Práctica 1: Análisis de Algoritmos

Adrián Grassin Luis

11 de febrero de 2025

1. Principios SOLID

Los principios SOLID son guías fundamentales para el diseño orientado a objetos que mejoran la mantenibilidad y escalabilidad del software. El principio de responsabilidad única (SRP) establece que una clase debe tener una única razón para cambiar, es decir, debe encapsular una única funcionalidad o responsabilidad del sistema.

2. Patrón Estrategia

El patrón estrategia permite definir una familia de algoritmos intercambiables en tiempo de ejecución. Cada algoritmo se encapsula en una clase separada que implementa una interfaz común, permitiendo que los algoritmos varíen independientemente de los clientes que los utilizan. En esta práctica, se aplica este patrón para implementar diferentes estrategias de multiplicación de matrices.

3. Análisis de Algoritmos

3.1. Arquitectura de Ejecución

- **CPU:** AMD Ryzen 5600X (6 cores, 12 threads)
- **GPU:** AMD Radeon RX 7800 XT (16GB VRAM)
- **RAM:** 32GB DDR4
- **Framework:** .NET 8.0

3.2. Resultados Experimentales

3.3. Análisis y Conclusiones

Los resultados muestran tres comportamientos distintivos según el tamaño de las matrices:

- **Matrices pequeñas (500x500):** La implementación por filas es superior debido a la mejor localidad de memoria y menor overhead. La versión por columnas muestra un rendimiento significativamente peor debido a los fallos de caché.
- **Matrices medianas (500x500-1000x1000):** La GPU comienza a mostrar ventajas, alcanzando hasta 105 GFlops en matrices de 750x750, comparado con 0.77 GFlops de la CPU.
- **Matrices grandes (1000x1000):** La diferencia entre implementaciones se amplifica. La versión por columnas muestra el peor rendimiento (0.11 GFlops), mientras que la GPU mantiene un rendimiento superior (11-129 GFlops).

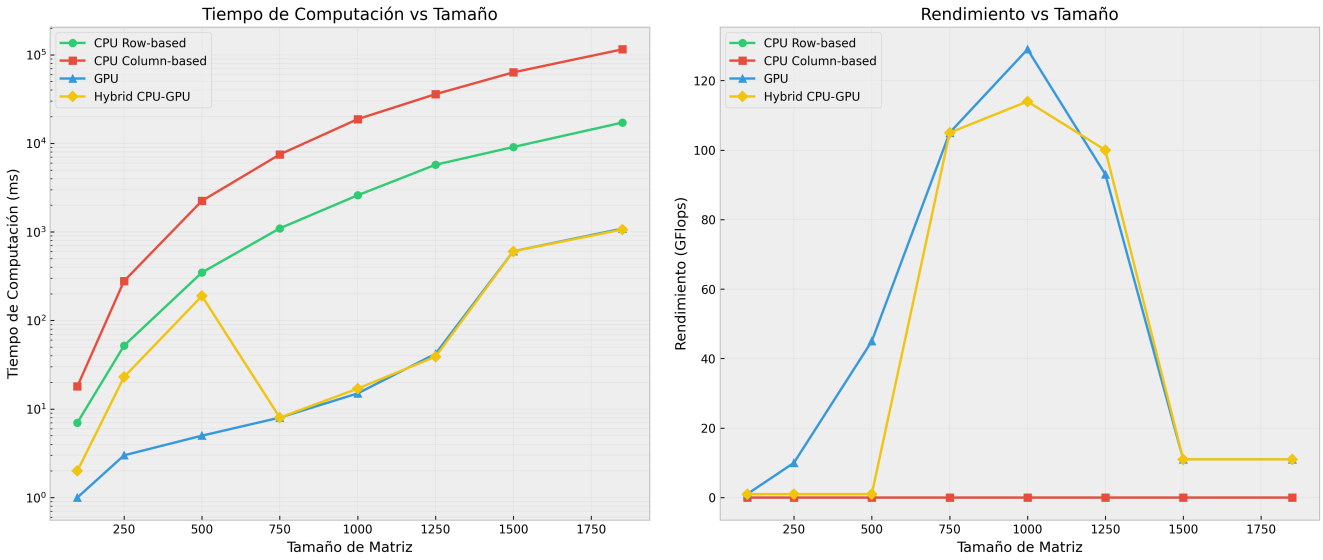

Figura 1: Comparación de rendimiento: Tiempo de computación (izq.) y GFlops (der.)

La implementación por filas supera consistentemente a la versión por columnas debido a la arquitectura de memoria de la CPU, que favorece accesos secuenciales. La GPU muestra una clara ventaja en matrices grandes debido a su capacidad de procesamiento paralelo masivo.