SISTEMAS OPERATIVOS

EVALUACIÓN DE RENDIMIENTO

Profesionales en Formación

JUAN FELIPE GALVIS

ANDRÉS PÉREZ

JUAN DIEGO REYES

17 DE MAYO DE 2024

BOGOTÁ D.C

Índice

1. Resumen

El problema abordado en este informe es la evaluación del rendimiento de algoritmos de multiplicación de matrices en diferentes configuraciones y entornos de ejecución. Se propone comparar el rendimiento de dos sistemas de cómputo mediante la multiplicación de matrices, utilizando los paradigmas de computación en serie y en paralelo. La metodología empleada fue experimental, realizando 30 repeticiones para cada configuración de tamaño de matriz y número de hilos, y calculando los promedios de los tiempos de ejecución obtenidos. Los resultados indicaron que el algoritmo de transpuesta en paralelo es superior para matrices grandes, mientras que para matrices no hay un cambio significativo.

2. Introducción

El propósito de este informe es documentar el proceso seguido para evaluar el rendimiento de dos algoritmos de multiplicación de matrices: el clásico y el de transpuesta. Se pretende analizar y comparar la eficiencia de estos algoritmos tanto en ejecución en serie como en paralelo, considerando factores como el tamaño de las matrices y el número de hilos de ejecución. La evaluación se llevó a cabo en entornos con sistemas operativos Linux, y los resultados obtenidos se utilizaron para extraer conclusiones y proporcionar recomendaciones.

3. Contexto sistemas operativos

Para llevar a cabo los experimentos de evaluación de rendimiento, se utilizaron entornos con sistemas operativos Linux. En particular, se emplearon máquinas virtuales con las siguientes especificaciones:

- Hardware:
  - 4 núcleos de CPU
  - 4 GB de memoria RAM

- Software:
  - Sistema operativo: Linux Ubuntu 20.04 LTS

Estos entornos permitieron realizar pruebas controladas y reproducibles, asegurando que los resultados fueran consistentes y comparables.

4. Comparación experimental de sistemas de cómputo

Se compararon diferentes sistemas de cómputo para evaluar su rendimiento al ejecutar los algoritmos de multiplicación de matrices. Los sistemas seleccionados para la comparación incluyeron tanto máquinas virtuales como sistemas nativos con diferentes configuraciones de hardware. Los principales aspectos evaluados fueron la eficiencia en el uso de los recursos del sistema y el tiempo de ejecución de los algoritmos.

Comparación	Linux	Windows
Características	Código abierto, terminal de línea de comandos, menor consumo de recursos, flexibilidad y personalización	Interfaz gráfica, orientado a usuarios domésticos y empresariales, costo de licencias, cerrado
Conclusiones en el contexto	Para la comparación de los algoritmos de multiplicación de matriz, Linux es el más correcto, debido a su eficiencia en la ejecución y compilación, por su bajo consumo en recursos es la mejor opción, la manera en la que las salidas de los archivos es muy rápida sin importar el tamaño de la mtriz o número de hilos.	Para poder utilizar windows en este contexto, es muy complicada su compilación en C, teniendo en cuenta la generación de archivos, puede ser muy pesado y poco eficiente, no obstante por medio de replit en un navegador web es posible hacer su compilación, sin embargo, no es ideal para este caso.

Tabla inspirada en: (Cuadro Comparativo de los Sistemas Operativos Windows y LINUX, 2023)

5. Comparación basada en implementación de Algoritmos de Multiplicación de Matrices

La implementación de los algoritmos de multiplicación de matrices se realizó en el leguaje C. Se eligió C por su alto desempeño y eficiencia El lenguaje de programación C es conocido por su alto desempeño y eficiencia. C se compila directamente a código máquina, lo que resulta en una ejecución más rápida en comparación con los lenguajes interpretados. Además, C proporciona un control detallado sobre los recursos del sistema, como la gestión de memoria y el uso del procesador, y permite realizar operaciones de bajo nivel, como manipular directamente punteros y registros, lo cual es crucial para maximizar la eficiencia en tareas computacionalmente intensivas como la multiplicación de matrices. De igual manera, C soporta el paralelismo a través de diversas bibliotecas y extensiones, como pthreads. Estas herramientas permiten escribir programas que pueden aprovechar múltiples núcleos y procesadores, lo que es esencial para mejorar el rendimiento en la multiplicación de matrices de gran tamaño.

En esta investigación se utilizaron dos algoritmos de multiplicación de matrices: el clásico y el de la transpuesta. El algoritmo clásico de multiplicación de matrices realiza la multiplicación de

matrices de forma directa, iterando sobre las filas de la primera matriz y las columnas de la segunda. El algoritmo clásico de multiplicación de matrices es un método fundamental que toma dos matrices, A y B, y produce una tercera matriz C tal que C = A*B. Este algoritmo es aplicable cuando el número de columnas de la matriz A es igual al número de filas de la matriz B. Si A es de tamaño m*n, y B es de tamaño n*p, la matriz resultante C será de tamaño m*p. Este algoritmo tiene una complejidad computacional de $O(n^3)$. (Universidad Nacional de Colombia, s. f.).

El algoritmo de la transpuesta para la multiplicación de matrices es una variación del algoritmo clásico que mejora la eficiencia del acceso a la memoria. Al transponer una de las matrices antes de la multiplicación, se pueden realizar accesos secuenciales a la memoria, lo que es más eficiente que los accesos dispersos típicos del algoritmo clásico. Este algoritmo funciona transponiendo la matriz B antes de realizar la multiplicación, lo que permite realizar accesos secuenciales a la memoria en lugar de accesos dispersos. Esta mejora es significativa en sistemas con jerarquías de memoria caché. La complejidad computacional de este algoritmo es de O(n*m*p), siendo m, n, y p los tamaños de las matrices como se explicaron en el algoritmo de multiplicación clásica. El rendimiento práctico puede ser significativamente mejor en sistemas modernos con jerarquías de memoria caché, debido a la mejora en la eficiencia del acceso a la memoria. (Rodó, 2022).

6. Paradigmas de computación Serie Paralelo

Se analizaron los paradigmas de computación en serie y en paralelo para entender cómo afectan el rendimiento de los algoritmos de multiplicación de matrices.

Computación en serie: En este paradigma, las operaciones se realizan de manera secuencial, una tras otra. Aunque es más sencillo de implementar, puede ser ineficiente para tareas que pueden beneficiarse del paralelismo, para el caso de las matrices 1 hilo.

Computación en paralelo: En este paradigma, las operaciones se dividen en múltiples tareas que se ejecutan simultáneamente en diferentes hilos o procesadores. Esto puede reducir significativamente el tiempo de ejecución, especialmente para tareas intensivas en cálculos como la multiplicación de matrices ya que podemos usar de 2 hasta n hilos.

Se implementaron ambas versiones de los algoritmos y se compararon los tiempos de ejecución para diferentes configuraciones en la sección 9 y sección10 de este documento.

(Tinetti & Denham, s. f.)

7. Metodología experimental basada en los grandes números

La metodología experimental se basó en la ley de los grandes números bajo la teoría de la probabilidad, para obtener resultados promediados y representativos. Esta ley indica que el promedio de una muestra de números en una sucesión se puede relacionar entre sí, probado por Bernoulli en 1713. Para cada combinación de tamaño de matriz y número de hilos, se realizaron 30 repeticiones de los experimentos. Esto permitió minimizar el impacto de variaciones aleatorias y obtener una medida más precisa del rendimiento.

Procedimiento (mostrados exhaustivamente en la sección 9):

- Se compilaron los archivos fuente utilizando GCC.

- Se ejecutaron los algoritmos para diferentes tamaños de matrices y números de hilos.

- Los resultados de cada ejecución se registraron y promediaron para obtener valores representativos como se muestra en la sección 9.

(*La Ley de los Grandes Números*, s. f.)

8. Herramientas/plataformas utilizadas

Se hizo uso de diferentes plataformas y herramientas, las cuales se describirán a continuación.

Se utilizó una máquina virtual con sistema operativo Ubuntu para correr los scripts y programas necesarios para la evaluación del rendimiento de los algoritmos de multiplicación de matrices. a máquina virtual estaba configurada con una CPU de 4 núcleos, memoria RAM de 4 GB y un sistema operativo Ubuntu 20.04 LTS. La VM se empleó para ejecutar el script en Perl que automatiza la ejecución de los algoritmos de multiplicación de matrices y para obtener los resultados de los tiempos de ejecución.

De igual manera se utilizó Replit. La cual es una plataforma de desarrollo en línea que permite escribir, compilar y ejecutar código en múltiples lenguajes de programación. En este proyecto, Replit se utilizó para desarrollar y probar los algoritmos de multiplicación de matrices en lenguaje C. Para facilitar la escritura y depuración del código en C en un entorno colaborativo y accesible desde cualquier lugar con conexión a Internet.

También se utilizó un script en Perl para automatizar la ejecución de los algoritmos de multiplicación de matrices. El script Perl se encargaba de ejecutar cada algoritmo 30 veces con diferentes tamaños de matrices y números de hilos especificados. Se usó este script para automatizar la repetición de los experimentos para capturar datos de rendimiento consistentes y reducir el error humano. El script Perl generó archivos de resultados que contienen los tiempos de ejecución para cada configuración de tamaño de matriz y número de hilos.

Además, se creó un programa en C++ para procesar los resultados. Es te programa en C++ lee los archivos de resultados generados por el script Perl. El programa calcula los promedios de los tiempos de ejecución de cada configuración de tamaño de matriz y número de hilos y guardaba estos promedios en archivos CSV. El resultado de este programa son unos archivos CSV que contienen los promedios de los tiempos de ejecución, listos para ser importados en herramientas de análisis y visualización de datos.

Otra herramienta utilizada es Microsoft Excel. Microsoft Excel es una aplicación de hoja de cálculo que se utilizó para analizar y visualizar los datos obtenidos de los experimentos. Los archivos CSV generados por el programa en C++ se importaron a Excel para crear tablas y gráficos. Esto facilitó el análisis de los resultados promediados y permitió crear diagramas y tablas que permiten una interpretación visual clara de los datos y comparaciones entre las diferentes configuraciones de los algoritmos.

Estas herramientas y plataformas trabajaron en conjunto para facilitar el desarrollo, ejecución, automatización, procesamiento y análisis de los experimentos de rendimiento de los algoritmos de multiplicación de matrices.

9. Análisis exhaustivo de los datos

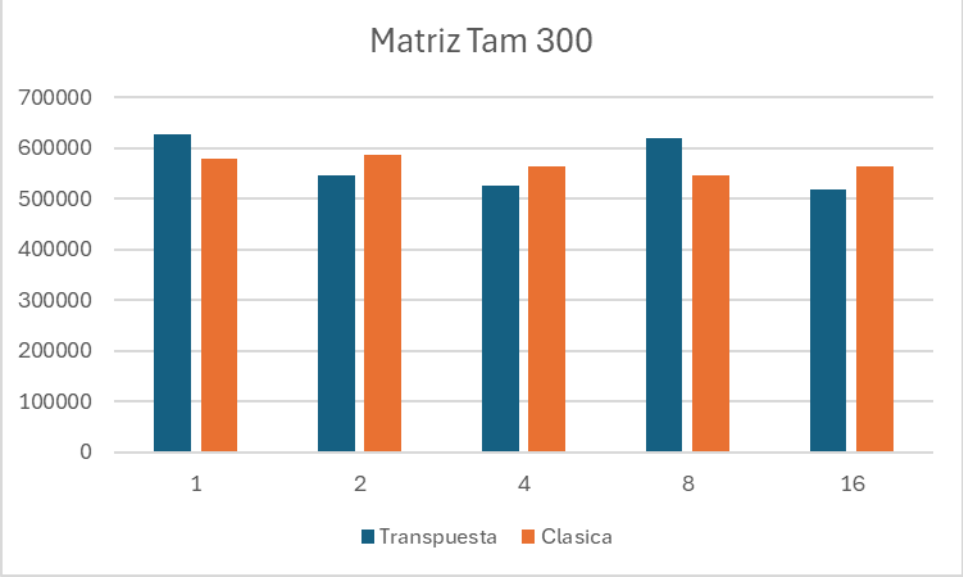
Los datos obtenidos de los experimentos se analizaron haciendo uso del script lanzar.pl que correrá repetidamente los programas a probar en distintas configuraciones de matrices y de hilos introduciendo distintos argumentos de entrada, almacenando las salidas de estos en archivos de texto nombrados según los argumentos usados. A continuación, se presentan algunos de los resultados más relevantes:

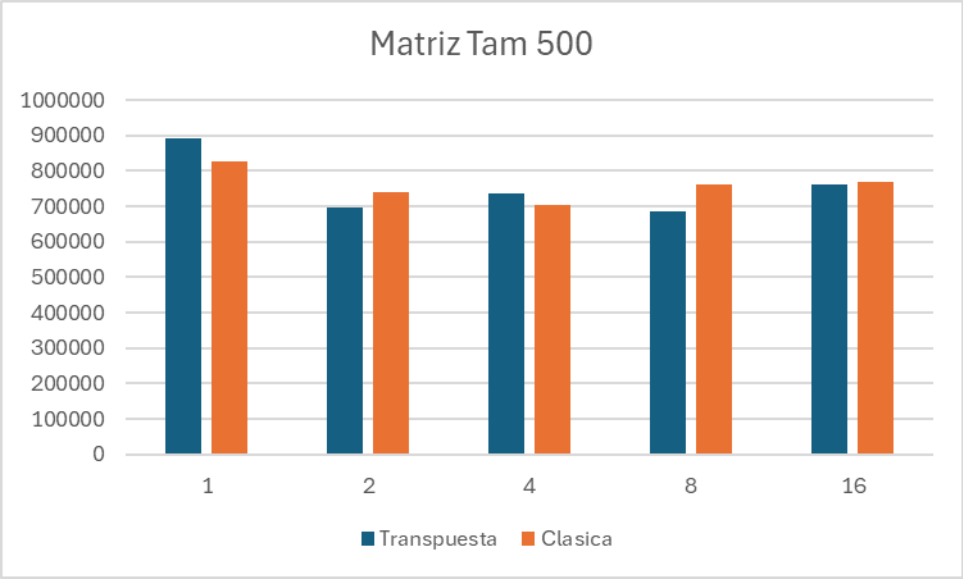
- Tiempos de Ejecución:
    - Se compararán los tiempos de ejecución de ambos algoritmos para diferentes tamaños de matrices y números de hilos para validar que tanto impacto tenían estos sobre el resultado final.
    - Estos tiempos son obtenidos de los archivos de texto generados por el script lanzar.pl
- Repetición:
    - En los archivos con la información de los tiempos se encuentran registrados estos tiempos en múltiples ejecuciones del programa con los mismos argumentos para tener más certeza de que los datos son precisos.
    - Una vez recibidos estos datos se calcula el promedio de estos para poder establecer sobre qué tiempos se suele mantener el programa en cierta configuración.
- Tablas y Gráficos:

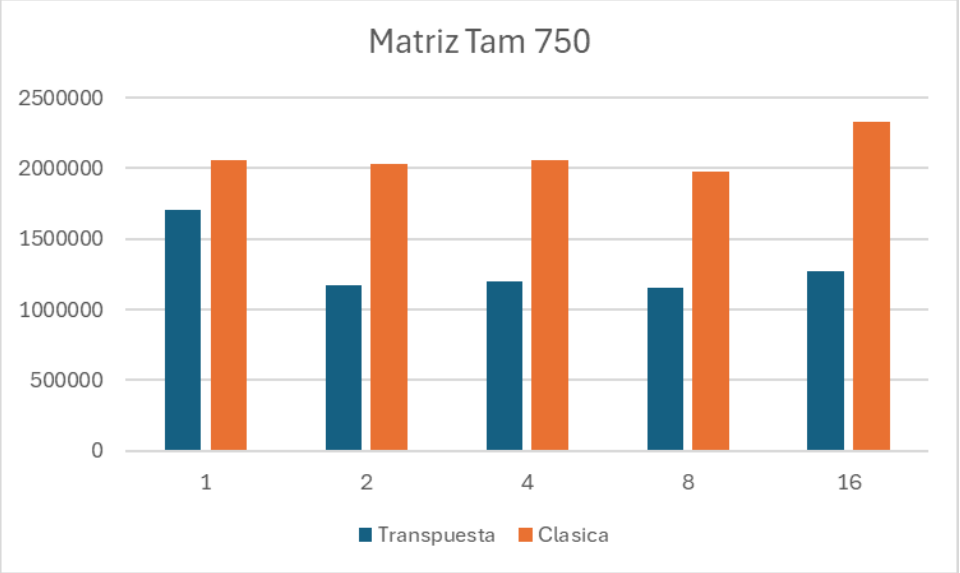
- Una vez calculados los promedios se realiza una tabla para los dos programas que marca el tiempo promedio de estos en cada configuración de hilos y tamaño de matriz.

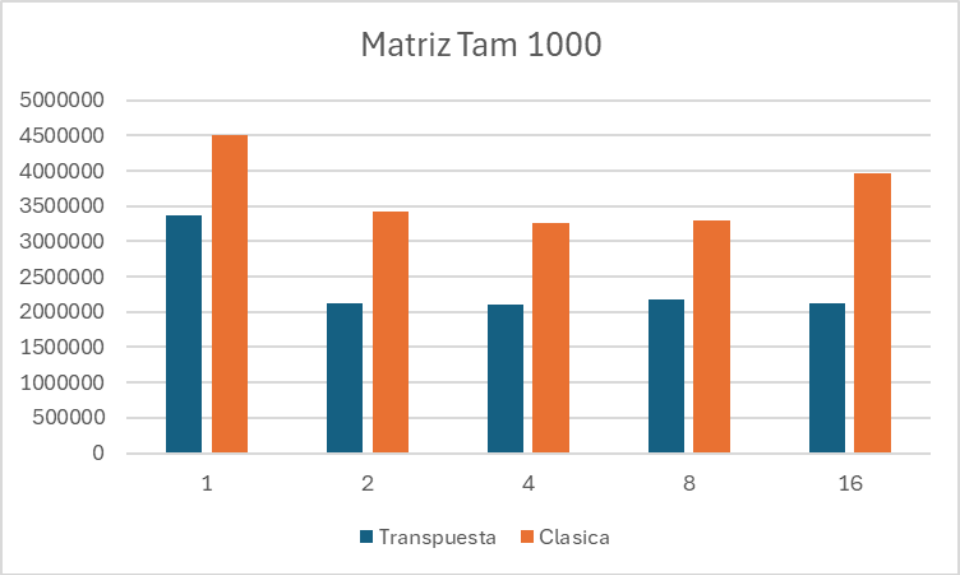
10. Presentación de resultados

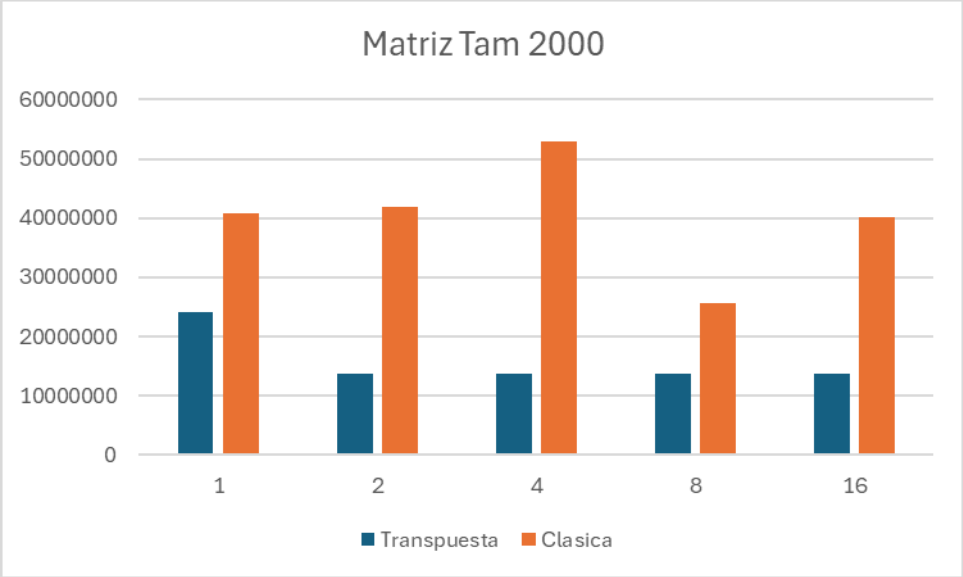
Aquí se ven graficados en la coordenada y los tiempos promedio de ejecución en microsegundos y en la x el número de hilos usados para un tamaño concreto de matriz.

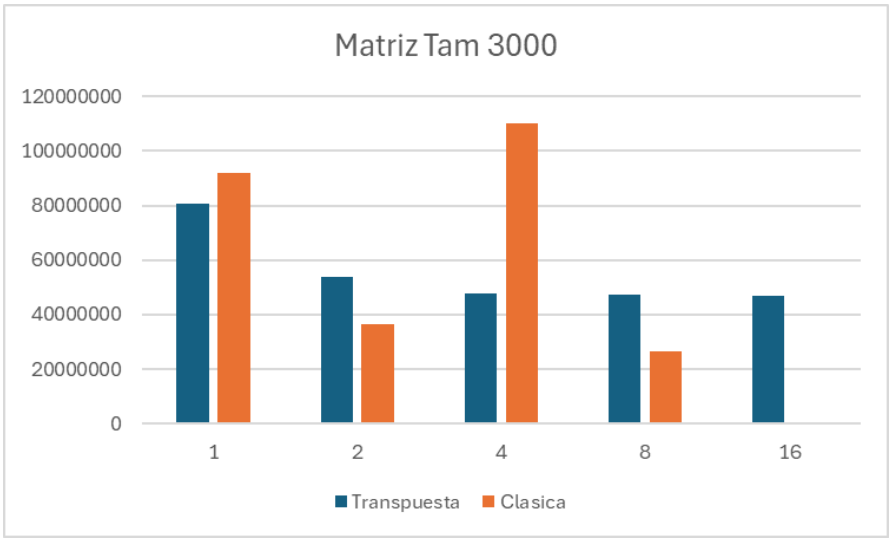


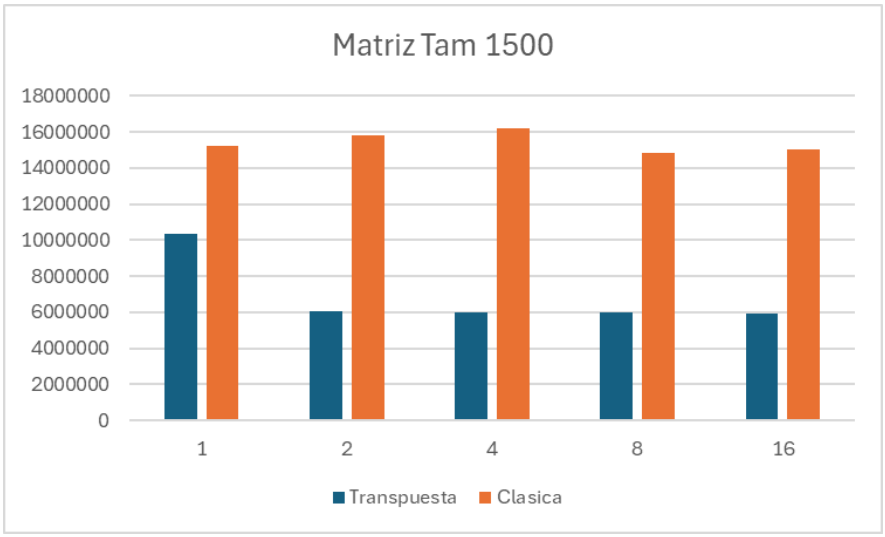

Matriz Tam 500
1000000
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
1
2
4
8
16
Transpuesta
Clasica

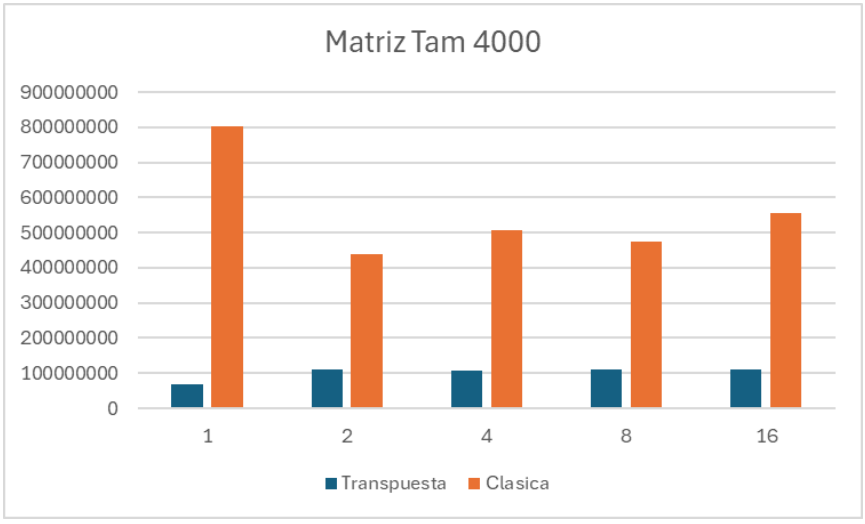

Matriz Tam 750
2500000
2000000
1500000
1000000
500000
0
1
2
4
8
16
Transpuesta
Clasica


Matriz Tam 1000
5000000
4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
1
2
4
8
16
Transpuesta
Clasica


Matriz Tam 2000
60000000
50000000
40000000
30000000
20000000
10000000
0
1
2
4
8
16
Transpuesta
Clasica


Matriz Tam 3000
120000000
100000000
80000000
60000000
40000000
20000000
0
1
2
4
8
16
Transpuesta
Clasica


Matriz Tam 1500
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
1
2
4
8
16
Transpuesta
Clasica


Matriz Tam 4000
900000000
800000000
700000000
600000000
500000000
400000000
300000000
200000000
100000000
0
1
2
4
8
16
Transpuesta
Clasica

11. Conclusiones

- Eficiencia del Algoritmo de Transpuesta: El algoritmo de transpuesta para la multiplicación de matrices demostró ser más eficiente que el algoritmo clásico, especialmente cuando se trabajó con matrices de gran tamaño. Esta eficiencia se debe a la mejora en el acceso a la memoria que permite usar secuenciales de datos, optimizando el rendimiento en sistemas con jerarquías de memoria caché.
- Impacto del Tamaño de la Matriz: Para matrices pequeñas (menores a 1000x1000), las diferencias en rendimiento entre el algoritmo clásico y el de transpuesta no fueron significativas. Sin embargo, a medida que el tamaño de la matriz aumentaba, el algoritmo de transpuesta mostró un rendimiento considerablemente mejor. Esto se debe a que los accesos dispersos a la memoria en el algoritmo clásico se vuelven más costosos a medida que la matriz crece en tamaño.
- Ventajas del Paralelismo: La implementación de ambos algoritmos en paralelo demostró una mejora significativa en los tiempos de ejecución. El uso de múltiples hilos permitió reducir el tiempo de cálculo, demostrando la efectividad del paralelismo en tareas computacionalmente intensivas como la multiplicación de matrices. En algunos casos, el tiempo de ejecución se redujo hasta cinco veces cuando se utilizaron 16 hilos en comparación con un solo hilo.
- Utilización de Recursos: Aunque el uso de múltiples hilos mejoró el rendimiento, también incrementó la utilización de recursos del sistema, especialmente en términos de CPU. Es importante balancear el número de hilos utilizados para evitar un uso excesivo de recursos que pueda afectar el rendimiento general del sistema.

- Variabilidad de los Resultados: La metodología experimental basada en la ley de los grandes números permitió obtener resultados más representativos al promediar los tiempos de ejecución sobre múltiples repeticiones. Esto ayudó a minimizar el impacto de variaciones aleatorias y proporcionó una medida más precisa del rendimiento de los algoritmos.

12. Recomendaciones

- Ampliar el Rango de Experimentos: Es recomendable realizar más experimentos con diferentes configuraciones de hardware y software para validar los resultados obtenidos. Esto incluye probar en sistemas con diferentes arquitecturas de CPU, tamaños de memoria y configuraciones de caché.
- Explorar Optimizaciones Adicionales: Se deben explorar posibles optimizaciones adicionales en los algoritmos, como el uso de técnicas de bloqueo de matrices para mejorar aún más la eficiencia del acceso a la memoria y reducir el tiempo de ejecución.

- Utilizar Bibliotecas Especializadas: Considerar el uso de bibliotecas y frameworks paralelos específicos, como BLAS (Basic Linear Algebra Subprograms) y OpenMP, que están optimizados para operaciones de álgebra lineal y pueden mejorar la eficiencia en aplicaciones del mundo real.
- Análisis de Costos y Beneficios del Paralelismo: Realizar un análisis de costos y beneficios del uso de paralelismo en términos de utilización de recursos del sistema y el impacto en otros procesos que puedan estar ejecutándose en paralelo. Esto ayudará a determinar el número óptimo de hilos a utilizar.

13. Repositorio Personal

https://github.com/felo312/Sistemas-Operativos/tree/main/Evaluaci%C3%B3nRendimiento

14. Referencias

- Cuadro comparativo de los sistemas operativos Windows y LINUX. (2023, 10 diciembre). [Diapositivas]. SlideShare. https://es.slideshare.net/slideshow/cuadro-comparativo-de-los-sistemas-operativos-windows-y-linux/264513012
- Rodó, P. (2022, 24 noviembre). *Matriz traspuesta*. Economipedia. https://economipedia.com/definiciones/matriz-traspuesta.html
- Universidad Nacional de Colombia. (s. f.). *Universidad Nacional de Colombia: Clase 2. Parte 2. Multiplicación de matrices*. https://ciencias.medellin.unal.edu.co/cursos/algebra-lineal/clases/8-clases/39-clase-2-parte2.html
- Tinetti, F., & Denham, M. (s. f.). Paralelización y Speedup Superlineal en Supercomputadoras Ejemplo con Multiplicación de Matrices. *Unlp*.
- *La ley de los grandes números*. (s. f.). EGADE. https://egade.tec.mx/es/egade-ideas/opinion/la-ley-de-los-grandes-numeros