Universidad de la República
Facultad de Ingeniería, INCO
Montevideo, Uruguay.

Práctico 3:

GPGPU

Grupo 10

Integrantes:

Diego Amorena - 5.011.502-7 - diegoamorenag@gmail.com

Federico Gil - 5.198.750-6 - federico.gil@fing.edu.uy

Ejercicio 1

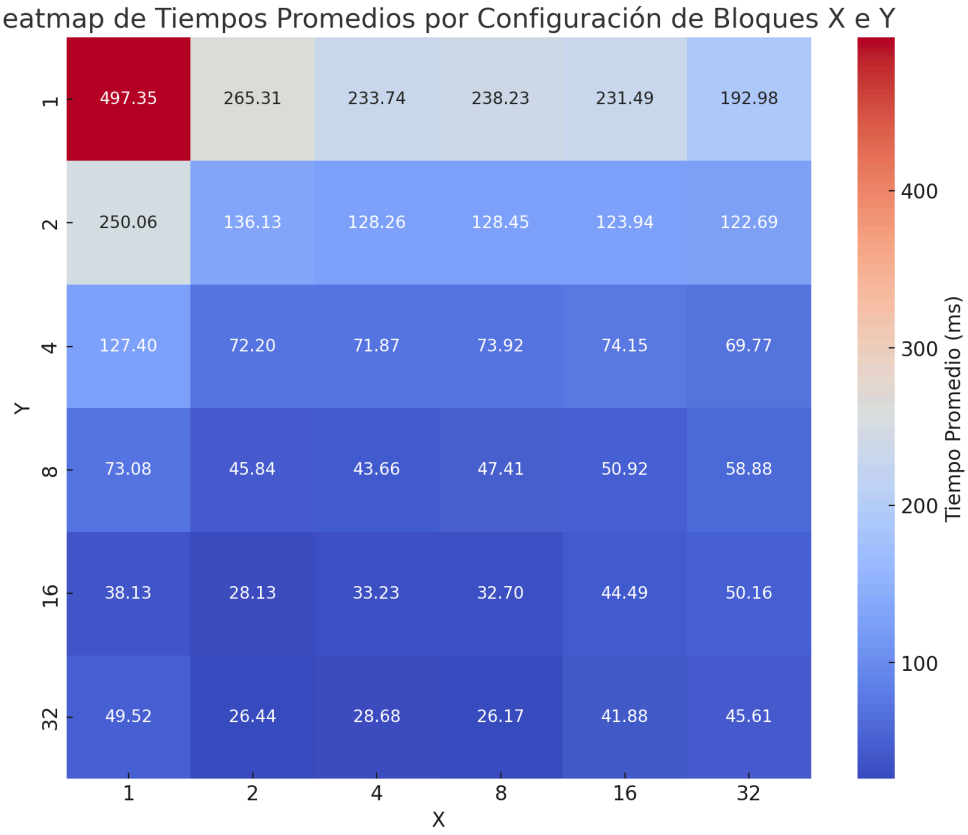
Para las partes A y B del ejercicio, se realizaron pruebas utilizando matrices de dimensiones 16k x 16k. Los valores asignados a cada celda de las matrices se determinaron mediante la fórmula (x + 16k × y), x e y representan las coordenadas de fila y columna de la celda respectivamente.

Parte a

En la parte a, se probó transponer la matriz a partir de bloques de 32 x 32, el tiempo promedio en 10 ejecuciones fue de 46.04 ms, con variación estándar de 1.06 ms. Puesto que se están usando bloques de 32x32 sucede que las lecturas son alineadas, si se supone almacenamiento por filas, y las escrituras desalineadas.

Parte b

Para la segunda parte del ejercicio, se probó todo par de (i,j) con i y j <=32, ambos siendo potencia de dos. El siguiente gráfico muestra para todas las combinaciones de tamaños de bloque los promedios de tiempo obtenidos:



Se observó que los bloques altos y angostos resultaron más eficientes en comparación con otros tipos de bloques. Esto se atribuye a que, durante la trasposición de la matriz, bloques

de estas características tienden a escribir en posiciones de memoria cercanas entre sí, optimizando así el proceso de acceso y escritura en memoria, lo que mejora significativamente el rendimiento.

Como en cada acceso se traen 128 Bytes, donde entran 32 enteros, se puede decir que la dimensión en Y debe ser 32 para aprovechar esto, mientras que la dim en X se observa empíricamente que el mejor valor es 2. Con lo que se elige el tamaño 2x32 el cual tiene una media de: 26.4357 ms y un desvío de 0.1789.

Ejercicio 3

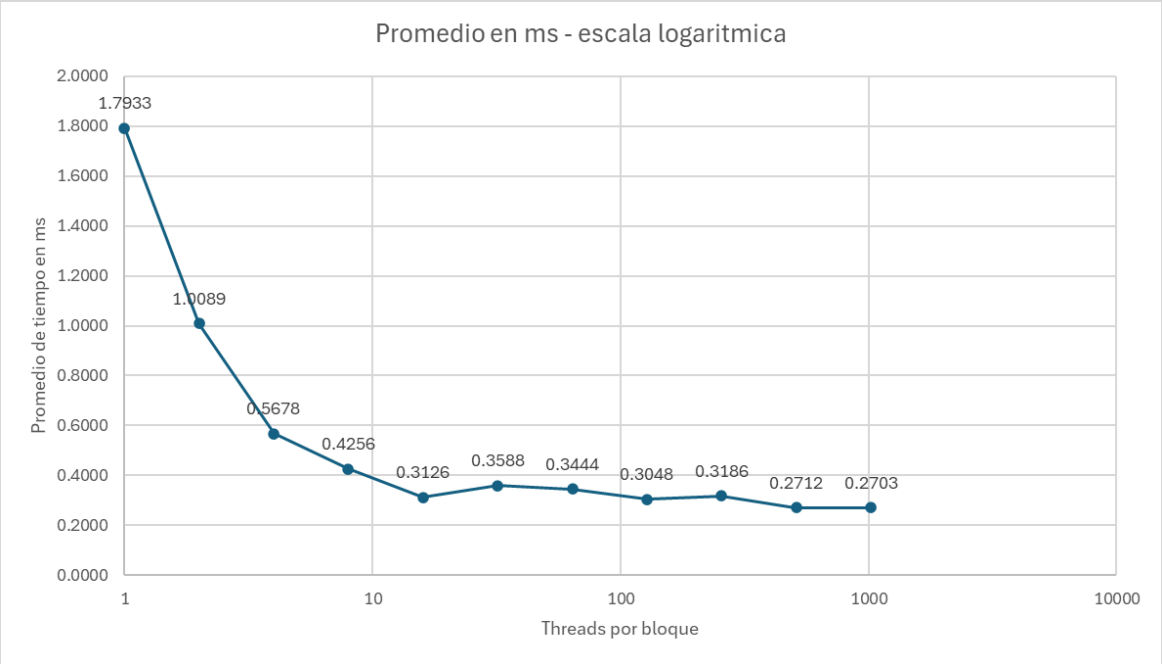
Parte a

Para esta parte se implementó un kernel donde cada hilo será responsable de calcular el producto interno entre una fila de la matriz y el vector. Se tomó la decisión de tomar un hilo por bloque. En la parte b se discutirá optimizaciones y por que esto puede no ser lo mejor. A continuación, los tiempos obtenidos para las ejecuciones. Tiempo promedio: 1.78876 ms, desvío 0.00986998. Para esto se ejecutó la prueba con 100 muestras.God

Parte b

Para esta parte se decidió variar la cantidad de hilos por bloque, notamos que al aumentarla, el tiempo total de ejecución disminuye, esto se debe principalmente a que se está aprovechando del acceso coalescente, ya que los datos que están accediendo los hilos están cercanos unos con otros, en este caso, todo el vector v es el mismo, con lo que los accesos de los hilos pueden ser combinados en menos transacciones de memoria, lo que reduce significativamente la latencia y el ancho de banda necesario. También se hace un uso eficiente de los Warps y de los SM, al tener una cantidad más alta de hilos, se llenan completamente los warps, disminuyendo la cantidad de ciclos de reloj que se desperdiciarian por warps parcialmente llenos. Adicionalmente, mayor ocupación de los SM implica que haya más hilos activos y listos para ejecutarse cuando algunos hilos están esperando por operaciones de acceso a memoria.

A continuación se muestran los resultados obtenidos para 100 muestras de cada tamaño de bloque en escala logarítmica:



En la misma se puede apreciar el fenómeno que mencionamos.