Informe de Prácticas

Sesión 1

Computación de Altas Prestaciones

Máster en Ingeniería Informática

Universidad de Oviedo

Autor:

Rubén Martínez Ginzo, UO282651@uniovi.es

Abril 2025

Índice

Índice de figuras

Índice de tablas

1. Introducción

En esta sesión de prácticas de laboratorio se evalúan distintos esquemas de acceso a memoria aplicados a la multiplicación de matrices. Concretamente, se analiza el rendimiento de esta operación matemática utilizando tres estrategias de acceso diferentes:

- *Row-Major order*
- *Column-Major order*
- *Z-Order*

El objetivo principal es determinar cuál de estas estrategias ofrece un mejor desempeño en términos de tiempo de ejecución y eficiencia en el uso de memoria.

1.1. Desarrollo

Para el desarrollo de esta práctica, se han seguido las indicaciones recogidas en el guion de la sesión correspondiente. Se han implementado las tres estrategias de acceso en el código fuente, y se han llevado a cabo diversas pruebas para medir su rendimiento. Con el fin de comparar los tiempos de ejecución, se ha desarrollado un programa en lenguaje *C* con fines de *benchmarking*, tal y como se detalla en el capítulo siguiente. Todo el código fuente se encuentra disponible públicamente en el siguiente repositorio de GitHub, así como en el archivo *zip* asociado a esta entrega.

1.2. *Benchmarking*

Además del trabajo realizado durante la sesión de laboratorio, se ha desarrollado un programa en *C* que automatiza la medición de los tiempos de ejecución para las distintas estrategias de acceso a memoria.

Este programa, ubicado en el subdirectorio *benchmark*, permite especificar diferentes tamaños de matriz y un número fijo de iteraciones para cada experimento. Durante la ejecución, se realiza la multiplicación de matrices utilizando cada uno de los esquemas de acceso, registrando el tiempo requerido por cada estrategia. En el caso particular del algoritmo *zorder*, se evalúan automáticamente todos los tamaños de bloque posibles para cada tamaño de matriz, llevando a cabo la multiplicación con cada configuración. Cada multiplicación se repite tantas veces como iteraciones se hayan especificado, calculando el tiempo medio de ejecución de todas ellas. Esto permite reducir significativamente el ruido en las mediciones y obtener resultados más precisos.

Al finalizar, el programa devuelve un *log* en formato CSV con los resultados otenidos. Dichos tiempos de ejecución constituyen la base de los análisis y comparativas que se presentan en los capítulos siguientes.

2. Expectativas y análisis inicial

Antes de analizar los resultados obtenidos, se comentan las expectativas iniciales respecto al rendimiento de cada uno de los esquemas de acceso a memoria aplicados al producto de matrices.

2.1. *Row-Major order*

El esquema *Row-Major order* se basa la lectura de matrices por filas, lo que, traspolado al producto de matrices ($C = A \times B$) implica que la matriz resultado **C** se recorrerá de dicha forma. Esto es, para realizar el cálculo de cada elemento de la primera fila de la matriz resultado, se recorrerá la primera file de la matriz **A** y todas las columnas de la matriz **B**. Este proceso se repetirá para cada una de las filas de la matriz **C**.
Se espera que este esquema de acceso a memoria sea muy eficiente, ya que minimiza la cantidad de fallos en caché. Concretamente, para el cálculo de cada fila de la matriz **C**, se espera que se produzcan:

- 1 único fallo de caché para la matriz **C**, al acceder al primer elemento de su primera fila.
- 1 único fallo de caché para la matriz **A**, al acceder al primer elemento de su primera fila.
- $N \times N$ fallos de caché para la matriz **B**, ya que se accede columna a columna y cada columna tiene N elementos.

2.2. *Column-Major order*

El esquema *Column-Major order* se basa la lectura de matrices por columnas, lo que, traspolado al producto de matrices ($C = A \times B$) implica que la matriz resultado **C** se recorrerá de dicha forma. Esto es, para realizar el cálculo de cada elemento de la primera columna de la matriz resultado, se recorrerá la primera columna de la matriz **B** y todas las filas de la matriz **A**. Este proceso se repetirá para cada una de las columnas de la matriz **C**.
Se espera que este esquema de acceso a memoria sea menos eficiente que el anterior, ya que se producen más fallos de caché. Concretamente, para el cálculo de cada columna de la matriz **C**, se espera que se produzcan:

- N fallos de caché para la matriz **C**, ya que se accede a N elementos de su primera columna.
- N fallos de caché para la matriz **A**, ya que se accede a N veces a cada una de sus filas.
- $N \times N$ fallos de caché para la matriz **B**, ya que se accede N veces a los N elementos de su primera columna.

2.3. *Z-Order*

El esquema *Z-Order* se basa en la división recursiva de las matrices en submatrices de tamaño $2^n \times 2^n$ y su posterior recorrido en orden Z. Traspolado al producto de matrices ($C = A \times B$), implica que la matriz resultado **C** se recorrerá en orden Z para cada una de las submatrices en las que se haya dividido.
Este esquema permite una mejor utilización de la memoria caché en casos donde las matrices a multiplicar son de grandes dimensiones, ya que se accede a los elementos de forma más local. Sin embargo, su rendimiento puede verse afectado por el tamaño de bloque utilizado. Se espera, por tanto, que sea el esquema más eficiente en casos de matrices de grandes dimensiones. Predecir empíricamente el número de fallos de caché en el uso de este esquema es algo muy complejo, por lo que su estudio se limita a la próxima experimentación y comparación de resultados.

3. *Row-Major order* vs *Column-Major order*

En esta sección se comparan los dos métodos de recorrido de matrices, *Row-Major order* y *Column-Major order*, en cuanto a su rendimiento en el producto matricial. Para ello, se ha realizado dicha operación para los siguientes tamaños de matrices: {2, 4, 8, 16, 32, 64, 128, 256, 398, 512, 636, 774, 892, 1024, 1152, 1280, 1408, 1536}. Cada producto se ha realizado un total de 32 veces y se ha medido el tiempo medio de ejecución. Los resultados obtenidos se presentan en la Figura 1.

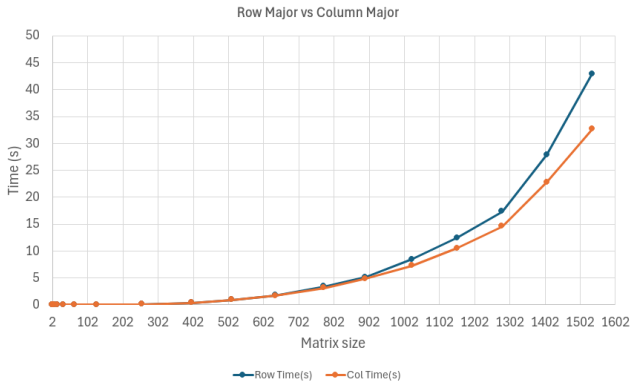

Figura 1: Comparación de tiempos de ejecución para *Row-Major order* y *Column-Major order*

Sorprendentemente, el recorrido por columnas es más rápido que el recorrido por filas. Esta diferencia comienza a ser notable a partir de un tamaño de matriz relativamente grande, aproximadamente de 1024×1024. Dicha diferencia crece a medida que aumenta el tamaño de la matriz, alcanzando unos $10s$ de diferencia en el mayor tamaño de matriz probado (1536×1536). Para representar la diferencia de rendimiento entre ambos métodos, se ha calculado la aceleración de *Row-Major order* respecto a *Column-Major order* para cada tamaño de matriz. Dicha aceleración se ha graficado en la Figura 2.

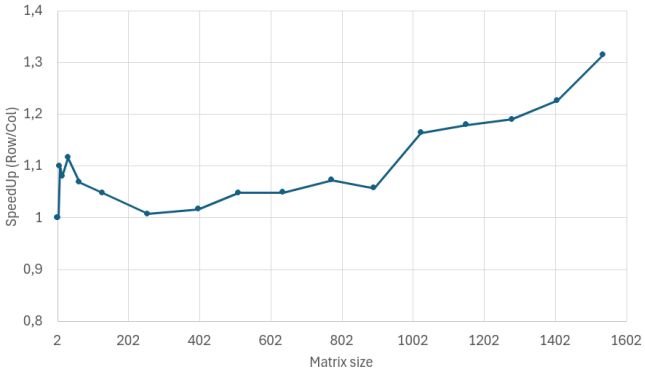

Figura 2: Aceleración de *Row-Major order* respecto a *Column-Major order*

Como puede observarse, *Column-Major order* es más rápido que *Row-Major order* para todos los tamaños de matriz probados, siendo notablemente mejor conforme aumenta el tamaño de la matriz. Esto se ve reflejado en la tendencia creciente de la aceleración graficada.

4. *Z-Order*

5. Conclusiones y dificultades