

Computación Paralela y Distribuida - Practica #1

Angel David Corredor Morales
adcorroedorm@unal.edu.co

Sebastian Felipe Delgado Perez
sfdelgadop@unal.edu.co

I. INTRODUCCIÓN

Este documento recopila los resultados presentados al realizar el primer laboratorio de la materia Computación paralela y distribuida del departamento de ingeniería de sistemas e industrial dictada a la carrera de ingeniería de sistemas y computación, el taller consta de desarrollar el efecto de distorsión en múltiples hilos usando POSIX bloqueante.

II. EFECTO DE DIFUMINADO

El efecto de difuminado se logra usando una matriz de convolución kernel, también conocido como máscara, básicamente cada píxel pasa por el kernel, el cual podríamos considerar un “filtro”, y el nuevo píxel es calculado usando los píxeles adyacentes en conjunto con el kernel como se ve en la imagen 1.

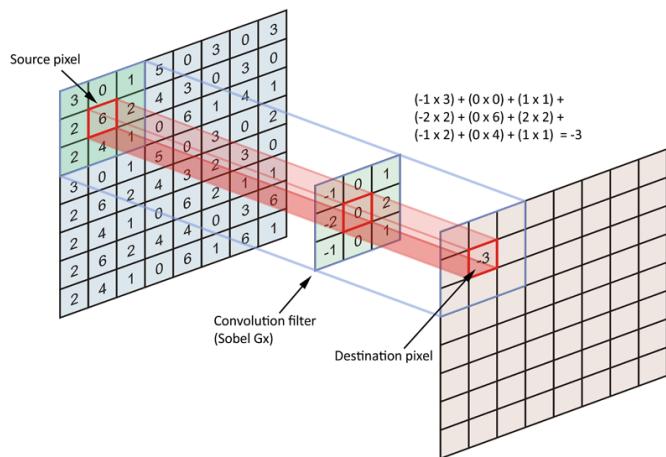


Fig. 1: Operación de Convolucion basada en Kernel
https://github.com/OKStateACM/AI_Workshop/wiki/Computer-Vision-with-Convolution-Networks

El cálculo del nuevo píxel es una convolución entre dos matrices, en este caso en particular se decidió por un promedio de los píxeles circundantes para generar el efecto de difuminado.

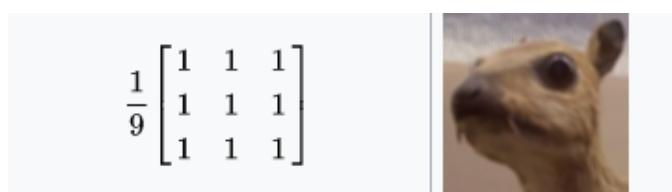


Fig. 2: Ejemplo de kernel de difuminado tamaño 3
[https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

En la figura 3b se puede observar el resultado de aplicar el filtro de difuminado con un tamaño de kernel igual a 15.



(a) Imagen Original



(b) Imagen Difuminada con un kernel de 15

III. OBJETIVO Y ENTORNO DE TRABAJO

El objetivo de la práctica es ver cómo se comporta el paralelismo realizado con diferentes hilos y kernels de tamaño distinto. El equipo en el que se realizaron las pruebas es un n1-standard-16 (16 vCPUs, 60 GB de memoria) El cual es una máquina virtual funcionando en la plataforma de Google Cloud, como el nombre indica cuenta con 16 núcleos y 60Gb de memoria. También es importante mencionar que todos los experimentos se ejecutaron 3 veces para evitar la aleatoriedad

que puede ocurrir al momento de ejecutarlos, y los resultados que se ven a continuación son el promedio de los 3 resultados.

IV. RESULTADOS

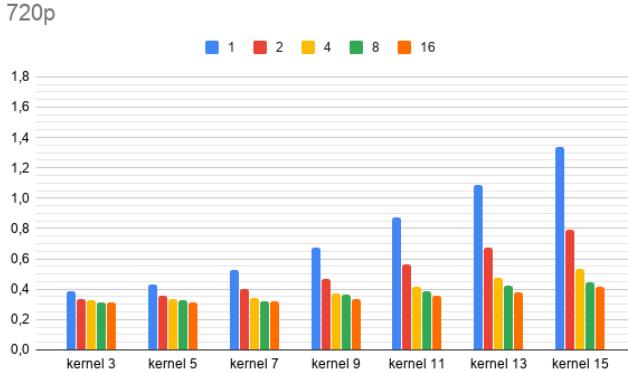


Fig. 4: Resultados obtenidos para una imagen de 720p

Con la imagen a 720p podemos ver que en los tamaños más pequeños de kernel, la mejoría es apenas apreciable, cosa distinta a lo que se ve con un kernel de 15 el cual pasa de 0.79 con dos hilos a 0.41 usando 16, una mejoría de un 50% aprox, usando 8 veces más núcleos, mientras que usando 2 veces más núcleos pasa de 0.79 a 0.53 una mejora del 32%, el cual es el salto más drástico mientras duplicamos la capacidad de cómputo.

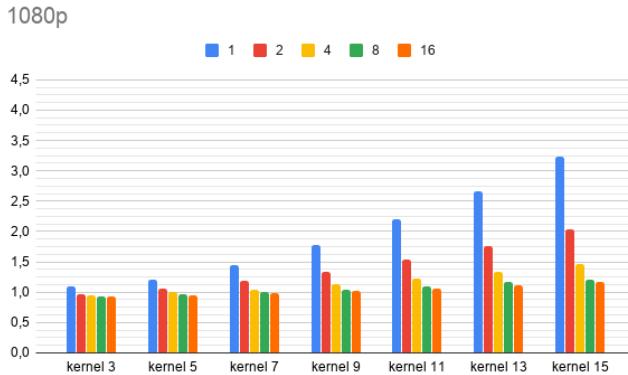


Fig. 5: Resultados obtenidos para una imagen de 1080p

Similar al caso anterior en los kernel más pequeños la mejoría entre núcleos es bastante pequeña, en cuanto al caso más notable, el del kernel de tamaño 15 las diferencias porcentuales empiezan a ser algo más pequeñas, de 2 a 4 hilos hay una diferencia del 28.2% mientras que de 2 a 16 una de 42%. además la diferencia entre 8 y 16 hilos empieza a ser bastante pequeña con solo una mejoría de 4% en el mejor de los casos.

Se sigue la tendencia que ya se tenía con la diferencia de que en este caso las mejoras interesantes se empiezan a ver desde un tamaño de kernel 9, mientras que en los casos anteriores

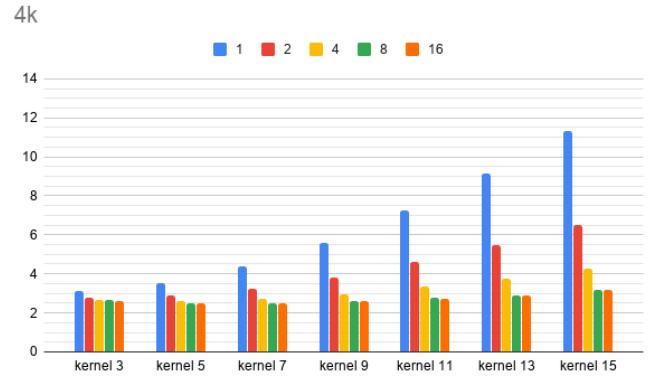


Fig. 6: Resultados obtenidos para una imagen de 4k

empezaban en el 11, volviendo al caso más interesante, el del kernel de tamaño 15, de 2 a 4 hilos se ve una diferencia de un 34.66%, el más alto en todos los experimentos, y siguiendo con esa línea de 2 a 16 hilos hay una mejoría del 51.53% la cual es también la más alta. Pero de 8 a 16 es en dónde se presenta la diferencia más pequeña, de tan solo un 0.42%. Además es importante notar que el aumento en tiempo usando 2 hilos entre un kernel de 13 y 15 aumenta en un 18%, el más grande entre las distintas imágenes.

A. SPEEDUP

Finalmente se calcula el speedUp alcanzado por el algoritmo debido a la paralelización, el cual se presenta en las tablas I, II y III:

Hilos	Tamaño del kernel						
	3	5	7	9	11	13	15
2	1,1555	1,2108	1,3171	1,4383	1,5379	1,6145	1,6903
4	1,1730	1,2906	1,5433	1,8151	2,0772	2,3041	2,5152
8	1,2224	1,3298	1,6287	1,8573	2,2494	2,5711	2,9873
16	1,2190	1,3730	1,6306	1,9994	2,4278	2,8417	3,2458

TABLE I: Speedup alcanzado en la imagen 720p

Hilos	Tamaño del kernel						
	3	5	7	9	11	13	15
2	1,1369	1,1413	1,2160	1,3397	1,4274	1,5185	1,5920
4	1,1589	1,2076	1,3960	1,5734	1,7997	1,9945	2,2126
8	1,1721	1,2439	1,4554	1,7019	2,0217	2,2740	2,6878
16	1,1843	1,2707	1,4859	1,7546	2,0768	2,3956	2,7435

TABLE II: Speedup alcanzado en la imagen 1080p

Hilos	Tamaño del kernel						
	3	5	7	9	11	13	15
2	1,1169	1,2125	1,3388	1,4707	1,5812	1,6629	1,7303
4	1,1680	1,3440	1,5895	1,8832	2,1763	2,4230	2,6467
8	1,1851	1,4106	1,7421	2,1622	2,6304	3,1343	3,5627
16	1,1937	1,4223	1,7537	2,1696	2,6690	3,1329	3,5667

TABLE III: Speedup alcanzado en la imagen 4k

V. CONCLUSIONES

- En los kernel pequeños las diferencias son especialmente pequeñas si las comparamos con kernel más grandes.
- La diferencia entre 8 y 16 hilos es imperceptible en las imágenes 4k.
- El salto más grande en rendimiento siempre se presenta en el salto de 2 a 4 hilos de ejecución y así mismo se notó más en la imagen a 4k que en los casos anteriores.
- En todos los casos el tiempo aumenta de manera más dramática con el aumento de kernel son los 2 hilos.
- Teniendo en cuenta que la diferencia entre 8 y 16 hilos es tan pequeño no es un salto rentable comparado con el beneficio que se consigue.