Multiplicación de Matrices

Reporte técnico

Iván Aram González Su

A01022584

Agosto 2018

Resumen

En este trabajo se presenta un problema que en el área de solución de problemas con programación es bastante trivial, pero con una solución y un análisis diferente al que comúnmente se hace, el cual es multiplicación de matrices. El problema fué resuelto de 3 maneras diferentes, la primera es de manera sequencial como se ha hecho en muchas ocaciones, la segunda fué con hilos del CPU; o sea tomando en cuenta el numero de procesadores que se tienen. La tercera fué la manera por la cual se inclina esta clase, usando hilos de la tarjeta gráfica tomando en cuenta la tarjeta gráfica de la computadora.

1. Introducción

Para hacer la parte sequencial utilicé 3 ciclos anidados, como normalmente se haría. Para los hilos con CPU utilicé 8 hilos porque mi computadora tiene 4 núcleos físicos y 4 núcleos virtuales y tener 8 hilos hacía la multiplicación de la manera más optima. Para realizar la versión de CUDA utilicé bloques de 512 por 1 e hilos de el techo del número de filas entre 512 por el numero de columnas, esta fué la configuración más optima después de hacer varias pruebas cambiando el número de bloques y de hilos.

2. Desarrollo

Desarrollé 4 códigos, el primer llamado MatrixMultiplication.cpp el cual tiene la versión de CPU sin hilos (la más tardada). El segundo llamado MatrixMultiplicationThreads.cpp que contiene la versión de CPU con hilos que es menos tardada pero igual toma tiempo en resolver multiplicaciones de matrices de una escala mayor, el tercero lllamado MatrixMultiplicationCuda.cu en el cual se encuentra la multiplicación de matrices hecha en CUDA (la más rápida). El cuarto código realizado es en donde se comparan los 3 métodos y se llama MatrixMultiplicatiónComparison.cu en el cual se puede ver “gráficamente” la comparación de las 3 versiones y se puede ver si las matrices son las mismas tanto en CPU sin hilos vs GPU como en CPU con hilos vs GPU.

Las siguientes tablas muestran la comparación de los tiempos tomados para realizar la multiplicación en las 3 diferentes vesiones (no son datos tomados de MatrixMultiplicationComparison.cu sino de cada uno de los códigos por separado):

En milisegundos:

Versión	Tiempo (1000 x 1000)	Tiempo (2000 x 2000)	Tiempo (4000 x 4000)
CPU sin hilos	3509.547607 ms	49358.679688 ms	492576.750000 ms
CPU con hilos	1483.176758 ms	14813.812500 ms	137040.921875 ms
GPU	39.174431 ms	435.257874 ms	3642.102051 ms

En segundos:

Versión	Tiempo (1000 x 1000)	Tiempo (2000 x 2000)	Tiempo (4000 x 4000)
CPU sin hilos	3.51 s	49.36 s	492.58 s (8.21 min)
CPU con hilos	1.48 s	14.81 s	137.04 s (2.28 min)
GPU	0.04 s	0.44 s	3.64 s

Igual obtuve el Speedup con las siguiente fórmula:

$$Speedup = \frac{Tiempo\,secuencial}{Tiempo\,paralelo}$$

Versión	Speedup (1000 x 1000)	Speedup (2000 x 2000)	Speedup (4000 x 4000)
CPU sin hilos vs CPU con hilos	2.37	3.33	3.59
CPU vs GPU	89.59	113.40	135.25

Hice un Makefile para compilar los 4 códigos sin problema el cual tiene la regla ‘all’ para compilar todos, la regla ‘rebuild’ para borrar los ejecutables y volver a compilar los códigos y la regla ‘clean’ para borrar los ejecutables.
También hice un script de bash para poder correr las 3 versiones con N igual a 1000, 2000 y 4000 en un comando el cual es ./run.sh, este va a correr make rebuild para generar los ejecutables y luego va a ejecutar matrix_multiplication_comparison con N igual 1000, después con N igual a 2000 y al final con N igual a 4000.