Computación Paralela y Distribuida
Práctica CUDA

Angel Rendón, amrendonsa@unal.edu.co
Andrés Forero, afforeroc@unal.edu.co
Universidad Nacional Bogotá, Colombia

Figura 1: Escudo de la Universidad

***Resumen*—Informe de práctica que muestra las ventajas de CUDA como modelo de paralelización sobre GPU, aplicando el efecto borroso (Blur) en tres imágenes con distintos tipos de resolución: 720p, 1080p y 4K; y con variación del tamaño del kernel entre 3 y 15. Mostrando los análisis de tiempo de respuesta y speedup. El código fuente implementa CUDA, memoria dinámica y estructuras de datos en lenguaje C.**

I. Introducción

Partiendo del concepto de computación paralela, se dispuso a implementar un algoritmo de efecto borroso para imágenes de alta calidad y sobre este, estudiar el efecto combinado de hilos y del kernel sobre el tiempo de respuesta y speedup. Lo anterior se hace de suma importancia, debido a la eficiencia en el procesamiento de imágenes que tienen como condiciones: 1) los archivos visuales de gran tamaño deben soportados por el hardware que se disponible actualmente en el modelo de computación clásica y 2) se debe considerar los límites de paralelización que están determinados por la ley de Amdahl.

Junio 25 de 2019

II. Efecto borroso

Cuando desenfocamos una imágen, hacemos que la transición entre colores de los pixeles que la componen sea más suave.

- Efecto borroso promedio: Toma un pixel referente y sobre un conjunto de pixeles alrededor promedia cada uno de los canales de colores de RGB. El conjunto de pixeles alrededor varía en tamaño; esto se llama kernel. Y el promedio se realiza sobre todos los pixeles de la imágen.
- Efecto borroso gaussiano: Es similar al efecto borroso promedio pero el promedio de los canales de colores se calcula con distintos pesos en función de la cercanía del píxel referencia.

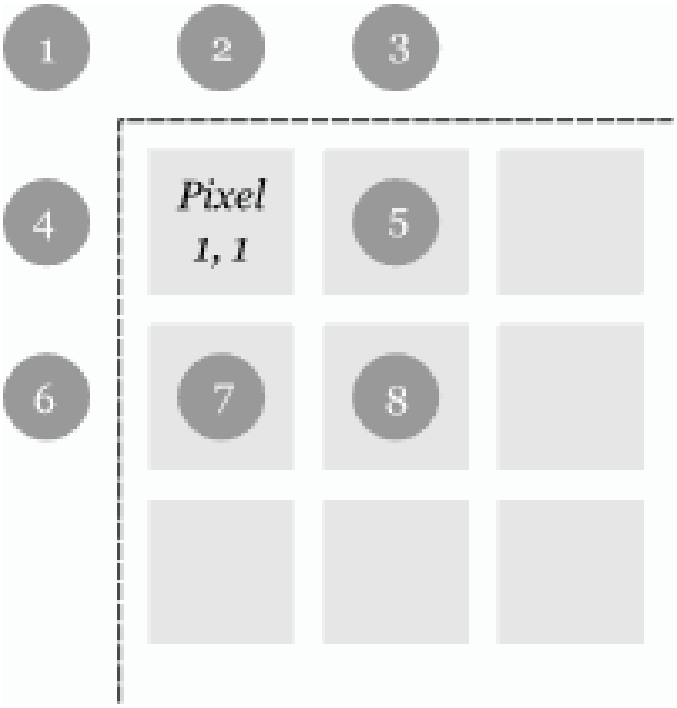

Figura 2: Efecto Borroso

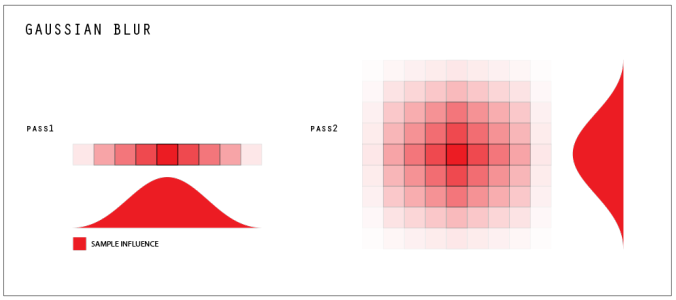

Figura 3: Efecto Borroso Gaussiano

Figura 4: División de la imagen por hilos

III. Paralelización del algoritmo

En el programa de paralelización se implementó la asignación block-wise y se tomó como referencia el largo de la imagen para lanzar la cantidad de hilos. Por ejemplo, en el caso de la imagen de 720p que tiene como dimensiones 1280x720, se lanzaron 1280 hilos. Para la imagen de 1080p se lanzaron 1920 hilos y de forma similar para la imagen de 4K se lanzaron 3840 hilos.

IV. Experimentos y resultados

Para la práctica de CUDA se utilizó un equipo de cómputo disponible por Google Colab con las siguientes características: un procesador Intel(R) Xeon(R) CPU 2.3 GHz dual core; y una GPU "Tesla T4 (40) Multiprocessors, (128) CUDA Cores/MP: 5120 CUDA Cores y GPU Max 1.59 GHz. Sobre el Notebook de Python en Colab se clonó el repositorio: https://github.com/afforeroc/blur-cuda que contiene dos scripts para compilar y ejecutar los dos programas de efecto borroso: 1) En el caso de CUDA, solo variando el tamaño el kernel y 2) en el caso de OpenMP variando el tamaño del kernel y el número de hilos. Los programas toman estos valores como argumento, paralelizan y finalmente anexan todos los tiempos de respuesta de procesamiento al documento *.csv correspondiente al tipo de imagen. Finalmente, los valores se manipularon en hojas de cálculo para obtener las gráficas de tiempo de respuesta y de speedup.

V. Conclusiones

- Hay una convergencia en tiempo de respuesta para CUDA, donde se lanza el máximo número de hilos.
- En las gráficas de Speedup vs Kernel se evidencia una mejora en el uso de la GPU en comparación a los hilos que puede lanzar en CPU con OpenMP.
- Se evidencia la estabilización del Speedup en el tamaño de kernel 13 para la imagen de 4K.
- Para las imágenes de 1080p y 720p aún no se logra alguna estabilización de speedup sobre algún tamaño de kernel en el uso de CUDA.

VI. Bibliografía

- https://www.w3.org/Talks/2012/0125-HTML-Tehran/Gaussian.xhtml
- http://www.jhlabs.com/ip/blurring.html
- https://bit.ly/2Hror6D
- https://intel.ly/2VOCM5K

VII. Anexos

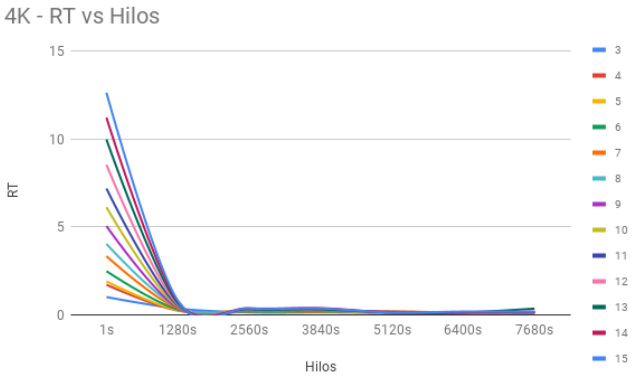

Figura 5: CUDA - 4K - RT vs Hilos

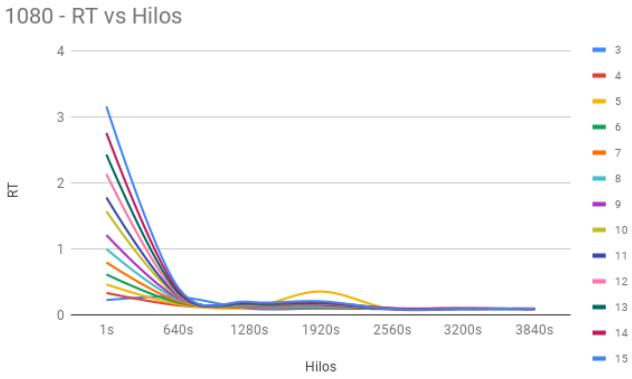

Figura 6: CUDA - 1080p - RT vs Hilos

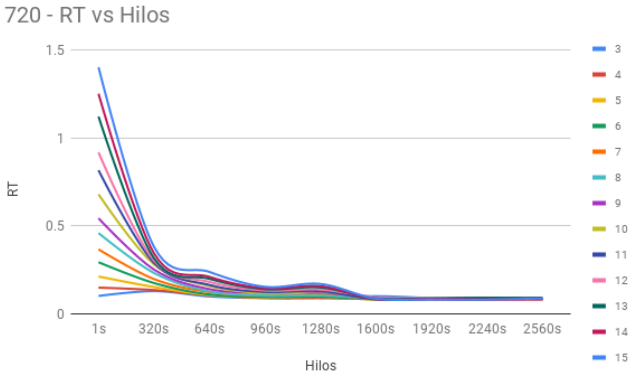

Figura 7: CUDA - 720p - RT vs Hilos

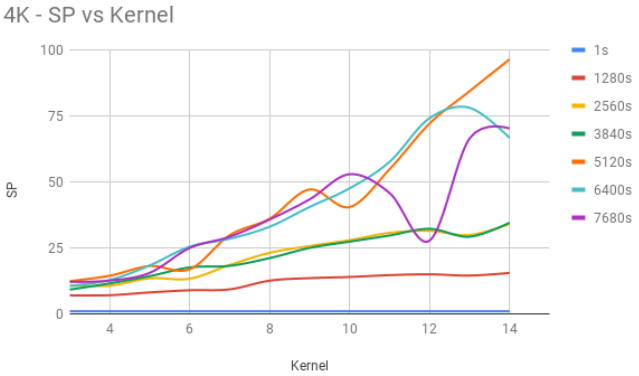

Figura 8: CUDA - 4K - RT vs Kernel

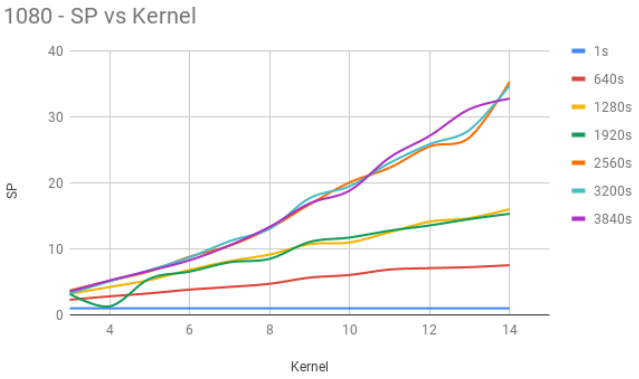

Figura 9: CUDA - 1080p - RT vs Kernel

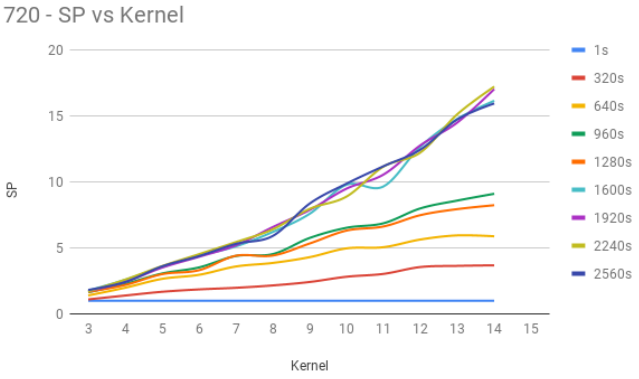

Figura 10: CUDA - 720p - RT vs Kernel

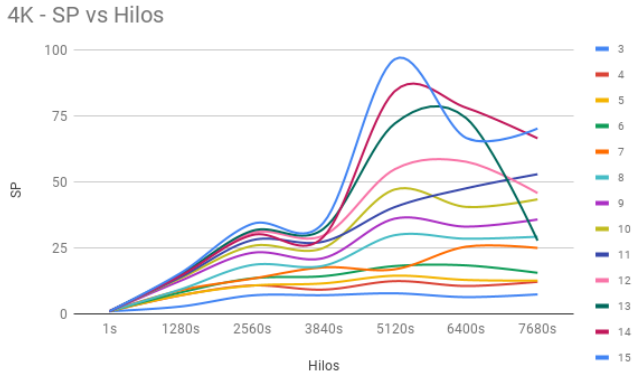

Figura 11: CUDA - 4K - SP vs Hilos

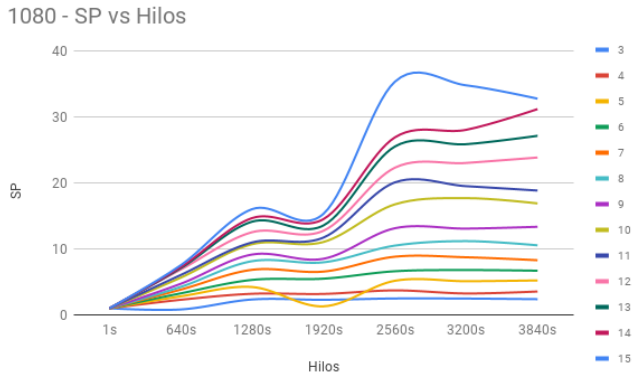

Figura 12: CUDA - 1080p - SP vs Hilos

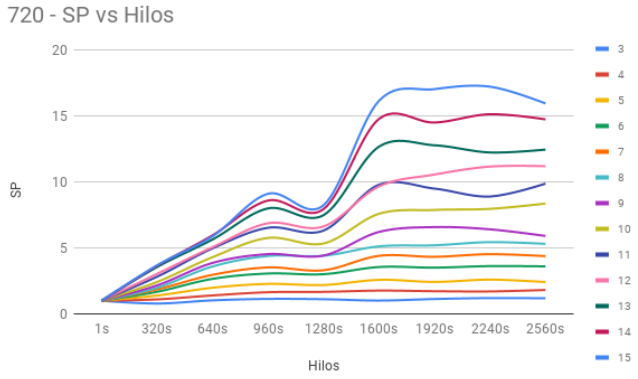

Figura 13: CUDA - 720p - SP vs Hilos