Ciencia de la Computación

Computación Paralela y Distribuida

Docente Alvaro Henry Mamani Aliaga

Análisis de Cache and Programs

Entregado el 24/08/2023

Luis Fredy Huachaca Vargas

Semestre VIII

2023-2

"El alumno declara haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo"

En este informe analizaremos cómo las especificaciones de mi PC están afectando la eficiencia de la ejecución del programa y los resultados observados, sin olvidar mencionar como las nuevas

generaciones de computadoras están hechas para el paralelismo y como la forma de iterar de cierta forma afecta en el tiempo y es mejor para el paralelismo o no.

Especificaciones de PC:

- CPU: Intel Core i7-11800H de 11.a generación (16 núcleos, 8 subprocesos por núcleo)
- Velocidad del reloj de la CPU: base de 2,30 GHz, hasta 4,60 GHz como máximo
- Arquitectura: x86_64
- Niveles de caché: L1d: 384 KiB, L2: 10 MiB, L3: 24 MiB

Análisis del programa:

El programa son iteraciones sobre una matriz usando bucles anidados. Calcula el tiempo necesario para las órdenes de acceso a la matriz tanto de fila principal como de columna principal. El orden de fila principal accede a elementos consecutivos en la memoria a lo largo de filas, mientras que el orden de columna principal accede a elementos consecutivos a lo largo de columnas.

Resultados:

Matrix Size	Row-Major Time	Column-Major Time
100	210265	225537
1000	15545857	21255546
10000	615591078	2431201094
20000	2942936216	13467746379
30000	7379159816	32987929790

Output del programa después de ejecutar.

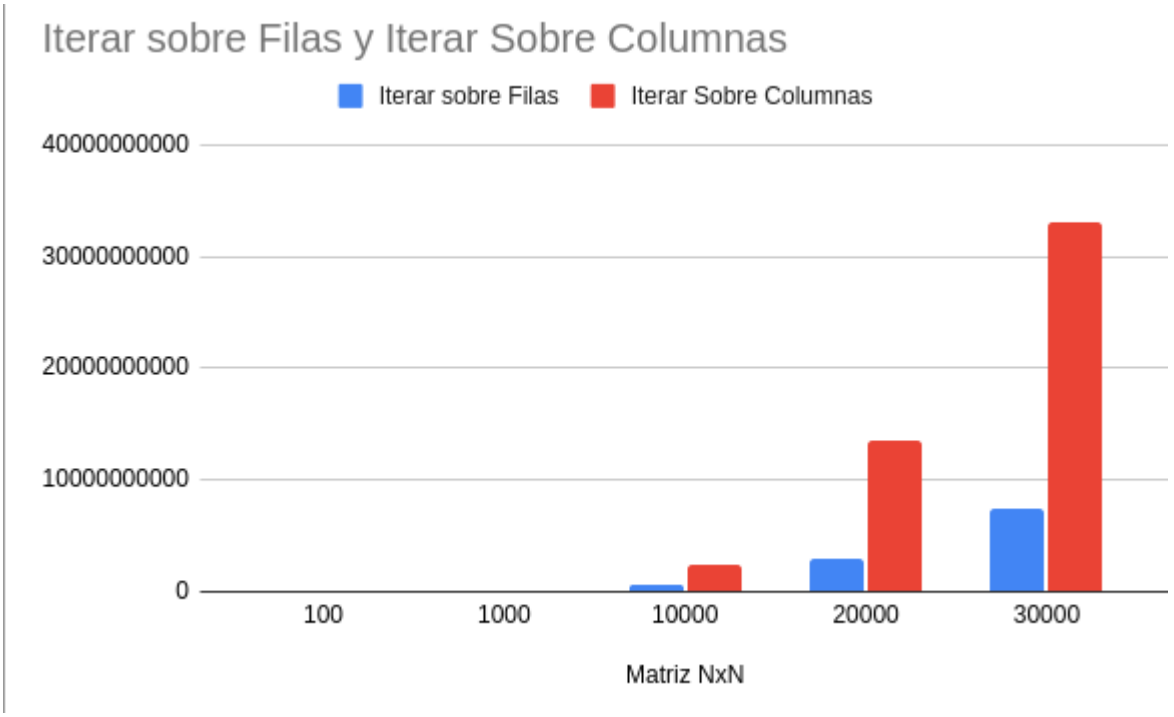

Gráfico Indicando la eficiencia de uno sobre otro.

Observaciones:

- A medida que aumenta el tamaño de la matriz, también aumenta el tiempo de ejecución de ambos bucles, lo cual es esperado ya que el número de cálculos crece.
- El primer bucle (orden principal de fila) funciona consistentemente ligeramente mejor que el segundo bucle (orden principal de columna) para todos los tamaños de matriz.

Explicación:

1. Localidad de caché: Los niveles de caché de la CPU (L1, L2 y L3) desempeñan un papel crucial en la eficiencia del acceso a la memoria. Acceder a datos que ya están en el caché es significativamente más rápido que recuperar datos de la memoria principal. El orden de fila principal exhibe una mejor localidad de caché porque accede a los elementos secuencialmente en la memoria, haciendo un mejor uso de las líneas de caché. El orden de las columnas principales salta más por la memoria, lo que provoca errores de caché y provoca una ejecución más lenta.

2. Patrones de acceso a la memoria: El orden de las filas principales tiene una mejor localidad espacial, ya que se accede a los elementos vecinos uno tras otro en la memoria. En el orden de las columnas principales, cada acceso a un nuevo elemento puede requerir la recuperación desde una ubicación de memoria diferente, lo que genera errores de caché y un acceso más lento.

Análisis con diferentes Computadores y Especificaciones:

1. Diferentes PC con diferentes tamaños de caché y pueden afectar su el programa en términos de rendimiento de caché. Los cachés más grandes conducen a una mejor utilización del caché, lo que reduce los tiempos de acceso a la memoria. Las memorias caché más pequeñas pueden provocar una destrucción de la memoria caché, lo que ralentiza el rendimiento. La jerarquía de caché, la asociatividad y la arquitectura del Caché influyen en el acceso a los datos y la latencia de la memoria. El tamaño de la línea de caché afecta las recuperaciones de memoria. Las políticas de almacenamiento en caché afectan el comportamiento de escritura. Estos factores influyen colectivamente en la velocidad de ejecución del script.

Conclusiones:

1. Optimizar los patrones de acceso a la memoria: El orden de las filas principales tiende a funcionar mejor debido a la localidad de la caché. Si es posible, estructure sus algoritmos y diseños de datos para aprovechar esto favoreciendo el acceso secuencial a la memoria.
2. Hay patrones de acceso más eficientes que otros, por ejemplo el acceso por filas, esto es útil al momento de plantearse problemas de paralelismo futuros.
3. La optimización del rendimiento es un proceso complejo que puede implicar experimentación, pruebas y ajustes para encontrar las mejores soluciones para su caso de uso y configuración de hardware específicos.

Repositorio:
-https://github.com/Luis-Huachaca-HV/ParalelaCourse