PRACTICA 4

José Teodosio Lorente Vallecillos

Tarjeta Grafica GTX 1650 DDR6

Ejercicio 1:

*Effective Bandwidth (GB/s):*192.0 GB/s
GFLOP/s Effective=2,849 TFLOP/s

TEORICOS

*Effective Bandwidth (GB/s):*192.0 GB/s

PRACTICOS

*Effective Bandwidth (GB/s):*176.0 GB/s
GFLOP/s Effective=29,56 GFLOP/s

Como se puede observar el Bandwidth de la tarjeta grafica de las especificaciones del fabricante coincide con el calculado teóricamente, a su vez que el calculo practico no dista demasiado de los números teóricos; sin embargo, en los GFLOP/s efectivos el fabricante nos indica un valor de 2,849 TFLOP/s y el numero obtenido en los prácticos dista mucho del mismo siendo 29,56 GFLOP/s

Ejercicio 2:

Para la realización de este ejercicio he usado una función encontrar máximo la cual lo calcula en la GPU, haciendo uso del vector de numero aleatorios creado, del máximo a devolver, y de un mutex, estrictamente necesario para la obtención del máximo ya que sin el no se podría garantizar la exclusión mutua en la sentencia atómica de la asignación del nuevo máximo a la variable global max; también haciendo uso como apoyo de stride y offset que nos permiten hacer que si el tamaño de bloque fuese 100 y el de hilo 10, con estos el primer thread se encargará de la primera entrada del vector global de la 101, de la 201 en adelante, lo que lo hace mas flexible y escalable. Y también uso una reducción del vector para que al finalizar el mayor valor de entre los 512 inicializados como ejemplo, se encuentre en la primera posición de la cache

Ejercicio 3:

En este ejercicio el planteamiento ha sido que dadas dos matrices una A de dimensiones A*B pasadas por parámetro y otra B de dimensiones C*D pasadas a su vez por parámetro, siendo B y C iguales para así cumplir la compatibilidad matemática; haciendo uso de los identificadores de bloque y hebra y también del tamaño de bloque, tanto de x como de y para así simular la profundidad de la matriz ya que en CUDA en el estándar sobre el que se desarrolla necesita saber el número de columnas antes de compilar el programa. Por lo tanto, es imposible cambiarlo o establecerlo en medio del código.
Sin embargo, con un poco de reflexión llegamos a la conclusión que esto no es un gran problema. No podemos usar la cómoda notación A[i][j], pero no nos costará tanto porque podemos indexar filas y columnas correctamente, usando lo previamente descrito los identificadores de hebras y bloque y su tamaño.
Y la forma más fácil de linealizar una matriz 2D es apilar cada fila a lo largo, desde la primera hasta la última, siendo lo que he realizado.
Y con eso se realiza la multiplicación de cada fila de la matriz A por la columna de la matriz B y su correspondiente sumatoria en función de su posición.