

Reporte técnico

Seung Hoon Lee - A01021720

Septiembre 2018

Resumen

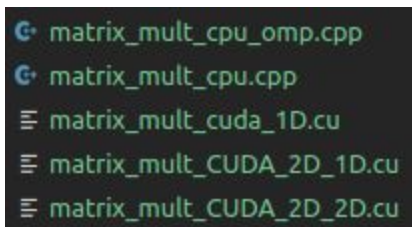
El siguiente documento es un documento que hace comparaciones de tiempo de procesamiento entre CPU, CPU usando threads y GPU con bloques e hilos. Lo que se va a hacer es un programa que haga multiplicaciones de matrices, como fue mencionado con las 3 diferentes formas y comprar sus tiempos de ejecución para ver cómo difieren.

1. Introducción

Las necesidades de cómputo de numerosas aplicaciones obligan a desarrollar software eficiente y seguro para plataformas multiprocesador. Además, el auge de los procesadores multinúcleo y de las redes de ordenadores ha aumentado la difusión del procesamiento paralelo, que cada vez está más al alcance del público en general. No obstante, para utilizar los sistemas paralelos y/o distribuidos de forma eficiente es necesaria la programación paralela.

Para eso hemos creado 5 diferentes programas en esta tarea. Uno utilizara el CPU de manera normal para hacer la multiplicación de matrices. Otro utilizara el CPU pero con threads para que se pueda utilizar openmp que es una librería de C para ejecutarlo paralelamente. Y al final se utilizaran 3 diferentes programas de GPU 1D, 2D_1D y 2D_2D con diferentes hilos y bloques para poder ver cómo son diferentes los resultados.

La tarea contiene 5 archivos principales:



```
matrix_mult_cpu_omp.cpp
matrix_mult_cpu.cpp
matrix_mult_cuda_1D.cu
matrix_mult_CUDA_2D_1D.cu
matrix_mult_CUDA_2D_2D.cu
```

2. Desarrollo

Para el primer desarrollo que es el programa que utiliza solo el procesador (matrix_mult_cpu.cpp) se hizo que se corriera el programa 10 veces y que se tomara un tiempo promedio:

```
seunglee@seunglee-G750JX ~/Desktop/Multinucleo/Tareas/a-1-matrix-multiplicat
seu95 $ ./MatrixMultCPU
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 4774 ms
Cantidad de repeticiones hechas: 10
Tamano de matriz: 1000 x 1000
```

```
seunglee@seunglee-G750JX ~/Desktop/Multinucleo/Tareas/a-1-matrix-multiplicat
seu95 $ ./MatrixMultCPU
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 36532 ms
Cantidad de repeticiones hechas: 10
Tamano de matriz: 2000 x 2000
```

```
seunglee@seunglee-G750JX ~/Desktop/Multinucleo/Tareas/a-1-matrix-multiplicat
seu95 $ ./MatrixMultCPU
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 197855 ms
Cantidad de repeticiones hechas: 10
Tamano de matriz: 4000 x 4000
```

Para el segundo desarrollo que es el programa que utiliza solo el procesador con programacion paralela (matrix_mult_cpu_omp.cpp) se hizo que se corriera el programa 10 veces y que se tomara un tiempo promedio:

```
seu95 $ ./MatrixMultOMP
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 11614 ms
Cantidad de repeticiones hechas: 10
Tamano de matriz: 1000 x 1000
```

```
seu95 $ ./MatrixMultOMP
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 112603 ms
Cantidad de repeticiones hechas: 10
Tamano de matriz: 2000 x 2000
```

```
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 197137 ms
Cantidad de repeticiones hechas: 10
Tamano de matriz: 4000 x 4000
```

Para el tercer desarrollo que son los programas que utilizan la GPU con diferentes bloques e hilos se hicieron varios diferentes ejecuciones para ver cómo eran afectados por los hilos y bloques:

```
Using Device 0: GeForce GTX 770M
Matrix size: nx 1000 ny 1000
multMat elapsed 3930.645264 ms
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 1670 ms
Cantidad de repeticiones hechas: 1
Tamano de matriz: 1000 x 1000
Matrix multiplications from host and GPU match!.
```

N = 1000	1D	Tiempo en (ms)
x = 16	y = 1	1670
x = 32	y = 1	2392
x = 64	y = 1	2930
x = 128	y = 1	2350
x = 256	y = 1	2383

```
./MatrixMultCUDA 2D1D Starting...
Using Device 0: GeForce GTX 770M
Matrix size: nx 1000 ny 1000
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 3148 ms
Cantidad de repeticiones hechas: 1
Tamano de matriz: 1000 x 1000
```

N = 1000	2D_1D	Tiempo en (ms)
x = 16	y = 1	3148
x = 32	y = 1	5399
x = 64	y = 1	5522
x = 128	y = 1	5465
x = 256	y = 1	5506

```
./MatrixMultCUDA 2D2D Starting...
Using Device 0: GeForce GTX 770M
Matrix size: nx 1000 ny 1000
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 477 ms
Cantidad de repeticiones hechas: 1
Tamano de matriz: 1000 x 1000
Matrix multiplications from host and GPU match!.
```

N = 1000	2D_2D	Tiempo en (ms)
x = 16	y = 16	477
x = 32	y = 32	857
x = 64	y = 64	N/A
x = 128	y = 128	N/A
x = 256	y = 256	N/A
x = 256	y = 1	2698

No se pudo correr después de $x = 64$ y $y = 64$ y se hizo una ultima prueba de $x = 256$ y $y = 1$.

3. Características del Sistema

Asus GJX-J750

Tarjeta Gráfica: GTX 770M

Procesador: Intel Core - i7 4th generation

4. Conclusión

Como se puede observar en los resultados, es obvio que utilizar programación paralela puede ser increíblemente eficiente. Sin embargo, algo que también se pudo observar fue que las tarjetas gráficas son una gran manera de poder ejecutar esta programación en paralelo. Es por eso que podemos ver como ha crecido el crypto-mining que son básicamente miles de tarjetas gráficas corriendo todo el tiempo.

También se puede observar que los diferentes bloques e hilos que se le ponen a la tarjeta gráfica pueden llegar a afectar la velocidad de ejecución. No siempre es mejor tener muchos bloques e hilos o viceversa, siempre es mejor pensar el problema y pensar como sería mejor dividido para que se ejecute más rápido.

Referencias

Programación Paralela. (n.d.). Retrieved from http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela_teoría/index.html

Programación paralela. (n.d.). Retrieved from https://www.ecured.cu/Programación_paralela