Reporte Técnico

Alejandro Herce Bernal

Septiembre 2018

Resumen

La tarea consiste en medir las diferencias entre procesar la multiplicación de 2 matrices de NxN en CPU, CPU con paralelismo y GPU. Las matrices deberán ser de N = 1000 y 2000. Cada multiplicación tiene que ser validada en GPU y en CPU.

1. Introducción

CUDA es una arquitectura de cálculo paralelo de NVIDIA que aprovecha la gran potencia de la GPU para proporcionar un incremento extraordinario del rendimiento del sistema.

Los sistemas informáticos están pasando de realizar el "procesamiento central" en la CPU a realizar "coprocesamiento" repartido entre la CPU y la GPU. Para posibilitar este nuevo paradigma computacional, NVIDIA ha inventado la arquitectura de cálculo paralelo CUDA.

Para nuestras pruebas también estaremos utilizando OpenMP. La diferencia fundamental entre OpenMP y CUDA es que OpenMP ofrece paralelismo a O(100), mientras que CUDA ofrece O(10,000). El procesamiento en GPU también ofrece ventajas como mayor bandwith de memoria.

El programa consiste de un solo archivo, en el cual están las 3 operaciones: Multiplicación en CPU, multiplicación en CPU con OpenMP y multiplicación en GPU.

2. Desarrollo

a. Especificaciones Técnicas

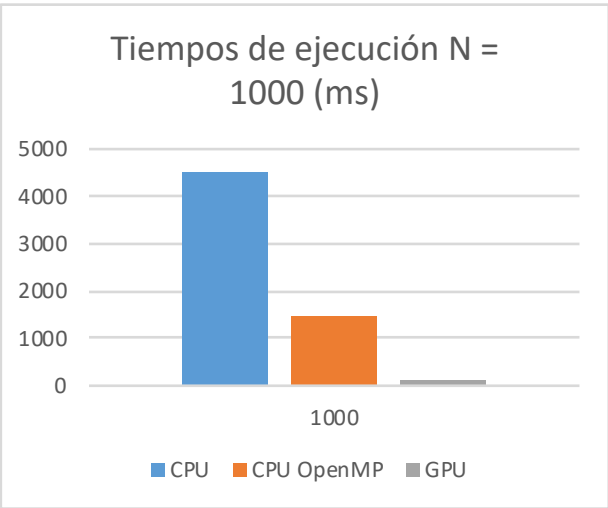
- Computadora de escritorio
- Procesador:
    - Intel Core i7-4790k
    - Frecuencia: 4.4Ghz
    - Cores físicos: 4
    - Cores lógicos: 8
    - L1 cache: 256 KB
    - L2 cache: 1.0 MB
    - L3 cache: 8.0 MB
- Tarjeta de video:
    - Nvidia Geforce GTX 980 Ti
    - Memoria: 6 GB GDDR5
    - CUDA cores: 2816

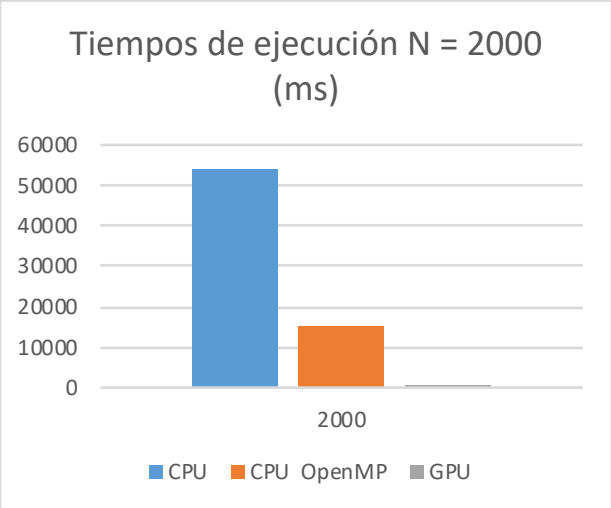
- GPU clock speed: 1.24 GHz
- Memory clock speed: 3505 MHz

```
 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 980 Ti"
  CUDA Driver Version / Runtime Version          9.2 / 9.2
  CUDA Capability Major/Minor version number:    5.2
  Total amount of global memory:                 6144 MBytes (6442450944 bytes)
  (22) Multiprocessors, (128) CUDA Cores/MP:     2816 CUDA Cores
  GPU Max Clock rate:                            1241 MHz (1.24 GHz)
  Memory Clock rate:                             3505 Mhz
  Memory Bus Width:                              384-bit
  L2 Cache Size:                                 3145728 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  CUDA Device Driver Mode (TCC or WDDM):         WDDM (Windows Display Driver Model)
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            No
  Supports Cooperative Kernel Launch:            No
  Supports MultiDevice Co-op Kernel Launch:      No
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 9.2, CUDA Runtime Version = 9.2, NumDevs = 1
Result = PASS
```

b. Pruebas

Para las pruebas, se repitieron las simulaciones 10 veces por cada tamaño de matriz y se calculó el promedio.





N	CPU	CPU OPENMP	GPU
1000	4517 ms	1501 ms	121 ms
2000	54228 ms	14859 ms	825 ms

Para la configuración de OpenMP, se intentó usar diferentes números de threads. De los 8 disponibles, se intentó con 4, 6 y los 8. En los 3 casos, la diferencia era mínima, de unos pocos milisegundos.

Para la configuración del GPU, se utilizó un block de (16, 8) y un grid de (256, 256). Fue la configuración más rápida que se encontró. Variar los tamaños del block de 8 en 8 agregaba alrededor de 20 milisegundos a las pruebas. El grid de 256 fue necesario para la prueba de N = 2000, ya que un número menor hacía que la multiplicación fuera incorrecta. En N = 1000 cambiar el grid a (128, 128) no afecto los tiempos significativamente.

3. Conclusión

Los resultados obtenidos demuestran como el paralelismo puede reducir tiempos de computo drásticamente. El GPU pudo procesar ambas matrices sin rebasar el límite de 1000 milisegundos, mientras que los tiempos en CPU cambiaron drásticamente.

Usando el GPU se tiene que usar el número de threads y blocks cuidadosamente, pues cambiarlos afecta el performance significativamente. Y también la regla de “que sobre a que falte” no aplica, al contrario, en unos casos el performance era mucho peor usando de más.

Para las pruebas de N = 4000, elegir los tamaños de blocks y threads fue casi imposible, el kernel mandaba errores con casi todas las combinaciones de threads y blocks que se intentaron. Las únicas configuraciones que se lograron probar regresaban una matriz incorrecta.