INF-351: Computación de Alto Desempeño
Laboratorio 1
Desarmando... el rompecabezas?

Prof. Álvaro Salinas

02 de Mayo de 2020

1. Descripción y Marco Teórico

En el siguiente laboratorio, usted realizará un análisis del paralelismo ofrecido por CUDA al resolver un determinado problema. Para lograrlo, deberá implementar CUDA kernels que permitan ejecutar un procesamiento paralelo sobre un conjunto de datos, para posteriormente comparar su desempeño con una versión secuencial que se ejecute en la CPU. Los objetivos principales de esta experiencia consisten en evaluar sus conocimientos sobre los fundamentos básicos de la programación en CUDA, tales como la elaboración de kernels, manejos de memoria, distribución de trabajo en CUDA threads, indexación y mediciones de tiempo.

Rompecabezas

En este laboratorio vamos a continuar con el trabajo sobre imágenes que comenzamos en las clases prácticas. En esta ocasión, vamos a considerar que la imagen a procesar corresponde a un rompecabezas que debemos desarmar según un orden de piezas específico que recibirá nuestro algoritmo. Es importante tener en cuenta las siguientes consideraciones:

- La imagen siempre será una imagen cuadrada de tamaño $N \times N$ píxeles.
- Las piezas del rompecabezas siempre serán cuadradas de tamaño $S \times S$ píxeles. S siempre será divisor de N, así evitamos que hayan piezas incompletas.
- Considerando los dos puntos anteriores, el rompecabezas estará conformado por $P \times P$ piezas, donde $P = N/S$.

Llamaremos "orden" a un conjunto de P^2 números enteros i consecutivos, donde $0 \leq i \leq P^2 - 1$. Cada valor de i corresponde a un identificador de una determinada pieza. El conjunto de valores considera la concatenación de las filas de piezas.

El orden del rompecabezas armado (imagen de entrada) corresponde al conjunto ordenado de forma ascendente, es decir, $\langle 0, 1, 2 \ldots, P^2 - 2, P^2 - 1 \rangle$.

El rompecabezas debe ser desarmado siguiendo un nuevo orden que le será especificado al algoritmo. Este nuevo orden corresponderá al mismo conjunto de números, pero desordenado. El objetivo es indicar en qué posición debe quedar cada pieza de acuerdo a dónde se encuentra su identificador i dentro de este conjunto.

Dado que la explicación anterior pudo haber sido confusa, analicemos el siguiente ejemplo para comprender este concepto:

Consideremos $N = 90$, $S = 30$ y $P = 3$, es decir, una imagen de 90×90 píxeles dividida en 9 piezas de 30×30 píxeles, cada una asociada a un identificador entre 0 y 8. A continuación, la primera imagen a la izquierda, muestra el orden del rompecabezas ordenado, es decir, la imagen de entrada para nuestro algoritmo. Las dos imágenes restantes corresponden a posibles imágenes de salida de acuerdo al orden especificado.

0	1	2
3	4	5
6	7	8

Orden: $\langle 0, 1, 2, 3, 4, 5, 6, 7, 8 \rangle$

3	6	1
5	2	8
7	4	0

Orden: $\langle 3, 6, 1, 5, 2, 8, 7, 4, 0 \rangle$

8	4	0
6	7	1
2	3	5

Orden: $\langle 8, 4, 0, 6, 7, 1, 2, 3, 5 \rangle$

El algoritmo a desarrollar recibirá una imagen de entrada y un orden para la imagen de salida. Un ejemplo ilustrativo de lo que debe realizar nuestro código se muestra a continuación:

Imagen de entrada $\longrightarrow$ Imagen de salida

La imagen del ejemplo corresponde a una imagen de 400×400 píxeles dividida en 16 piezas de 100×100 píxeles. El orden especificado al algoritmo que permitió generar la imagen de salida observada fue $\langle 1, 2, 5, 3, 14, 7, 8, 11, 9, 13, 12, 10, 0, 15, 6, 4 \rangle$.

2. Archivo de Entrada

El archivo de texto que su programa recibirá tiene el siguiente formato:

- La primera línea contiene 2 enteros correspondientes a los valores de N y S.
- La segunda línea contiene P^2 enteros correspondientes al orden especificado para generar la imagen resultante.
- Las líneas 3, 4 y 5 contienen cada una $N \times N$ valores flotantes entre 0 y 1 correspondientes a los canales R, G y B de la imagen respectivamente. Estos valores consideran la concatenación de filas de píxeles de la imagen de entrada. Estas tres líneas son equivalentes a las líneas 2, 3 y 4 de los archivos utilizados en las clases prácticas.

Junto a este enunciado, encontrará un script en lenguaje Python que le permitirá generar los archivos necesarios a partir de una imagen. Dicho script genera automáticamente 5 imágenes (junto a los archivos de texto correspondientes) de tamaño 100×100, 200×200, 400×400, 800×800 y 1600×1600. El nombre de los archivos generados es `img<N>x<N>.txt`, donde `<N>` es reemplazado por el valor de N correspondiente, por ejemplo: `img100x100.txt`.

3. Archivos de Salida

Se le recomienda utilizar el formato visto en las clases prácticas del curso. La única diferencia respecto a dicho formato es que, ahora que trabajamos con imágenes cuadradas, la primera línea del archivo de salida solo contendrá un único entero con el valor de N (en vez de M y N). Gracias a esto, puede reutilizar la función `Write` presente en el código incompleto que le fue compartido para la actividad, teniendo que realizar mínimas modificaciones.

El mismo script que genera los archivos de entrada es capaz de generar las imágenes resultantes a partir de los archivos de salida. Si desea utilizar la celda de ejemplo para generar dichas imágenes, procure que sus archivos de salida sean nombrados `img<N>x<N>CPU.txt` y `img<N>x<N>GPU.txt`, donde `<N>` nuevamente debe ser reemplazado por el valor de N correspondiente, por ejemplo: `img100x100CPU.txt` y `img100x100GPU.txt`. Debe tener los 10 archivos generados para poder utilizar dicha celda del script. Estos 10 archivos corresponden a 2 archivos (resultados de CPU y GPU) para cada uno de los 5 tamaños generados como archivos de entrada.

4. Desarrollo

1. Como primeros pasos, cree una función de C/C++ que permita leer los archivos de entrada. También, en el main, genere los arreglos necesarios, reserve la memoria y copie la información necesaria a GPU.

2. Implemente una función secuencial de C/C++ que genere la imagen resultante a partir de la imagen de entrada y el orden especificado.

3. Implemente un CUDA kernel que realice el mismo procedimiento que la función creada en el punto anterior. Preocúpese de seguir buenas prácticas y tener una correcta indexación.

   [Pregunta] Antes de ejecutar su código, ¿cuál de las dos implementaciones cree usted que presentará un mejor desempeño? ¿Depende de algo? Argumente su respuesta.

4. Ejecute su solución para cada archivo de entrada y mida los tiempos de cada caso (recuerde medir el tiempo utilizando las funciones adecuadas para cada caso).

   [Informe] Presente en su informe la imagen de entrada que utilizó (un solo tamaño de ésta es suficiente) y las 5 imágenes resultantes obtenidas (usted debe asegurarse de que ambas implementaciones, CPU y GPU, entreguen el mismo resultado, pero no necesita mostrar estos duplicados en el informe).

   [Informe] Realice un gráfico o tabla en donde se muestren los 10 tiempos obtenidos (cada una de las 2 implementaciones para cada uno de los 5 archivos de entrada). Si opta por realizar un gráfico, obtendrá 2 curvas distintas (una por cada implementación), conteniendo el eje x los valores de N, mientras que el eje y corresponderá a los tiempos medidos. *Hint: considere utilizar una escala logarítmica si las curvas no se aprecian correctamente.*

   [Pregunta] Analice los tiempos obtenidos. ¿Qué puede concluir? Comente sobre el comportamiento observado y relaciónelo al enfoque de paralelismo (trabajo de cada hebra) que utilizó.

   [Pregunta] ¿Se le ocurre otra manera de diseñar su implementación? De ser afirmativa su respuesta, explique brevemente (no necesita implementarlo). ¿Cree usted que sería mejor que la solución que diseño inicialmente? Explique las ventajas y desventajas. *Hint: analice las posibilidades de un trade-off entre el uso de memoria y el tiempo de ejecución.*

5. Reglas y Consideraciones

Entrega

- La entrega debe realizarse en un archivo de nombre Lab1-X.tar.gz (formatos rar y zip también son aceptados), donde X debe ser reemplazado por el número de su grupo. Diríjase a la inscripción de grupos para consultar su número.
- El archivo de entrega debe contener un informe en formato pdf junto con el código implementado para resolver el laboratorio. Se le ruega entregar un código ordenado.
- El informe debe contener:
  - Título y número del laboratorio.
  - Nombre y rol de todos los integrantes del grupo.
  - Modelo y compute capability de la tarjeta gráfica que fue utilizada para ejecutar el código. Si se probó con más de una tarjeta gráfica, incluya los datos de todas y específique qué tarjeta se utilizó en cada pregunta y resultado reportado.
  - Desarrollo. Preocúpese de incluir cada ítem señalado con los tag **[Pregunta]** e **[Informe]** en el enunciado.
  - Conclusiones. Incluya aprendizajes, comentarios, observaciones o supuestos que surgieron durante el desarrollo del laboratorio.
- El descuento por día de retraso es de 30 puntos, con un máximo de 1 día de retraso. No se aceptarán entregas posteriores.
- En caso de copia, los grupos involucrados serán evaluados con nota 0. No hay problema en generar discusión y compartir ideas de implementación con sus compañeros, pero códigos copiados y pegados no serán aceptados.
- El incumplimiento de estas reglas implica descuentos en su evaluación.
- La fecha de entrega es el día Martes 12 de Mayo. Se habilitará la opción de entrega en Aula.

Consideraciones

- Trabaje con flotantes de precisión simple (`float`).
- Para mediciones de tiempo, utilice `clock()` de la librería time.h en caso de mediciones en CPU y `cudaEvent()` para códigos de GPU. Trabaje con milisegundos $[ms]$.
- En caso de optar por la realización de gráficos, puede optar por la herramienta que más le acomode. La librería matplotlib de Python siempre es una buena opción.
- Utilice 256 hebras por bloque en su implementacion.