Computación Paralela
Practica 1

Jairo Suárez
Universidad Nacional Bogotá, Colombia
Email:jaasuarezga@unal.edu.co

Figura 1: Escudo de la Universidad.

***Resumen*—En este documento se muestra el análisis, resultados, gráficas y conclusiones de la aplicación de un algoritmo para generar el efecto borroso de una imagen comparando su rendimiento de ejecución con implementación de hilos.**

I. Introducción.

La paralelización permite dividir un problema grande en problemas pequeños que se pueden resolver simultaneamente, con esta practica se hace un análisis para comparar el rendimiento de un algoritmo paralelizable para generar el efecto borroso de una imagen, en este caso el algoritmo empleado para la practica que se uso fue el Normalized Box filter.

Para la prueba del algoritmo se utilizaron imagenes de resolución 720p, 1080p, y 4k.

October 26, 2017

I-A. Efecto borroso de una imagen.

El suavizado, también llamado borrado, es una operación de procesamiento de imágenes simple y de uso frecuente.

I-A1. Método.: Normalized Box filter - Este algoritmo usado para la generación del efecto borroso de una imagen consiste en un filtro lineal, en el que el valor de un píxel de salida se determina como una suma ponderada de valores de píxeles de entrada, es decir:

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$$

Donde i,j es la posición del pixel y h(k,l) es el kernel asociado, es decir los coeficientes del filtro. La salida de cada pixel es la media de sus vecinos en el kernel, en este caso el kernel es una cuadricula alrededor del pixel, que contiene todos sus vecinos.

En la siguiente imagen se muestra un ejemplo de la cuadricula generada para un pixel, el cual se sombrea de amarillo, el tamaño de la cuadricula para el ejemplo es de 5.

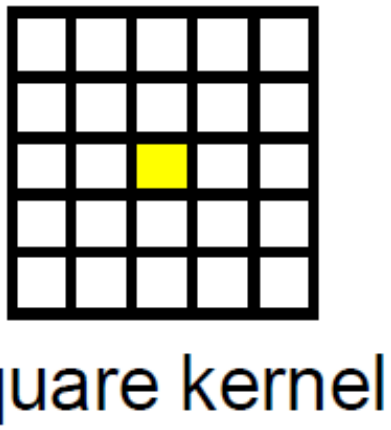

Figura 2: Representación gráfica del kernel de un pixel.

II. Paralelización del algoritmo.

Para la paralelización del algoritmo se particionó la imagen por bloques de igual tamaño, cada bloque corresponde a un hilo, de esta manera cada hilo se encarga de ejecutar el algoritmo por el bloque que tiene asginado la imagen, luego de que todos los hilos realizaron el efecto borroso de la imagen, se juntan los bloques para conformar de nuevo la imagen.

Figura 3: Imagen Original.

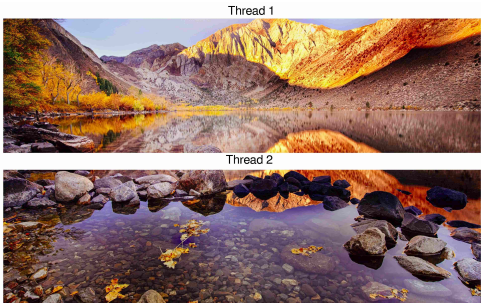

Figura 4: Imagen particionada en dos bloques.

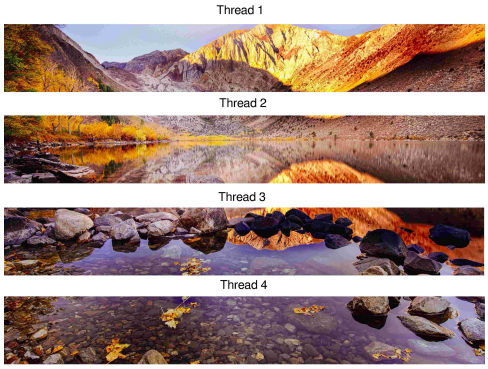

Figura 5: Imagen particionada en cuatro bloques.

III. Experimentos y resultados.

Para comprobar el algoritmo y su eficiencia con diferentes cantidad de hilos, se utilizó imagenes con resolución 720p, 1080p y 4k a las cuales se les aplicara la función de blur. Para la manipulación de las imágenes se uso la libreria opencv, el algoritmo fue implementado en el lenguaje de programación C++.

La ejecución del programa se realizó en un computador portatil con procesador Intel Core i7-4510U CPU de 2.00GHz y con 4 núcleos. Se utilizaron tres métodos diferentes para ejecutar la paralelización para luego comparar, la primera utilizando hilos POSIX, la segunda utlizando la libreria OpenMP y por último utlizando GPU con la plataforma de procesamiento paralelo CUDA. A continuación se listarán estos tres métodos y sus resultados.

- **POSIX:** Los resultados que se muestran a continuación son los tiempos de ejecución por cada imagen con 1, 2, 4, 8 y 16 hilos usando hilos POSIX con la libreria $< pthread.h >$.

Cuadro I: Tiempo de respuesta imagen 720p

Imagen con resolución 720p,				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
1	*9*	1.52	21	8.23
2	*9*	0.87	21	4.54
4	*9*	0.81	21	4.27
8	*9*	0.81	21	4.24
16	9	0.81	21	4.18

Cuadro II: Tiempo de respuesta imagen 1080p.

Imagen con resolución 1080p				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
1	*9*	3.49	21	18.60
2	*9*	1.94	21	10.28
4	*9*	1.85	21	9.69
8	*9*	1.86	21	9.70
16	9	1.84	21	9.60

Cuadro III: Tiempo de respuesta imagen 4k.

Imagen con resolución 4k				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
1	*9*	13.64	21	1:14.64
2	*9*	7.62	21	0:42.28
4	*9*	7.34	21	0:43.19
8	*9*	7.36	21	0:43.10
16	9	7.32	21	0:43.78

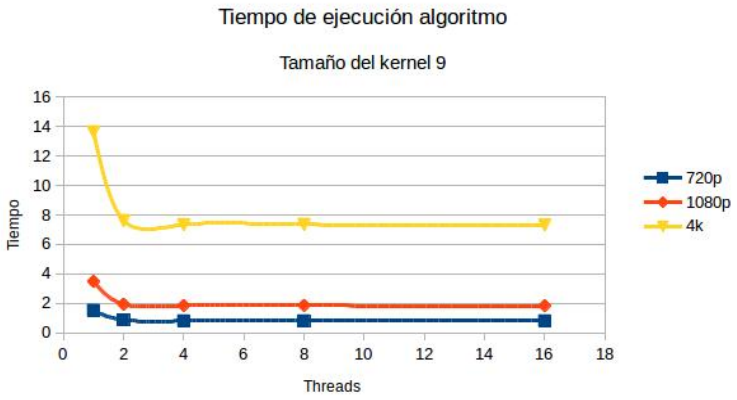

Figura 6: Tiempos de respuesta con Kernel 9.

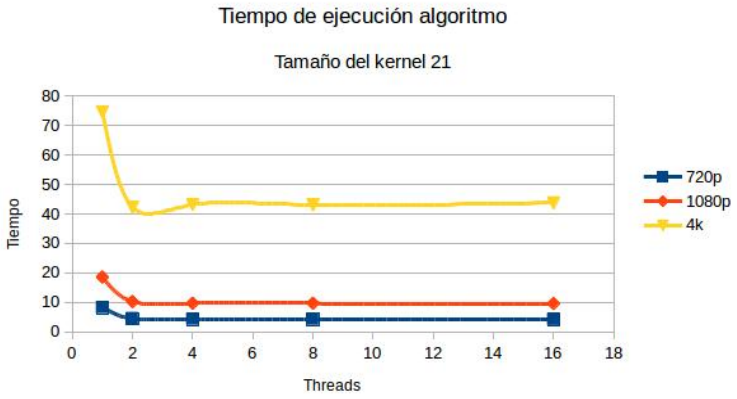

Figura 7: Tiempos de respuesta con Kernel 21.

Para hallar los tiempos del Speedup de los experimentos con kernel 9 y 21 se utilizó la siguiente función:

$$S_p(n) = \frac{T*}{T_p(n)}$$

Al hallar el speedup para cada imagen con su respectivo kernel se obtuvieron los siguientes resultados representados en gráficas:

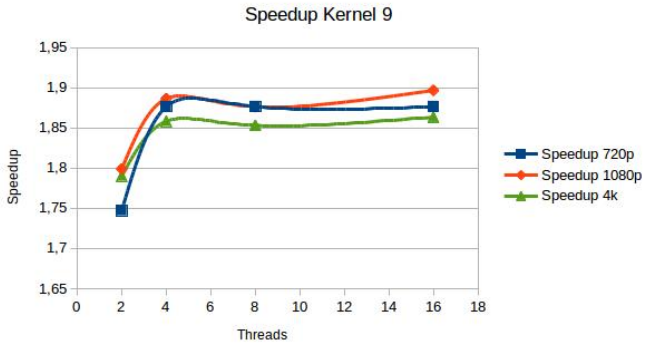

Figura 8: Speedup con Kernel 9

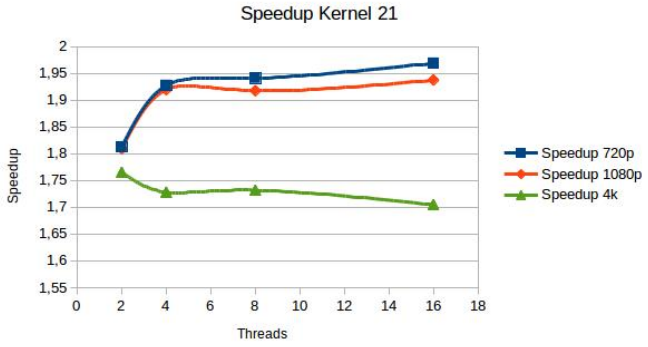

Figura 9: Speedup con Kernel 21

- **OpenMP:** Los resultados que se muestran a continuación son los tiempos de ejecución por cada imagen con 1, 2, 4, 8 y 16 hilos usando la libreria OpenMP y con la forma de separamiento de cargas **Loop construct.**

Cuadro IV: Tiempo de respuesta imagen 720p

Imagen con resolución 720p,				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
1	*9*	1.57	21	8.34
2	*9*	0.92	21	4.96
4	*9*	0.89	21	4,63
8	*9*	0.91	21	4.75
16	9	0.92	21	4.82

Cuadro V: Tiempo de respuesta imagen 1080p.

Imagen con resolución 1080p				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
1	*9*	3.51	21	19.05
2	*9*	2.09	21	11,23
4	*9*	2.00	21	10.53
8	*9*	2.05	21	10.90
16	9	2.08	21	10.85

Cuadro VI: Tiempo de respuesta imagen 4k.

Imagen con resolución 4k				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
1	*9*	13.75	21	1:15.52
2	*9*	8.24	21	0:45.44
4	*9*	7.83	21	0:44.92
8	*9*	8.06	21	0:47.95
16	9	8.16	21	0:51.35

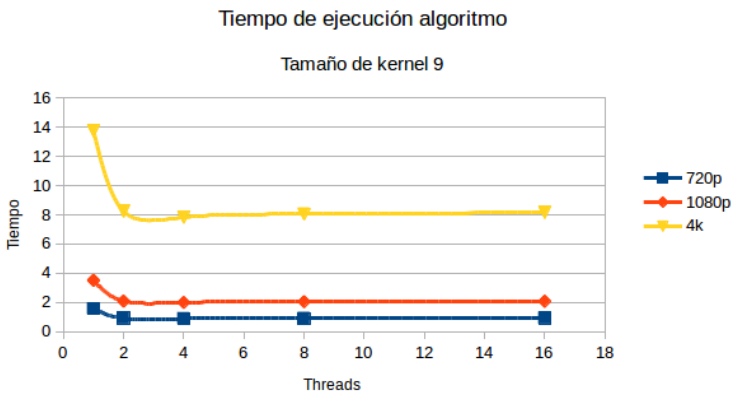

Figura 10: Tiempos de respuesta con Kernel 9.

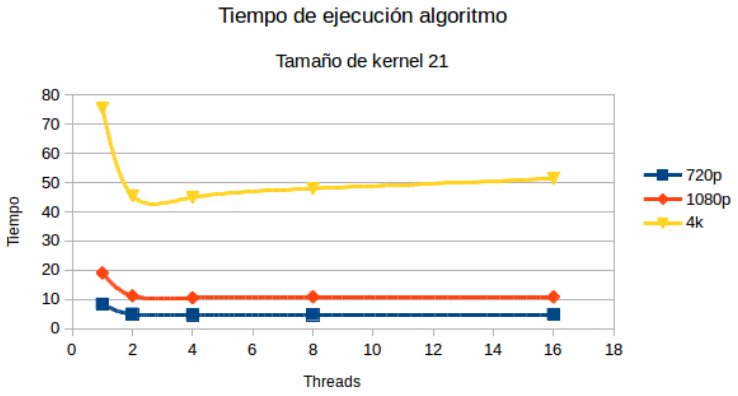

Figura 11: Tiempos de respuesta con Kernel 21.

Al hallar el speedup para cada imagen con su respectivo kernel se obtuvieron los siguientes resultados representados en gráficas:

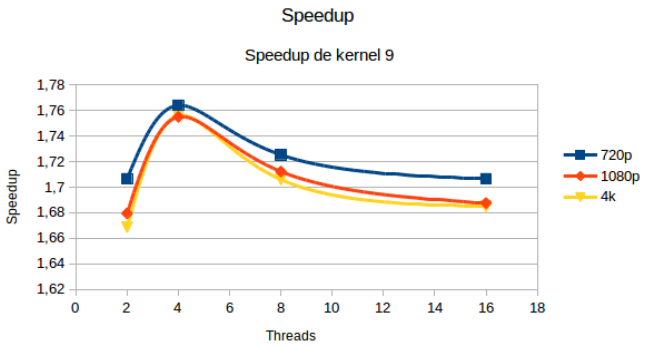

Figura 12: Speedup con Kernel 9

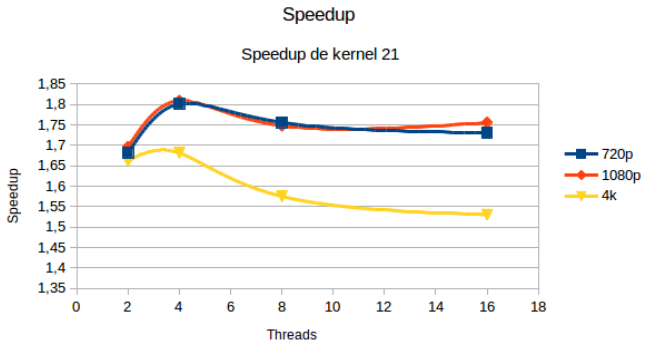

Figura 13: Speedup con Kernel 21

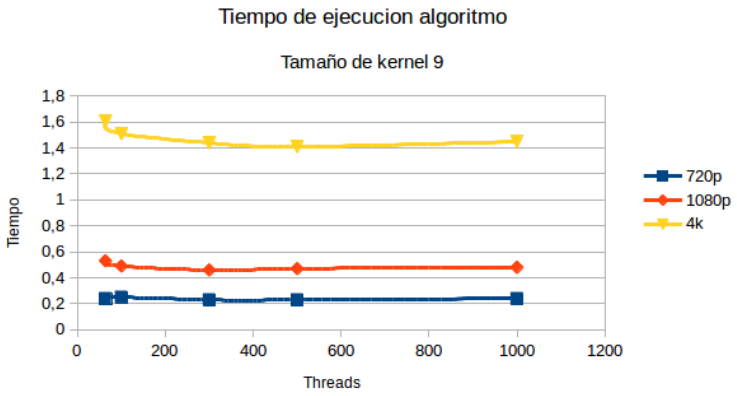

Figura 14: Tiempos de respuesta con Kernel 9.

- **CUDA:** Los resultados que se muestran a continuación son los tiempos de ejecución en GPU por cada imagen con 64, 100, 300, 500,y 1000 hilos usando GPU NVIDIA con 2 MP (Multiprocesadores) y 64 core en cada multiprocesador con 1.25 GHz y con la plataforma CUDA.

Cuadro VII: Tiempo de respuesta imagen 720p

Imagen con resolución 720p,				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
64	*9*	0.24	21	0.67
100	*9*	0.25	21	0.75
300	*9*	0.23	21	0.70
500	*9*	0.23	21	0.72
1000	9	0.24	21	0.85

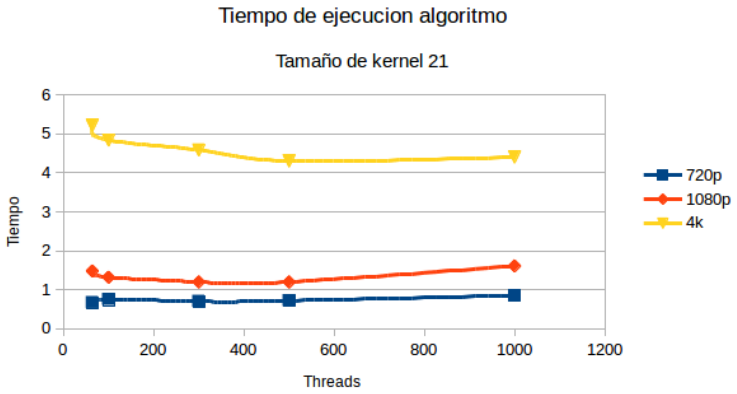

Figura 15: Tiempos de respuesta con Kernel 21.

Cuadro VIII: Tiempos de respuesta imagen 1080p.

Imagen con resolución 1080p				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
64	*9*	0.53	21	1.48
100	*9*	0.49	21	1.32
300	*9*	0.46	21	1.2
500	*9*	0.47	21	1.2
1000	9	0.48	21	1.61

Al hallar el speedup para cada imagen con su respectivo kernel se obtuvieron los siguientes resultados representados en gráficas:

Cuadro IX: Tiempo de respuesta imagen 4k.

Imagen con resolución 4k				
Numero de Threads	**Kernel**	**Tiempo (s)**	**Kernel**	**Tiempo (s)**
64	*9*	1.61	21	5.22
100	*9*	1.51	21	4.84
300	*9*	1.44	21	4.59
500	*9*	1.41	21	4.31
1000	9	1.45	21	4.41

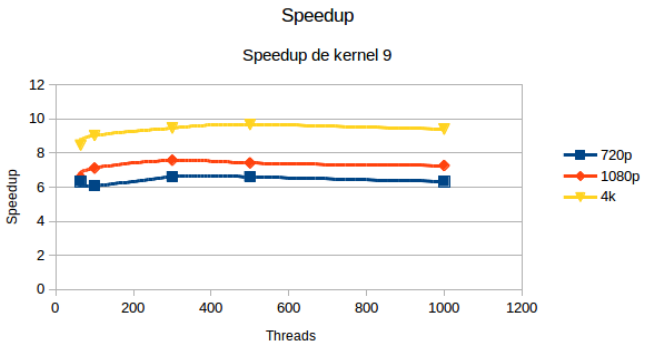

Figura 16: Speedup con Kernel 9

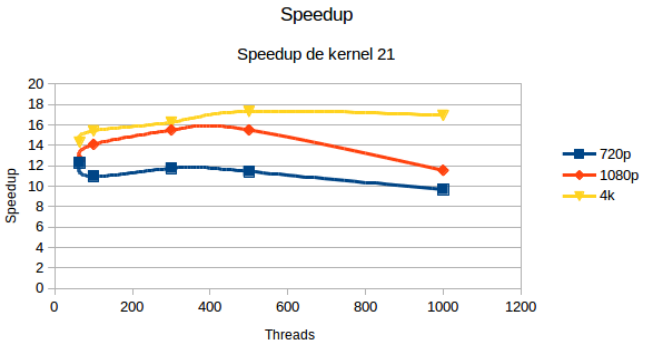

Figura 17: Speedup con Kernel 21

CONCLUSIONES.

Con esta práctica se evidenció la eficiencia de aplicar la paralelización a un algoritmo que genera el efecto borroso de una imagen. Se pudo notar que mientras más hilos hayan, los tiempos de respuesta serán menores debido al particionamiento del problema con hilos. En la experimentación debido a que el procesador tenía 4 núcleos no se notó mucho la diferencia de tiempos entre 4, 8 y 16 hilos como si se noto la diferencia entre 1, 2 y 4 hilos respectivamente.

A diferencia de la GPU con la CPU, en esta se evidenció un notorio cambio en los tiempos debido a la cantidad de cores de una respecto a la otra, mientras la CPU contaba con 4 cores, la GPU tenía 128, la diferencia es muy grande, por esto también los resultados proporcionados por cada método con CPU y GPU varian bastante.

También se pudo notar que entre mayor resolución tiene la imagen, mayor son los tiempos de respuesta, sin embargo, los tiempos comparados entre hilos se reducen considerablemente.