Informe de Prácticas
Sesión 3
Computación de Altas Prestaciones
Máster en Ingeniería Informática

Universidad de Oviedo

Autores:

Rubén Martínez Ginzo, UO282651@uniovi.es
Alejandro Rodríguez López, UO281827@uniovi.es

Abril 2025

Índice

Índice de figuras

Índice de tablas

1. Introducción

En esta sesión de prácticas de laboratorio se aborda la programación en C/CUDA. Para ello, se implementarán y analizarán dos problemas:

- Producto matricial.
- Resolución de sistemas de ecuaciones utilizando el método de *Gauss-Jordan*.

El objetivo principal de esta sesión es la implementación de ambos problemas en C/CUDA y su posterior análisis de rendimiento respecto a la implementación en CPU.

1.1. Desarrollo

Para llevar a cabo el desarrollo de esta práctica, se han seguido las indicaciones recogidas en el guion de la sesión correspondiente. Cada uno de los dos alumnos involucrados se ha centrado en la resolución de uno de los problemas, siendo el producto matricial el problema asignado al alumno Rubén Martínez Ginzo y la resolución de sistemas de ecuaciones el problema asignado al alumno Alejandro Rodríguez López.

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*

2. Expectativas iniciales

3. Producto matricial en GPU

En esta sección se analiza el rendimiento del producto matricial en GPU. Para ello, se ha implementado un *kernel* que realiza el producto de dos matrices de tamaño $N \times N$ y se ha medido el tiempo de ejecución en función del tamaño de la matriz. Cada tamaño de matriz se ha ejecutado con todos los bloques de hilos posibles $(1, 2, 4, 8, 16, 32)$.

3.1. CPU vs GPU

Se compara a continuación el rendimiento del producto matricial en CPU y en GPU. Para ello, se grafican los resultados obtenidos con el algoritmo *Z-Order* en CPU y con el *kernel* paralelizado en GPU utilizando 1 *thread per block*. Dichos resultados se muestran en la Figura 1.

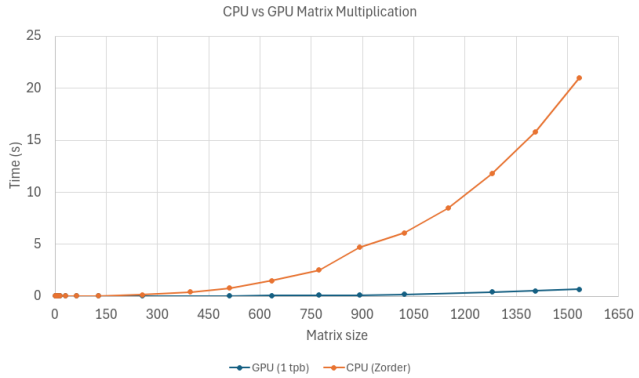

Figura 1: Rendimiento del producto matricial en CPU y GPU

Como se puede observar, el rendimientro del producto matricial en GPU es infinitamente superior al de CPU. Además, debe tenerse en cuenta que se está utilizando el algoritmo probado en CPU que mejores resultados ha dado (*Z-Order*) respecto a la configuración menos eficiente en GPU (1 *thread per block*).

3.2. *Threads per block*

En esta sección se analiza el rendimiento del producto matricial en GPU en función de la cantidad de *threads per block* utilizados. Los resultados se grafican en la Figura 2.

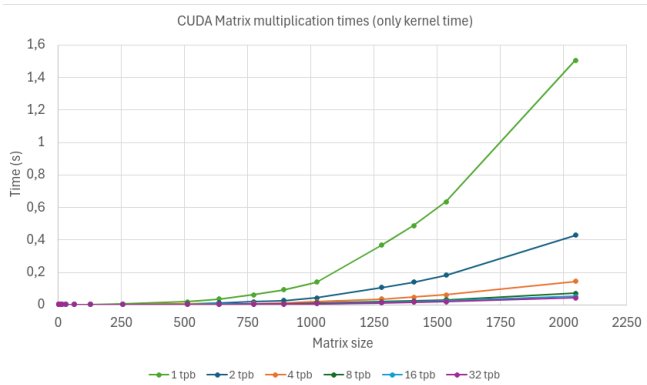

Figura 2: Rendimiento del producto matricial en GPU en función de los *threads per block*

Como es de esperar, el rendimiento mejora a medida que se incrementa el número de *threads per block*. Esta mejora de rendimiento se hace mucho más notable conforme aumenta el tamaño de la matriz, aunque en este caso, a partir de 8 *threads per block* puede considerarse despreciable. Esto se debe a que el máximo tamaño de matriz probado ($N = 2048$) es relativamente pequeño y no se aprovechan al máximo las capacidades de la GPU. En la Figura 3 se observa el rendimiento del producto matricial en GPU con matrices mucho mayores.

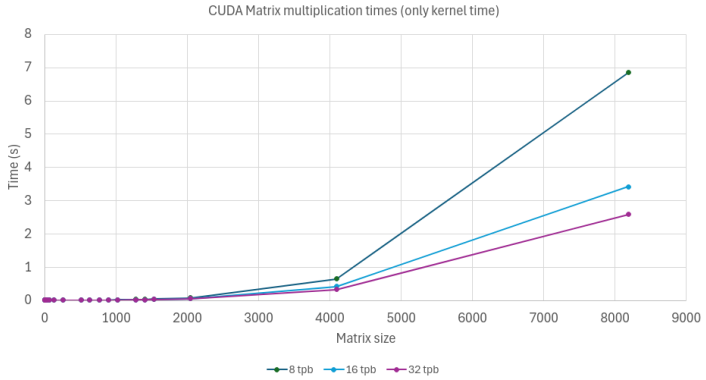

Figura 3: Rendimiento del producto matricial en GPU con matrices de tamaño $N > 2048$

Ahora sí que la diferencia de rendimiento con más de 8 *threads per block* es notable, y será aun mayor utilizando matrices de más tamaño.

3.3. Reserva de memoria y copia de resultados

Una peculiaridad de la GPU es que su memoria no es accesible desde la CPU. Por lo tanto, es necesario reservar memoria en la GPU para almacenar los resultados y luego copiarlos a la memoria de la CPU. En las gráficas anteriores sólo se ha tenido en cuenta el tiempo de ejecución del *kernel* y no el tiempo de reserva de memoria ni el tiempo de copia de resultados. En una sitación real, el tiempo de reserva de memoria y el tiempo de copia de resultados deberían ser tenidos en cuenta, ya que de poco sirve generar resultados si no son accesibles desde la CPU. En la Figura 4 se muestra la comparativa de ambas situaciones.

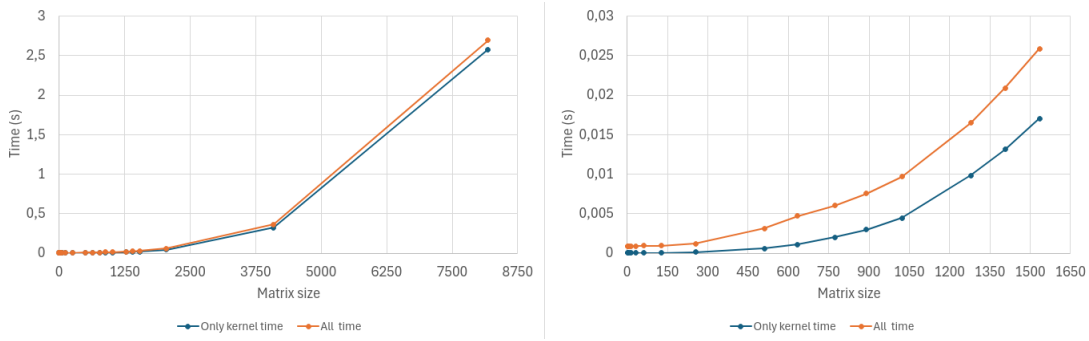

Figura 4: Sólo el tiempo de ejecución del *kernel* vs el tiempo total

Sorprendentemente, el tiempo de reserva de memoria y copia de resultados está muy optimizado ya que, con los tamaños de matriz probados, no hay apenas diferencia entre únicamente el tiempo de ejecución del *kernel* y el tiempo total. Aun así, si se probase con matrices de mucho mayor tamaño (poco funcional para esta práctica) seguramente la diferencia sería notable.

4. *Equations*

5. Conclusiones