GPGPU - 2015

Informe Laboratorio

Javiel, Gonzalo - 4.666.259-5
Velazquez, Joaquin - 4.775.364-6

Consigna

Parte 1

Realizar una versión secuencial del algoritmo. El resultado debe ser una imagen en la que cada pixel se encuentre pintado con el color correspondiente al centro más cercano, el cual debe ser un promedio de los pixels pertenecientes a una ventana de tamaño 5 x 5 centrada en dicho centro.

La función que realiza la versión secuencial se llama “voronoi_CPU”, implementada en el archivo voronoi.cu.
Se utiliza en esta función el sorteo de los centros pedida en la Parte 3 para poder realizar las comparaciones entre el algoritmo secuencial y el algoritmo paralelo en igualdad de condiciones.

Imagen “fing.pgm”

Imagen resultado de ejecutar el algoritmo secuencial para la imagen “fing.pgm” con 6000 centros.

Parte 2

Realizar un kernel en CUDA que tome como entrada una imagen en escala de grises y calcule la imagen promedio para un tamaño de ventana de 5x5. Esta operación es un caso particular de la convolución vista en el práctico del curso, en el cual la máscara está formada por unos y se divide el resultado de la convolución entre el tamaño de la máscara.
Para realizar la operación DEBE utilizarse la memoria compartida.

El kernel que implementa esta parte es kernelParte2 en el archivo voronoi.cu.
Como se solicitó se utilizó la memoria compartida, "cargando" la misma de la manera sugerida en la letra. A su vez el acceso a memoria es *coalesced* para lograr un acceso más eficiente.

Imagen promedio resultado de ejecutar el kernel Parte 2 para la imagen "fing.pgm" con 6000 centros.

Parte 3

Construir una función que realice el sorteo de las coordenadas correspondientes a los centros (la función debe recibir la cantidad de centros como parámetro) y luego invoque un kernel en CUDA que tome como entrada la imagen promedio calculada en la parte anterior, y un arreglo con las coordenadas de cada centro; y devuelva otra imagen con el diagrama de Voronoi correspondiente.

La función que implementa el sorte de centros es “sorteoCentros en el archivo voronoi.cu. Para el sorteo se utilizó la función random de C.

Es pertinente mencionar que si la cantidad de centros es menor a 6000, se guarda el arreglo de coordenadas en memoria compartida para lograr optimizar el acceso a las entradas del mismo por todos los hilos.

Con el fin el de evitar copias a memoria innecesarias se toma como imagen de entrada de este kernel, la imagen promedio resultado de ejecutar el kernel de la parte anterior.

Imagen resultado de ejecutar el kernel Parte 3 para la imagen “fing.pgm” con 6000 centros.

Parte 4

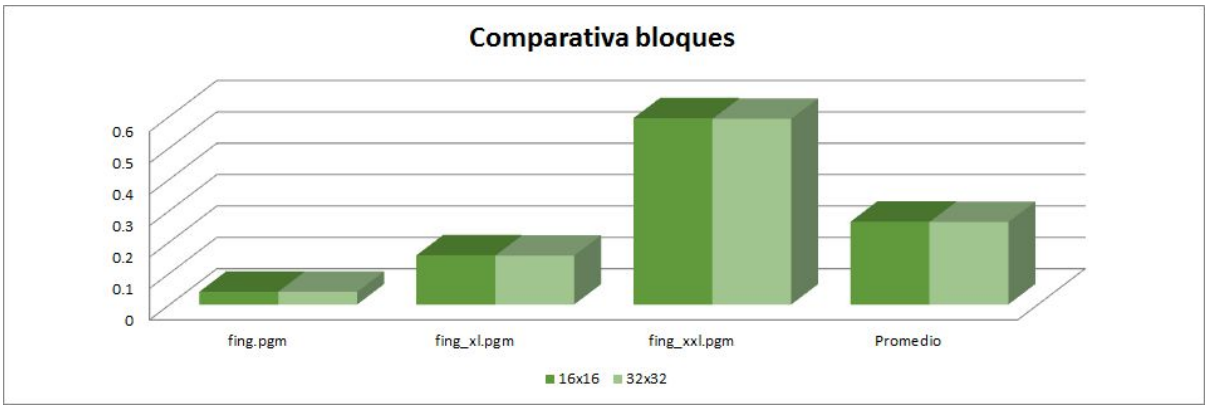
Compare los tiempos de ejecución obtenidos por las versiones secuencial (Parte 1) y paralela (Parte 2 + Parte 3) para las tres imágenes proporcionadas con tamaños de bloque de 16x16, 32x32. Para el mejor tamaño de bloque compare los tiempos de ejecución obtenidos al variar la cantidad de centros.

En primer lugar se realizaron seis ejecuciones con 1000 centros, que contienen todas las combinaciones posibles entre las tres imágenes y los tamaños de los bloques 16x16 y 32x32. Tomando el promedio para cada configuración, se puede concluir que por una mínima diferencia el mejor tamaño de bloque es 32x32.

A continuación se presentan los resultados de la prueba:

Cantidad de centros: 1000

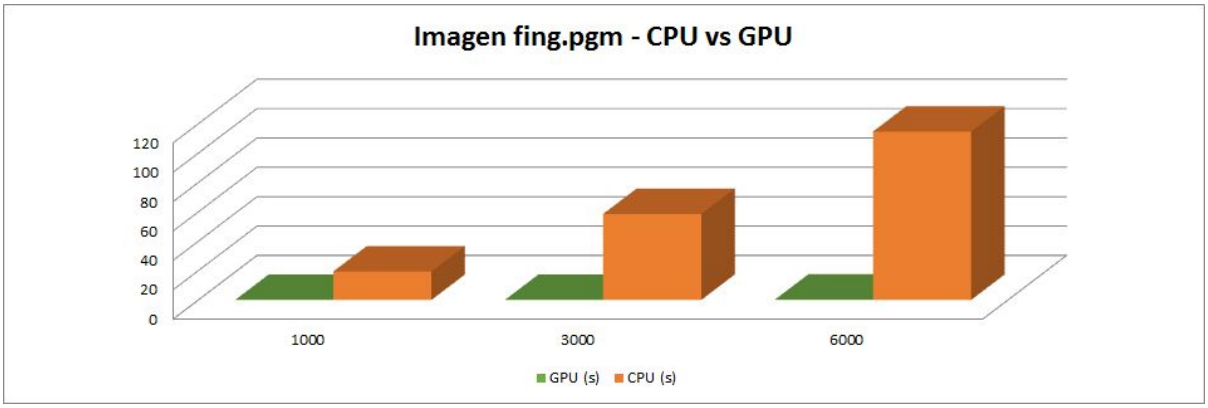
Tamaño de bloque	fing.pgm	fing_xl.pgm	fing_xxl.pgm	Promedio
16x16	0.04083	0.156789	0.594008	0.263875667
32x32	0.040716	0.156043	0.591914	0.262891



Utilizando un tamaño de bloque de 32x32 (mejor configuración) y una mascara de 5, se midieron los resultados de ejecutar el algoritmo secuencial y el algoritmo paralelo variando los centros entre 1000,6000,10000.

Imagen fing.pgm

# Centros	CPU (s)	GPU (s)
1000	19.414265	0.040681
3000	58.583384	0.106731
6000	114.660591	0.196817



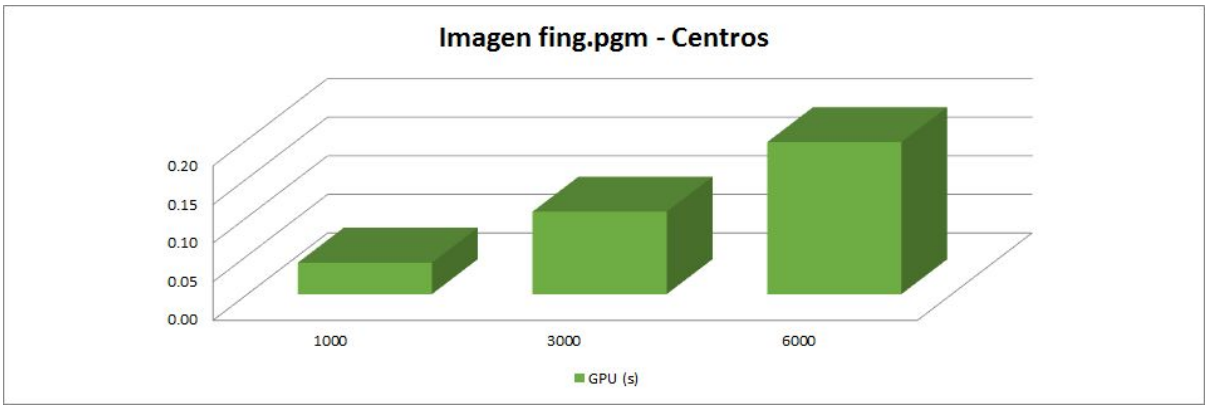
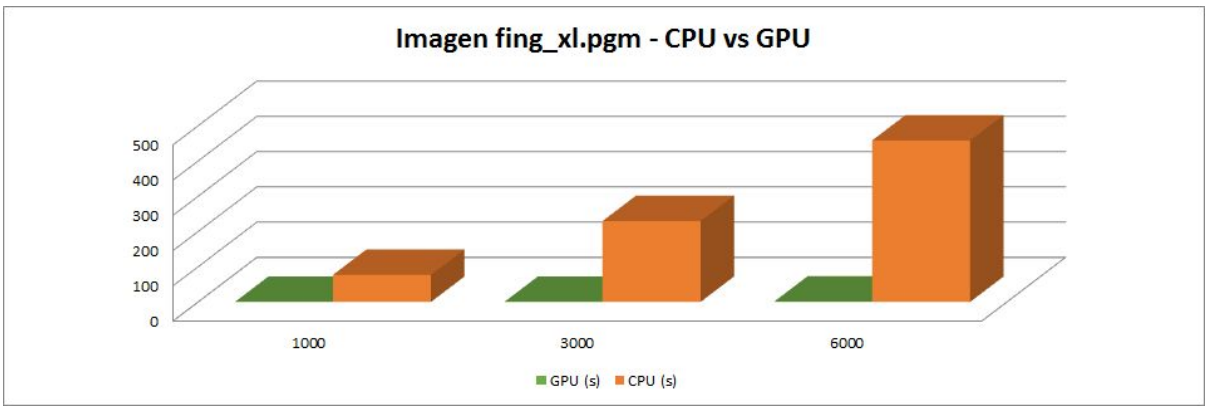


Imagen fing_xl.pgm

# Centros	CPU (s)	GPU (s)
1000	76.922203	0.155937
3000	229.561426	0.400690
6000	458.182715	0.769126



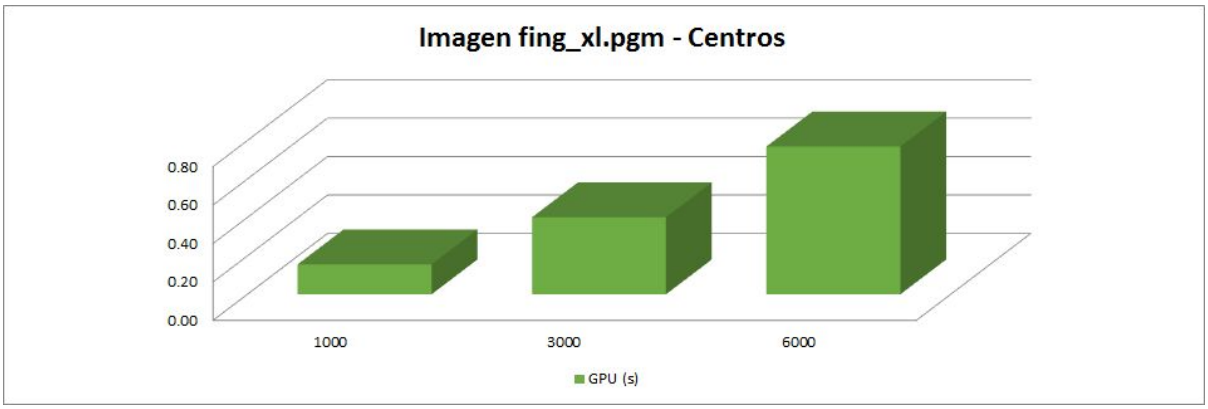
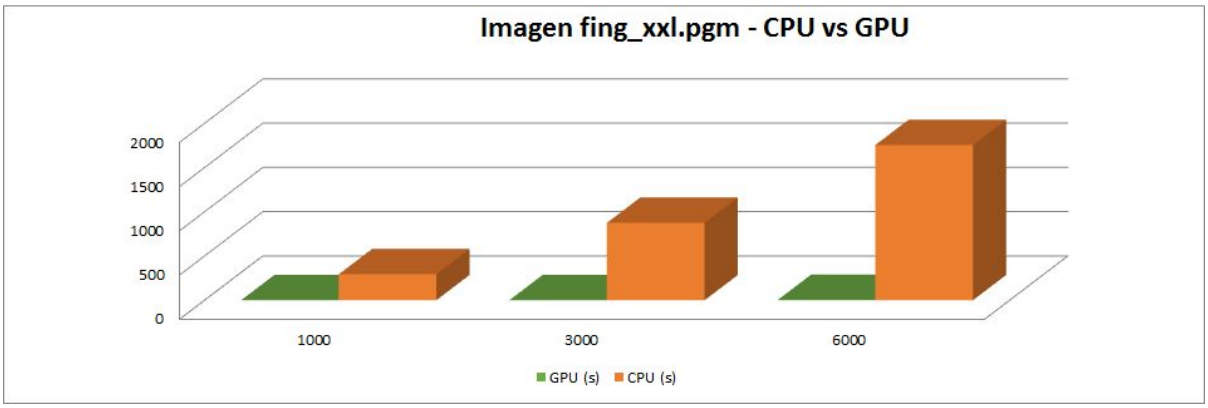
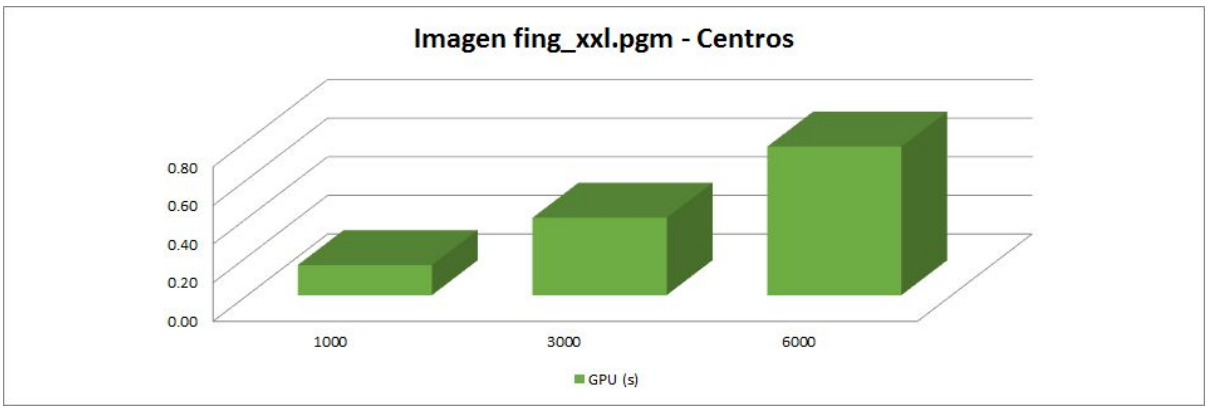


Imagen fing_xxl.pgm

# Centros	CPU (s)	GPU (s)
1000	293.474907	0.588668
3000	876.040566	1.520121
6000	1755.187662	3.637830





Problemas encontradas

Al aumentar la cantidad de centros a un número mayor a 9500 el sistema operativo bloqueaba la ejecución del algoritmo en la GPU, para solucionar este inconveniente se modificó el registro de windows por sugerencia del docente Ernesto Dufrechou.
Link a la solución https://msdn.microsoft.com/en-us/library/ff569918(v=vs.85).aspx

A su vez fue necesario ejecutar el algoritmo como usuario administrador para una cantidad de centros elevada ya que sistema operativo bloqueaba la ejecución para usuarios sin derechos.