INF-351: Computación de Alto Desempeño
Laboratorio 2
Decodificando Imágenes

Prof. Álvaro Salinas

27 de Mayo de 2021

1. Descripción y Marco Teórico

En el siguiente laboratorio serán evaluados sus conocimientos sobre uso de memoria global y accesos de memoria coalescentes.

Imágenes Codificadas

Hemos diseñado una forma de codificar el contenido de una imagen. Dado un número entero P, podemos representar cada píxel de una imagen mediante P valores p_i, con $1 \leq i \leq P$, de tal manera que los canales R, G y B de dicho píxel se calcules según:

$$R = \sum_{i=1}^{P} r_i p_i \qquad G = \sum_{i=1}^{P} g_i p_i \qquad B = \sum_{i=1}^{P} b_i p_i$$

donde los valores r_i, g_i y b_i son valores constantes utilizados para el cálculo de todos los píxeles.

Archivo de Entrada

Junto al enunciado de este laboratorio podrán encontrar el archivo `img.txt` que corresponde al input del programa a realizar. El formato de este archivo es:

- Primera línea: 3 valores enteros M, N y P, donde $M \times N$ corresponde al tamaño de la imagen y P es la cantidad de valores utilizados en la codificación según se explicó anteriormente.
- Segunda línea: P valores enteros correspondientes a las constantes r_i.
- Tercera línea: P valores enteros correspondientes a las constantes g_i.
- Cuarta línea: P valores enteros correspondientes a las constantes b_i.
- Siguientes $M \times N$ líneas: Una línea por píxel, cada una conteniendo P flotantes correspondientes a los valores p_i.

Los píxeles están ordenados en el archivo según filas concatenadas, es decir, los primeros N píxeles son la primera fila de la imagen, los siguientes N píxeles corresponden a la segunda fila, y así sucesivamente. De esta forma, al generar un archivo de salida con este mismo orden, podrán utilizar el mismo script de python utilizado para las clases prácticas con el fin de visualizar el resultado final.

Considere el archivo `img.txt` como solamente un ejemplo de entrada. Su código debería funcionar para cualquier valor de P.

2. Desarrollo

a) Desarrolle una función de CPU que se encargue de generar los canales R, G y B de la imagen resultante a partir de los datos de entrada. Mida el tiempo que demora en procesar la imagen correspondiente al archivo de ejemplo entregado con la tarea.

b) Implemente un CUDA kernel que resuelva el mismo problema utilizando un ordenamiento de datos correspondiente a AoS (array of structures). Recuerde que esto implica tanto organizar los datos en el arreglo de entrada como manejar de una determinada forma los índices para acceder a los datos. Mida el tiempo que demora la ejecución de su kernel para la imagen de ejemplo.

c) Desarrolle un CUDA kernel que decodifique la imagen, pero esta vez utilizando SoA (structure of arrays). Nuevamente, mida el tiempo que demora la ejecución para la imagen de ejemplo.

[Informe] Incluya en su informe la imagen resultante que obtuvo. No es necesario que presente las 3 imágenes idénticas obtenidas con cada implementación. Basta con que usted se cerciore de que todos los códigos estén funcionando correctamente.

[Informe] Incluya en su informe una tabla o gráfico con los tiempos obtenidos para cada implementación.

[Pregunta] ¿Cuál resultó ser la implementación más eficiente? ¿Qué puede concluir sobre lo observado? Comente al respecto y argumente porqué cree usted que el código más eficiente tuvo un rendimiento superior.

[Pregunta] ¿Qué cree usted que ocurriría con sus implementaciones si $P \geq 32$? Argumente su respuesta.

3. Reglas y Consideraciones

Entrega

- La entrega debe realizarse en un archivo de nombre Lab2-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 resolver cada ítem señalado con el tag **[Informe]** o **[Pregunta]** en el enunciado.
  - Conclusiones. Incluya 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 no cumplimiento de estas reglas implica descuentos en su evaluación.
- La fecha de entrega es el día Lunes 7 de Junio. 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 sus implementaciones.