Matrix Multiplication

Pablo Macías Landa
Septiembre 2018

Resumen

Se realizó una multiplicación de matrices usando diferentes modificaciones de threads en CPU y GPU para comparar los diferentes resultados obtenidos y decidir la configuración más rápida y eficiente.

Introducción

Para las pruebas de la multiplicación de matrices se utilizó el equipo con las siguientes especificaciones:

- Laptop
- 8th Gen Intel Core i7-8750H processor, 6 Cores/12 Threads, 2.2GHz/4.1GHz (Base/Max Turbo), 9MB Cache
- Mobile Intel HM370 Chipset
- NVIDIA® GeForce® GTX 1060 Max-Q Design (6GB GDDR5 VRAM, Optimus™ Technology)
- 16GB dual-channel SO-DIMM (DDR4, 2667MHz)
- 256GB M.2 SSD (NVMe PCIe 3.0 x4)

Desarrollo

1. **CPU sin Threads**
    - **1000**

```
Matrix size: nx 1000 ny 1000
multMatrixOnHost elapsed 3459.964844 ms
```

    - **2000**

```
Matrix size: nx 2000 ny 2000
multMatrixOnHost elapsed 37785.804688 ms
```

    - **4000**
2. **CPU con threads**
    - **1000**

```
Matrix size: nx 1000 ny 1000
multMatrixOnHost elapsed 831.854065 ms
```

    - **2000**

```
Matrix size: nx 2000 ny 2000
multMatrixOnHost elapsed 12131.022461 ms
```

    - **4000**

```
Matrix size: nx 4000 ny 4000
multMatrixOnHost elapsed 122267.710938 ms
```

3. **CUDA threads**
    - En todas las diferentes variaciones del tamaño del bloque me dieron resultados muy similares, pero siempre con 128 me dio el mejor resultado por 1 o 2 ms.
    - **1000**

```
Using Device 0: GeForce GTX 1060 with Max-Q Design
Matrix size: nx 1000 ny 1000
multMatrixOnGPU1D <<<(8,1), (128,1)>>> elapsed 269.958496 ms
```

    - **2000**

```
Using Device 0: GeForce GTX 1060 with Max-Q Design
Matrix size: nx 2000 ny 2000
multMatrixOnGPU1D <<<(16,1), (128,1)>>> elapsed 1098.737427 ms
```

    - **4000**

```
Using Device 0: GeForce GTX 1060 with Max-Q Design
Matrix size: nx 4000 ny 4000
multMatrixOnGPU1D <<<(32,1), (128,1)>>> elapsed 4739.149414 ms
```

4. **CUDA blocks**
    - Igual que en el ejercicio pasado, los resultados no variaron mucho, pero con las dimensiones 32x32 daba el mejor resultado por unos cuantos ms.
    - **1000**

```
Using Device 0: GeForce GTX 1060 with Max-Q Design
Matrix size: nx 1000 ny 1000
multMatrixOnGPU1D <<<(32,32), (32,32)>>> elapsed 20.231327 ms
```

    - **2000**

```
Using Device 0: GeForce GTX 1060 with Max-Q Design
Matrix size: nx 2000 ny 2000
multMatrixOnGPU1D <<<(63,63), (32,32)>>> elapsed 137.476990 ms
```

    - **4000**

```
Using Device 0: GeForce GTX 1060 with Max-Q Design
Matrix size: nx 4000 ny 4000
multMatrixOnGPU1D <<<(125,125), (32,32)>>> elapsed 976.744202 ms
```

Conclusión

Pudimos observar como progresivamente los tiempos iban mejorando en cada uno de los códigos, desde el más simple como CPU normal hasta utilizar bloques en CUDA. Los resultados son bastante claros siendo CUDA con bloques la más eficiente por mucho, especialmente comparado con las opciones en CPU.

También pudimos jugar con las configuraciones dentro del mismo CUDA para cambiar los tamaños de los bloques permitiendonos ver los diferentes tiempos que no varían mucho, pero a

la larga pueden causar una gran diferencia. Aún no me queda muy claro como saber cual será la mejor configuración, las diferentes pruebas las hice a prueba y error pero es algo que me interesaría saber más para no hacerlo tan rudimentario.