Ejercicio de paralelización con CUDA

Pablo Renero Balgañón, Antonio Román López

Índice

1. Introducción

1.1. Objeto del documento

El objeto del presente documento es mostrar los resultados obtenidos en el proceso de análisis, optimización y paralelización del código secuencial proporcionado en la asignatura con el fin de obtener el mínimo tiempo de cómputo posible.

1.2. Metodología

La metodología seguida para la realización de la práctica ha sido la siguiente:

- Análisis del código secuencial.
- Paralelización con *CUDA*
- Optimización de aspectos secuenciales del código.

1.3. Realización

Los primeros pasos que hemos realizado han sido generar un kernel para la fase de actualización, copia y la de relajación sin ninguna optimización sobre el secuencial.

Tras la realización de esta primera versión , analizando el código, que el vector layer_copy no tenía sentido mantenerlo e inicializarlo en el host. En el caso del vector layer no era necesario copiarlo desde el device al host y cargarlo de nuevo al device, simplemente con descargarlo para buscar los máximos valía.

Tras entrar al leader board pero sin superar la referencia uno nos pusimos a realizar las reducción sobre los maximos.

1.3.1. Reducción

La idea principal de la reducción era generar un vector compartido del doble del tamaño del bloque, para cargar en las posiciones pares el valor que se compara y en las impares la posición en la que se encuentra. Tras incluir en memoria shared aquellos valores entre (0-layer_size) y rellenar de ceros el resto, procedimos a recorrer los vectores como se muestra en el siguiente esquema 1.

Investigando nos dimos cuenta que si cambiabas la forma de recorrer el vector podíamos evitar algún calculo menos. Esta forma de recorrer el

vector, como se muestra en la imagen 2, nos dio lugar a una pequeña mejora temporal.

Tras las comparaciones obtenemos un vector intermedio (maximosTemp) de máximos. El hilo con id global igual a 0 busca el máximo del vector. Problema: debido a los accesos a memoria que se hacen es más lento, por lo que, decidimos sacarlo al secuencial.

2. Optimizaciónes finales

Tras todo el proceso el proceso de paralelización y con una versión ya válida nos dispusimos realizar algunas optimizaciones en el código. La más obvia de todas es quitar la copia y dependendiendo de la ejecución en la que se encuentra se cambia el orden en el que se les pasan layer_copy y layer, lo cual es más rápido que hacer el intercambio de punteros.

Tras investigar y leer un par de instigaciones sobre reducciones en cuda, nos dimos cuenta que medida que avanza la reducción el número de hilos activos disminuye. Esto provoca que cuando la iteración llega al punto que es menor o igual de 32 solo hay un warp activo, dando la posibilidad de desenrollar el bucle en las 6 últimas iteraciones sin necesidad de sincronizar los hilos.

3. Anexo I

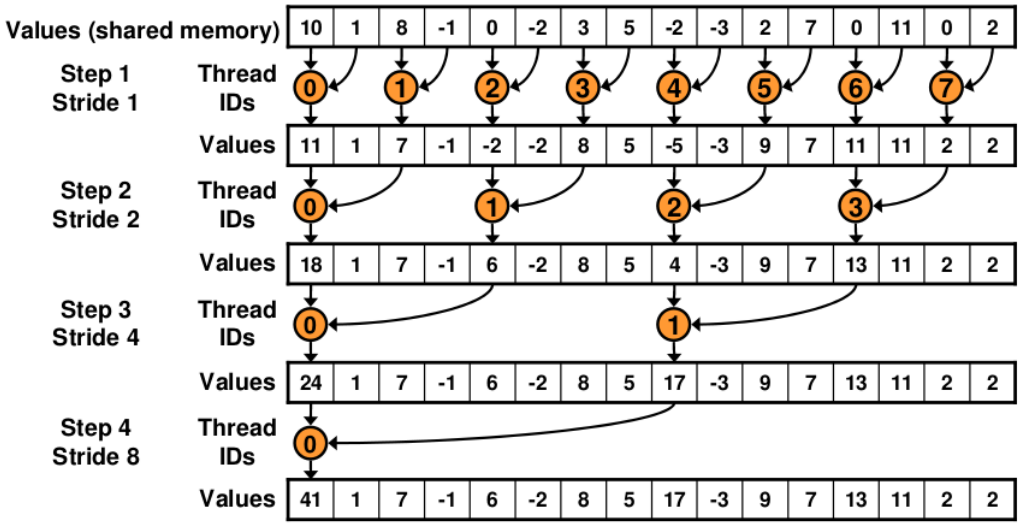

Figura 1: Esquema para recorrer el vector V1

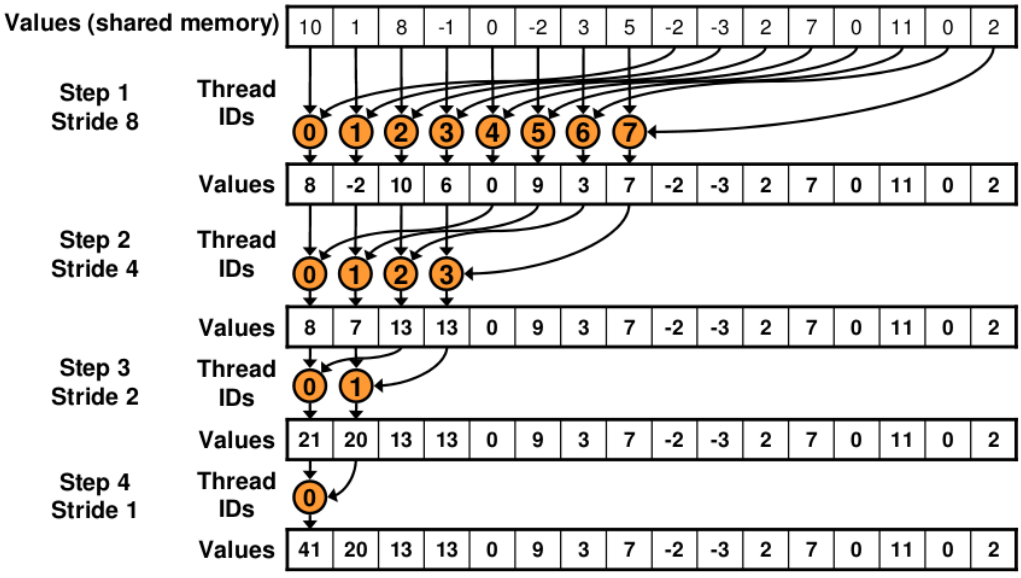

Figura 2: Esquema para recorrer el vector V2

Referencias

[1] «Optimizing Parallel Reduction in CUDA». [Online]. Disponible en http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/reduction/doc/reduction.pdf.