Computación Paralela y Distribuida
Practica 1

Integrantes:
Cristian Alejandro Mantilla Duque - camantillad
David Stephan Ramirez Camacho - dsramirezc
Duvan Camilo Albarracin vargas - dcalbarracinv

Abstract

En el presente documento exponemos el desarrollo de la práctica 1 de la materia computación paralela y distribuida, para la cual se solicitó desarrollar un programa que ejecutara el efecto blur sobre una imagen dado un tamaño de kernel y un número de hilos.

Para el desarrollo de la práctica inicialmente se contempló el uso de del gaussian blur estándar, el cual como su nombre lo indica define el kernel bajo la función de Gauss, vista a continuación:

$$G(x,\ y)\ = 1\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Su implementación se realizó haciendo uso del triángulo de pascal, multiplicando la fila n del triángulo como vector fila y vector columna para luego normalizar la matriz nxn resultante dividiendo por la suma al cuadrado de los valores del triángulo. Obteniendo por ejemplo el siguiente kernel n=5.

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Fig 1. Gauss-blur kernel n=5

Sin embargo se evidencio que el tiempo de ejecución de dicho algoritmo era muy deficiente por lo que se prosiguió a buscar un algoritmos con mayor eficiencia. Tras un periodo de búsqueda dimos con un artículo online “Fastest Gaussian Blur (in linear time)” escrito por Ivan Kutskir, en el cual basándose en el trabajo “Fast image convolutions”[1] de Wojciech Jarosz implementó algoritmos de blurring más eficientes. De este tomamos referencia para utilizar el Algoritmo 3 de complejidad O(n*r) el cual hace uso de una función boxBlur para aproximar una pasada del algoritmo de Gauss, definida como:

$$bb[i,j]\ = \sum_{y=i-br}^{i+br}\ \sum_{x=j-br}^{j+br} f[x,y]/(2*br)^2$$

Se dio inicio a su uso pero no se obtenían los resultados esperados pues este efecto ha de ser utilizado sobre cada canal de color, respuesta que obtuvimos a partir de los comentarios del artículo hechos por el mismo autor.

Siendo así entonces procedimos a separa la imagen en base a sus canales RGB, una vez

hecho esto obtuvimos los resultados de blurring esperados. Siguiente a esto se empezó el proceso de paralelización, el cual se contempló como blockwise dividiendo así cada uno de los canales en bloques tales que cada hilo procesaría aquel segmento.

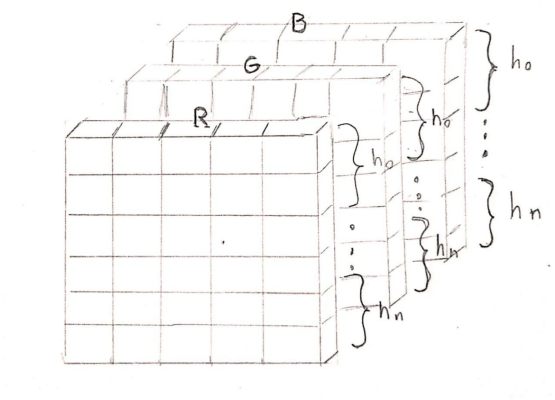

Fig 2. Segmentación en canales RGB y paralelización blockwise

Una vez implementado se obtuvieron los siguientes resultados en tiempo, cabe destacar que las espcificaciones de la maquina corresponden a un computador HP con procesardor AMD A8-6410 APU with AMD Radeon R5 Graphics.

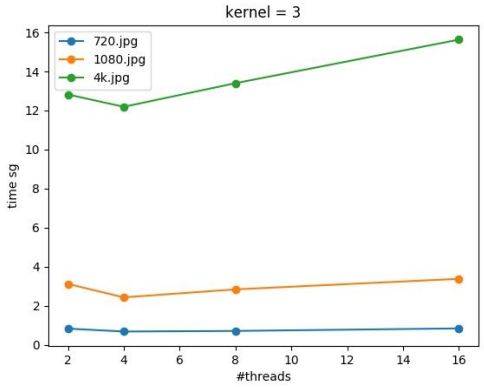

Fig 3. Tiempo para kernel n = 3

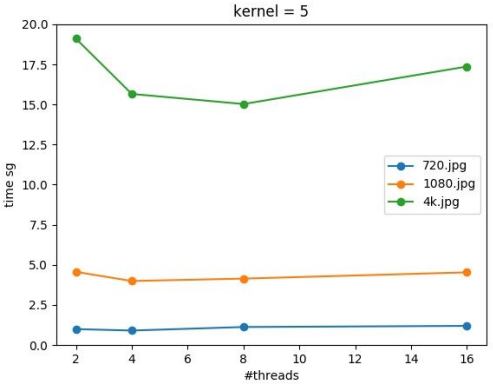

Fig 4. Tiempo para kernel n = 5

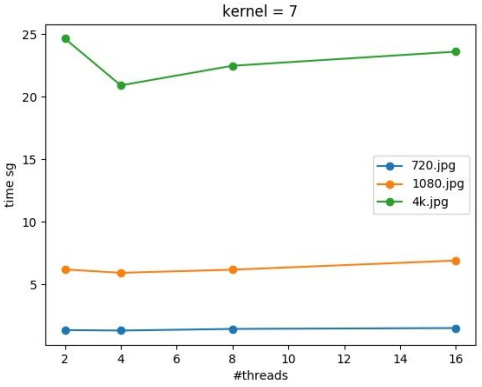

Fig 5. Tiempo para kernel n = 7

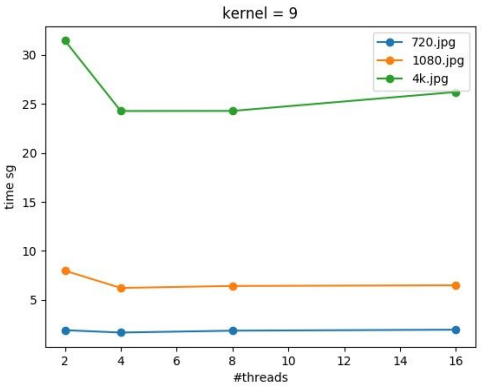

Fig 7. Tiempo para kernel n = 7

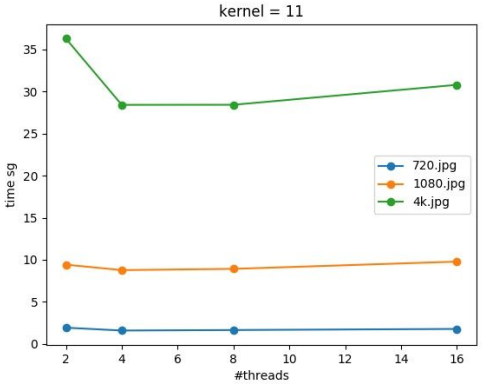

Fig 4. Tiempo para kernel n = 11

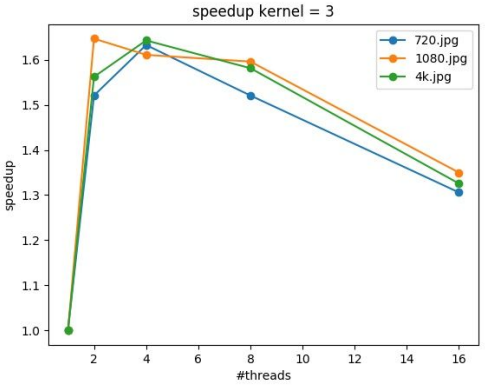

Fig 7. Speedup para kernel n = 3

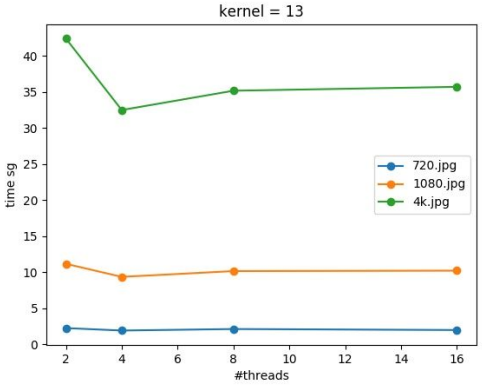

Fig 5. Tiempo para kernel n = 13

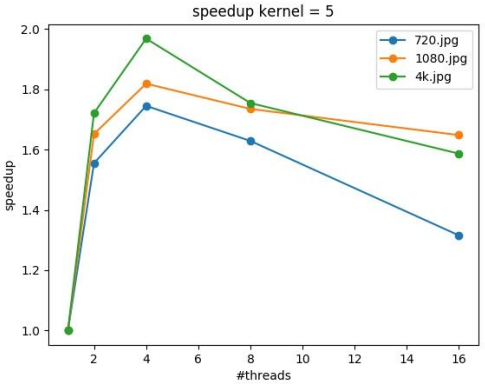

Fig 8. Speedup para kernel n = 5

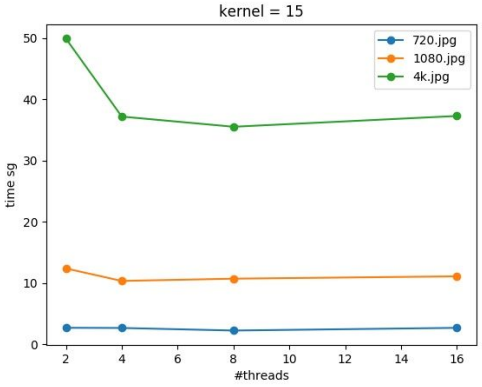

Fig 6. Tiempo para kernel n = 15

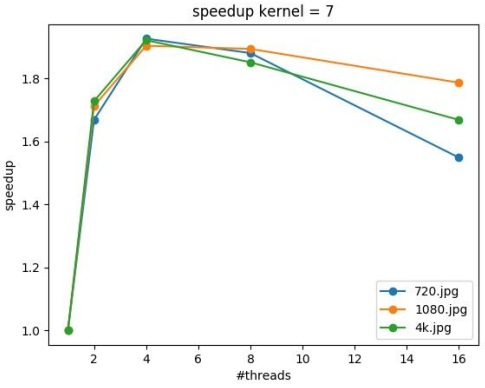

Fig 9. Speedup para kernel n = 7

Y los siguientes resultados en speedup

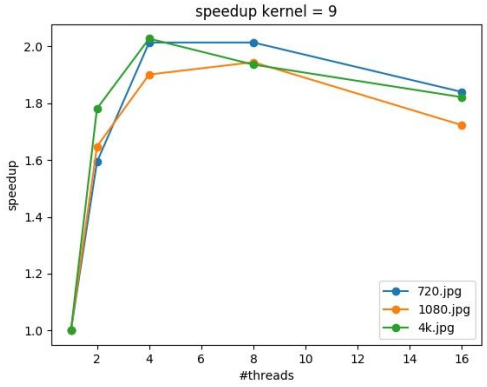

Fig 8. Speedup para kernel n = 9

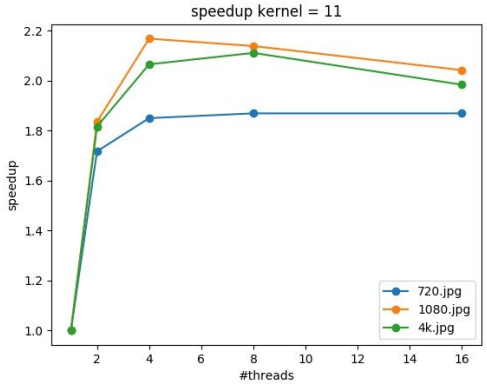

Fig 9. Speedup para kernel n = 11

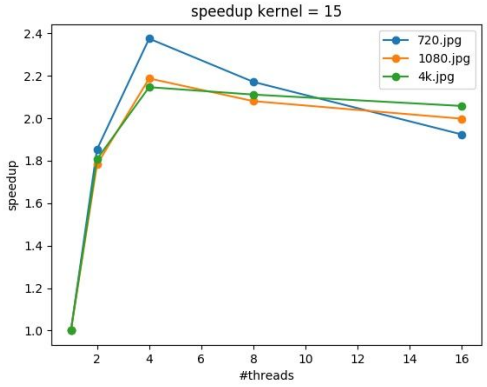

Fig 10. Speedup para kernel n = 15

Tras analizar los resultados se observa que se mas o menos con lo esperado, debido a que para ciertos casos se observa que el tiempo para mayor número de threads en ocasiones puede ser mayor que para un número más pequeño del mismo, sin embargo se puede contemplar una explicación en que el computador cuenta con solo 4 nucleos, asi que al correr sobre un mayor número de núcleos se observa que el tiempo de ejecución puede ser mayor pues la tarea no se reparte unitariamente sino que un núcleo ha de procesar la tarea que en un principio había sido asignada a otro. Con ello podemos da paso a la reflexión acerca de la gran herramienta que constituye la paralelización en el mejoramiento del performance de un programa que cumple con la condición de se paralelizable.

Referencias

[1] Fastest Gaussian Blur (in linear time), Ivan Kutskir Recuperado de: http://blog.ivank.net/fastest-gaussian-blur.html