Universidad del Valle de Guatemala
Facultad de Ingenieria
Departamento de Ciencias de la Computación
CC3054 Organización de Computadoras y Assembler

Introducción a Assembler ARM
Ciclo 2 de 2025

Laboratorio 09

CUDA: múltiples bloques y múltiples Streams

INSTRUCCIONES: realiza esta actividad en grupos de 2 personas máximo. Puedes trabajar individualmente si así lo deseas.

Número de Grupo en Canvas: _____

Nombre: Sarah Rachel Estrada Banilla **Carné:** 24347

Nombre: Angel Gabriel Chávez Otzoy **Carné:** 24248

Ejercicio 1. 25 pts: 2.5 c/uno. Analiza el código Lab09_1S.ipynb, e investiga las siguientes funciones:

INSTRUCCION	EXPLICACION DE FUNCIONALIDAD Y SALIDA GENERADA	PARÁMETROS DE ENTRADA
fopen	Es una función que tama dos argumentos y devuelve un flujo de archivos asociados con el archivo especificado	Nombre archivo modo
fclose	Es una función para cerrar un archivo finalizando el guardado del contenido en el archivo y se libera la memoria.	Puntero creado por fopen.
fgets	Esta función lee un máximo de caracteres del flujo de archivo dado y los almacena en la matriz a la que apunta.	str: puntero cantor: max caracteres arroyo: flujo de archivo
strtok	Es una función que devuelve el siguiente token en una cadena	str: puntero a la cadena a tokenizar delim: delimitador
atof	Se utiliza para convertir una cadena que representa un número de punto flotante en un valor de punto flotante de doble precisión.	str: cadena representando punto flotante
CudaMemcpy Async	Esta función copia datos entre el host y el dispositivo, puede devolver un success o un error	dst: destino, count: src: fuente tamaño kind: tipo bytes. stream: IO stream
CudaStream Synchronize	Bloquea el host hasta que se hayan completado todas las operaciones en el flujo especificado. retorna éxito o error.	stream: IO stream
cudaEventCreate	Crea un evento que se puede utilizar para realizar un seguimiento de la finalización de tareas en un flujo. retorna éxito o error.	event: evento
cudaEventRecord	Registra un evento. Captura en el evento el contenido del stream en el momento de la llamada.	event: stream:
CudaEventElap sedTime	Calcula el ~~contenido~~ tiempo transcurrido entre eventos. Devuelve éxito o error.	ms start end

/ /

Ejercicio 2

¿Qué hace el programa? Explique su funcianamiento.
El programa calcula el promedio de de las columnas que contienen el cambio de temperatura por añeo.
El programa primero abre el archivo csv, descarta los encabezados, lee las 4,318 filas y extrae las ultimas 10 columnas. Crea 10 vectores que representan cada columna. Luego utiliza una funcion dividir cada linea por comas y convertir los números de la cadena en punto flotante. Reserva memoria en la GPU para los 10 vectores y otro para guardar los resultados y luego canfigura los eventos para medir tiempos de cada etapa. Se utiliza un hilo por fila donde se suman los 10 valores y calcula el promedio. Se calculan 256 hilos por bloque. Todo esto en el lanzamiento del Kernel. Luego se miden los tiempos de host a device, del lanzamiento del Kernel y del device al host. Imprime los tiempos de ejecución. imprime los primeros 20 promedios y libera la memoria.

¿Porqué leer linea por linea del CSV y no cargar todo el documento en una sola operación?
Primero, para evitar que se sobrecarge la RAM, ademas de que el archivo es muy grande como para leerlo completo, Es más eficiente ya que aunque hubiera un error en una linea, el programa continuaria y no habria sobrecarga.

/ /

¿Qué pasaría si el dataset creciera a 200,000 filas? ¿Qué cambiaría?

Se podría cambiar el hecho de solo utilizar 10 vectores a matrices ya que son bastantes datos, o utilizar mayor cantidad de vectores para guardar los datos y luego calcularlos. Así se puede evitar que se sobrecargue la RAM.

¿Porqué el kernel asigna un hilo por fila en lugar de 1 hilo por columna?

Se aprovecha mejor el uso del GPU, ya que se puede paralelizar mejor al ser 2318 hilos y no solo 10 hilos y se evitan condiciones de carrera ya que cada fila es independiente.

Ejercicio 2. 20 pts: 5 c/una Responde las siguientes preguntas:

¿Qué hace el programa? Explica el funcionamiento
¿Por qué leer línea por línea del CSV y no cargar todo el documento en una sola operación?
¿Qué pasaría si el dataset creciera a 200,000 filas? ¿Qué cambiaría?
¿Por qué el kernel asigna 1 hilo por fila en lugar de 1 hilo por columna?

Ejercicio 3. 10 pts. Agrega al código la funcionalidad que identifique cuál es el país con el valor de promedio más alto calculado. Es decir, que imprima el nombre del país.

¿Este proceso se debe hacer en el Host o en el Device? Explica tu respuesta. Es mejor hacer el calculo en el host ya que se debe de buscar el país con mayor promedio y son relativamente pocas filas.

Ejercicio 4. 5 pts. Modifica el CSV para que en lugar de tener 4318 elementos, este contenga una cantidad significativa de filas (mínimo duplicar la cantidad de filas). Esto nos ayudará a que se vea mejor el comportamiento de los streams.

Ejercicio 5. 40 Pts: 10 cada inciso.

a. **Modifica el código para que al momento de ejecutar el programa (Asegúrate de quitar cudaDeviceSynchronize() entre etapas):**
   - El GPU utilice 2 streams.
   - El GPU utilice 4 streams.
   - El GPU utilice 8 streams.

b. **Completa la siguiente tabla:**

Cantidad de Streams	Tiempo Host->Device	Tiempo Kernel	Tiempo Device->Host
1	0.165888	0.100960	0.044256
2	0.175424	0.108480	0.051200
4	0.191424	0.104448	0.71680
8	0.302176	0.137760	0.120256

c. **Genera gráficas con las siguientes características:**
   - En el eje X la cantidad de Streams, eje Y Tiempo Host->Device
   - En el eje X la cantidad de Streams, eje Y Tiempo Kernel
   - En el eje X la cantidad de Streams, eje Y Tiempo Device->Host

d. **¿Puedes observar algún patrón de comportamiento? En cuál de los tres tiempos se ve una mayor influencia de cambio de acuerdo con la cantidad de streams utilizados. Explica por qué las gráficas se comportan de la forma que se muestra en tus dibujos.** Se puede observar que al transferir la memoria, al utilizar multiples streams, hay una disminución del tiempo. Esto porque los streams tienen un flujo independiente que permite el paralelismo entre transferencias de memoria y la ejecución de kernels.