GPGPU – Laboratorio 3

Introducción

El objetivo de este laboratorio es comprender cómo el patrón de acceso a memoria global, así como la elección de los parámetros de la grilla puede afectar el desempeño. Uno de los conceptos clave para resolver los ejercicios es el de acceso coalesced, es decir, el mecanismo mediante el cual se juntan varios pedidos de memoria de los threads de un warp en una sola transacción. En dispositivos de la generación 6.0 en adelante, la regla para unificar los accesos es sencilla y podría escribirse de la siguiente manera: *los accesos a memoria simultáneos de un warp son agrupados en la menor cantidad de transacciones de 32B necesarias para satisfacer todos los accesos*.

Para este práctico siempre puede asumir que las dimensiones de la matriz son múltiplos del tamaño de bloque.

Los tiempos de ejecución deben tomarse como el promedio de 10 ejecuciones (presentando únicamente el promedio y desviación estándar en el informe). Para obtener este dato fácilmente es conveniente utilizar la herramienta NsightSystems de la siguiente forma:

```
sbatch launch_single.sh nsys profile --stats true ./programa
```

Ejercicio 1

Construir un kernel que reciba una matriz de enteros alojada en memoria global y devuelva la matriz transpuesta. Reserve dos espacios de memoria distintos para las matrices de entrada y salida. El kernel debe ser sencillo y no debe utilizar la memoria compartida (todas las lecturas y escrituras deben realizarse en memoria global).
La grilla debe ser bidimensional y los bloques también deben ser bidimensionales. Esto permite usar directamente las variables threadIdx, blockIdx y blockDim para acceder a los distintos elementos, obteniendo un código más simple.

a) Ejecute el kernel con un tamaño de bloque de 32x32 y analice el patrón de acceso a memoria global que se da en las lecturas y escrituras. Mida el tiempo de ejecución del kernel.

b) Modifique el tamaño de bloque para favorecer el acceso coalesced en la escritura. Justifique adecuadamente la elección de tamaño de bloque. Ejecute el kernel nuevamente y compare el tiempo de ejecución con el caso anterior.

Ejercicio 2

a) Construya un kernel que reciba una matriz de enteros alojada en memoria global y a cada elemento (x,y) de la matriz le sume el elemento que se encuentra en la posición (x+4,y). Elija un tamaño de bloque adecuado y ejecute el kernel, registrando el tiempo de ejecución.

b) Analice el patrón de acceso a memoria de la parte anterior. Modifique el kernel y los parámetros de la grilla para que, aún reduciendo el paralelismo, se mitigue el efecto del acceso desalineado a la memoria global.

Ejercicio 3

a) Construya un kernel sencillo que reciba una matriz **A** y un vector **v** de enteros alojados en memoria global, y devuelva el vector **x=Av**[1]. La matriz **A** es de 10240 filas x 256 columnas. Ejecute con un tamaño de bloque adecuado y mida el tiempo de ejecución.

b) Analice el patrón de acceso a memoria de la parte anterior y proponga al menos una optimización. Ejecute el kernel y compare el desempeño con la versión base.

Entregar

El código correspondiente a cada parte y un informe en formato pdf con el análisis de los resultados experimentales.

---

1 $x(i)= \Sigma_j(A(i,j)*v(j))$