Multiplicación de Matrices

Cynthia Castillo Millán

Septiembre 2018

Resumen

El proyecto consiste en generar un programa multi-núcleo para el cálculo de una multiplicación entre matrices. Éste debe implementarse de tres maneras diferentes: a través del CPU sin hilos, CPU con hilos e implementando CUDA a través de bloques e hilos. La finalidad de las implementaciones es evaluar el desempeño de cada una de éstas.

.

1. Introducción

CUDA es una plataforma de computación paralela y un modelo de programación desarrollado por Nvidia para computación general en sus propias GPU (unidades de procesamiento de gráficos). CUDA permite a los desarrolladores acelerar las aplicaciones consumo computacional para al aprovechar la potencia de las GPU para la parte paralelizable del cálculo.

2. Desarrollo

Capacidades de la Computadora utilizada para el desempeño en CPU

Sistema

Procesador:	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
Memoria instalada (RAM):	8.00 GB (7.89 GB utilizable)
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64

Capacidades del CUDA device

```
Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 670"
  CUDA Driver Version / Runtime Version          9.0 / 7.5
  CUDA Capability Major/Minor version number:    3.0
  Total amount of global memory:                 1996 MBytes (2093023232 bytes)
  ( 7) Multiprocessors, (192) CUDA Cores/MP:     1344 CUDA Cores
  GPU Max Clock rate:                            980 MHz (0.98 GHz)
  Memory Clock rate:                             3004 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 524288 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 1 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
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Exclusive Process (many threads in one process is able to use ::cudaSetDevice() with this device)
>

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 9.0, CUDA Runtime Version = 7.5, NumDevs = 1, De
vice0 = GeForce GTX 670
Result = PASS
```

3. Ejemplo de referencias y figuras

CPU sin threads

Matrix 1000

```
time seq (ms): 21840.732422
time seq (ms): 21959.326172
time seq (ms): 21882.951172
time seq (ms): 21853.785156
time seq (ms): 22575.744141
time seq (ms): 21964.386719
time seq (ms): 21860.193359
time seq (ms): 21853.982422
time seq (ms): 21940.998047
time seq (ms): 21966.498047
AVG (ms): 21969.859375
```

Matrix 2000

```
*** SEQUENTIAL 2000*****
AVG (ms): 286860.156250
```

Matrix 4000

Tras 6 horas de correr, el cálculo de la matriz secuencial no había terminado por lo que se detuvo el experimento.

CPU con OMP

Matrix 1000

```
time omp (ms): 4666.204102
time omp (ms): 4602.200684
time omp (ms): 4611.632812
time omp (ms): 4613.465332
time omp (ms): 4593.815918
time omp (ms): 4640.253906
time omp (ms): 4669.996582
time omp (ms): 4607.210449
time omp (ms): 4748.097168
time omp (ms): 4705.888672
AVG (ms): 4645.876465
```

Matriz 2000

```
*** PARALLEL 2000*****
AVG (ms): 88703.179688
```

Matriz 4000

Tras 6 horas de correr, el cálculo de la matriz secuencial no había terminado por lo que se detuvo el experimento.

GPU – 1D1D

1D1D

Matrix 1000

```
Matrix size: nx 1000 ny 1000
multMatrixOnHost elapsed 3865.115234 ms
multMatrixOnGPU1D <<<(4,1), (256,1)>>> elapsed 413.277130 ms
Arrays match.
```

Matrix 2000

```
A01374530@alien1-lab:~/.../GPU_MatrixMult$ ./a.out 2000
Matrix size: nx 2000 ny 2000
multMatrixOnHost elapsed 50247.453125 ms
multMatrixOnGPU1D <<<(8,1), (256,1)>>> elapsed 1569.121948 ms
Arrays match.
```

Matriz 4000

```
Matrix size: nx 4000 ny 4000
Killed
```

1D2D

Matrix 1000

```
A01374530@alien1-lab:~/.../GPU_MatrixMult$ ./D1D2 1000
Matrix size: nx 1000 ny 1000
multMatrixOnHost elapsed 4051.699951 ms
multMatrixOnGPU1D <<<(4,1000), (256,1)>>> elapsed 45.045265 ms
Arrays match.
```

Matrix 2000

```
A01374530@alien1-lab:~/.../GPU_MatrixMult$ ./D1D2 2000
Matrix size: nx 2000 ny 2000
multMatrixOnHost elapsed 51978.152344 ms
multMatrixOnGPU1D <<<(8,2000), (256,1)>>> elapsed 359.600800 ms
Arrays match.
```

2D2D

Matrix 1000

```
A01374530@alien1-lab:~/.../GPU_MatrixMult$ ./D2D2 1000
Matrix size: nx 1000 ny 1000
multMatrixOnHost elapsed 4346.540039 ms
AVG (ms): 41.516487
multMatrixOnGPU1D <<<(32,32), (32,32)>>> elapsed 4346.540039 ms
Arrays match.
```

Matrix 2000

```
A01374530@alien1-lab:~/.../GPU_MatrixMult$ ./D2D2 2000
Matrix size: nx 2000 ny 2000
multMatrixOnHost elapsed 53419.414062 ms
AVG (ms): 290.147919
multMatrixOnGPU1D <<<(63,63), (32,32)>>> elapsed 53419.414062 ms
Arrays match.
```

Matrix 4000

```
A01374530@alien1-lab:~/.../GPU_MatrixMult$ ./D2D2 4000
Matrix size: nx 4000 ny 4000

Killed
```