Instituto Politécnico Nacional

“Escuela Superior de Computo”

Integrantes:

- Ignacio Cortés Atzin Maxela
- Ríos Rivera Fernanda Anahí

Grupo: 6BM2

Unidad de aprendizaje: Cómputo paralelo

Profesor: Luis Alberto Ibáñez Zamora

“Ensayo sobre CUDA”

Índice

Introducción

En el contexto actual de la computación de alto rendimiento, los sistemas multiprocesador, en particular aquellos basados en GPUs (Unidades de Procesamiento Gráfico), han transformado significativamente la forma en que los problemas complejos son resueltos. La programación en paralelo se ha establecido como un nuevo enfoque para aprovechar el poder de estos sistemas, permitiendo la ejecución simultánea de múltiples instrucciones y tareas. Este paradigma es especialmente relevante en el caso de las GPUs, que están diseñadas para ejecutar grandes volúmenes de operaciones en paralelo, lo que las hace ideales para aplicaciones que requieren una enorme capacidad de procesamiento, como la mecánica computacional, e incluso la inteligencia artificial en áreas como el aprendizaje de máquina y la visión artificial.

A diferencia de la programación secuencial, en la que las instrucciones se ejecutan una tras otra, la programación paralela divide un problema en partes más pequeñas que pueden resolverse de manera concurrente. Esto no solo mejora el rendimiento al distribuir la carga de trabajo entre varios procesadores o hilos, sino que también permite superar las limitaciones impuestas por la frecuencia de reloj de las CPUs tradicionales. La programación en CUDA (Compute Unified Device Architecture), una plataforma de programación paralela desarrollada por NVIDIA, aprovecha la arquitectura de las GPUs para ejecutar algoritmos de manera eficiente. CUDA permite a los programadores utilizar un lenguaje de programación similar a C, pero con la capacidad de escribir código que se ejecute directamente en la GPU.

El avance de la computación paralela mediante CUDA ha permitido que las GPUs no solo se utilicen para el procesamiento gráfico, sino también como coprocesadores en tareas de computación científica. A través de la arquitectura de CUDA, se ha posibilitado el desarrollo de aplicaciones más rápidas y potentes, optimizando el rendimiento en tareas como simulaciones físicas, procesamiento de imágenes, y aprendizaje profundo, entre otras. Este trabajo explora el entorno de programación CUDA, su implementación en aplicaciones de mecánica

computacional, y cómo la integración de CPUs y GPUs en un modelo de cómputo combinado ha transformado el panorama de la programación en paralelo.

¿Qué es CUDA a nivel hardware?

En las tarjetas gráficas usuales, los procesadores de vértices y de fragmentos se encargan de diferentes tareas, pero cada uno tiene instrucciones y capacidades distintas. Esto puede generar desequilibrios en la carga de trabajo y dificultar la eficiencia.

CUDA, desde la arquitectura de los GPU, representa un diseño unificado que maximiza el aprovechamiento del paralelismo masivo. Este enfoque incluye Streaming Multiprocessors (SM), que agrupan núcleos CUDA (o Streaming Processors, SP), encargados de ejecutar instrucciones de manera concurrente. En esencia, proporciona la infraestructura física que soporta el modelo de programación y permite la ejecución eficiente de aplicaciones paralelas.

En este contexto, CUDA se puede analizar desde dos perspectivas principales: como modelo de programación y como arquitectura de cómputo.

Desde la perspectiva del modelo de programación, CUDA es una plataforma que permite aprovechar el paralelismo de las GPUs NVIDIA. Este modelo introduce una jerarquía de hilos estructurada en bloques y mallas (grids), lo que facilita dividir problemas computacionales complejos en tareas más pequeñas que pueden ser ejecutadas en paralelo. Los desarrolladores escriben funciones denominadas kernels, que son ejecutadas de forma concurrente por múltiples hilos en la GPU. Además, proporciona un manejo detallado de la memoria, organizándola en espacios globales, compartidos, locales y constantes, lo que permite optimizar el rendimiento del programa a nivel de software. Su principal enfoque es abstraer los detalles del hardware, permitiendo un mejor enfoque en estructurar y optimizar las aplicaciones para que sean eficientes en términos computacionales.

Por otro lado, como arquitectura de cómputo, CUDA se refiere al diseño físico de las GPUs y su capacidad para ejecutar operaciones masivamente paralelas. Una

GPU compatible con CUDA está formada por múltiples Streaming Multiprocessors (SM), que contienen núcleos CUDA (o Streaming Processors, SP). Estos núcleos son los encargados de realizar cálculos aritméticos y lógicos. La arquitectura también incluye una jerarquía de almacenamiento, que abarca memoria global, compartida, local y registros, optimizando el acceso a los datos para reducir la latencia y maximizar el ancho de banda. Un aspecto destacado de esta arquitectura es la organización de los hilos en grupos de 32, denominados warps, que se ejecutan físicamente en paralelo. Esta unión en el diseño de la arquitectura permite escribir código más sencillo y optimizado.

Componentes principales de la arquitectura CUDA

Con la evolución hacia la arquitectura CUDA, se unifican estos procesadores en una única unidad de procesamiento, los Streaming Multiprocessors (SM), lo que permite un manejo más eficiente y homogéneo de las tareas.

Estas unidades de procesamiento tienen una serie de núcleos que son los encargados de ejecutar las instrucciones. A su vez, están interconectadas entre sí por una región de memoria compartida. Cada SM está formado por núcleos de procesamiento denominados 'núcleos CUDA' o Streaming Processors (SP) que se encargan de ejecutar las instrucciones.

Ilustración 1. Arquitectura de una tarjeta gráfica CUDA-enabled

Dentro de la jerarquía de ejecución, los hilos se organizan en bloques y estos bloques forman mallas. Cada bloque de hilos puede ejecutar instrucciones de manera concurrente, y la unidad de distribución de trabajo se encarga de distribuir los bloques entre los diferentes SM. Cada hilo tiene asignados registros privados y no hay penalización por cambio de contexto, lo que mejora la eficiencia de la ejecución.

Los bloques pueden tener una, dos o tres dimensiones, lo que es útil para trabajar con datos multidimensionales, como en el procesamiento de imágenes o en la resolución de ecuaciones diferenciales en varias dimensiones.

Otro componente importante en esta arquitectura es la ejecución en warps. Un warp es un grupo de 32 hilos que se ejecutan en paralelo y deben seguir la misma ruta de ejecución. Si todos los hilos de un warp siguen la misma instrucción, se alcanza la máxima eficiencia. Si los hilos siguen diferentes instrucciones (debido a bifurcaciones), puede haber una pérdida de eficiencia.

Respecto a la memoria, los hilos acceden a los datos a diversos niveles dentro de una jerarquía de memoria. Haciendo que cada hilo tenga una zona reservada de memoria local y cada bloque tenga un área de memoria compartida visible por todos los hilos de su mismo bloque. Concluyendo en un mismo espacio de memoria global al que todos los hilos pueden acceder ubicado en un chip externo de memoria DRAM.

Una de las claves de la programación en CUDA es el kernel, que es una función o conjunto de instrucciones que se ejecutan de manera paralela en los núcleos CUDA. Se define el número de hilos por bloque y el número de bloques. Los hilos se agrupan en bloques y forman una malla, lo que facilita la programación y organización de los datos. Este modelo de programación es sencillo porque permite pensar en las operaciones como una distribución de trabajo que se ejecuta en paralelo.

Evolución de CUDA

Desde su aparición en 2006, CUDA (Compute Unified Device Architecture) ha revolucionado la computación paralela al aprovechar la capacidad de procesamiento masivo de las GPU de NVIDIA. A lo largo de los años, esta arquitectura ha evolucionado a través de diferentes generaciones, cada una marcando hitos importantes en cuanto a rendimiento, eficiencia y soporte para nuevas aplicaciones como inteligencia artificial y simulaciones científicas.

Se puede describir una línea del tiempo de las principales generaciones de arquitectura CUDA:

La primera generación, conocida como Tesla (2006), marcó el inicio del cómputo general en GPU (GPGPU). Aunque rudimentaria comparada con las siguientes, Tesla introdujo el concepto de núcleos CUDA y permitió que los programadores accedieran directamente al paralelismo masivo de las GPU para acelerar cálculos científicos y de ingeniería. Fue un punto de partida clave que demostró que las GPU podían hacer mucho más que solo gráficos.

En 2010, NVIDIA presentó Fermi, una arquitectura que representó un gran salto en capacidades programables. Fermi introdujo por primera vez una jerarquía de caché (L1 y L2) en la GPU, mejoró la ejecución en doble precisión (FP64), y añadió soporte para corrección de errores (ECC) en memoria, lo cual fue crucial para aplicaciones críticas. Además, aumentó la cantidad de registros disponibles por hilo, permitiendo algoritmos más complejos.

Dos años después, en 2012, llegó Kepler, una arquitectura orientada a mejorar la eficiencia energética y la capacidad de procesamiento simultáneo. Kepler introdujo características innovadoras como el Parallel Dynamic Kernel Execution (ejecución dinámica de kernels) y Hyper-Q, que permitió ejecutar múltiples flujos de trabajo de manera concurrente, mejorando así la utilización de los recursos en entornos multiusuario o multitarea.

En 2014, Maxwell refinó aún más la eficiencia por vatio y reorganizó la jerarquía de memoria, unificando la memoria compartida con la caché L1, lo que simplificó la

gestión de datos y redujo la latencia. Esta arquitectura también mejoró la compresión de memoria y el rendimiento general en aplicaciones gráficas y de cómputo. Maxwell fue la base de muchas GPU comerciales ampliamente usadas en aplicaciones de consumo y profesionales.

Con el lanzamiento de Pascal en 2016, NVIDIA introdujo tecnologías avanzadas como la memoria HBM2 de alta velocidad y el NVLink, una interconexión de alta banda ancha entre GPU y CPU o entre GPUs. Pascal también ofreció un rendimiento superior en operaciones de precisión mixta, especialmente en FP16, lo que lo convirtió en una plataforma ideal para los inicios del aprendizaje profundo moderno.

La arquitectura Volta, lanzada en 2017, supuso un punto de inflexión para la inteligencia artificial. Introdujo los primeros Tensor Cores, unidades especializadas para multiplicaciones de matrices que aceleran drásticamente redes neuronales profundas. Volta también mejoró significativamente el rendimiento en operaciones de precisión mixta y ofreció una nueva capacidad para ejecutar modelos de IA a gran escala.

En 2018, Turing amplió esta idea con la incorporación de RT Cores, núcleos dedicados al trazado de rayos en tiempo real (Ray Tracing), lo que marcó un avance en la fidelidad gráfica en tiempo real. Además, los Tensor Cores de segunda generación ofrecieron un rendimiento aún mayor para inferencias de IA. Turing consolidó a CUDA como una arquitectura no solo para cómputo científico, sino también para gráficos de última generación y renderizado híbrido.

Ampere, presentada en 2020, fue diseñada para ser versátil tanto en centros de datos como en videojuegos. Incluyó Tensor Cores de tercera generación y mejoró el soporte para nuevos formatos de precisión como TF32 y FP16, optimizados para aprendizaje profundo. Esta generación también permitió una mayor escalabilidad, adaptándose desde GPUs para laptops hasta supercomputadoras.

Finalmente, en 2022, Hopper representó un enfoque completamente centrado en inteligencia artificial de alto rendimiento. Equipadas con Tensor Cores

especializados en FP8 y un nuevo Transformer Engine, las GPUs Hopper están optimizadas para entrenar y ejecutar modelos de lenguaje masivos como GPT. Con interconexiones NVLink 4.0 y memoria HBM3, Hopper está diseñada para liderar la nueva era de la computación acelerada y la IA generativa.

Últimos avances tecnológicos

En la última década, la aceleración del cómputo mediante GPU ha transformado profundamente campos como la inteligencia artificial, la simulación científica, los gráficos en tiempo real y el análisis de big data. NVIDIA ha liderado esta revolución a través de su plataforma CUDA, que ha evolucionado en paralelo con las arquitecturas de sus GPU. La generación más reciente, Hopper, representa la cima de esta evolución, al integrar tecnologías altamente especializadas que responden directamente a las demandas computacionales del presente y del futuro.

La arquitectura Hopper, presentada en 2022 con la GPU H100, está diseñada específicamente para cargas de trabajo centradas en inteligencia artificial y deep learning a gran escala, tiene 144 módulos de silicio (SM) en total, con 128 núcleos FP32, 64 núcleos FP64, 64 núcleos INT32 y cuatro núcleos Tensor por módulo de silicio. A diferencia de generaciones anteriores como Ampere o Turing, que equilibraban gráficos e IA, Hopper se enfoca exclusivamente en ofrecer el mayor rendimiento posible para el entrenamiento e inferencia de modelos masivos. Uno de sus avances más destacados es la inclusión de Tensor Cores de cuarta generación con soporte para el formato FP8, que permite ejecutar operaciones con una precisión extremadamente eficiente, reduciendo el uso de energía y memoria sin sacrificar la calidad de los resultados.

Además, Hopper introduce el Transformer Engine, un bloque de hardware optimizado específicamente para ejecutar redes neuronales de tipo Transformer, como las que se utilizan en modelos de lenguaje extensos (LLMs) como GPT y BERT. Este motor combina precisión mixta y algoritmos adaptativos para acelerar de forma inteligente las operaciones en FP8 o FP16 según lo requiera la red. Como resultado, el entrenamiento de modelos que antes tomaba semanas ahora puede

reducirse significativamente a días u horas, lo cual es vital en contextos empresariales y científicos donde el tiempo es crítico.

En paralelo, otras innovaciones de hardware se integran perfectamente a Hopper. La arquitectura incluye soporte para memoria HBM3, que proporciona un ancho de banda excepcionalmente alto para flujos de datos intensivos. Asimismo, NVIDIA NVLink 4.0, la interconexión de alta velocidad entre GPUs, permite escalar múltiples H100 de forma eficiente, eliminando los cuellos de botella asociados con buses tradicionales como PCIe. Esto es crucial para clústeres de alto rendimiento y supercomputadoras donde la cooperación entre decenas o cientos de GPUs es esencial.

Otro avance fundamental de Hopper es su compatibilidad con tecnologías como Unified Memory, que simplifica el acceso a la memoria entre la CPU y la GPU. En lugar de copiar datos manualmente entre ambas, Unified Memory permite a los desarrolladores trabajar con un único espacio de direcciones compartido, lo que reduce la complejidad del código y mejora el rendimiento en ejecuciones mixtas. Esta característica es particularmente beneficiosa en entornos donde las aplicaciones están distribuidas entre distintas capas de procesamiento.

En cuanto al ecosistema de software, Hopper mantiene una integración total con los principales marcos de inteligencia artificial, incluyendo TensorFlow, PyTorch y JAX. Los desarrolladores pueden utilizar estas plataformas con mínimos ajustes para aprovechar automáticamente la aceleración por Tensor Cores y otras capacidades avanzadas de la GPU. Además, NVIDIA proporciona bibliotecas especializadas como cuDNN, TensorRT y NCCL, optimizadas para correr sobre Hopper, lo que permite un desarrollo más rápido y eficaz en proyectos de IA de última generación.

La arquitectura Hopper no solo representa una mejora incremental, sino una redefinición del cómputo acelerado. Al integrar núcleos especializados, memoria de ultra alto rendimiento, conectividad avanzada y un ecosistema de software maduro, Hopper establece un nuevo estándar en el diseño de GPUs para inteligencia artificial y aplicaciones científicas avanzadas, NVIDIA ha anunciado ya nuevas líneas de investigación que apuntan a futuras generaciones aún más optimizadas, con mayor

densidad de cómputo, menor consumo energético y una integración más profunda con aplicaciones de IA generativa, cómputo cuántico simulado y sistemas autónomos. Con estos avances, CUDA se consolida como una plataforma esencial para el futuro de la computación a gran escala.

Aplicaciones y relevancia actual

Lo que comenzó como una herramienta para facilitar el acceso a la capacidad de cómputo masivo de las GPU, se ha transformado en la base fundamental que impulsa una amplia gama de disciplinas tecnológicas, desde la visión computacional y el aprendizaje profundo, hasta las simulaciones científicas más complejas y la industria de los videojuegos.

En el campo de la visión por computadora, CUDA ha sido crucial para acelerar algoritmos de detección de objetos, segmentación semántica, reconocimiento facial y reconstrucción 3D. Estos procesos, que tradicionalmente eran computacionalmente costosos en CPU, se benefician enormemente de la arquitectura paralela de CUDA, permitiendo que tareas que antes tomaban minutos o segundos ahora se ejecuten en milisegundos. Esto ha sido esencial para aplicaciones en tiempo real como los vehículos autónomos, la vigilancia inteligente y los sistemas biométricos, donde la latencia mínima es crítica.

En el ámbito del aprendizaje profundo, la importancia de CUDA es aún más evidente. Frameworks como TensorFlow, PyTorch y Keras están diseñados para correr sobre CUDA, aprovechando los núcleos CUDA y Tensor Cores para entrenar modelos con millones o incluso billones de parámetros. Gracias a esta plataforma, el entrenamiento de redes neuronales convolucionales, transformers o modelos generativos que antes requerían semanas en CPU, ahora puede realizarse en horas o días. Esta aceleración ha permitido el auge de tecnologías como la IA generativa, los asistentes conversacionales y los sistemas de recomendación inteligente, todos impulsados por la potencia computacional de CUDA.

La computación de alto rendimiento (HPC) es un campo que se centra en la agregación de potencia computacional para resolver problemas complejos en

ciencia, ingeniería o negocios con mayor rapidez. Los núcleos CUDA, con sus capacidades de procesamiento paralelo, desempeñan un papel fundamental en la HPC. La computación de alto rendimiento a menudo requiere la ejecución de un gran número de operaciones matemáticas, una tarea ideal para la arquitectura paralela de los núcleos CUDA. Cada núcleo CUDA puede ejecutar una sola instrucción a la vez, pero al combinarse en miles, como en las GPU modernas, pueden procesar grandes conjuntos de datos en paralelo, reduciendo significativamente el tiempo de computación. Por ejemplo, en simulaciones científicas, que a menudo implican la resolución de modelos matemáticos complejos, las capacidades de procesamiento paralelo de los núcleos CUDA se pueden aprovechar para realizar cálculos en grandes conjuntos de datos simultáneamente. Esto puede reducir significativamente el tiempo necesario para completar la simulación, permitiendo a los científicos realizar simulaciones más complejas y detalladas.

El aprendizaje automático y la inteligencia artificial (IA) son campos que requieren una alta potencia computacional debido a la complejidad de los algoritmos y al tamaño de los conjuntos de datos involucrados. Los núcleos CUDA, con sus capacidades de procesamiento paralelo, desempeñan un papel fundamental en estos campos. Los algoritmos de aprendizaje automático, en particular los de aprendizaje profundo, implican la realización de un gran número de multiplicaciones de matrices. Estas operaciones se pueden paralelizar y ejecutar eficientemente en los núcleos CUDA. Por ejemplo, al entrenar un modelo de aprendizaje profundo, la salida de cada neurona en una capa se puede calcular de forma independiente. Esto significa que la tarea se puede dividir entre varios núcleos CUDA, y cada núcleo calcula la salida de una neurona diferente simultáneamente. Este procesamiento paralelo reduce significativamente el tiempo de entrenamiento, lo que permite entrenar modelos más complejos o utilizar conjuntos de datos más grandes. Además de acelerar el entrenamiento de los modelos de aprendizaje automático, los núcleos CUDA también intervienen en la fase de inferencia. La

inferencia implica utilizar un modelo entrenado para realizar predicciones a partir de nuevos datos. Esta tarea también se puede paralelizar y ejecutar eficientemente en núcleos CUDA, lo que resulta en tiempos de respuesta más rápidos en aplicaciones que requieren predicciones en tiempo real, como la conducción autónoma o el reconocimiento de voz. La inteligencia artificial, especialmente en áreas como el procesamiento del lenguaje natural y la visión artificial, también se beneficia de las capacidades de procesamiento paralelo de los núcleos CUDA. Tareas como el reconocimiento de imágenes o la traducción de idiomas implican la realización simultánea de un gran número de cálculos, una tarea ideal para las capacidades de los núcleos CUDA.

Conclusión

Para concluir con este trabajo, se puede decir que CUDA ha aportado diversos avances en el campo de la computación de alto rendimiento al desarrollar una plataforma que permite combinar una arquitectura de hardware optimizada con un modelo de programación que está diseñado para maximizar el uso del cómputo paralelo. Además de que su estructura cuenta con componentes como los Streaming Multiprocessors (SM), Streaming Processors (SP), jerarquías de memoria, warps, kernels, entre otros, que permiten ejecutar tareas de alta complejidad con una buena eficiencia. Estas características son valiosas para campos como la inteligencia artificial, ya que permiten optimizar el rendimiento en procesos intensivos como el entrenamiento y la inferencia de modelos, maximizando el aprovechamiento del paralelismo masivo que ofrecen las GPUs.

También, se puede notar su evolución desde la serie Tesla hasta generaciones más recientes como Hopper, que incorpora grandes innovaciones con los núcleos tensoriales que han incrementado su alcance en áreas como inteligencia artificial o simulaciones científicas.

Con el continuo desarrollo de las GPUs, CUDA sigue avanzando en el rendimiento computacional, permitiendo abordar problemas cada vez más complicados de

manera más rápida y eficiente. Por lo que CUDA es una herramienta que se volverá clave para enfrentar los desafíos computacionales del futuro.

Bibliografía

- Sabri, O. (2010). Programación en el entorno CUDA en aplicaciones de mecánica computacional. Biblioteca de la Escuela Superior de Ingenieros de Sevilla. https://biblus.us.es/bibing/proyectos/abreproy/11926/fichero/Segunda+parte+TECNOLOGIA+CUDA%252Fi_cuda.pdf

- Represa Pérez, C., Cámara Nebreda, J. M., & Sánchez Ortega, P. L. (2016). Introducción a la programación en CUDA (v3.1). Área de Tecnología Electrónica, Departamento de IngenieríaIng.eniería Electromecánica, Universidad de Burgos.
- An Even Easier Introduction to CUDA. (2022). Recuperado de https://developer.nvidia.com/blog/even-easier-introduction-cuda/

- Morgan, T. P. (2022, September 21). Deep dive into Nvidia's "Hopper" GPU architecture. The Next Platform. https://www.nextplatform.com/2022/03/31/deep-dive-into-nvidias-hopper-gpu-architecture/
- Arquitectura de GPU NVIDIA Hopper. (n.d.). NVIDIA. https://www.nvidia.com/es-la/data-center/technologies/hopper-architecture/

- García-García, A. (n.d.). Sesión 1: Inicios y Evolución · cuda. https://blitzman.gitbooks.io/cuda/content/problemas-1-c%C3%A1lculo-de-rendimientos.html

- Rao, R. (2024, February 6). Understanding Nvidia CUDA Cores: A Comprehensive guide. Wevolver. https://www.wevolver.com/article/understanding-nvidia-cuda-cores-a-comprehensive-guide