Laboratorio 3

Melisa Bonias

Parte 1: Procesamiento de Imágenes

Introducción

Este problema consiste en realizar unas tareas de propósito general sobre una imagen dada. En particular se pide poder espejar verticalmente, horizontalmente y rotar en un ángulo específico la imagen, aprovechando efectivamente la concurrencia al trabajar cada pixel independientemente.

Respecto de las decisiones adoptadas, en consideración de que ciertas variables varían en función de la tarea a realizar (ya sea un espejado o una rotación) y de que cada tarea representa una instancia a los dispositivos definida separadamente del resto, se decidió escribir el programa en torno a un bucle central cuyas iteraciones representan intentos de realizar las tareas de rotación, espejado vertical y espejado horizontal.

Dado que cada tarea puede o bien ser requerida o no, y hay en total tres tareas independientes, existen en total 8 posibles inicializaciones del programa. Cuál es la inicialización requerida, se sabe desde el comienzo de la ejecución, ya que esta información es pasada mediante los argumentos del programa. En caso de que la tarea de rotación sea requerida, tres parámetros extra (ángulo, y centro de rotación) son pedidos al usuario mediante stdin.

En cada iteración del bucle principal, se pide memoria y se crean estructuras de OpenCL particulares a la tarea elegida, como un Programa y un Kernel. Estas estructuras son liberadas al finalizar cada iteración.

Resta aclarar que dada la naturaleza bidimensional de las imágenes, las dimensiones del espacio indexado es de 2, y cada work-item cuenta con dos variables de identificador del tipo (xi, yi).

Parte 2: Simulación Física

Este problema consiste en realizar una simulación física de la propagación del calor a través de una placa cuadrada uniforme.

Al igual que en el problema de la parte 1, el problema puede modelarse a través de una matriz, contando con un espacio indexado en dos dimensiones y cada work-item trabajando en un elemento particular de la matriz resultante. En este caso particular, la tarea de cada work-item es calcular la temperatura resultante de una fuente de calor.

Dado que no hay garantía de que la velocidad de todos los work-items sea equivalente, resulta importante asegurar como precondición que cada work-item cuenta, al comienzo de cada iteración, con los valores anteriores de sus vecinos ya calculados. Para esto, es necesario utilizar una barrera al final de una iteración, y de esta forma evitar que un work-item rápido comience con una iteración sin los valores de los que depende ya calculados.

Respecto de la cuestión de un diseño abstraído del tamaño de work-group del hardware, un primer enfoque podría sugerir dividir la imagen en áreas repartidas entre los work-group, y de esta forma asegurar la sincronización dentro de cada área. Sin embargo, el problema con este enfoque es no considerar la sincronización inter-área (la sincronización intra-área es la que resuelven las barreras), ya que los valores límite dentro de un área dependen de valores de otras áreas. En otras palabras, las áreas no son totalmente independiente entre ellas. Una solución a este problema es sacrificar la total concurrencia para asegurar la correcta sincronización entre áreas: Dividir la imagen en áreas lo más grandes posible. Luego utilizar un único work-group (en lugar de uno por cada área) que realice el trabajo por cada área de manera secuencial.

Conclusión final

En nuestros días saber cómo aprovechar la capacidad de cómputo del hardware, en particular de las tarjetas gráficas, es un conocimiento que muchas veces no resulta lo suficientemente difundido, a pesar de que estos dispositivos tienen un costo que pagamos completamente. Me quedo con una gran gratificación al haber aprendido una herramienta extensamente útil, que pertenece la gran rama de la computación paralela y que tiene incontables aplicaciones el cálculo numérico y en particular problemas de simulación física.