Programación básica y Programación de algoritmos paralelos con CUDA

Pablo Liste Cancela, Jorge Lojo Abal

Programación de Arquitecturas Emergentes
Universidad de Santiago de Compostela
Ingeniería Informática

Marzo 2025

Índice general

1. Introducción

En la era actual de la computación de alto rendimiento, las unidades de procesamiento gráfico (GPUs) han transformado radicalmente el panorama del cómputo paralelo, ofreciendo capacidades sin precedentes para resolver problemas computacionalmente intensivos. Gracias a su arquitectura altamente paralela, las GPUs permiten acelerar significativamente aplicaciones en diversas áreas, como aprendizaje profundo, simulaciones científicas, análisis de grandes volúmenes de datos y renderizado de gráficos. En este contexto, la presente práctica se centra en la **Programación de Algoritmos Paralelos con CUDA**, conformando el segundo bloque del laboratorio dentro de la asignatura de **Programación de Arquitecturas Emergentes** del Grado en Ingeniería Informática.

El objetivo principal de esta práctica es explorar e implementar algoritmos paralelos fundamentales utilizando la arquitectura CUDA (*Compute Unified Device Architecture*) en las GPUs NVIDIA A100 disponibles en el **Centro de Supercomputación de Galicia** (CESGA). Estas unidades representan una de las arquitecturas más avanzadas de NVIDIA, integrando numerosas innovaciones como los *Tensor Cores* de tercera generación, memoria HBM2 de alta velocidad y la arquitectura Ampere. En conjunto, estos avances permiten un rendimiento excepcional en aplicaciones de computación paralela, con especial énfasis en operaciones de aprendizaje profundo, simulación y análisis de datos a gran escala.

A diferencia de prácticas anteriores centradas en el desarrollo de algoritmos secuenciales y en la introducción al entorno de trabajo, este trabajo aborda la implementación y optimización de tres algoritmos fundamentales que explotan diferentes características y patrones de acceso a memoria de las GPUs. La selección de estos algoritmos se fundamenta en la diversidad de desafíos que presentan en términos de paralelismo, uso eficiente de memoria y optimización del rendimiento:

- **DAXPY (Double-precision $\mathbf{A \cdot X}$ Plus Y)**: Un algoritmo de álgebra lineal básico que combina operaciones de multiplicación escalar y suma vectorial. Se implementará utilizando memoria unificada para evaluar sus beneficios en operaciones vectoriales. Se analizará cómo la transferencia de datos entre CPU y GPU influye en el rendimiento y se estudiarán técnicas de optimización para mejorar la eficiencia computacional.
- **Histograma**: Un algoritmo con patrones de acceso irregular y desafíos de sincronización, implementado en dos versiones (memoria global y memoria compartida) para analizar comparativamente el impacto de la jerarquía de memoria en el rendi-

miento. Se estudiarán estrategias de reducción de colisiones en la memoria global y el uso de memoria compartida para mejorar la eficiencia.

- **Convolución**: Un algoritmo fundamental en procesamiento de señales e imágenes con patrones de acceso regulares pero intensivos en memoria. Se implementará tanto con una solución propia utilizando memoria global como con la librería especializada NPP de NVIDIA. Se analizará la eficiencia de cada enfoque y su escalabilidad en función del tamaño de entrada.

La arquitectura NVIDIA A100 proporciona un escenario ideal para estas implementaciones, con sus 108 *Streaming Multiprocessors* (SMs), 6912 CUDA *cores*, 432 *Tensor Cores* y 40 GB de memoria HBM2 con un ancho de banda de hasta 1.6 TB/s. Esta configuración *hardware* permite explorar los límites reales de paralelismo y evaluar cómo diferentes decisiones de diseño afectan al rendimiento final de los algoritmos.

Este estudio no solo busca lograr implementaciones eficientes, sino también comprender en profundidad cómo las características arquitectónicas de las GPUs modernas influyen en el rendimiento de diferentes patrones algorítmicos. Se analizarán aspectos críticos como la configuración óptima de bloques e hilos, la gestión eficiente de la jerarquía de memoria, la ocupancia de los multiprocesadores y los límites de la capacidad computacional en cada algoritmo. Además, se evaluarán estrategias de optimización como la minimización de accesos a memoria global y el uso de memoria compartida, entre otros.

Los resultados obtenidos se documentarán mediante un análisis comparativo, abarcando diversas métricas clave que permitan evaluar el rendimiento y la eficiencia de las implementaciones desarrolladas. En particular, se medirán los factores de aceleración (*speedup*) con respecto a ejecuciones secuenciales y optimizadas, así como los tiempos detallados de cada una de las fases del cómputo, incluyendo la transferencia de datos entre CPU y GPU, la inicialización de estructuras, la ejecución de los *kernels* y la recolección de resultados.

Además, se estudiarán en profundidad los patrones de acceso a memoria, diferenciando entre el uso de memoria global, compartida, cachés y registros, con el fin de determinar su impacto en el rendimiento. Se analizará la eficiencia en la utilización de los recursos computacionales, evaluando métricas como la ocupación de los multiprocesadores, la latencia de acceso a memoria y el balance de carga entre los hilos de ejecución. Asimismo, se explorarán posibles cuellos de botella que puedan surgir en la ejecución y se compararán estrategias de optimización para mitigarlos.

Finalmente, se discutirán las observaciones y conclusiones derivadas de la experimentación con distintas configuraciones de ejecución y tamaños de problema, proporcionando una visión integral sobre la programación eficiente en GPUs modernas. Este análisis permitirá no solo validar la efectividad de las estrategias implementadas, sino también extraer pautas generales para el desarrollo de algoritmos paralelos altamente optimizados en arquitecturas CUDA.

2. Laboratorio 2.1

2.1. Compilar y ejecutar programas

2.2. Bloque 1: Programación básica con CUDA

Compila y ejecuta el código `00_dev_query.cu`

¿Cuál es el número máximo de hilos que puede ejecutar la GPU de forma simultánea?