Informe de Rendimiento

Sistemas Operativos

Presentado por:
Santiago Chitiva Contreras
Martin Medina Novoa
Andrés David Rueda

PONTIFICIA UNIVERSIDAD JAVERIANA

BOGOTA D.C

2024

Contenido

Resumen

El problema abordado es la evaluación del rendimiento de diferentes sistemas de cómputo mediante la ejecución de un programa intensivo en cálculos, como la multiplicación de matrices. Se implementaron dos versiones de un programa en C: una secuencial y otra paralela con hilos. Estos programas se ejecutaron en dos sistemas diferentes, uno con arquitectura x86_64 y otro con arquitectura ARM. Se midió el tiempo de ejecución en ambos sistemas y se compararon los resultados para determinar cuál tiene mejor rendimiento en tareas intensivas en cálculos.

Introducción

Cuando se quiere elegir un dispositivo para trabajar y realizar actividades diarias, siempre se busca el mejor equipo que cumpla con nuestras expectativas. Para lograr esto, es fundamental considerar tanto el software como el hardware del dispositivo. El rendimiento de un sistema de cómputo puede evaluarse a través de diversas pruebas, una de las más comunes es la ejecución de programas intensivos en cálculos, como la multiplicación de matrices. Esta prueba permite medir el tiempo de respuesta del sistema y determinar su eficiencia bajo condiciones de máxima carga. En este contexto, es relevante comparar el rendimiento de diferentes arquitecturas de procesadores, como x86_64 y ARM, para identificar cuál se desempeña mejor en tareas computacionales exigentes. Por ello, se implementaron dos versiones de un programa en C que realiza la multiplicación de matrices, una utilizando un enfoque secuencial y otra un enfoque paralelo con hilos, y se ejecutaron en dos sistemas de cómputo diferentes para evaluar y comparar sus rendimientos.

Evaluación de rendimiento

Para estimar el tiempo promedio que se demora en ejecutar el promedio se seleccionaron dos sistemas de cómputo (SCP), cada uno con diferentes componentes y características, el número de matrices usadas en cada uno fue el mismo, pero la cantidad de hilos es diferente debido a que el hardware es distinto.

1. SCP1: El primer sistema de cómputo se basa en un sistema con arquitectura x86_64, un procesador Intel Core i5 de doceava generación con 6 nucleos físicos donde se tienen 2 hilos por cada uno, para un total de 12 hilos en total.

2. SCP2: El segundo sistema de cómputo elegido posee una arquitectura ARM el cual funciona de manera donde se usan conjuntos de instrucciones reducidas y su rendimiento está centrado en el software, este diseño es más conocido como RISC, esto permite que las operaciones se manejen de manera más rápida y eficiente en comparación a las tecnologías de instrucciones complejas o CISC. Además de esto la maquina posee 8 núcleos en su procesador M2 donde 4 son de rendimiento y 4 de eficiencia y por último también es importante nombrar que posee 8 GB de RAM las cuales se manejan de manera eficiente gracias al sistema operativo.

Ahora bien, para los programas que se implementaron para realizar las pruebas de rendimiento, basados en la multiplicación de matrices, esto ocurre debido a que las operaciones complejas ayudan a diferentes ciencias e ingenierías donde se analizan datos, simulan diferentes situaciones y en general ayudan a solucionar problemas muy complejos. Estos casos necesitan gran potencia de cómputo y para probar el límite en algunas ocasiones se utilizan las multiplicaciones de matrices ya que son operaciones intensivas en cálculos, especialmente para matrices de gran tamaño. Esto hace que sea una buena prueba para medir el rendimiento del hardware y obtener un resultado cercano a la verdad, además actúa como uso intensivo para la CPU como el acceso y la manipulación de memoria de la máquina.

También se usan porque estamos acostumbrados por historia, ya que se usaron como benchmarking desde los primeros días de la computación científica. Esto ha llevado a una gran cantidad de datos históricos y comparativos que pueden ser utilizados para evaluar el progreso y las mejoras en la tecnología del hardware y software.

Paradigma de programación

1. **Serie:** El programa mm_clasico.c sigue un enfoque secuencial tradicional para realizar la multiplicación de matrices. En este paradigma, las operaciones se ejecutan de manera lineal, una tras otra, en un solo hilo de ejecución.
   La función principal mult_thread realiza los cálculos iterando sobre las filas y columnas de las matrices y acumulando el resultado en la matriz de salida mC.

2. **Paralelo:** El programa mm_transpuesta.c utiliza un enfoque paralelo con hilos (threads) para realizar la multiplicación de matrices. En este paradigma, el trabajo se divide en múltiples hilos que se ejecutan simultáneamente, aprovechando así la capacidad de procesamiento paralelo de los sistemas modernos con múltiples núcleos. La función mult_thread es similar a la del programa secuencial, pero se ejecuta en múltiples hilos, cada uno encargado de calcular una porción de la matriz de salida mC. La división del trabajo se realiza asignando a cada hilo un rango de filas de la matriz para procesar.

Metodología

Se utilizaron lenguajes como C y Perl, el primero se utiliza debido a que es el lenguaje de programación que más proporciona control y rendimiento tanto del software como del hardware del computador.

El programa mm_clasico.c sigue un enfoque secuencial tradicional, donde las operaciones se ejecutan de manera lineal en un solo hilo de ejecución. Por otro lado, el programa mm_transpuesta.c utiliza un enfoque paralelo con hilos, dividiendo el trabajo entre múltiples hilos que se ejecutan simultáneamente, aprovechando así la capacidad de procesamiento paralelo de los sistemas modernos con múltiples núcleos.

Estos programas se ejecutaron en dos sistemas de cómputo diferentes, uno con arquitectura x86_64 (Intel Core i5 de 12ª generación con 6 núcleos físicos y 12 hilos) y otro con arquitectura ARM (procesador M2 con 8 núcleos, 4 de rendimiento y 4 de eficiencia, y 8 GB de RAM).

Para automatizar la ejecución de los programas con diferentes tamaños de matrices y números de hilos, se utilizó un script en Perl llamado lanza.pl. Este script se encarga de ejecutar ambos programas (clásico y transpuesta) con diferentes combinaciones de tamaños de matriz (100, 200, 300, 400, 800, 1000, 2000 y 4000) y números de hilos independiente para cada computador. Cada combinación se ejecuta 30 veces debido a la teoría de los grandes números que indica que si se quiere acercarse a la verdad hay que evaluar un experimento este número de veces, y los resultados de tiempo de ejecución se almacenan en archivos separados con un formato específico.

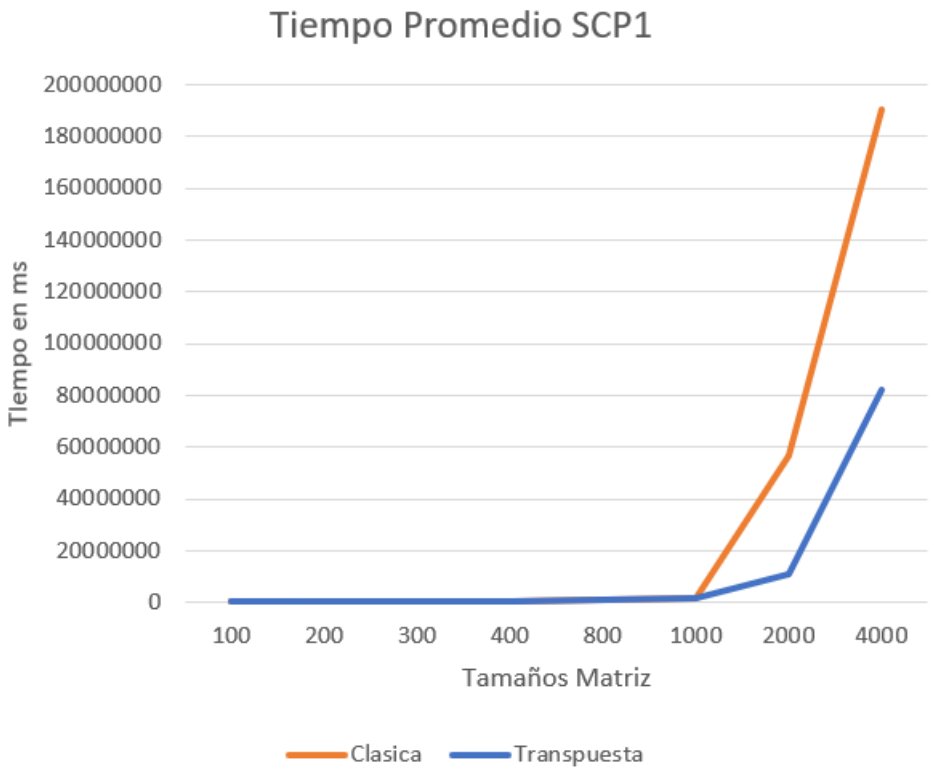
Evaluación exhaustiva

1. Sistema de cómputo 1:

   Resultados:

Tamaños	100	200	300	400	800	1000	2000	4000
Tiempos Clasica	616089	531304	449671	565950	1090698	1761593	56845684	190372349
Tiempos Transpuesta	278409	552157	564141	571859	1012704	1712183	10865271	82312893

   Gráfica:



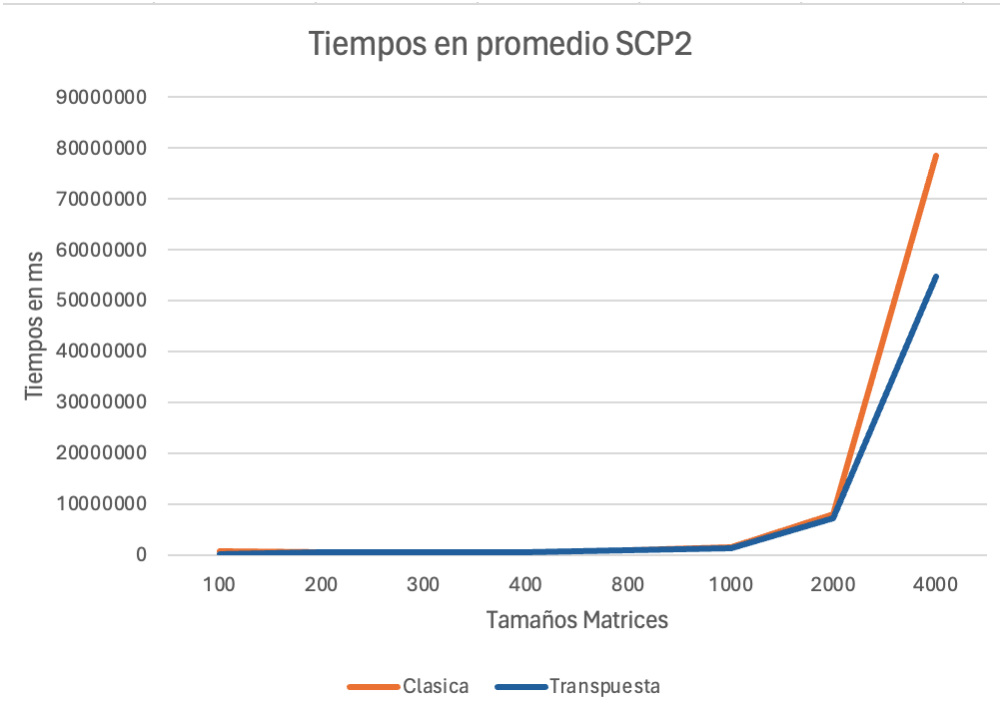
Como se puede observar, se muestra la relación entre el tamaño de las matrices y el tiempo de ejecución para cada uno de los dos códigos. Analizando los resultados, podemos decir que para matrices pequeñas (hasta la 1000x1000) no hay mucha diferencia entre los dos algoritmos, exceptuando la 100x100, donde claramente es más rápido el algoritmo clásico. No obstante, cuando empiezan los tamaños grandes de matrices, el algoritmo transpuesto toma mucha ventaja en cuestión de tiempo de ejecución (por tanto, eficiencia) con respecto al clásico.

2. Sistema de cómputo 2:

Resultados:

Tamaños	100	200	300	400	800	1000	2000	4000
TiemposClasica	625602	472078	591229	542063	970954	1410678	8049548	78477791
TiemposTranspuesta	188776	525727	472845	562110	946076	1344020	7170177	54600905

Gráfica:



Como se puede observar la tabla y el gráfico muestran los tiempos de ejecución de los dos algoritmos distintos para multiplicar matrices con los tamaños asignados.
Los resultados muestran que, para matrices pequeñas (hasta tamaño 400 aproximadamente), el algoritmo clásico es más rápido que el algoritmo de traspuesta. Sin embargo, a medida que aumenta el tamaño de las matrices, el algoritmo de traspuesta se vuelve más eficiente y supera al algoritmo clásico.
Para matrices de tamaño 4000, el algoritmo de traspuesta es significativamente más rápido que el algoritmo clásico. Esto se puede observar tanto en los valores numéricos de la tabla

como en las líneas del gráfico, donde la línea naranja (algoritmo clásico) crece más rápidamente que la línea azul (algoritmo de traspuesta) para tamaños grandes de matrices.

Conclusiones y recomendaciones

- La comparación entre los sistemas de cómputo con arquitectura x86_64 y ARM mostró diferencias significativas en rendimiento dependiendo del tamaño de las matrices. El sistema con arquitectura x86_64 (SCP1) demostró un mejor rendimiento con matrices más pequeñas, y el ARM (SCP2) lo hizo con matrices grandes, por su diseño de procesamiento RISC y el uso eficiente de núcleos de rendimiento y eficiencia.

- La implementación paralela del algoritmo transpuesto aprovechó de manera más efectiva los cores de ambos sistemas, especialmente en matrices grandes. En ambas arquitecturas, el enfoque paralelo superó al secuencial cuando las matrices alcanzaron tamaños grandes, lo que subraya la importancia de la programación paralela en aplicaciones de cálculos y gran cantidad de datos.

- Al ejecutar las pruebas 30 veces por combinación de tamaño de matriz y número de hilos, los resultados obtenidos fueron consistentes, validando la metodología utilizada y la aplicación de la teoría de los grandes números para obtener conclusiones fiables sobre el rendimiento de los sistemas.

- El uso de la multiplicación de matrices como prueba de benchmarking tiene un respaldo en la historia que proporciona una base sólida para la comparación de rendimiento. Esto permite evaluar el progreso y las mejoras en tecnologías al pasar del tiempo.

- Es recomendable optimizar los programas de cálculo como estos, considerando las características específicas de la arquitectura del sistema. Para el sistema con arquitectura ARM, se deben aprovechar los cores y la eficiencia para tareas adecuadas, mientras que en el sistema x86_64, la gestión de hilos debe estar optimizada para aprovechar al máximo los cores físicos y lógicos.

- En aplicaciones con grandes cantidades de datos, se debe implementar un enfoque de programación paralela, ya que así se mejora el rendimiento y también permite una mayor escalabilidad a medida que aumentan las demandas del sistema de cómputo.

- Continuar utilizando y mejorando scripts de automatización como el script en Perl (lanza.pl) para realizar pruebas de rendimiento en diferentes configuraciones, para asegurar una evaluación que facilite la recolección de datos necesarios para los análisis.

Referencias

- Leibovich, F., Gallo, S., De Giusti, L., Chichizola, F., Naiouf, M., & De Giusti, A. (2011). Comparación de paradigmas de programación paralela en cluster de multicores: Pasaje de mensajes e híbrido. Un caso de estudio. In *Proc. Congreso Argentino de Ciencias de la Computación (CACIC 2011)* (pp. 241-250). https://d1wqtxts1xzle7.cloudfront.net/90947578/Documento_completo-libre.pdf?1662985253=&response-content-disposition=inline%3B+filename%3DComparacion_de_paradigmas_de_programacio.pdf&Expires=1716522166&Signature=TaJibm9cSVfxKBVaSLz0jdGq~7k7trD93uemjgECq2Jh9arawuoET1d

qqPeQ8mJqKM~Tofw2oKHLAo77zXjGClSuA~okwGcw-KKoOWOYIp6ww~xDg9MmF1d5KlkTyJVPDn0diD1PfuS4MOl5pZkx0m5-B0KhPrFXjPncaWmXTcXa1k~cGbab89a~0jtu7dR1ztD2pxvk7tlrUfo8DxFQIlTeYc7-S2aWUVXvX8MdLdhVNP1UD4Y3ncVn3-YibhO4NPNTVoHZWoa4GLwQ53WIvpnRG~DSkUHcW9wMDIJ~mGYNZMDrZ04t44xnjTUArC6S66MUo5Veo86ZrveGcoAjIQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

- O'Connor, J. J., & Robertson, E. F. (2000). Matrices y determinantes. *Revista de educación matemática*, *15*(1).

- Guadalupe, M. S. A. (2021). Matrices: Una descripción general de sus operaciones y tipos. *Journal de Objetos y Objetivos Matemáticos Número 4. Enero-Junio de 2021*, 12. https://www.joom.org.mx/files/JOOM%204.pdf#page=14

- Lay, D. C. (2007). *Álgebra lineal y sus aplicaciones*. Pearson educación. https://books.google.es/books?hl=es&lr=&id=lTIVrKT9CMIC&oi=fnd&pg=PR14&dq=multiplicacion+de+matrices&ots=NAWZoNlAB4&sig=RIElubd7QO98mGFrXTgFMBGn9j4#v=onepage&q=multiplicacion%20de%20matrices&f=false

- de Cárdenas Cristia, A. (2006). El benchmarking como herramienta de evaluación. *Acimed*, *14*(4), 0-0. http://scielo.sld.cu/scielo.php?pid=S1024-94352006000400015&script=sci_arttext