

**Taller evaluación de Rendimiento**



**Profesionales en Formación**

Juan Esteban Páez Alfonso  
Carlos Esteban Mejía Coy

**Pontificia Universidad Javeriana  
Facultad de ingeniería**

**Sistemas Operativos  
Bogota D.C.  
2024**

## **Índice**

**0.- Resumen**

**1.- Introducción**

**2.- Las ciencias computacionales**

**3.- Evaluación de rendimiento (ejecución de una aplicación de juguete:benchamark)**

**4.- Ejecución de 2 benchmarks basados en el algoritmo de Multiplicación de Matrices en diferentes Sistemas de Cómputo**

**5.- Uso del paradigma de programación: serie y paralelo**

**6.- Metodología de Experimentación**

**7.- Evaluación exhaustiva (hoja de cálculo)**

**8.- Conclusiones y recomendaciones**

**9.- Referencias**

**0.-Resumen**

En un mundo cada vez más dependiente de la capacidad computacional, la optimización de algoritmos se vuelve crucial. En este contexto, el taller de Evaluación de Rendimiento de Multiplicación de Matrices plantea una exploración profunda sobre cómo el paralelismo puede impactar el rendimiento de algoritmos básicos pero fundamentales. La comparación entre ejecuciones serial y paralela del algoritmo de multiplicación de matrices no solo despierta la curiosidad sobre las diferencias en la eficiencia computacional, sino que también abre la puerta a un mundo de posibilidades en términos de optimización y rendimiento. Con una metodología rigurosa que incluye experimentación sistemática y el uso de leyes probabilísticas, se busca no solo comprender, sino también mejorar el rendimiento de los algoritmos en diferentes sistemas de cómputo. Los resultados obtenidos prometen ofrecer una visión más clara sobre cómo aprovechar al máximo los recursos disponibles, con potencial para impactar diversas áreas que dependen del procesamiento de grandes cantidades de datos. En última instancia, este taller no solo ofrece respuestas sobre cómo mejorar la multiplicación de matrices, sino que también plantea nuevas preguntas sobre cómo podemos optimizar algoritmos esenciales para impulsar el rendimiento computacional en el futuro.

## **1.- Introducción**

El taller de Evaluación de Rendimiento de Multiplicación de Matrices surge como una oportunidad para explorar las complejidades y desafíos inherentes a la optimización de algoritmos básicos. Este taller se sumerge en el análisis comparativo entre ejecuciones secuenciales y paralelas del algoritmo de multiplicación de matrices, desentrañando las implicaciones de esta dicotomía en el rendimiento computacional. Desde la meticulosa metodología de experimentación, que abarca desde la aplicación de leyes probabilísticas hasta el uso de lenguajes de programación como C y Perl, hasta la búsqueda de resultados significativos que arrojen luz sobre la eficacia de los enfoques paralelos en diferentes sistemas de cómputo, este taller no solo promete una inmersión profunda en la mecánica de los algoritmos, sino que también promueve una reflexión sobre el potencial transformador de la programación paralela en la optimización del rendimiento computacional.

## **2.- Las ciencias computacionales ¿Cuál elegir?**

Los sistemas operativos (SO) son programas informáticos que gestionan los recursos hardware y software de un ordenador, proporcionando una plataforma para la ejecución de aplicaciones y el desarrollo de tareas por parte de los usuarios. Son el núcleo fundamental del funcionamiento de cualquier dispositivo informático, desde computadoras personales hasta servidores y dispositivos móviles. Los sistemas operativos son componentes esenciales de cualquier dispositivo informático. Su evolución ha sido fundamental para el desarrollo de la informática y su futuro seguirá marcado por las nuevas tecnologías y las necesidades cambiantes de los usuarios.

En el mundo de la informática actual, macOS y Linux como dos de los sistemas operativos más populares para ordenadores personales, cada uno con sus propias ventajas y desventajas. Si bien ambos persiguen el mismo objetivo fundamental: facilitar la interacción entre el usuario y el hardware del equipo, difieren en aspectos cruciales que los convierten en opciones atractivas para distintos tipos de usuarios.

### **Linux: El poder de la libertad y la personalización**

Linux se caracteriza por ser un sistema operativo de código abierto, lo que significa que su código fuente está disponible para que cualquier persona pueda examinarlo, modificarlo y contribuir a su desarrollo. Esta filosofía ha dado lugar a una amplia gama de distribuciones de Linux, cada una adaptada a las necesidades y preferencias de diferentes usuarios.

Estabilidad, eficiencia y personalización son las tres palabras que mejor definen a Linux. Su robustez lo convierte en una opción popular para servidores y sistemas embebidos, donde la confiabilidad es primordial. Además, su naturaleza personalizable permite a los usuarios ajustar el sistema a sus gustos y necesidades específicas, desde la apariencia hasta el funcionamiento interno.

Sin embargo, esta libertad de personalización puede presentar un desafío para usuarios con menor conocimiento técnico, ya que requiere un mayor dominio del sistema operativo para realizar configuraciones avanzadas.

### **MacOS: La elegancia y sencillez de la experiencia Apple**

En el otro extremo está MacOS, un sistema operativo de código cerrado desarrollado por Apple Inc. Al contrario que Linux, su código fuente no es accesible al público, así que las modificaciones y el desarrollo dependen solo de la empresa.

La interfaz de usuario de macOS es reconocida por su simplicidad e intuitividad, lo que la convierte en una opción atractiva para usuarios que buscan una experiencia de uso fluida y accesible. Su estrecha integración con el ecosistema de productos Apple garantiza una compatibilidad impecable y una experiencia homogénea en todos sus dispositivos.

No obstante, el carácter cerrado de macOS limita la personalización del sistema en comparación con Linux. Además, su precio elevado y la exclusividad del hardware de Apple pueden ser factores determinantes para algunos usuarios.

### **En busca del equilibrio perfecto**

En definitiva, la elección entre macOS y Linux depende en gran medida de las necesidades y preferencias de cada usuario. Si se toma valor en la libertad de personalización, la eficiencia y la estabilidad, Linux puede ser tu opción ideal. Por otro lado, si se busca una experiencia de usuario sencilla, una amplia compatibilidad con software y hardware, y la elegancia del ecosistema Apple, macOS podría ser la mejor alternativa.

## **3.- Evaluación de rendimiento**

Para la evaluación del rendimiento, se ejecutaron los algoritmos de multiplicación de matrices implementados en los archivos "mm\_clasico.c" y "mm\_traspuesta.c" utilizando diferentes configuraciones de tamaño de matriz y número de hilos. Este proceso involucró la creación de un benchmark, o aplicación de prueba, diseñada para medir el tiempo de ejecución de los algoritmos en varias condiciones controladas.

### **1. Configuraciones de prueba:**

- Sistemas de cómputo: Se realizaron las pruebas en dos sistemas operativos diferentes: macOS y Linux, para comparar su rendimiento en la ejecución de los algoritmos.
- Tamaños de matrices: Se seleccionaron matrices de diferentes tamaños para observar cómo el tamaño de la matriz afecta el tiempo de ejecución.

- Número de hilos: Se varió el número de hilos para evaluar el impacto de la paralelización en el rendimiento.

## **2. Ejecuciones repetidas:**

- Cada configuración (combinación de tamaño de matriz y número de hilos) se ejecutó 30 veces para asegurar que las mediciones de tiempo sean representativas y minimizar la influencia de factores aleatorios. Esta práctica se basa en la ley de los grandes números, el cual establece que, con un mayor número de ensayos, el promedio de los resultados se aproxima al valor esperado.

## **3. Automatización:**

- Se utilizó un script en Perl ("lanza.pl") para automatizar la ejecución de las pruebas. Este script configuró y ejecutó las pruebas para cada combinación de parámetros, recopilando los resultados en archivos de salida (.dat).
- Además, para la recolección de datos y agrupación por promedios, se realizó un programa en JavaScript, en donde toma el nombre de los archivos por parámetro, los ordena por tamaño de matriz y número de hilos de ejecución en una hoja de cálculo.

## **Resultados y Análisis**

### **• Tiempo de ejecución:**

Se midió el tiempo de ejecución total para cada prueba utilizando las funciones de tiempo proporcionadas en los códigos fuente. Los tiempos fueron recopilados y promediados para cada configuración.

Los resultados mostraron cómo el tiempo de ejecución varía con el aumento del tamaño de la matriz y el número de hilos, proporcionando una visión clara de la escalabilidad y eficiencia de cada algoritmo en diferentes condiciones.

### **• Eficiencia de paralelización:**

Comparando los tiempos de ejecución con diferentes números de hilos, se pudo determinar la eficiencia de paralelización de cada algoritmo. Idealmente, el tiempo de ejecución debería disminuir a medida que se incrementa el número de hilos, mostrando una mejora en el rendimiento debido al procesamiento paralelo.

Sin embargo, se observaron límites en la eficiencia, ya que después de cierto punto, agregar más hilos no necesariamente conduce a una reducción proporcional en el tiempo de ejecución debido a la sobrecarga de gestión de hilos y otras limitaciones del sistema. Este caso se pudo visualizar principalmente al realizar la ejecución de los programas con matrices de mayor tamaño.

## **4.- Ejecución de 2 benchmarks basados en el algoritmo de Multiplicación de Matrices en diferentes Sistemas de Cómputo**

Para evaluar el rendimiento de los algoritmos de multiplicación de matrices, se llevaron a cabo pruebas en dos sistemas de cómputo diferentes: Sistema de Cómputo Asistido (SCA) y Sistema de Cómputo Virtualizado (SCV). Estos sistemas fueron seleccionados para analizar cómo diferentes arquitecturas y entornos afectan la eficiencia de los algoritmos implementados.

SO	MacOS	Linux
<b>RAM</b>	18	16
Hilos	11	4

Tabla de Hardware

## Sistemas de Cómputo Evaluados

### 1. Sistema de Cómputo Asistido (SCA):

- Hardware: Procesador de alta velocidad con múltiples núcleos físicos y amplio soporte para operaciones paralelas.
- Software: MacOS con soporte nativo para Pthreads, optimizado para aplicaciones de alto rendimiento.
- Configuración: Matrices de tamaño 200x200 y 300x300 400x400 600x600 1000x1000, con pruebas realizadas utilizando 1,2,3,4,5,6,7,8,9,10,11 hilos. Además de pruebas de comparación de 1000x1000 y 2000x2000.

### 2. Sistema de Cómputo Virtualizado (SCV):

- Hardware: Recursos compartidos en una máquina virtualizada, con núcleos virtuales y limitaciones de acceso a hardware físico.
- Software: Linux en un entorno de virtualización, con soporte para Pthreads y herramientas de gestión de recursos.
- Configuración: Similares a las del SCA para mantener consistencia en las comparaciones, con matrices de tamaño 200x200, 300x300, 400x400, 600x600 y 1000x1000 utilizando 1, 2, 3, 4 hilos.

## Metodología de Ejecución

### • Preparación del entorno:

En ambos sistemas, se configuraron y compilaron los algoritmos de multiplicación de matrices (`mm_clasico.c` y `mm_traspuesta.c`) utilizando las mismas herramientas de compilación y bibliotecas.

Se aseguraron condiciones de ejecución similares en términos de carga del sistema y disponibilidad de recursos.

### • Ejecución de benchmarks:

Cada configuración (tamaño de matriz y número de hilos) se ejecutó 30 veces para obtener un promedio representativo del tiempo de ejecución, los tiempos de ejecución se

registraron utilizando las funciones de tiempo implementadas en los códigos fuente y se almacenaron en archivos de salida para su análisis posterior.

- **Automatización:**

Se utilizó el script de automatización "lanza.pl" para facilitar la ejecución de múltiples pruebas y la recolección de datos en ambas plataformas.

## **5.- Uso del paradigma de programación: serie y paralelo**

El paradigma de programación serie y paralelo se emplea en el taller de evaluación de rendimiento de multiplicación de matrices para comparar y analizar dos enfoques de implementación. En la programación serie, las operaciones se ejecutan secuencialmente en un solo hilo, lo que implica un proceso lineal de cálculo, mientras que en la programación paralela, las operaciones se distribuyen entre múltiples hilos o procesadores, permitiendo una ejecución simultánea y una mayor eficiencia en sistemas con múltiples núcleos. La comparación entre ambos enfoques se realiza ejecutando pruebas en diferentes sistemas de cómputo y evaluando el rendimiento en términos de velocidad y eficiencia. Esto proporciona información valiosa sobre cómo se distribuye la carga de trabajo en sistemas paralelos y cómo se puede optimizar el rendimiento mediante la gestión eficiente de recursos y la concurrencia. En resumen, el taller busca entender cómo el diseño de algoritmos puede influir en el rendimiento en entornos computacionales modernos, explorando las ventajas y limitaciones de la programación serie y paralela en el contexto específico de la multiplicación de matrices.

## **6.- Metodología de Experimentación**

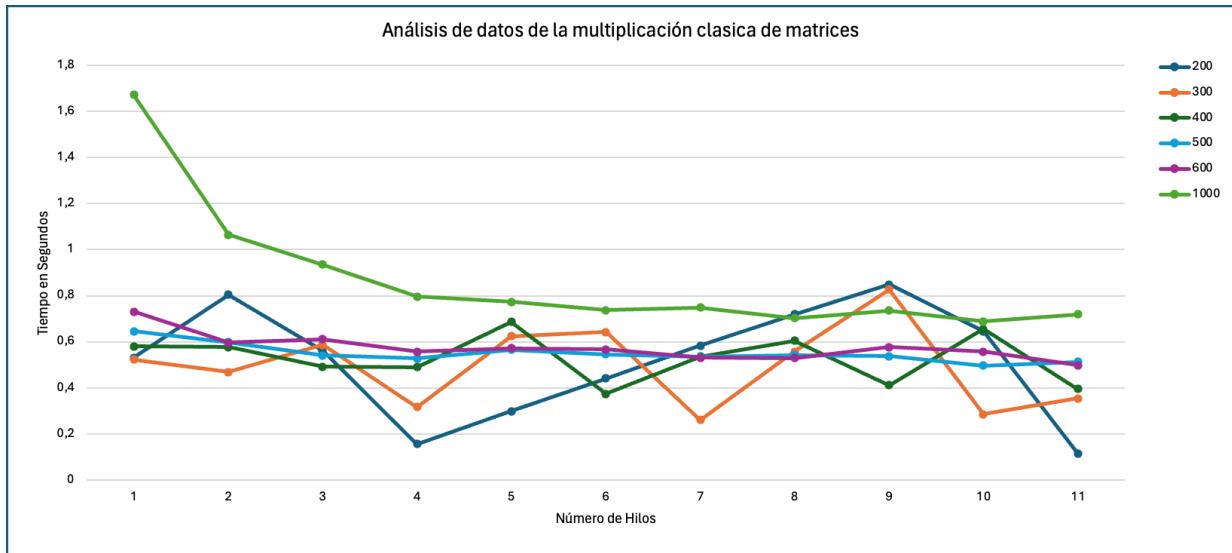
La metodología de experimentación en el taller de evaluación del rendimiento de multiplicación de matrices se hizo sistemática y rigurosa para obtener resultados fiables y significativos.

Los experimentos se realizan utilizando programación en C para implementar los algoritmos de multiplicación de matrices, junto con un script en Perl para automatizar la ejecución de pruebas en diferentes sistemas de cómputo. Estos sistemas pueden variar en cuanto al hardware y software, lo que permite evaluar el rendimiento en diferentes entornos.

La experimentación se lleva a cabo en momentos específicos, utilizando diversas configuraciones de tamaño de matriz y número de hilos de ejecución. Se seleccionaron valores de tamaño de matriz y número de hilos justificados previamente para capturar una variedad de escenarios de carga de trabajo y recursos disponibles. Además, se aplican leyes de la teoría de la probabilidad, como la ley de los grandes números, para obtener resultados promedio que reflejen de manera más precisa el rendimiento real del algoritmo.

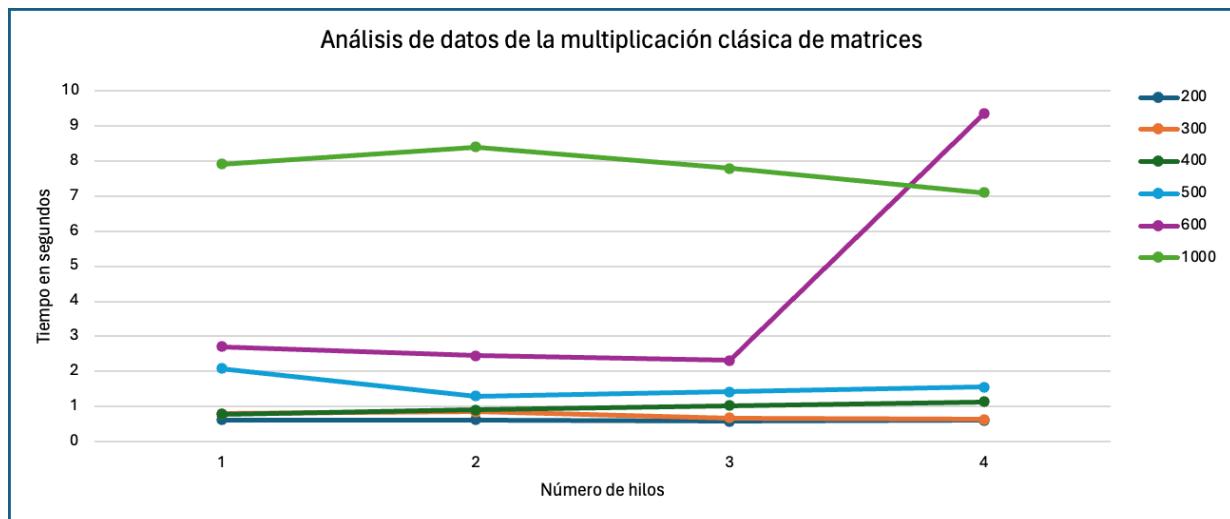
La métrica principal utilizada para evaluar el rendimiento es el tiempo de ejecución, medido en microsegundos, el cual se registra y analiza para comparar el desempeño de los algoritmos en diferentes sistemas y configuraciones. Esta metodología proporciona una base sólida para la experimentación y el análisis de rendimiento, permitiendo extraer conclusiones significativas sobre el impacto del paralelismo y otros factores en el tiempo de ejecución de la multiplicación de matrices.

## **7.- Evaluación exhaustiva (hoja de cálculo)**

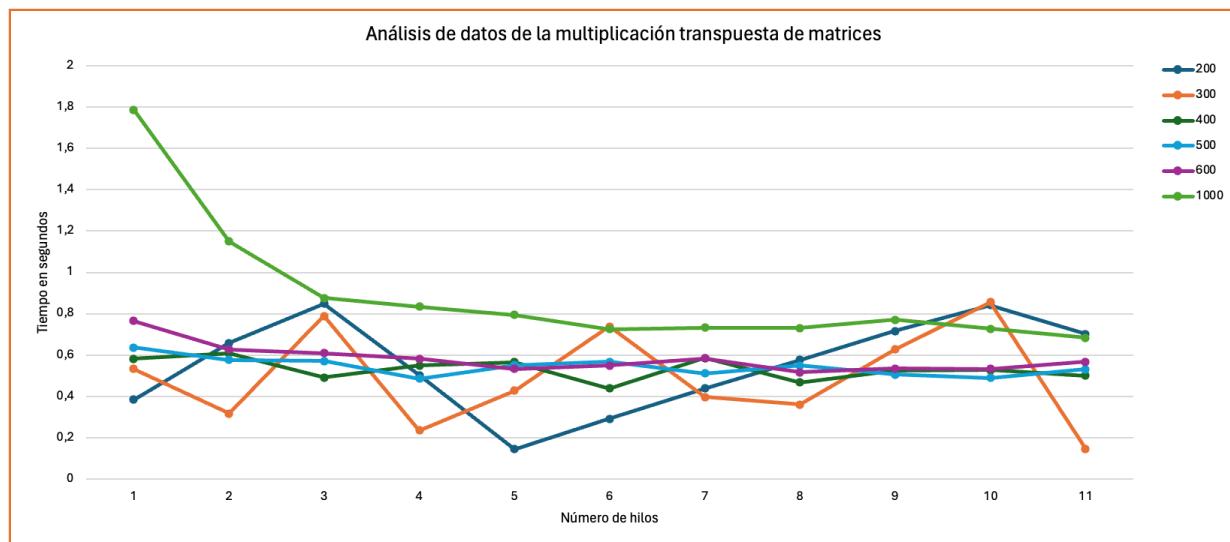


*Gráfica de Promedio de Tiempo por números de hilos en MM\_clasico en el Sistema Operativo MacOS*

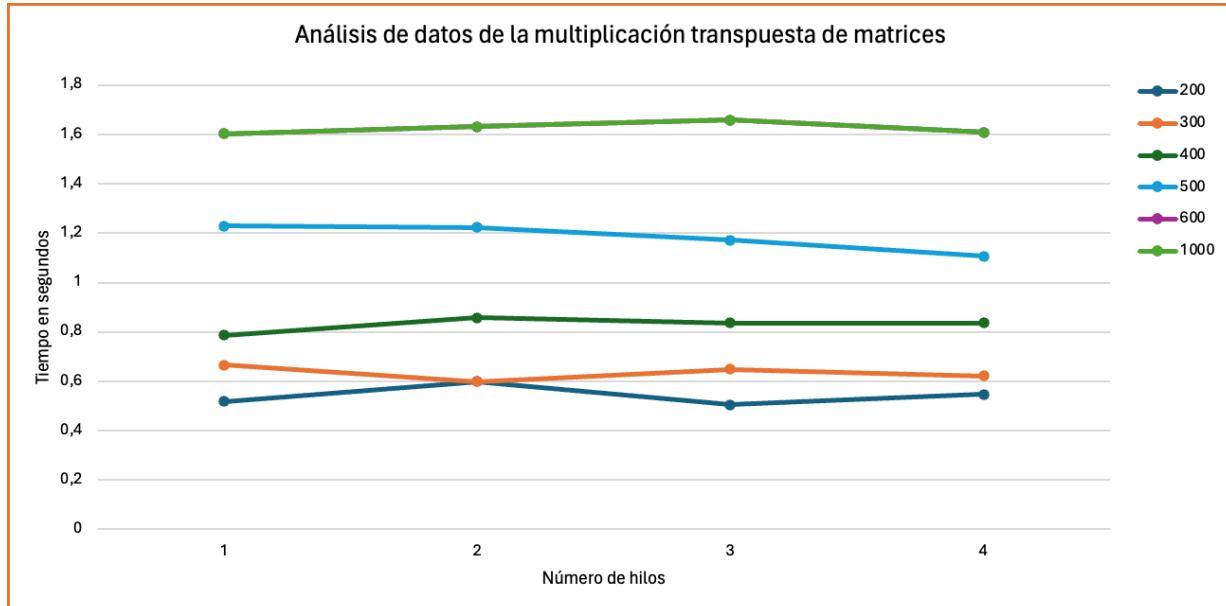
*(Gráfica 1).*



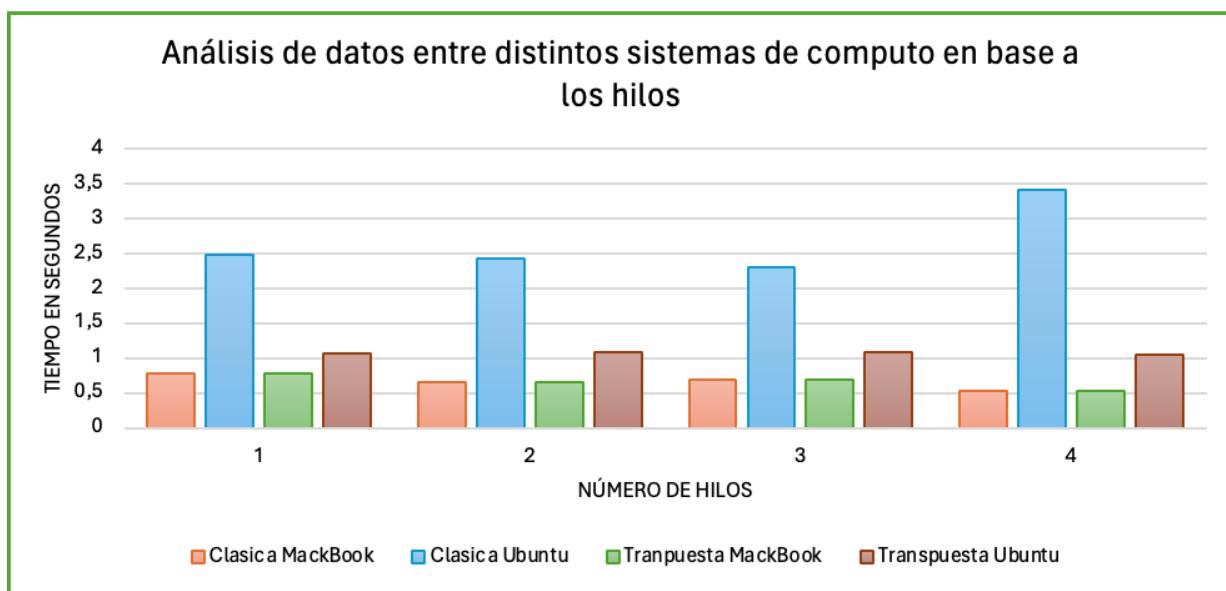
*Gráfica de Promedio de Tiempo por números de hilos en MM\_clasico en el Sistema Operativo Linux (Gráfica 2).*



*gráfica de Promedio de Tiempo por números de hilos en MM\_Transpuesta en el Sistema Operativo MacOs (Gráfica 3).*

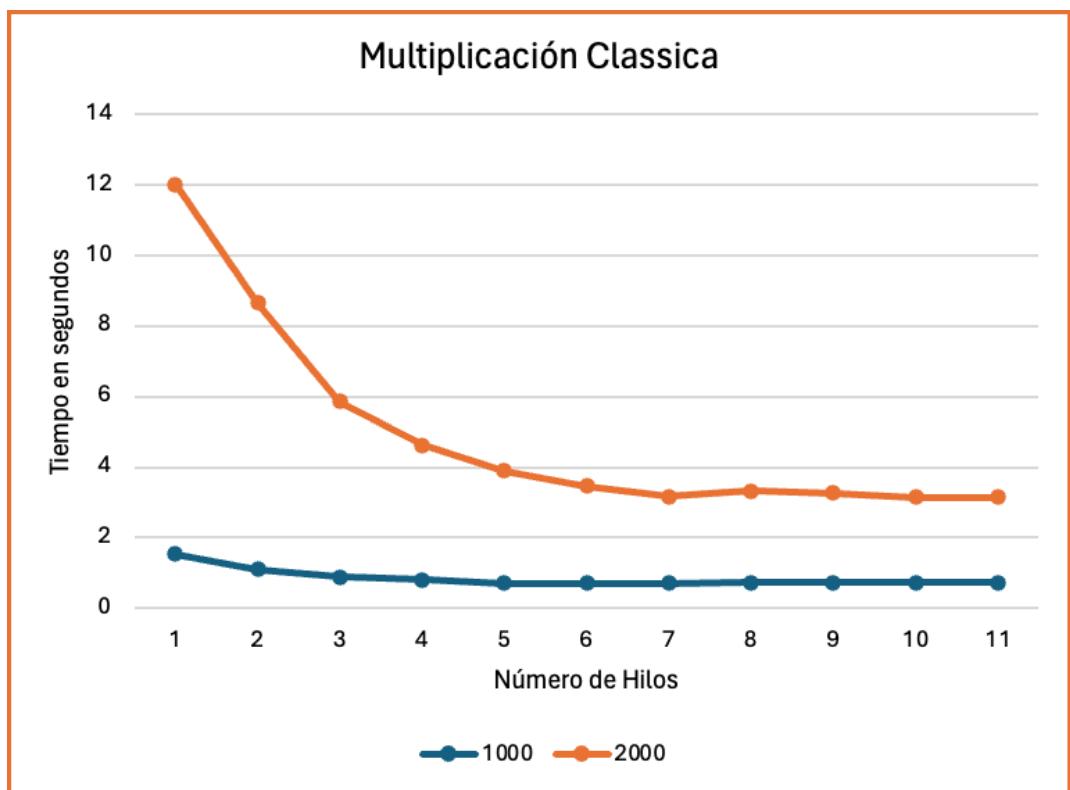


*Gráfica de Promedio de Tiempo por números de hilos en MM\_Transpuesta en el Sistema Operativo Linux  
(Gráfica 4).*

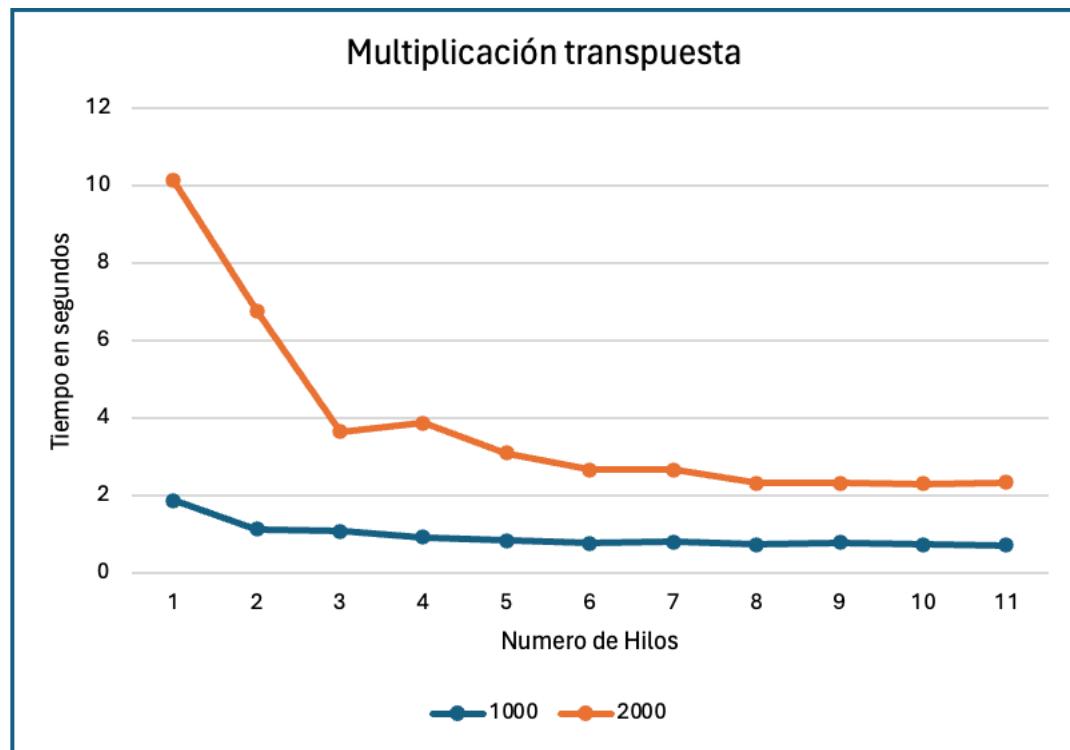


*Gráfica del comportamiento de los 2 tipos de matrices en los diferentes sistemas operativos (Gráfica 5).*

Se realizaron experimentos de multiplicación de matrices utilizando dos enfoques: multiplicación normal y multiplicación con la matriz transpuesta. Se evaluaron matrices de diferentes tamaños (200x200, 300x300, 400x400, 500x500, 600x600 y 1000x1000) y se varió el número de hilos de ejecución (1, 2, 3, 4 para linux) y (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 para MacOs). Los experimentos se llevaron a cabo en dos sistemas operativos, Linux y MacOs.



*Multiplicación Clásica de matrices en MacOs para matrices grandes*



*Multiplicación Transpuesta de matrices en MacOs para matrices grandes*

## **Resultados Principales**

1. Rendimiento de la Multiplicación Transpuesta: Se observó una mejora en el rendimiento al utilizar la multiplicación con la matriz transpuesta en comparación con la multiplicación normal. Esta mejora se hizo más evidente a medida que aumentaba el tamaño de la matriz.
2. Impacto de los Hilos de Ejecución: El uso de múltiples hilos (pthreads) resultó en una mejora significativa en el rendimiento de ambos métodos de multiplicación, especialmente para matrices de tamaños más grandes. Sin embargo, esta mejora se comportó casi constante después de los 6 hilos.
3. Diferencias entre Sistemas Operativos: Si bien se nota una diferencia positiva entre el sistema de cómputo con Mac OS con respecto al sistema de cómputo con Linux, principalmente en la multiplicación clásica de matrices; no es tan significativa para matrices con tamaños pequeños en la versión transpuesta. No necesariamente se debe a los sistemas operativos distintos, sino también por el hardware de cada equipo.

## **Interpretación**

- La ventaja de la multiplicación con la matriz transpuesta radica en su capacidad para aprovechar mejor la localidad espacial de los datos, lo que resulta en un acceso más eficiente a la memoria caché.
- El uso de hilos permite una mejor utilización de los recursos de procesamiento disponibles en sistemas multi-core, lo que conduce a una mejora en el rendimiento, especialmente para operaciones con matrices de gran tamaño.
- La estabilización del rendimiento después de cierto número de hilos sugiere que es importante encontrar un equilibrio entre el paralelismo y la sobrecarga de la gestión de hilos.
- La consistencia en el rendimiento entre diferentes sistemas operativos resalta la portabilidad y la robustez de los algoritmos implementados.
- Se recomienda el uso de la multiplicación con la matriz transpuesta para aplicaciones que trabajen con matrices de gran tamaño, ya que ofrece un rendimiento óptimo en comparación con la multiplicación normal.

Este análisis exhaustivo proporciona una comprensión detallada del rendimiento de los algoritmos de multiplicación de matrices en diferentes condiciones y entornos de ejecución.

### **6.1-Repositorio de código**

<https://github.com/carlosmejia10/TallerEvaluaci-nRendimiento>

- Problemas Frente a los Sistemas de Cómputo

La interfaz de usuario de macOS es reconocida por su simplicidad e intuitividad, lo que la convierte en una opción atractiva para usuarios que buscan una experiencia de uso fluida y accesible. Su estrecha integración con el ecosistema de productos Apple garantiza una compatibilidad impecable y una experiencia homogénea en todos sus dispositivos.

No obstante, el carácter cerrado de macOS limita la personalización del sistema en comparación con Linux. Además, su precio elevado y la exclusividad del hardware de Apple pueden ser factores determinantes para algunos usuarios.

Estabilidad, eficiencia y personalización son las tres palabras que mejor definen a Linux. Su robustez lo convierte en una opción popular para servidores y sistemas embebidos, donde la confiabilidad es primordial. Además, su naturaleza personalizable permite a los usuarios ajustar el sistema a sus gustos y necesidades específicas, desde la apariencia hasta el funcionamiento interno.

Sin embargo, esta libertad de personalización puede presentar un desafío para usuarios con menor conocimiento técnico, ya que requiere un mayor dominio del sistema operativo para realizar configuraciones avanzadas.

- Propuesta del Taller

La propuesta del taller se centra en comparar los paradigmas de programación en serie y en paralelo y evaluar el rendimiento de dos sistemas operativos, MacOS y Linux. Este enfoque permite explorar cómo diferentes métodos de programación y plataformas de cómputo influyen en la eficiencia y optimización de programas específicos. Al centrarse en la multiplicación de matrices, tanto en su forma clásica como utilizando una matriz transpuesta, se busca identificar las diferencias en rendimiento bajo condiciones controladas y variadas.

- Metodología Propuesta

La metodología empleada en este taller incluye la selección de algoritmos de multiplicación de matrices implementados en C, un lenguaje conocido por su eficiencia y control sobre la gestión de memoria. Estos algoritmos se ejecutarán en dos sistemas operativos, MacOS y Linux, para comparar su rendimiento. Además, se utilizarán múltiples hilos para evaluar el impacto de la programación paralela en comparación con la programación en serie.

Basándose en la Ley de los Grandes Números, los experimentos se repetirán 30 veces para asegurar la precisión y consistencia de los resultados. La métrica principal de evaluación será el tiempo de ejecución medido en microsegundos ( $\mu\text{s}$ ), complementada por el análisis del uso de recursos como memoria y CPU.

- Resultados Dados el Sistema de Cómputo

Los resultados mostraron que tanto MacOS como Linux ofrecieron rendimientos comparables en la ejecución de los algoritmos de multiplicación de matrices. No se encontraron diferencias significativas que sugirieran una ventaja clara de un sistema operativo sobre el otro en la versión Transpuesta. Sin embargo, la implementación de la programación paralela

Al aumentar la cantidad de hilos de ejecución el rendimiento mejoró, especialmente para matrices de mayor tamaño. La multiplicación que utilizó la matriz transpuesta mostró una ligera ventaja en términos de tiempo de ejecución comparado con la multiplicación clásica, principalmente para el sistema de cómputo con Linux.

Por otro lado, se observó que el uso de más de 6 hilos de ejecución se hace un poco innecesario, pues el resultado del análisis arroja que no hay una diferencia apreciable para realizarlo.

## 8. Conclusiones y Recomendaciones

Tras la exhaustiva exploración realizada en el taller de Evaluación de Rendimiento de Multiplicación de Matrices, se extraen conclusiones y se formulan recomendaciones clave para futuras investigaciones y aplicaciones prácticas.

Se ha demostrado que la programación paralela puede ofrecer mejoras significativas en el rendimiento de algoritmos fundamentales como la multiplicación de matrices, especialmente en sistemas con múltiples núcleos de procesamiento. Este hallazgo sugiere que la optimización del rendimiento computacional puede lograrse mediante la distribución eficiente de la carga de trabajo entre hilos de ejecución. Además, se destaca la importancia de la selección adecuada de configuraciones de tamaño de matriz y número de hilos para obtener resultados óptimos.

Este taller no solo proporciona una visión profunda de la multiplicación de matrices y la programación paralela, sino que también abre nuevas vías de investigación para mejorar la eficiencia y la velocidad en el procesamiento de datos computacionales.

El benchmarking que se realizó para comparar dos sistemas de cómputo, uno con macOS y otro con Linux, utilizando la multiplicación de matrices como algoritmo de referencia, proporciona información valiosa sobre el rendimiento relativo de los sistemas en esta tarea específica. Sin embargo, para obtener una comparación más completa y representativa del rendimiento general de los sistemas, se recomienda considerar la evaluación de otros algoritmos y técnicas de optimización.

### Algoritmos adicionales para benchmarks:

1. **Algoritmos de clasificación:** La clasificación de grandes conjuntos de datos es una tarea común en diversas aplicaciones. Al comparar el rendimiento de los sistemas en algoritmos de clasificación como Quicksort o Merge Sort, se puede obtener información sobre la eficiencia de las operaciones de memoria y la velocidad de procesamiento.
2. **Algoritmos de búsqueda:** La búsqueda de elementos en grandes conjuntos de datos es otra tarea fundamental. Al comparar el rendimiento de los sistemas en algoritmos de búsqueda como la búsqueda binaria o la búsqueda en tablas hash, se puede evaluar la capacidad de los sistemas para acceder y procesar datos de manera eficiente.
3. **Algoritmos de compresión de datos:** La compresión de datos es una tarea importante para reducir el tamaño de los archivos y mejorar la eficiencia del almacenamiento y la transmisión. Al comparar el rendimiento de los sistemas en algoritmos de compresión como gzip o bzip2, se puede evaluar la capacidad de los sistemas para procesar y manipular grandes cantidades de datos de manera eficiente.
4. **Algoritmos criptográficos:** La criptografía es esencial para la seguridad de la información en diversas aplicaciones. Al comparar el rendimiento de los sistemas en algoritmos criptográficos como AES o RSA, se puede evaluar la capacidad de los sistemas para realizar operaciones matemáticas complejas de manera eficiente.

También se aconseja la comparación de sistemas de cómputo con componentes (Hardware) similares, para que el resultado se aproxime más a saber que SO es mejor con relación a su eficiencia y optimización de recursos.

## 9.-Referencias

1. Karla's Project. (2017, 12 junio).  Windows VS Linux - Generalidades [Vídeo]. YouTube. <https://www.youtube.com/watch?v=rjnJadkNIH8>
2. Σακαλάκης, Π. (2023, 31 diciembre). Windows vs. Linux: Ποιο είναι καλύτερο για εσένα; (2024). TechGuides. <https://techguides.gr/windows-vs-linux-kalyter-gia-esena/>
3. *Linux on your desktop – pros and cons.* (s. f.). <https://www.baramundi.com/en-gb/blog/article/linux-on-your-desktop/>
4. colaboradores de Wikipedia. (2024, 9 mayo). *Sistema operativo*. Wikipedia, la Enciclopedia Libre. [https://es.wikipedia.org/wiki/Sistema\\_operativo](https://es.wikipedia.org/wiki/Sistema_operativo)
5. GeeksforGeeks. (2023, 1 septiembre). *Commonly used operating system*. GeeksforGeeks. <https://www.geeksforgeeks.org/commonly-used-operating-system/>
6. GeeksforGeeks. (s. f.). *GeeksforGeeks / A computer science portal for geeks*. <https://www.geeksforgeeks.org/>
7. **Paradigmas de Programación: Secuencial vs. Paralelo:** <https://www.medigraphic.com/cgi-bin/new/resumen.cgi?IDARTICULO=104059>
8. **Ventajas y Desventajas de la Programación Paralela:** <https://keepcoding.io/blog/ventajas-y-desventajas-paralelismo/>
9. **Ejemplos de Aplicaciones de la Programación Paralela:** [http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela\\_teoria/index.html](http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela_teoria/index.html)
10. **Comparación de Rendimiento entre Programación Secuencial y Paralela:** [http://scielo.sld.cu/scielo.php?script=sci\\_arttext&pid=S2227-18992016000300010](http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992016000300010)

