Rendimiento de Multiplicación de Matrices con OpenMP

John Alexander Rodriguez Cuellar
Pontificia Universidad Javeriana
Estudiante
jjrodriguezc@javeriana.edu.co

***Abstract*— El rendimiento en la multiplicación de matrices con OpenMP es un área crucial en computación de alto rendimiento. Este estudio se centra en evaluar y mejorar la eficiencia de la multiplicación de matrices utilizando OpenMP, una API para la programación paralela. Se analizan distintas estrategias de paralelización para optimizar el rendimiento, incluyendo la asignación de tareas a múltiples hilos de ejecución, la gestión de la concurrencia y la minimización de cuellos de botella. El objetivo es mejorar la velocidad de cálculo de grandes conjuntos de datos matriciales para aplicaciones científicas, de aprendizaje automático, simulaciones y más, aprovechando las capacidades de paralelización que brinda OpenMP.**

Keywords—Matrices, Rendimiento, OpenMP, Optimización.

I. Introduction

El estudio se concentra en evaluar el rendimiento de la multiplicación de matrices mediante OpenMP en tres máquinas diferentes. Se analiza el impacto de variar el número de núcleos disponibles y se compara el rendimiento de dos algoritmos. Esta operación matricial es esencial en numerosas aplicaciones científicas y tecnológicas, por lo que optimizar su eficiencia es crucial para sistemas de alto rendimiento. OpenMP, una API de programación para cómputo paralelo, ha ganado importancia al permitir el uso eficaz de recursos multi-núcleo. El estudio de varios experimentos busca comprender cómo diferentes configuraciones de hardware y algoritmos influyen en el rendimiento de la multiplicación de matrices, ofreciendo información valiosa para la optimización y la toma de decisiones en sistemas que involucran cálculos matriciales..

II. Casos de Uso

A. *Maquinas disponibles.*

Para el análisis de rendimiento sobre la multiplicación de matrices, se utilizaron 3 diferentes maquinas con características de hardware diferentes, en las cuales se ejecutaron 2 diferentes algoritmos para multiplicación de matrices. En cada una de estas maquinas se probaron diferentes tamaños de matrices, además de realizar una carga progresiva a los diferentes nucleos.

- Maquina 1 (Cratos): Maquina que pertenece a la Universidad a la cual se accedió de forma remota que cuenta con un total de 40 núcleos para la ejecución de procesos.
- Maquina 2 (Sistemas): Maquina que pertenece a la Universidad a la cual se accedió de forma remota, la cual cuenta con un total de 20 núcleos para la ejecución de procesos.
- Maquina 3 (Local): Maquina personal en la cual se instalo una maquina virtual con el sistema operativo Linux Ubuntu, la cual cuenta con un total de 10 núcleos.

B. *Algoritmos*

Se generaron 2 algoritmos para comprobar la eficiencia del software a la hora de realizar la multiplicación:

- Algoritmo clásico de multiplicación de matrices filas por columnas. En este algoritmo se realiza la multiplicación de las filas de la primera matriz por las columnas de la segunda matriz en su respectivo orden, al finalizar este proceso se suman para obtener el resultado final.
- Algoritmo filas por filas, en esta variación del algoritmo, se multiplican las filas de la primera matriz por las filas de la segunda matriz en su respectivo orden, al finalizar este proceso se realiza la suma para obtener la matriz final.

Si bien la eficiencia no se determina por un tipo de algoritmo especifico ya que depende de otras variables como el hardware y el tamaño de la matriz, se cuenta con una hipótesis en el cual el algoritmo de filas por filas contara con mejor rendimiento, dado que realiza la multiplicación de una fila completa antes de salir del primer *for*.

C. *Algoritmos*

Se cuenta con 6 tamaños diferentes de matrices para comprobar la eficiencia de las máquinas y algoritmos (100, 200, 400, 600, 800 y 1000). Cada algoritmo se ejecutará un total de 30 veces por cada tamaño de matriz, esto para poder generar un tiempo promedio de cada ejecución y determinar cuáles características y algoritmos cuentan con la mejor eficiencia.

III. Resultados Experimentación

Con la información recopilada de los experimentos, se logra deducir:

1. Más núcleos no se traducen a una mejor eficiencia en el proceso, para ninguno de los 2 algoritmos.
2. Si bien, el cambio de algoritmo no garantiza un mejor desempeño, si se refuerza la hipótesis planteada, la cual nos dice que multiplicar matrices filas por filas, mejora los tiempos de procesamiento.
3. El número óptimo de núcleos utilizados se encuentra 21 (más o menos 3). Según las especificaciones de las maquinas Cratos y Sistemas, sin embargo, la maquina Local presento el número óptimo de núcleos para la tarea en 10 (Máximo de procesadores en la maquina).
4. Solo la maquina con Local con 10 procesadores genero una eficiencia mayor al 100%.

5. El mejor resultado de todas las combinaciones realizadas se encontró en 70.88 microsegundos para la siguiente combinación:
    a. Máquina de Sistemas.
    b. Algoritmo Filas por Filas.
    c. 17 núcleos.

En las figuras Adjuntas a este documento, se podrá ver el detalle 1 a 1 de las ejecuciones por *size* y por número de núcleos, además de las graficas de ejecución y el número óptimo de núcleos para la actividad por maquina y algoritmo.

IV. Conclusiones

- Como se implementa el software para las diferentes tareas impacta en el resultado de las actividades sin importar las características de las maquinas.
- El tener una maquina con recursos suficientes para optimizar los procesos es fundamental para actividades de alta carga en procesos, sin embargo, el tener recursos extra no significa que siempre deban utilizarse.
- El consolidar grandes cantidades de datos arrojadas por los experimentos realizados es un proceso que debe estar presente en la estimación de tiempos y costos de cualquier análisis.

Con esto se podría comprender que para cada actividad o tarea que se desee hacer y necesite un alto rendimiento, es necesario optimizar y encontrar la mejor combinación de componentes que permitan mejorar la eficiencia de todo el proceso.

V. Herramienta Utilizadas

- Visual Studio Code: Herramienta para edición de código fuente.
- Perl: Herramienta que se uso para la automatización de experimentos.
- Powershell: Herramienta para consolidación de resultados generados por experimentos.
- Excel: Herramienta para gráficos y análisis.

Si bien existen otras herramientas para las tareas realizadas, se decidió utilizar estas por potencia y sencillez.

VI. Anexos

Resultados Algoritmo Filas x Filas Maquina Sistemas

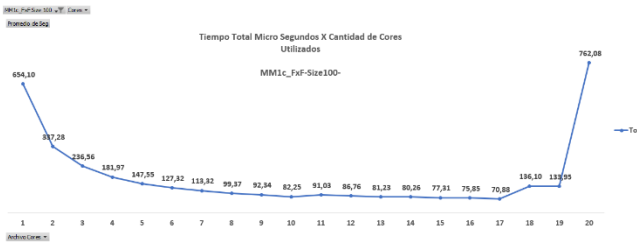


Fig MM1c_FxFSize100 S

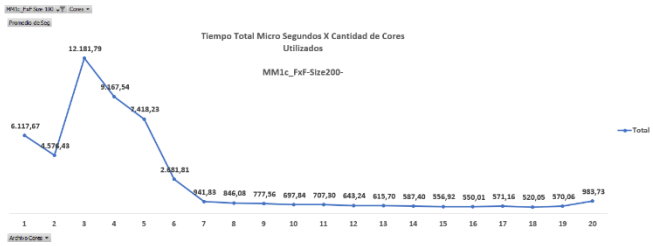


Fig MM1c_FxFSize200 S

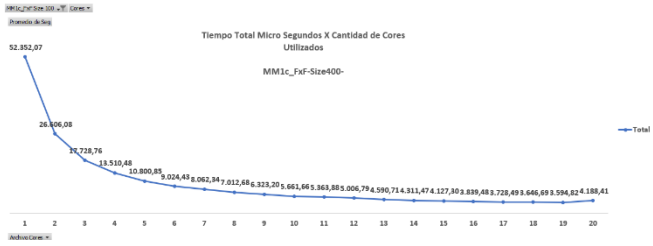


Fig MM1c_FxFSize400 S

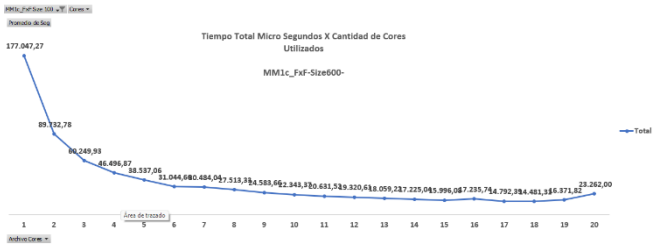


Fig MM1c_FxFSize600 S

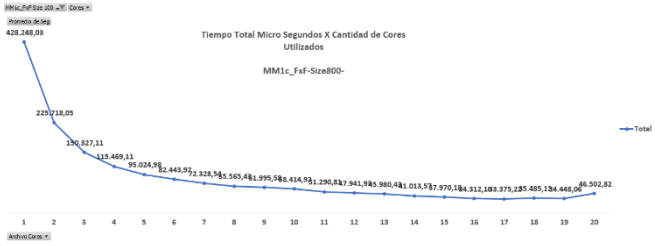


Fig MM1c_FxFSize800 S

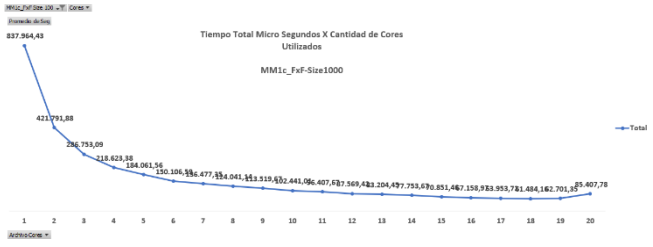


Fig MM1c_FxFSize1000 S

Resultados Algoritmo Filas x Columnas Maquina Sistemas

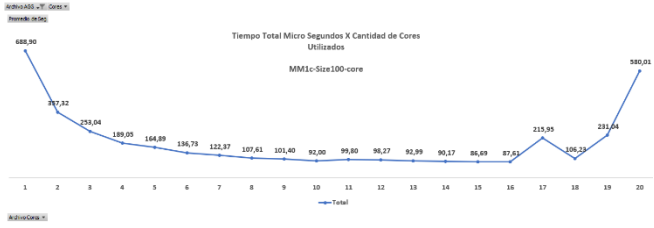


Fig MM1c_FxCSize100 S

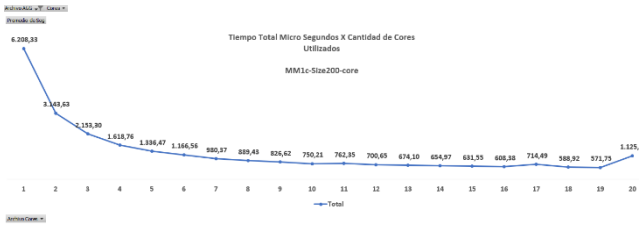


Fig MM1c_FxCSize200 S

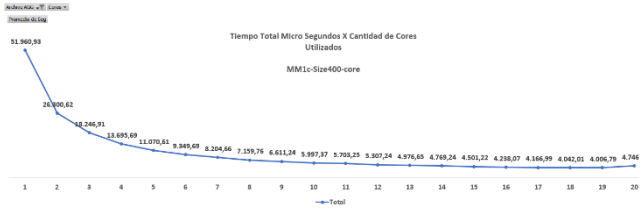


Fig MM1c_FxCSize400 S

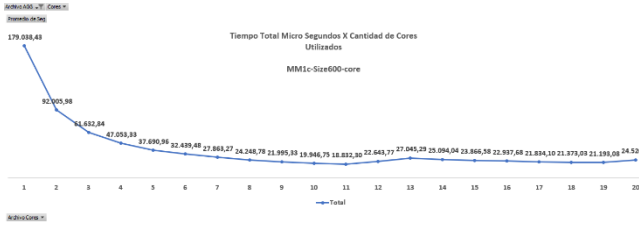


Fig MM1c_FxCSize600 S

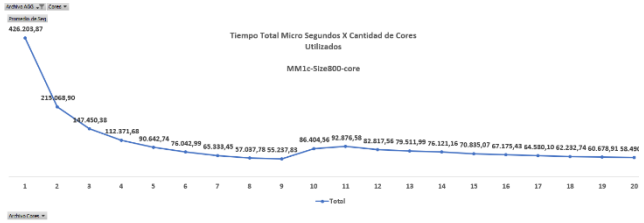


Fig MM1c_FxCSize800 S

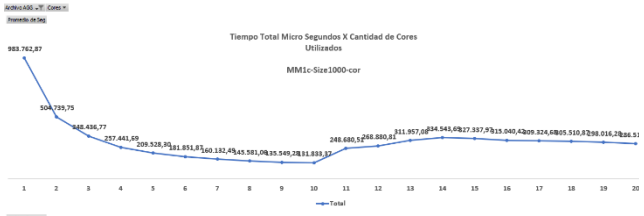


Fig MM1c_FxCSize1000 S

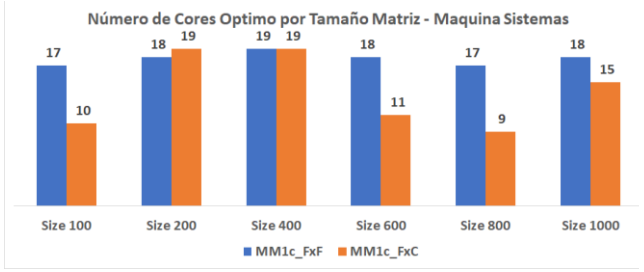
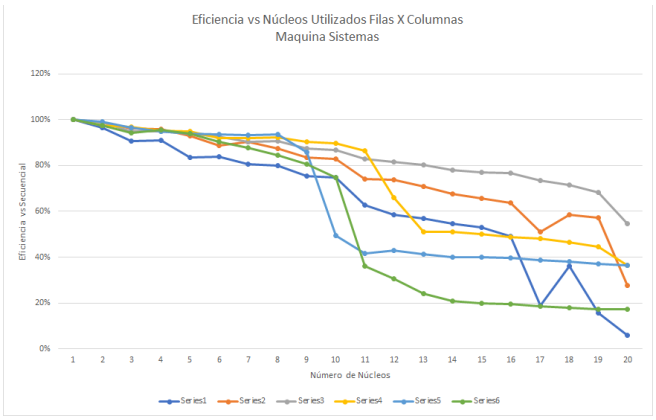
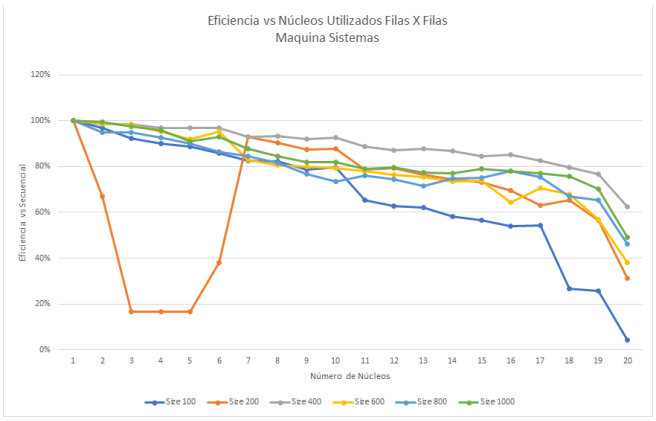


Fig FxC vs FxF Cores Optimos S





Resultados Algoritmo Filas x Filas Maquina Cratos

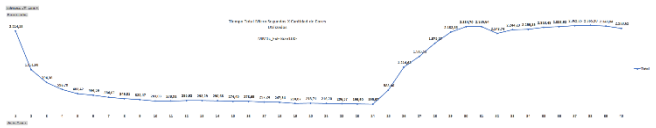


Fig MM1c_FxFSize100 C

Fig MM1c_FxFSize200 C

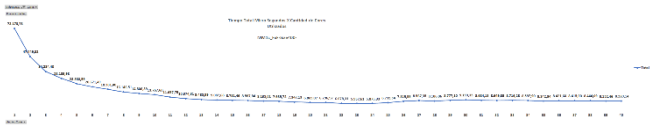


Fig MM1c_FxFSize400 C

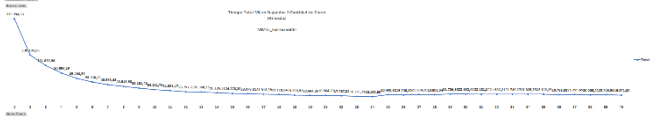


Fig MM1c_FxFSize600 C

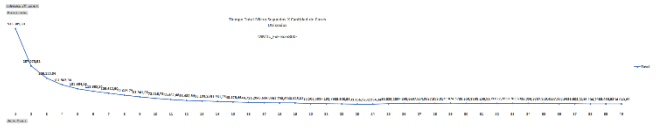


Fig MM1c_FxFSize800 C

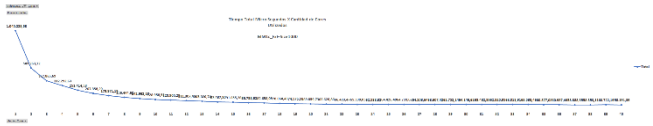


Fig MM1c_FxFSize1000 C

Resultados Algoritmo Filas x Columnas Maquina Cratos

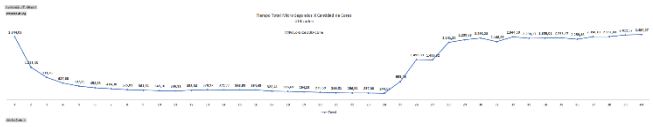


Fig MM1c_FxCSize100 C

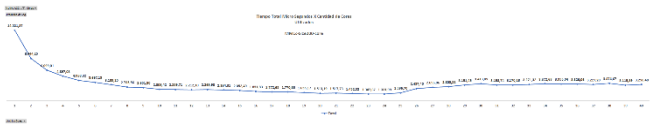


Fig MM1c_FxCSize200 C

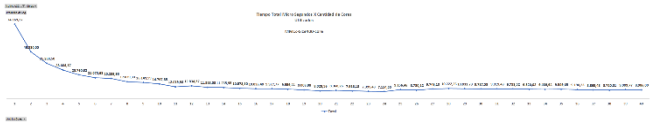


Fig MM1c_FxCSize400 C

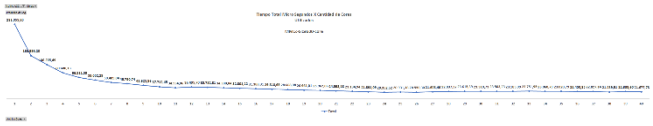


Fig MM1c_FxCSize600 C

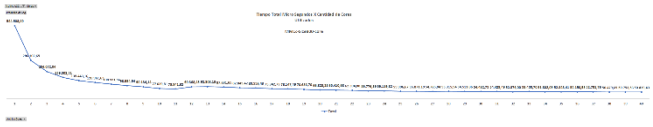


Fig MM1c_FxCSize800 C

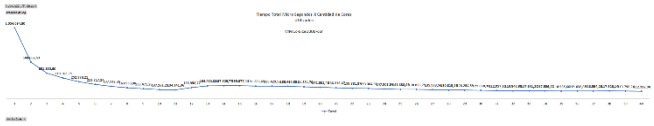


Fig MM1c_FxCSize1000 C

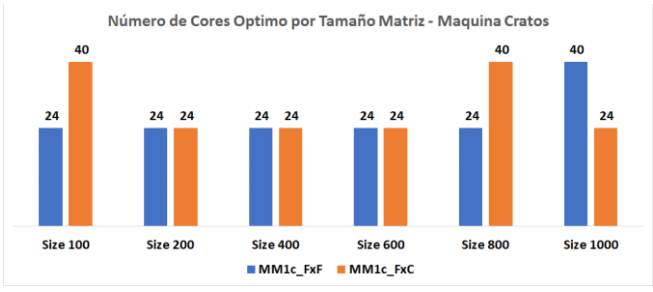
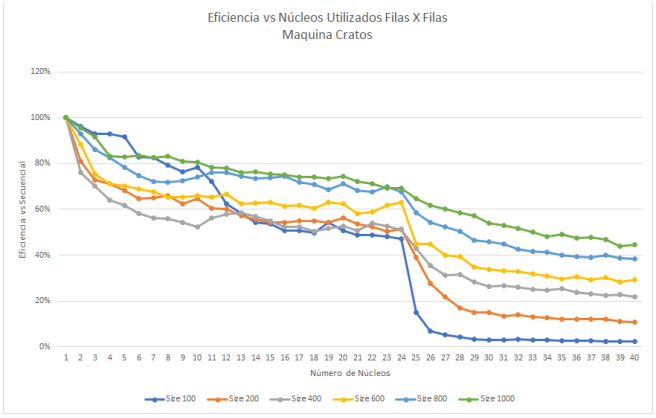
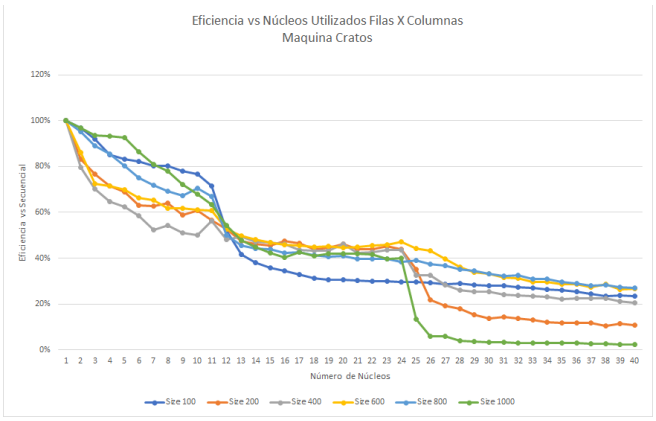


Fig FxC vs FxF Cores Optimos C





Resultados Algoritmo Filas x Columnas Maquina Local

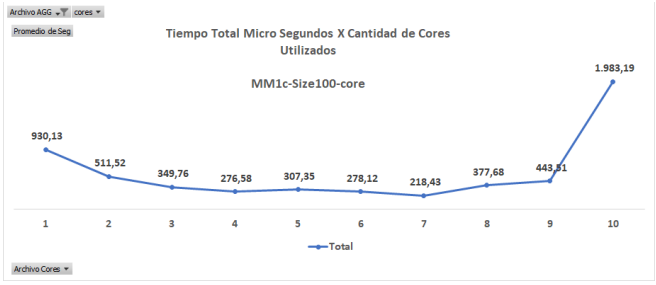


Fig MM1c_FxCSize100 L

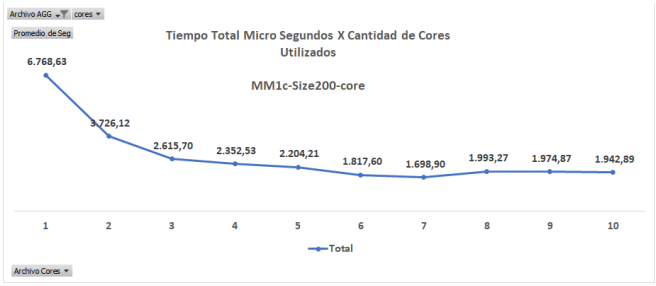


Fig MM1c_FxCSize200 L

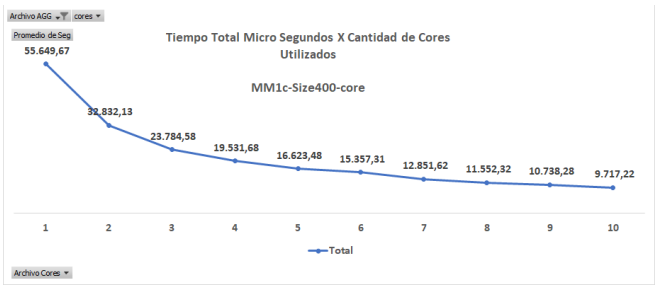


Fig MM1c_FxCSize400 L

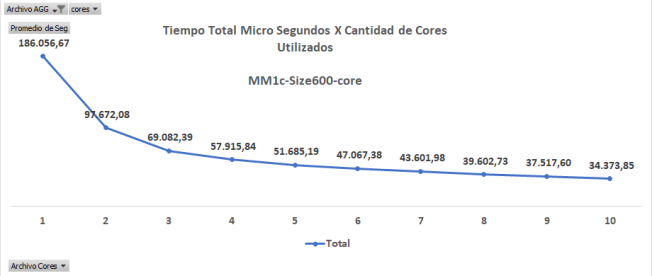


Fig MM1c_FxCSize600 L

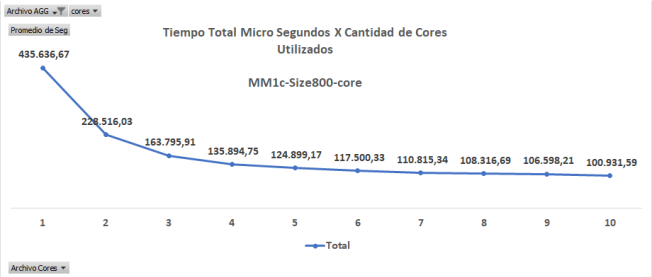


Fig MM1c_FxCSize800 L

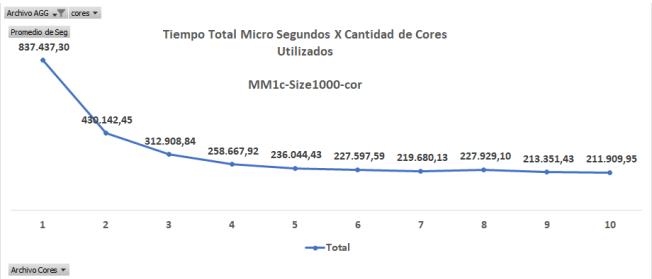


Fig MM1c_FxCSize1000 L

Resultados Algoritmo Filas x Filas Maquina Local

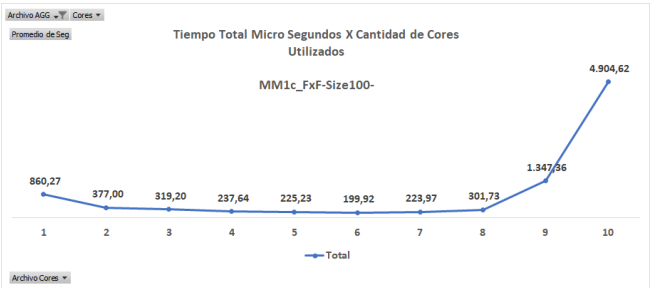


Fig MM1c_FxFSize100 L

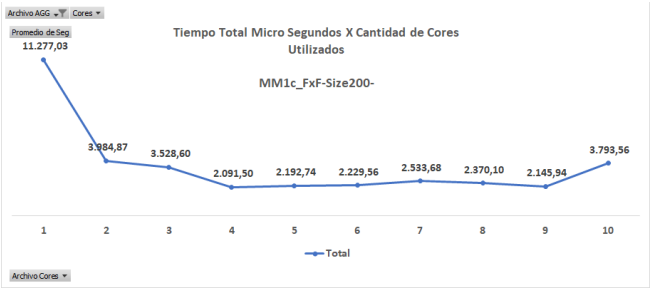


Fig MM1c_FxFSize200 L

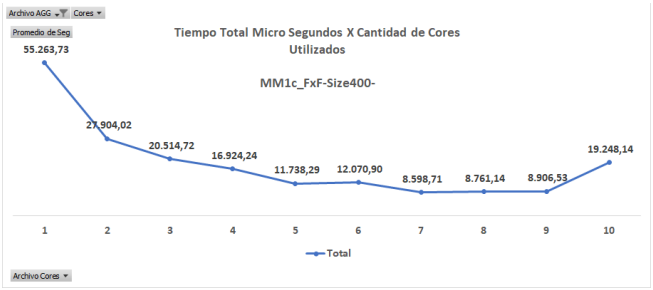


Fig MM1c_FxFSize400 L

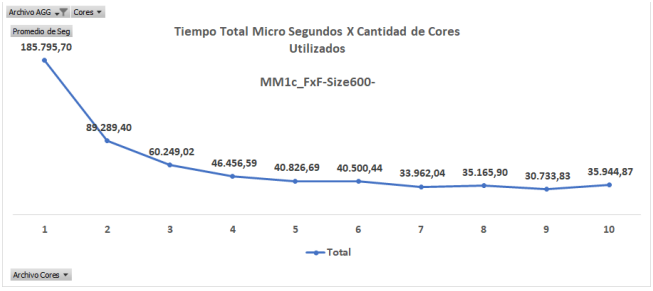


Fig MM1c_FxFSize600 L

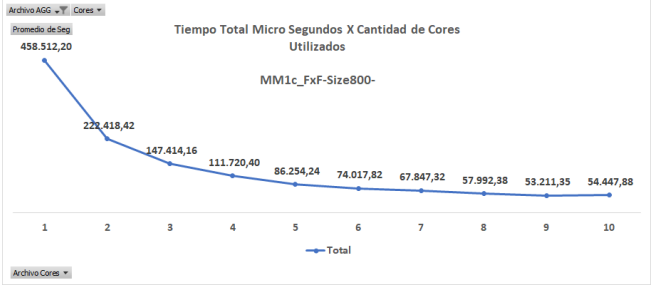


Fig MM1c_FxFSize800 L

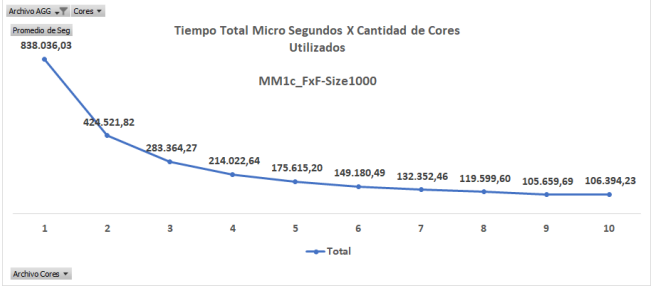


Fig MM1c_FxFSize1000 L

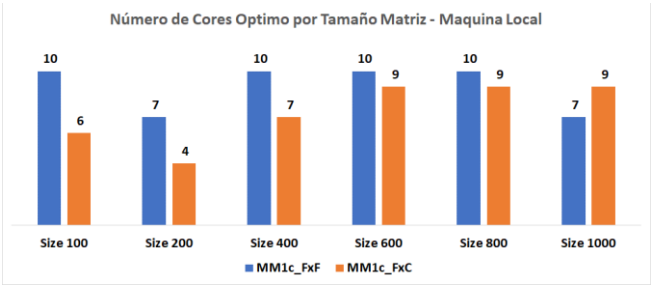
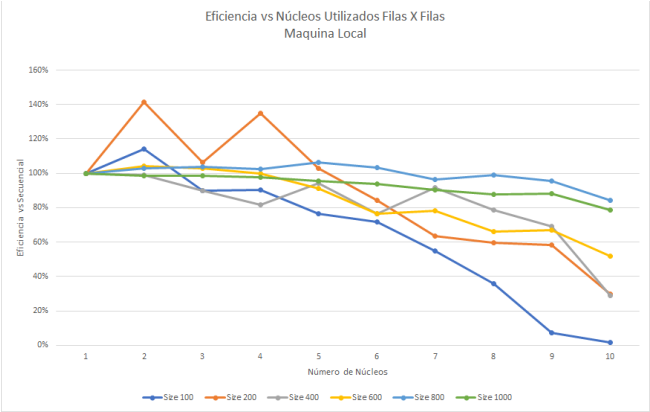
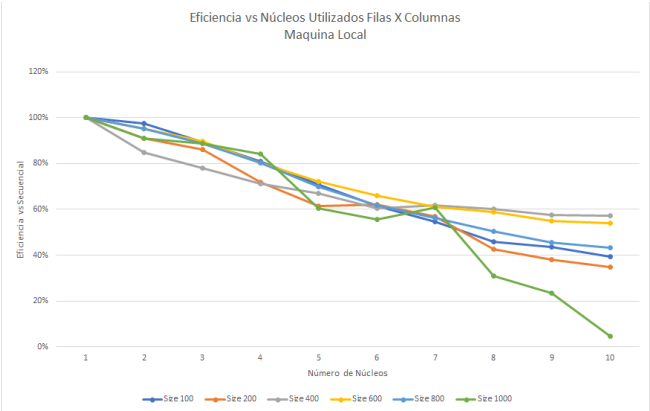


Fig FxC vs FxF Cores Optimos L





- **Se entrega como .Zip con el nombre "Anexo 1!, el cual contine la carpeta con todos los codigos y pruebas realizadas, incluyendo el script de powershell.**

Repositorio Git: Rendimiento OpenMP

No se utilizaron referencias externas para la realización de este trabajo.