Computación Paralela

Tema 4: Práctica Final

Mayo, 2023

El trabajo propuesto consiste en desarrollar un código en CUDA C/C++ que permita realizar una multiplicación de matrices mediante la aplicación de un conjunto de operaciones algebraicas. Para ello, se proponen una serie de ejercicios que deben realizarse en el orden indicado si se quiere obtener el código final que implemente de forma correcta el producto matricial.

Los alumnos que no dispongan de una GPU de NVIDIA pueden hacer uso de Google Colab para editar, compilar y ejecutar el código CUDA desarrollado. En este enlace se muestra un ejemplo de uso de este entorno de desarrollo ofrecido por Google. Tan sólo es necesario disponer de, al menos, una cuenta en GMail.

Ejercicio 1

Completa el código proporcionado en el directorio `scalarProd` de forma que, dados dos vectores, $v1 \in \Re^n$ y $v2 \in \Re^n$, permita realizar su producto escalar. La suma de los elementos obtenidos tras multiplicar las componentes de los vectores se debe llevar a cabo aplicando la operación de reducción implementada en el Ejercicio 4 (ítem 5) de la Práctica 2.

Ejercicio 2

Completa el código proporcionado en el directorio `transpose` de forma que, dada una matriz, $A \in \Re^{n \times n}$, permita obtener su traspuesta, A^T. Esta operación debe llevarse a cabo exclusivamente en la GPU. Comprueba el funcionamiento del código desarrollado con diferentes tamaños de problema.

Ejercicio 3

Completa el código proporcionado en el directorio `matrixMul` de forma que, dadas dos matrices cuadradas, A y B, permita obtener el producto matricial de ambas, $C = A \times B$. Esta operación se debe implementar de la siguiente forma:

- La matriz B se debe trasponer mediante el kernel implementado en el Ejercicio 2.
- La operación de multiplicación, $C = A \times B^T$, se debe realizar multiplicando cada fila de A por todas las filas de B^T. Para llevar a cabo el producto de los vectores fila de A y B^T, se debe utilizar el kernel implementado en el Ejercicio 1.

Observaciones:

- El producto matricial debe llevarse a cabo de forma asíncrona en la GPU, haciendo uso de streams, kernels concurrentes y memoria pinned.
- El código se debe ejecutar con diferentes tamaños de problema. Para comprobar su correcto funcionamiento, se debe comparar el resultado obtenido en la GPU con el que se obtendría si la operación se hubiese realizado exclusivamente en la CPU.