

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 le aplique un filtro de blur a través del método box blur a una foto lo cual sería una matriz de 1920x1080 o de diferentes tamaños. Lo que se va a hacer es multiplicar cada pixel de la matriz por la suma de otra matriz multiplicadora que sería el filtro en este caso. Para el blur en esta tarea se utilizará el box blur con una matriz de 3x3 alrededor del pixel. Esto significa que cada píxel será multiplicado por la suma de una matriz de 3x3 alrededor del pixel multiplicado por 1/9.

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.

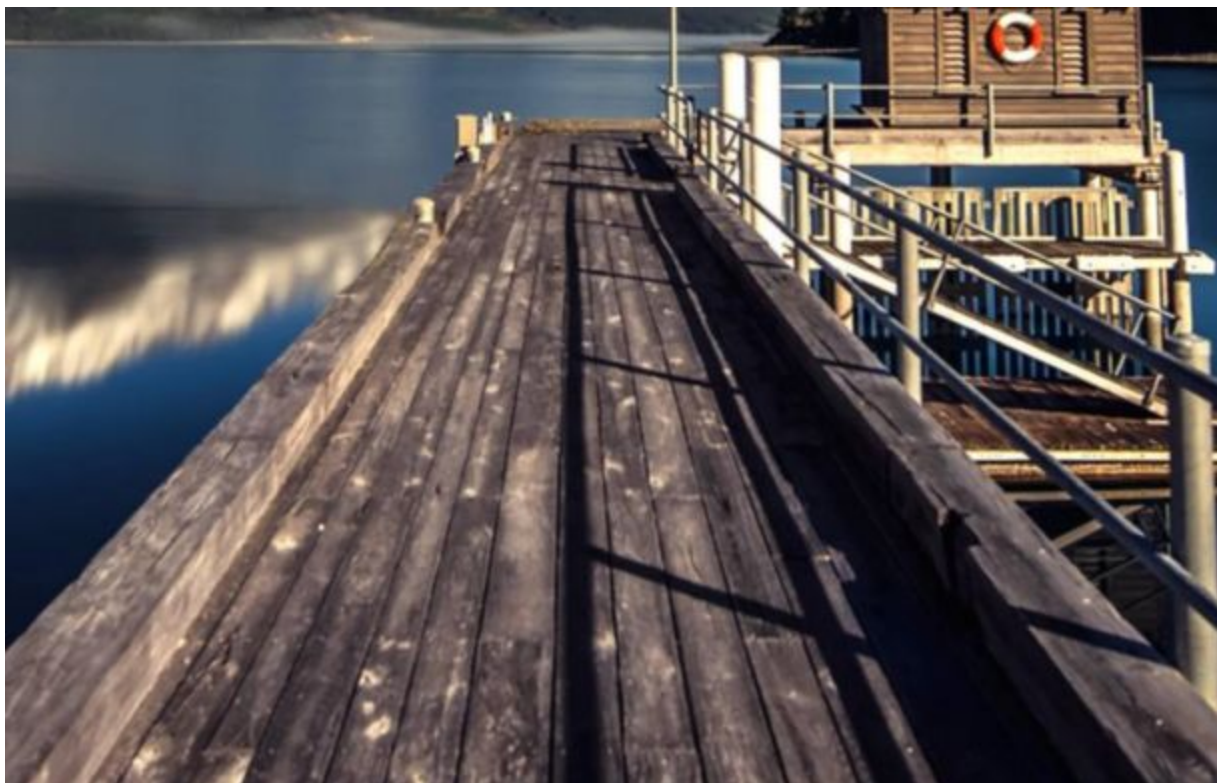
Para eso hemos creado 3 diferentes programas para esta tarea. Los archivos serán llamados blurring_CPU.cpp, blurring_OMP.cpp y blurring.cu junto con otros archivos en común como blurring.o y common.h. Lo que hará blurring_CPU.cpp será utilizar el algoritmo de box blur para aplicar el filtro de blur a una imagen (las imágenes se pueden observar también, se utilizaron 2 diferentes que son el image e imageHD. El imageHD es una imagen de 1920x1080 que fue la que se utilizó principalmente para el testing) como si fuera un programa normal sin threads y solo utilizando el CPU. El archivo o programa de blurring_OMP.cpp hará lo mismo que el programa de blurring_CPU.cpp excepto que en este se utilizarán threads o la librería de OpenMP para acelerar el proceso a través de programación paralela. Para finalizar el archivo de blurring.cu utilizará cuda para correr la multiplicación y suma de la matriz a través de la tarjeta gráfica usando bloques e hilos. Lo que se espera (con experiencia de la tarea pasada) es que blurring_CPU.cpp sea el más lento de los tres seguido de blurring_OMP.cpp y que blurring.cu sea absolutamente mucho más rápido que los otros dos.

La tarea contiene 3 archivos principales:

 blurring_CPU.cpp	9, U
 blurring_OMP.cpp	1, U
 blurring.cu	U
 blurring.o	U
 common.h	U
 image.jpg	U
 imageHD.jpg	U

2. Desarrollo

Para ver la diferencia entre las dos imágenes se tomó un screenshot:





Para el primer desarrollo que es el programa que utiliza solo el procesador (blurring_CPU.cpp) se corrió para la imagen de 1920x1080 (imageHD.jpg) y se tomó el tiempo:

Compilado con: `g++ -o blurring_CPU blurring_CPU.cpp -lopencv_core -lopencv_highgui -lopencv_imgproc -std=c++11`

```
seunglee@seunglee-G750JX ~/Desktop/Multinucleo/Tareas/assignment-2-image-blurring-l  
eeseu95 $ ./blurring CPU  
Input image step: 9216 rows: 1728 cols: 3072  
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 1015.122009 ms
```

Para el segundo desarrollo que es el programa que utiliza solo el procesador con programacion paralela (blurring_OMP.cpp) se corrió para la imagen de 1920x1080 (imageHD.jpg) y se tomó el tiempo:

Compilado con: `g++ -o blurring_OMP blurring_OMP.cpp -lopencv_core -lopencv_highgui -lopencv_imgproc -std=c++11`

```
seunglee@seunglee-G750JX ~/Desktop/Multinucleo/Tareas/assignment-2-image-blurring-l  
eeseu95 $ ./blurring OMP  
Input image step: 9216 rows: 1728 cols: 3072  
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 1009.548950 ms
```

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:

Compilado con: `make`

```
seunglee@seunglee-G750JX ~/Desktop/Multinucleo/Tareas/assignment-2-image-blurring-l
eeseu95 $ ./blurring.exe
Input image step: 9216 rows: 1728 cols: 3072
blur_kernel<<<(192, 108) , (16, 16)>>>
La cantidad de tiempo que se tarda cada ejecucion es alrededor de: 0.026908 ms con
bloque de 16 y 16
```

Se hicieron varios diferentes tests con diferentes dimensiones para los hilos/bloques:

x	y	Tiempo en (ms)
16	16	0.026908
32	16	0.027072
64	16	0.020041
128	16	0.021811
256	16	0.007347
512	16	0.006574
1024	16	0.007612
1024	32	0.008116
1024	64	0.006462
1024	128	0.006856
1024	256	0.006777
1024	512	0.006969
1024	1024	0.009112
512	512	0.006092
256	256	0.006801
128	128	0.006556
64	64	0.006966
32	32	0.022041
16	256	0.007687
16	512	0.007801
16	1024	0.006644

3. Características del Sistema

Asus GJX-J750

Tarjeta Gráfica: GTX 770M

- Cores: 960
- Max Frequency: 932MHz
- Memory Capacity: 3GB

Procesador: Intel Core - i7 4th generation

- Cores: 4
- Max Frequency: 3.8GHz
- Base Frequency: 2.8GHz

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.

De los resultados en sí se puede observar una mejora mínima entre el CPU y OMP ya que como la matriz del filtrado era bastante chica, es muy probable que el tiempo de mejora no sea tanto (en nuestro caso fueron 6 ms). Sin embargo la mejora entre el CPU y GPU (cuda) fue increíblemente alto. De 1000ms en promedio a 0.006ms. También desde los resultados se puede observar que dependiendo la cantidad de hilos y bloques que le fueron asignados a la gpu cambia el tiempo de procesamiento. Mientras que los peores fueron 0.02 ms (que de todos modos es una mejora increíblemente alta a comparación del CPU), los mejores fueron alrededor de 0.006ms. Casi 3 veces mejor. Esto demuestra que la utilización correcta de bloques e hilos importa mucho en cuanto a la programación en el GPU.

Referencias

Box blur. (2018, April 07). Retrieved from https://en.wikipedia.org/wiki/Box_blur

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