

INF-351: Computación de Alto Desempeño
Laboratorio 2
Intercalado Coalescente de Columnas

Prof. Álvaro Salinas

21 de Abril de 2019

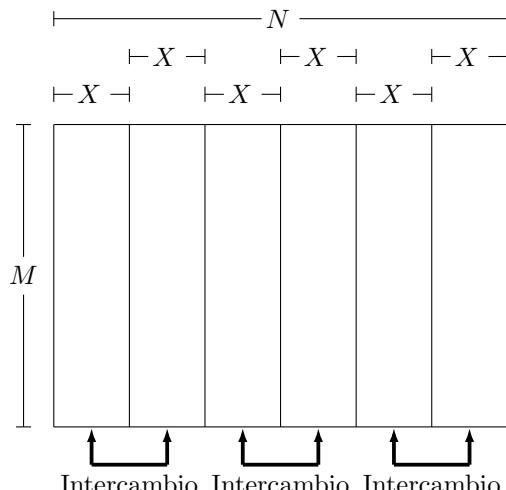
1. Descripción y Marco Teórico

En el siguiente laboratorio serán evaluados sus conocimientos sobre divergencia, uso de memoria global y accesos de memoria coalescentes. Adicionalmente, usted deberá demostrar un correcto manejo de la herramienta NVIDIA Visual Profiler.

Intercalado de Columnas en Imágenes

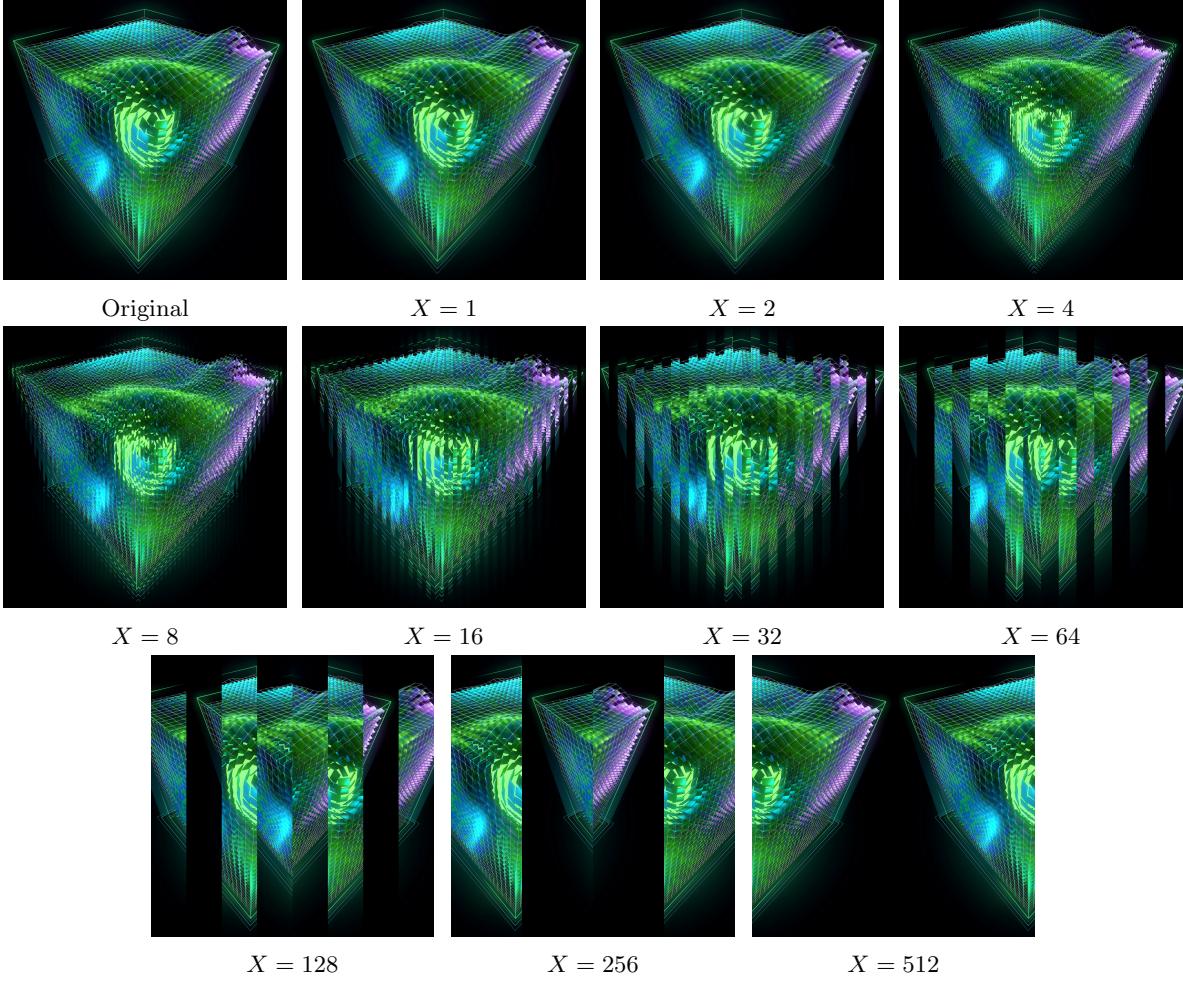
¡Así es! Seguiremos trabajando con imágenes...

En esta ocasión, intercalaremos subimágenes en forma de columnas para obtener una imagen resultante. Consideremos una imagen de $M \times N$ píxeles y X el ancho de una subimagen con $2X|N$, entonces tomaremos la subimagen compuesta por los $M \times X$ píxeles de las primeras X columnas, e intercambiaremos su posición con la subimagen compuesta por los $M \times X$ píxeles de las columnas de la X a la $2X - 1$. También intercambiaremos la posición de la subimagen entre las columnas $2X$ y $3X - 1$ con la subimagen entre las columnas $3X$ y $4X - 1$. Así continuaremos intercambiando las posiciones de todas las subimágenes en posiciones pares con la subimagen siguiente que corresponda.



Intercalado de subimágenes columnares.

Veamos un ejemplo práctico y el resultado que deben obtener en este laboratorio. Consideremos una imagen de tamaño 1016×1024 ($M = 1016$ filas de $N = 1024$ píxeles cada una). Entonces, para los siguientes valores de X , tendriamos las siguientes imágenes resultantes:



Archivos de Entrada

Esta vez recibirá un único archivo con el mismo formato usado en la clase práctica 1, es decir:

- La primera línea contiene dos enteros M y N , en dicho orden. M corresponde al número de filas de píxeles en la imagen y N el número de columnas, es decir, el largo de cada una de las M filas.
- Las tres filas restantes corresponden cada una a los canales R, G y B de la imagen. En cada línea se encuentran los $M \times N$ valores correspondientes a uno de los canales para todos los píxeles de la imagen. La imagen se encuentra concatenada por filas, es decir, primero están los N valores de la primera fila de píxeles, luego los N valores de la segunda fila, y así sucesivamente.

Se recomienda utilizar la imagen entregada junto con el enunciado, pues se ha escalado para que tenga un valor de N exactamente igual a 1024 (es la imagen del ejemplo anterior). De esta forma, utilizando bloques de 256 hebras, aseguramos que se necesiten 4 bloques por fila de píxeles. Adicionalmente, esto permite que los valores de X a utilizar multiplicados por 2 sean divisores exactos de N .

Archivos de Salida

Se le recomienda utilizar el formato visto en la primera clase práctica del curso. De esta forma, no debe realizar un nuevo código para la escritura del archivo ni tiene que crear una nueva herramienta para visualizar el resultado, pues puede utilizar la función `TXTtoRGB` presente en el script que se le compartió.

2. Desarrollo

1. Implemente un algoritmo de CPU que genere las 10 imágenes resultantes mostradas en el ejemplo a partir de la imagen de entrada. Esto es para los valores $X = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512$. Mida los tiempos de cada una de sus ejecuciones.

2. Sin modificar el orden de los píxeles (almacenando en un array las filas concatenadas), implemente un CUDA kernel que genere las mismas 10 imágenes resultantes que en el paso anterior. Preocúpese de tener accesos de memoria coalescentes, es decir, que las 32 hebras de un warp lean y escriban en la misma palabra de 128 bytes.

[Pregunta] ¿Encuentra algún problema en su implementación? De ser su respuesta afirmativa, antes de continuar, explique como se le ocurriría solucionarlo.

3. Manteniendo el orden otorgado por la concatenación de filas, implemente un CUDA kernel en donde las hebras de los primeros 2 bloques (de los 4 que procesan la fila completa) lean solamente los píxeles de las subimágenes que se moverán a la derecha (aquellas en posiciones pares comenzando desde 0). Por lo tanto, las hebras pertenecientes a los dos bloques restantes deberán trabajar con los píxeles que se moverán hacia la izquierda, es decir, las subimágenes impares. Vuelva a ejecutar su solución para todos los valores de X indicados anteriormente.

[Pregunta] Solo si su respuesta a la [Pregunta] anterior fue afirmativa, indique si esta implementación solucionó el error encontrado.

[Pregunta] ¿Cuál es el problema de esta implementación? ¿Ocurre en todos los casos? Indique qué condiciones se tienen que dar para que se presente este problema de optimización.

4. En esta ocasión, tendrá total libertad para modificar la forma de leer y escribir los archivos, así como la manera de almacenar la información en los arreglos. Manteniendo el hecho de que los dos primeros bloques procesen las subimágenes que se moverán a la derecha y los dos últimos las que se moverán a la izquierda, implemente un CUDA kernel que logre que hebras con índice consecutivo accedan a direcciones de memoria consecutivas para todos los valores de X .

[Pregunta] ¿Se arregló el problema que presentaba el caso anterior? ¿A qué se debe esto?

[Informe] Presente en su informe las 10 imágenes resultantes obtenidas (usted debe asegurarse de que las cuatro implementaciones entreguen el mismo resultado, pero no necesita mostrar imágenes duplicadas en el informe).

[Informe] Utilice la herramienta Visual Profiler para medir el rendimiento de las 3 versiones de GPU implementadas. Realice un gráfico o tabla en donde se muestren todos los tiempos obtenidos (cada una de las 4 implementaciones para cada uno de los 10 valores de X). Si opta por realizar un gráfico, obtendrá 4 curvas distintas (una por cada implementación), conteniendo el eje x los valores de X , mientras que el eje y corresponderá a los tiempos medidos. Hint: considere utilizar una escala logarítmica si las curvas no se aprecian correctamente.

[Informe] Utilizando nuevamente la herramienta Visual Profiler, compare el Bandwidth alcanzado en sus 3 implementaciones de GPU (esto lo puede encontrar en Perform Kernel Analysis > Perform Memory Bandwidth Analysis > Device Memory > Total).

[Pregunta] Analice los datos obtenidos para tiempos y uso de Bandwidth. ¿Qué puede concluir? ¿Se observa realmente una diferencia significativa? ¿A qué cree usted que se debe esto? Comente al respecto.

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 especifique 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 no cumplimiento de estas reglas implica descuentos en su evaluación.
- La fecha de entrega es el día Lunes 6 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 la herramienta Visual Profiler para códigos de GPU. Trabaje y presente resultados en milisegundos [`ms`].
- Al utilizar Visual Profiler recuerde que para obtener tiempos debe ejecutar el comando:
`nvprof -o archivo_salida ejecutable`
mientras que para realizar un análisis en profundidad de cada kernel se debe utilizar:
`nvprof --analysis-metrics -o archivo_salida ejecutable`
- 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.