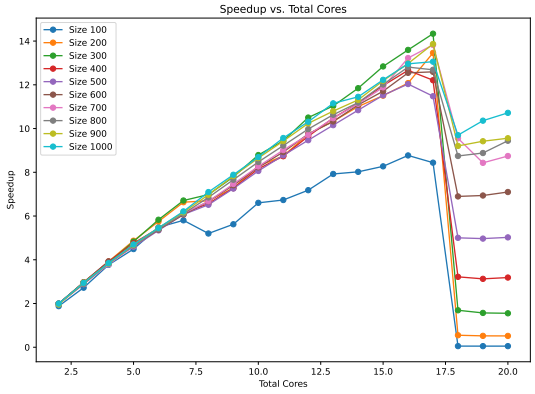

Figura 11: Aceleración o *speedup* contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 20 núcleos/hilos con el método de Filas por Filas.

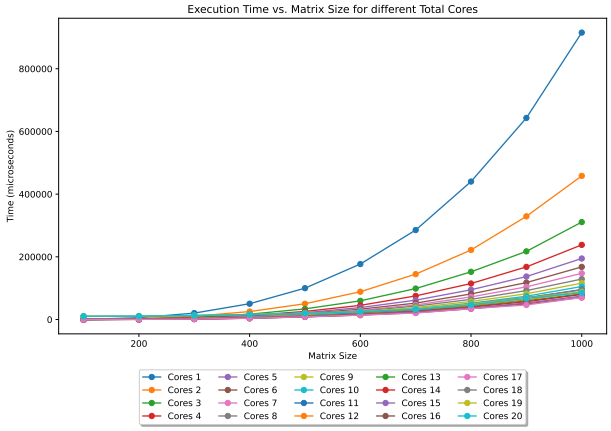

Figura 12: Tiempo contra diferentes tamaños matriz en la maquina de 20 núcleos/hilos con el método de Filas por Filas.

Tamaño de Matriz	Mejor Tiempo (µs)	núcleos/hilos Totales
100.0	122.097	24.0
200.0	906.200	24.0
300.0	2833.936	22.0
400.0	7304.620	24.0
500.0	13237.719	25.0
600.0	23794.633	37.0
700.0	37941.234	40.0
800.0	64127.754	40.0
900.0	95685.258	40.0
1000.0	119851.391	39.0

Tabla VI: Resultados de la multiplicación de matrices en diferentes tamaños en la maquina de 40 núcleos/hilos con el método de Filas por Columnas.

Tamaño de Matriz	Mejor Tiempo (µs)	núcleos/hilos Totales
100.0	88.307	24.0
200.0	473.344	24.0
300.0	1268.799	25.0
400.0	2794.314	24.0
500.0	5646.245	25.0
600.0	9508.978	24.0
700.0	15306.465	24.0
800.0	21773.646	27.0
900.0	28656.168	30.0
1000.0	35374.145	40.0

Tabla VII: Resultados de la multiplicación de matrices en diferentes tamaños en la maquina de 40 núcleos/hilos con el método de Filas por Filas.

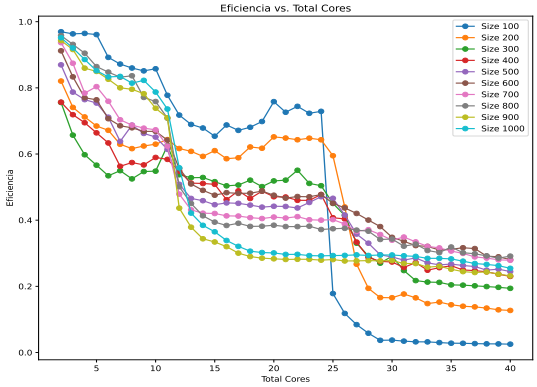

Figura 13: Eficiencia contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 40 núcleos/hilos con el método de Filas por Columnas.

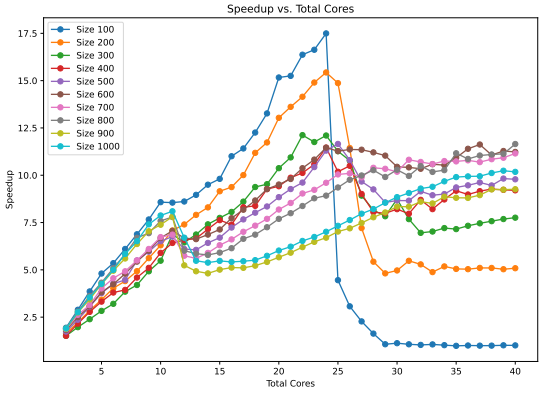

Figura 14: Aceleración o *speedup* contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 40 núcleos/hilos con el método de Filas por Columnas.

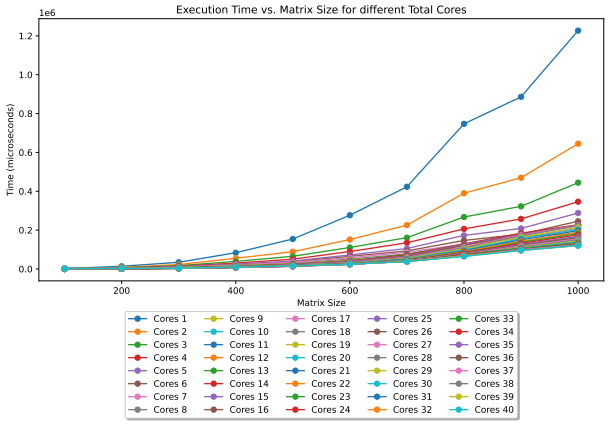

Figura 15: Tiempo contra diferentes tamaños matriz en la maquina de 40 núcleos/hilos con el método de Filas por Columnas.

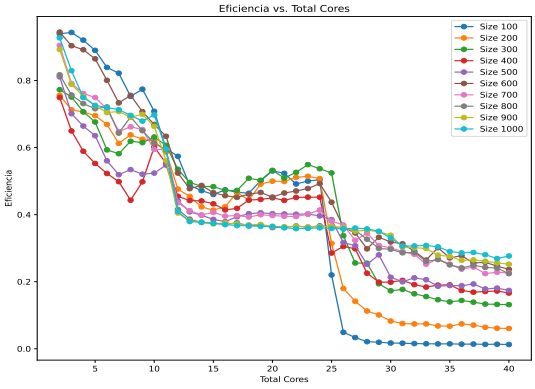

Figura 16: Eficiencia contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 40 núcleos/hilos con el método de Filas por Filas.

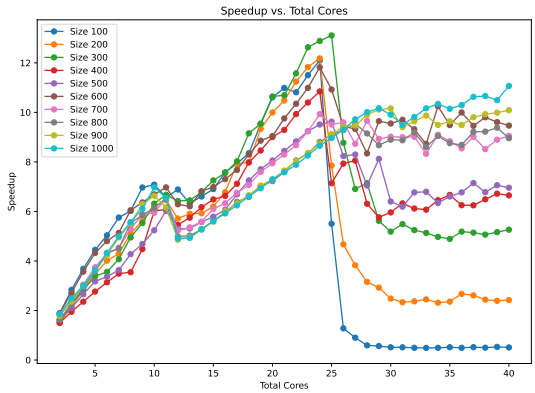

Figura 17: Aceleración o *speedup* contra el Total de núcleos/hilos en la multiplicación de matrices en diferentes tamaños en la maquina de 40 núcleos/hilos con el método de Filas por Filas.

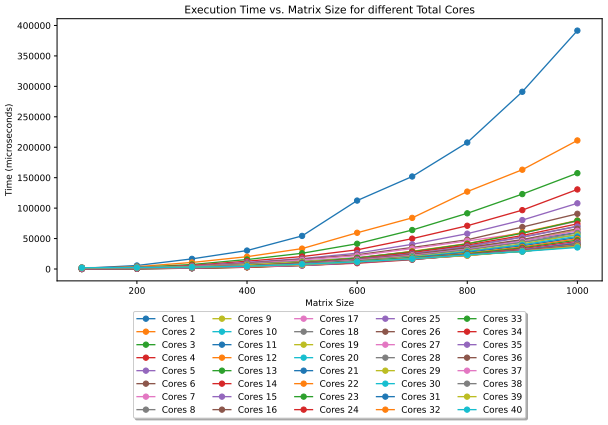

Figura 18: Tiempo contra diferentes tamaños matriz en la maquina de 40 núcleos/hilos con el método de Filas por Filas.

IV. Discusión

En esta sección, se interpretan parte de los resultados obtenidos, identificando patrones significativos y comparando los rendimientos de las dos estrategias de multiplicación de matrices: Filas por Columnas (FxC) y Filas por Filas (FxF). Se discuten las métricas de rendimiento seleccionadas, incluyendo el tiempo de ejecución, el speedup y la eficiencia, y cómo estos se relacionan con el tamaño de la matriz y la cantidad de núcleos/hilos utilizados en el procesamiento.

IV-A. Análisis de Resultados de Multiplicación de Matrices en Máquina de 10 Núcleos

Los resultados obtenidos en la máquina de 10 núcleos/hilos revelan diferencias notables en el rendimiento entre FxC y FxF. Se observa que FxC presenta tiempos de ejecución mayores que FxF para el mismo tamaño de matriz, lo que podría indicar una menor eficiencia en la gestión de los recursos computacionales. Sin embargo, FxC exhibe un mejor speedup y una eficiencia cercana a la unidad para matrices más grandes, lo que sugiere que la metodología aprovecha efectivamente la paralelización en escenarios de mayor complejidad computacional. Por otro lado, FxF demuestra ser consistentemente más eficiente en términos de tiempo de ejecución, manteniendo una eficiencia relativamente estable incluso a medida que el tamaño de la matriz aumenta. Esto implica que FxF podría ser más adecuado para aplicaciones que requieren un procesamiento eficiente de matrices de gran tamaño en máquinas con un número limitado de núcleos.

IV-B. Análisis de Resultados en Máquina de 20 Núcleos

Al extender el análisis a una máquina de 20 núcleos, se observa que FxC sigue un patrón similar al evidenciado en la máquina de 10 núcleos, con tiempos de ejecución que escalan considerablemente con el tamaño de la matriz. FxF, en cambio, no solo mantiene tiempos de ejecución más bajos, sino que también exhibe un speedup y una eficiencia superiores en matrices de mayor tamaño, subrayando su capacidad para escalar de manera más efectiva con el incremento de recursos de

procesamiento. La interpretación de los gráficos de eficiencia y speedup proporcionados refuerzan estos hallazgos. Mientras que FxC muestra una eficiencia decreciente con el aumento en el número de núcleos, FxF logra mantener una eficiencia y un speedup altos, indicando su idoneidad para sistemas con un número elevado de núcleos/hilos y matrices de gran tamaño.

IV-C. Análisis de Resultados en Máquina de 40 Núcleos

En la máquina de 40 núcleos, se confirma que FxF supera a FxC en términos de tiempos de ejecución y eficiencia, especialmente en matrices de tamaño considerable. A pesar de que FxC alcanza un speedup que resalta en matrices más pequeñas, este se reduce significativamente en matrices de mayor tamaño, lo que sugiere que sus ventajas se diluyen a medida que la complejidad computacional aumenta. Los gráficos proporcionados demuestran que FxF ofrece un rendimiento más consistente y robusto, manteniendo un speedup y una eficiencia elevados independientemente del tamaño de la matriz. Esto apunta a FxF como la estrategia preferible para el cálculo de multiplicaciones de matrices en sistemas de alto rendimiento.

V. Conclusión

Este trabajo ha proporcionado una evaluación de dos estrategias de multiplicación de matrices utilizando *OpenMP* en diferentes configuraciones de hardware. Los experimentos realizados han revelado que el enfoque de Filas por Filas (FxF) supera consistentemente al método tradicional de Filas por Columnas (FxC) en términos de tiempo de ejecución, eficiencia y speedup, especialmente cuando se trabaja con matrices de gran tamaño y en sistemas con un número elevado de núcleos.

El análisis detallado ha demostrado que la estrategia FxF es más eficiente en el uso de recursos computacionales y en la gestión de la memoria caché, lo que se traduce en una mejora significativa del rendimiento. Esto se ha observado en máquinas con 10, 20 y 40 núcleos, lo que indica que la superioridad del método Filas por Filas no es circunstancial sino consistente a través de diversas arquitecturas y configuraciones de hardware.

La importancia de estos hallazgos reside en su aplicabilidad en el campo de la computación de alto desempeño, donde la eficiencia y la capacidad de escalabilidad son fundamentales. Al optimizar operaciones tan usadas como la multiplicación de matrices, se pueden lograr avances significativos en la velocidad y eficiencia de los cálculos, lo que tiene implicaciones directas en campos tan variados como la simulación de modelos y la inteligencia artificial.

Para investigaciones futuras, sería interesante explorar la aplicación de estas estrategias de multiplicación de matrices en otras formas de paralelización, como la programación en GPU o el uso de *clusters* con tecnologías como MPI. Además, sería valioso investigar la adaptabilidad de estos métodos a otros tipos de operaciones matriciales que son comunes en aplicaciones de HPC.

Finalmente, este proyecto ha proporcionado una contribución significativa al entendimiento de la multiplicación de matrices en entornos paralelizados, demostrando la superioridad de la estrategia de Filas por Filas y sentando las bases para futuras investigaciones y aplicaciones en el campo de la computación de alto desempeño.

VI. Referencias

La documentación completa de los scripts, así como materiales adicionales, están disponibles en el siguiente repositorio: https://github.com/sergioarojasm98/Multiplicacion-de-Matrices-con-OpenMP

	Tamaño de Matriz	Mejor Speedup	Hilos Mejor Speedup	Mejor Eficiencia	Hilos Mejor Eficiencia
FxC - 10 Cores	100.0	5.985	8	1.153	3
	200.0	6.074	8	0.948	2
	300.0	6.564	8	0.959	2
	400.0	6.864	8	0.987	2
	500.0	6.621	8	0.981	2
	600.0	7.396	8	0.971	2
	700.0	7.458	10	0.976	2
	800.0	7.589	10	0.965	2
	900.0	7.500	8	0.980	2
	1000.0	7.907	10	0.970	2
FxF - 10 Cores	100.0	3.521	6	1.067	2
	200.0	5.303	7	0.912	4
	300.0	5.476	6	0.913	6
	400.0	6.187	7	0.943	4
	500.0	6.436	8	0.959	2
	600.0	6.494	7	0.954	5
	700.0	6.593	8	0.961	5
	800.0	6.527	7	0.968	2
	900.0	6.976	8	0.969	5
	1000.0	6.669	8	0.978	5
FxC - 20 Cores	100.0	11.153	17	0.995	2
	200.0	13.009	17	1.004	2
	300.0	12.841	17	0.999	2
	400.0	9.859	17	0.985	2
	500.0	9.237	17	0.994	2
	600.0	9.081	17	0.991	2
	700.0	7.886	17	0.953	2
	800.0	5.306	7	0.960	2
	900.0	6.183	7	1.015	3
	1000.0	6.196	7	1.024	2
FxF - 20 Cores	100.0	8.771	16	0.942	4
	200.0	13.459	17	0.993	2
	300.0	14.337	17	0.993	2
	400.0	12.662	16	0.994	2
	500.0	12.035	16	0.994	2
	600.0	12.593	17	0.998	2
	700.0	13.823	17	0.986	2
	800.0	12.810	16	0.992	2
	900.0	13.866	17	0.986	3
	1000.0	13.057	17	0.999	2
FxC - 40 Cores	100.0	17.496	24	0.970	2
	200.0	15.433	24	0.821	2
	300.0	12.120	22	0.756	2
	400.0	11.462	24	0.757	2
	500.0	11.654	25	0.870	2
	600.0	11.624	37	0.912	2
	700.0	11.152	40	0.937	2
	800.0	11.651	40	0.958	2
	900.0	9.262	40	0.943	2
	1000.0	10.239	39	0.952	2
FxF - 40 Cores	100.0	12.101	24	0.943	3
	200.0	12.179	24	0.756	2
	300.0	13.106	25	0.773	2
	400.0	10.846	24	0.749	2
	500.0	9.625	25	0.811	2
	600.0	11.821	24	0.944	2
	700.0	9.926	24	0.905	2
	800.0	9.540	27	0.817	2
	900.0	10.159	30	0.893	2
	1000.0	11.069	40	0.927	2

Tabla VIII: Resumen de las mejores métricas de Speedup y Eficiencia por Tamaño de Matriz