Programación con GPUs

Práctica 02 - Introducción a CUDA

Segundo Semestre, 2025

ATENCIÓN CON LAS PRÁCTICAS

En programación, la práctica es un componente fundamental del aprendizaje. Hacer estas prácticas te van a dar la oportunidad de poder aplicar lo aprendido en la teoría, entender mejor los conceptos y probar tus habilidades.
La recomendación de la cátedra es que:

- **Leas MUY bien el enunciado**: Analices el eneunciado y entiendas exactamente lo que hay que hacer. Parece sencillo, pero es común resolver un ejercicio diferente al planteado.
- **No busques soluciones óptimas inmediatamente**: Primero intentá resolver el problema y luego pensá si la solución es óptima.
- **Dediques tiempo a pensar**: No te desesperes si no se te ocurre la solución inmediatamente. Es común que las soluciones no salgan a la primera. Pensá en el problema, quizás volvé a leer la teória y fijate si se te ocurre. Son problemas complejos y a veces hay que darles tiempo para que se asienten.
- **¡No busques soluciones rápidas!**: No busques rápido en internet la solución o vayas a leer la solución a la guía de resoluciones inmediátamente. Cada solución que leas rápido te va a dar la falsa sensación de comprensión y te va a sacar la posibilidad de tener el momento **¡AHA!** donde realmente entendiste cómo resolver un problema. **¡Te entendemos, es difícil a veces!**, pero es parte del proceso de aprendizaje.
- **¡No te desanimes!**: Si volviste a pensarlo un tiempo y no se te ocurre nada, es momento de dejar el problema por un tiempo y retomarlo luego.
- **¡Volví al problema luego de un tiempo y no me sale!**: Si volviste a pensar el problema y no se te ocurre nada, es momento de leer la solución. No te sientas mal por esto, pero cuando te sientes a leer la solución. El proceso de leer la solución implica *entenderla*, y NO copiarla. Una vez que entiendas la solución, esperá un tiempo para escribrla y probarla.
- **¿Y si no entiendo la solución?**: Si no entendiste la solución, anotá las dudas, tratá de pensar qué es lo que te falta y preguntá a los docentes de la cátedra. ¡Nunca te quedes con la duda!
- **ChatGPT (cualquier LLM) lo resuelve todo**: ¡Es verdad!, pero como cualquier herramienta, cuanta más teoría sepamos, mejor podremos utilizarla.

¡Suerte en la práctica!

Contents

2.1 Ejercicio: Suma de dos vectores

Si quisiéramos utilizar un thread para calcular la suma de los dos vectores. ¿Cómo se calcularía el índice del thread?

i. `idx = threadIdx.x + threadIdx.y;`

ii. `idx = blockIdx.x + threadIdx.x;`

iii. `idx = blockIdx.x * blockDim.x + threadIdx.x;`

iv. `idx = blockIdx.x * threadIdx.x;`

2.2 Ejercicio: Sumar elementos adyacentes

Supongamos ahora que queremos usar un thread para sumar dos elementos contiguos de un vector (por simplicidad podemos suponer que el vector tiene un número par de elementos). ¿Cómo se calcularía el índice del thread?

i. `idx = blockIdx.x * blockDim.x + threadIdx.x + 2;`

ii. `idx = blockIdx.x * threadIdx.x * 2;`

iii. `idx = (blockIdx.x * blockDim.x + threadIdx.x) * 2;`

iv. `idx = blockIdx.x * blockDim.x * 2 + threadIdx.x;`

2.3 Ejercicio: Calcular el tamaño de los threads

Para un programa que suma dos vectores, supongamos que el tamaño de los vectores es 8000 y que el tamaño del bloque de threads es 1024. El programador configura que el kernel lance la menor cantidad de bloques de threads para cubrir todos los elementos de salida. ¿Cuántos threads habrá corriendo en el *grid*?

2.4 Ejercicio: cudaMalloc (parte 1)

Si queremos reservar un array de `v` elementos de tipo `int` en la memoria global del dispositivo CUDA, ¿cuál sería la expresión apropiada para el segundo argumento de la llamada a `cudaMalloc`?

2.5 Ejercicio: cudaMalloc (parte 2)

Si queremos reservar un array de `n` elementos del tipo `float` y tenemos una variable `d_A` que apunta a la memoria reservada, ¿cuál sería la expresión apropiada para el primer argumento de la llamada a `cudaMalloc`?

2.6 Ejercicio: Copia de memoria desde el *host* al *device*

Si queremos copiar 3000 bytes de datos desde el array del host `h_A` (donde `h_A` es un puntero al elemento 0 del array de origen) al array del dispositivo `d_A` (donde `d_A` es un puntero al elemento 0 del array de destino), ¿cuál sería la llamada a la API apropiada para esta copia de datos en CUDA?

2.7 Ejercicio: Manejo de errores

¿Cómo declararías una variable `err` que pueda recibir apropiadamente el valor devuelto por una llamada a la API de CUDA en caso de error?

2.8 Ejercicio: funciones en CUDA

Estás trabajando en un proyecto de CUDA y uno de tus compañeros se uqeja de que CUDA es muy tedioso ya que hubo que declarar muchas funciones que se planean ejecutar tanto en el host como en el device dos veces, una como función de host y otra como función de device. ¿Cuál sería tu respuesta?