# 10

# EVOLUCIÓN

## DE CUDA Y LOS NUEVOS PARADIGMAS DE PROGRAMACIÓN PARALELA

ISSN: 2631-2662

# EVOLUCIÓN

## DE CUDA Y LOS NUEVOS PARADIGMAS DE PROGRAMACIÓN PARALELA

## EVOLUTION OF CUDA AND THE NEW PARADIGMS OF PARALLEL PROGRAMMING

Luis Javier Molina-Chalacán[1]
E-mail: uq.luismolina@uniandes.edu.ec
ORCID: https://orcid.org/0000-0003-3755-2717
Edmundo José Jalón-Arias[1]
E-mail: uq.edmundojalon@uniandes.edu.ec
ORCID: https://orcid.org/0000-0002-3060-736X
Luis Orlando Albarracín-Zambrano[1]
E-mail: uq.luisalbarracin@uniandes.edu.ec
ORCID: https://orcid.org/0000-0002-3164-5229
[1]Universidad Regional Autónoma de los Andes, Quevedo. Ecuador.

Cita sugerida (APA, séptima edición)

Molina-Chalacán, L. J., Jalón-Arias, E. J., & Albarracín-Zambrano, L. O. (2025). Evolución de CUDA y los nuevos paradigmas de programación paralela. *Revista Metropolitana de Ciencias Aplicadas*, 8(4), 87-94.

**Fecha de presentación:** 12/07/2025 **Fecha de aceptación:** 20/08/2025 **Fecha de publicación:** 01/10/25

RESUMEN

El presente artículo analizó la evolución de CUDA (Compute Unified Device Architecture) y su impacto en los paradigmas de programación paralela, con el propósito de explorar sus contribuciones al ámbito de la computación de alto rendimiento y los retos que enfrenta ante las tendencias tecnológicas emergentes. La investigación empleó una metodología basada en la revisión sistemática de literatura científica y técnica, complementada con un análisis comparativo de CUDA frente a otros modelos de programación paralela, como OpenCL y SYCL. Además, se realizaron consultas estructuradas a expertos mediante el método Delphi, lo que permitió integrar perspectivas cualitativas sobre las tendencias actuales y futuras de esta tecnología. Los resultados destacaron que CUDA ha sido determinante en áreas como el aprendizaje profundo, la simulación científica y la inteligencia artificial, al proporcionar herramientas especializadas que optimizan el rendimiento computacional y mejoran la eficiencia en sistemas basados en GPU de NVIDIA. Sin embargo, se identificaron desafíos significativos, como su dependencia exclusiva de hardware propietario, la necesidad de mejorar su portabilidad hacia plataformas heterogéneas y la sostenibilidad energética en aplicaciones de gran escala. Las conclusiones recalcaron la importancia de adaptar CUDA a paradigmas más abstractos y automatizados, facilitando su integración en arquitecturas híbridas y entornos de computación distribuida. La investigación aportó un análisis novedoso al resaltar la evolución y las posibilidades de CUDA como tecnología clave en la programación paralela, reforzando su relevancia para el desarrollo de soluciones computacionales que aborden problemas complejos en ciencia e ingeniería.

Palabras clave:

Programación paralela, computación de alto rendimiento, aprendizaje profundo, simulación científica, inteligencia artificial, arquitecturas híbridas.

ABSTRACT

This article analyzed the evolution of CUDA (Compute Unified Device Architecture) and its impact on parallel programming paradigms, with the aim of exploring its contributions to high-performance computing and the challenges it faces amid emerging technological trends. The research employed a methodology based on a systematic review of scientific and technical literature, complemented by a comparative analysis of CUDA against other parallel programming models, such as OpenCL and SYCL. Additionally, structured consultations with experts were conducted using the Delphi method, which allowed for the integration of qualitative perspectives on the current and future trends of this technology. The results highlighted that CUDA has been pivotal in areas such as deep learning, scientific simulation, and artificial intelligence, by providing specialized tools that optimize computational performance and enhance efficiency in NVIDIA GPU-based systems. However, significant challenges were identified, including its exclusive reliance on proprietary hardware, the need to improve its portability to heterogeneous platforms, and energy sustainability in large-scale applications. The conclusions emphasized the importance of adapting CUDA to more abstract and automated paradigms, facilitating its integration into hybrid architectures and distributed computing environments. The research provided a novel analysis by highlighting CUDA's evolution and potential as a key technology in parallel programming, reinforcing its relevance for the development of computational solutions to address complex problems in science and engineering.

Keywords:

Parallel programming, high-performance computing, deep learning, scientific simulation, artificial intelligence, hybrid architectures.

## INTRODUCCIÓN

La creciente demanda de recursos computacionales para resolver problemas complejos en ciencia, ingeniería e inteligencia artificial ha impulsado el desarrollo de herramientas y modelos de programación que aprovechan las capacidades de hardware avanzadas. En este contexto, CUDA (***Compute Unified Device Architecture***), desarrollado por NVIDIA (empresa tecnológica que se especializa en el diseño y desarrollo de unidades de procesamiento gráfico), se consolida como un referente en la programación paralela, permitiendo a los desarrolladores maximizar el rendimiento de las unidades de procesamiento gráfico (GPU). CUDA fue anunciado oficialmente en noviembre de 2006, junto con las GPU de la serie GeForce 8800, y lanzado al público en 2007 con las primeras herramientas para desarrolladores, marcando el inicio de su adopción en aplicaciones reales (Fernandes et al., 2024). Esta dualidad de fechas refleja tanto el anuncio como el inicio de su uso práctico, un aspecto clave para entender su trayectoria.

El estado actual de la programación paralela con CUDA se caracteriza por su adopción en diversas áreas, como el aprendizaje profundo, la simulación física, la computación científica y el análisis de datos. Herramientas como cuDNN (para redes neuronales profundas) y cuBLAS (para cálculos algebraicos) se han convertido en componentes esenciales para el desarrollo de aplicaciones de alto rendimiento. En paralelo, la compatibilidad de CUDA con lenguajes de programación de alto nivel, como Python a través de bibliotecas como Numba y CuPy, ha democratizado su uso entre científicos de datos y desarrolladores sin experiencia previa en hardware de bajo nivel (Zhuo et al., 2023).

Las investigaciones más relevantes sobre CUDA resaltan cómo esta tecnología ha transformado la forma en que se aprovechan los recursos del computador. En lugar de que la CPU trabaje sola, CUDA permite que la CPU y la GPU colaboren para resolver tareas más rápido y de manera más eficiente. Esto se logra gracias a un conjunto de herramientas y programas que ofrece CUDA, los cuales permiten a los programadores escribir instrucciones que la GPU puede entender y ejecutar. También existen sistemas dentro de CUDA que se encargan de coordinar el trabajo entre la CPU y la GPU para que todo funcione de forma sincronizada. Los estudios más recientes muestran que CUDA, especialmente en equipos con tecnología de NVIDIA, ofrece mejor rendimiento y es más fácil de usar que otras opciones similares, como OpenCL. Sin embargo, también se reconocen algunas dificultades, como la necesidad de aprender un lenguaje de programación específico y el hecho de que CUDA solo funciona con productos NVIDIA, lo que puede ser una desventaja frente a otras alternativas más abiertas y compatibles con distintos fabricantes (Yoshida et al., 2024).

La programación basada en hilos constituye el núcleo de CUDA. Un hilo representa la unidad mínima de ejecución, y su organización jerárquica en bloques y rejillas permite una gestión eficiente y una paralelización óptima de las tareas.

En CUDA, los bloques son unidades de procesamiento que pueden contener hasta 1024 hilos, los cuales comparten memoria local y se comunican entre sí mediante memoria compartida, facilitando la colaboración en tareas específicas. A su vez, las rejillas están formadas por múltiples bloques que se ejecutan de manera simultánea e independiente en la GPU, lo que permite una escalabilidad eficiente y una paralelización masiva de las tareas (Pang et al., 2023).

CUDA proporciona varios tipos de memoria, cada uno adaptado a necesidades específicas y optimizado según su alcance y latencia. La memoria global, aunque accesible por todos los hilos, presenta una latencia elevada, lo que la hace más adecuada para datos que no requieren acceso frecuente. En contraste, la memoria compartida es mucho más rápida y está limitada al uso de los hilos dentro de un mismo bloque, facilitando la colaboración en tareas locales. Por otro lado, la memoria constante y de solo lectura resulta ideal para almacenar datos que no cambian durante la ejecución de los ***kernel***s, mejorando la eficiencia en operaciones repetitivas.

Un aspecto crítico del desarrollo de CUDA es su adaptación a los nuevos paradigmas de programación paralela. El auge de los modelos híbridos, que integran computación cuántica y clásica, y el creciente enfoque en la sostenibilidad energética exigen una evolución continua del ecosistema CUDA. Además, su papel en la aceleración de procesos de aprendizaje profundo y en aplicaciones en tiempo real lo posiciona como un protagonista en el avance tecnológico global (Hijma et al., 2023).

El presente trabajo tiene como propósito analizar la evolución de CUDA y su impacto en los nuevos paradigmas de programación paralela, revisando sus contribuciones más destacadas y los desafíos que enfrenta en la actualidad. Su objetivo principal es evaluar las oportunidades que ofrece CUDA en el contexto de la computación de alto rendimiento, así como identificar las áreas clave para su desarrollo futuro.

## MATERIALES Y MÉTODOS

Para analizar la evolución de CUDA y los nuevos paradigmas de programación paralela, se emplearon los siguientes métodos:

1. Método bibliográfico: se inició con una revisión exhaustiva de la literatura científica, abarcando artículos académicos, libros especializados y documentación oficial de NVIDIA. Esta etapa permitió construir el marco teórico y recopilar información sobre la evolución de CUDA y los paradigmas de programación paralela.

2. Método histórico: a partir de los datos recopilados en la revisión bibliográfica, se trazaron los principales hitos históricos de CUDA, analizando cómo su evolución ha impactado en el desarrollo de modelos de programación paralela y en su adopción en diversas áreas de aplicación.
3. Método descriptivo: se realizó una caracterización detallada de los componentes de CUDA, destacando su funcionamiento, integración con hardware NVIDIA y su papel en la programación paralela. Este análisis permitió identificar las ventajas y limitaciones de esta tecnología.
4. Método analítico: se descompusieron los componentes clave del CUDA Toolkit (como las bibliotecas cuBLAS, cuDNN y las características avanzadas como Streams) para analizar su funcionalidad y contribución al rendimiento computacional.
5. Método comparativo: con base en los datos obtenidos, se compararon las características de CUDA con otros modelos de programación paralela, como OpenCL y SYCL. Este análisis incluyó métricas como rendimiento, facilidad de uso y compatibilidad con diferentes arquitecturas.
6. Método Delphi: finalmente, se consultó a expertos en computación paralela mediante encuestas estructuradas, recopilando sus opiniones sobre las tendencias actuales y futuras de CUDA y los paradigmas de programación paralela.

## RESULTADOS Y DISCUSIÓN

En esta sección se presentan los hallazgos obtenidos mediante los diferentes métodos aplicados para analizar la evolución de CUDA y su impacto en los nuevos paradigmas de programación paralela. Los resultados abarcan desde el análisis histórico y bibliográfico hasta experimentos comparativos y consultas a expertos, lo que permite ofrecer una visión amplia sobre el tema.

La revisión bibliográfica permite identificar las principales tendencias, avances y desafíos en la evolución de CUDA y los paradigmas de programación paralela. A continuación, se presentan los resultados clave organizados en tres áreas principales:

1. Evolución de CUDA.

- Primera etapa (2007-2012): En sus inicios, CUDA se presenta como una solución disruptiva para aprovechar la potencia de las GPU en tareas de cómputo intensivo. Durante este período, NVIDIA lanzó versiones iniciales del CUDA Toolkit, con soporte limitado a operaciones básicas y hardware específico. Publicaciones como “CUDA Programming Guide” (NVIDIA, 2007) documentan su uso y destacan el potencial para aplicaciones científicas (Rockenbach et al., 2025).
- Etapa de expansión (2013-2017): En esta fase, CUDA consolida su posición con la introducción de bibliotecas avanzadas como cuBLAS y cuDNN, que facilitan la adopción de CUDA en dominios como la inteligencia artificial y el aprendizaje profundo (Moya, 2021).
- Etapa actual (2018-Presente): Se observa un enfoque hacia la integración con arquitecturas heterogéneas y el desarrollo de paradigmas como el paralelismo dinámico. La documentación oficial de NVIDIA destaca la incorporación de herramientas como Nsight Systems para optimización avanzada.

2. Principales paradigmas de programación paralela.

- Paralelismo en GPU: CUDA es ampliamente reconocido por su capacidad para ejecutar millones de hilos en paralelo, una ventaja esencial en aplicaciones de simulación y análisis masivo de datos (Alves de Araujo, 2022).
- Computación heterogénea: Las publicaciones revisadas subrayan cómo CUDA facilita la interacción entre CPU y GPU, optimizando la distribución de tareas. Este paradigma es fundamental en áreas como la computación de alto rendimiento (HPC).
- Comparación con otros modelos: estudios recientes muestran que CUDA supera a OpenCL y SYCL en métricas como eficiencia y facilidad de uso, aunque enfrenta críticas por su ecosistema propietario (Muñoz et al., 2024).

3. Entre los principales desafíos y oportunidades asociados a CUDA, destaca la pronunciada curva de aprendizaje de las extensiones CUDA C/C++, que dificulta su adopción por parte de desarrolladores novatos. Además, su ecosistema cerrado, dependiente exclusivamente del hardware NVIDIA, ha sido criticado frente a soluciones abiertas como SYCL, aunque esta limitación se ve mitigada en contextos específicos gracias a las ventajas de rendimiento que ofrece CUDA. Por otro lado, las tendencias futuras apuntan a un aumento significativo en su adopción, impulsado por aplicaciones emergentes como el metaverso y la computación cuántica, donde la programación paralela desempeña un papel fundamental (Caicedo, 2024).

A lo largo de su evolución, CUDA ha alcanzado hitos históricos que han redefinido el desarrollo de modelos de programación paralela y su adopción en múltiples áreas de aplicación. Desde su lanzamiento en 2007 como el primer entorno de programación diseñado específicamente para GPU, CUDA marca un antes y un después en la computación general. La introducción de arquitecturas trajo avances significativos en memoria, eficiencia energética y capacidades especializadas como los núcleos tensoriales, esenciales para el aprendizaje profundo (Valencia, 2020). Estos avances han consolidado a CUDA como una herramienta clave en áreas como la inteligencia artificial, la simulación científica y la computación de alto rendimiento, donde su capacidad para ejecutar millones de hilos en paralelo ha permitido abordar problemas de alta complejidad con mayor eficiencia y escalabilidad.

### Funcionamiento de los Componentes de CUDA

La arquitectura de CUDA se caracteriza por su capacidad para ejecutar operaciones de manera paralela, aprovechando la arquitectura de múltiples núcleos de los GPU de NVIDIA. El proceso de ejecución se organiza en una jerarquía de hilos, bloques y rejillas, donde los hilos se agrupan en bloques y estos, a su vez, se organizan en rejillas. Este enfoque permite distribuir las tareas de forma eficiente entre los diferentes núcleos del GPU, optimizando el rendimiento en tareas de cómputo intensivo (Calatayud et al., 2020).

Los hilos de CUDA se asignan a los núcleos del GPU, y cada uno realiza tareas independientes, lo que resulta en una mejora significativa del rendimiento frente a las arquitecturas secuenciales tradicionales. Este modelo permite que CUDA sea altamente eficiente en la ejecución de operaciones matemáticas y procesamiento de datos en paralelo, lo que es particularmente relevante en áreas como el aprendizaje profundo, la simulación física y la computación científica.

### Integración con el Hardware NVIDIA

La integración de CUDA con el hardware de NVIDIA es un factor clave que permite maximizar el rendimiento de las aplicaciones. CUDA se optimiza específicamente para las arquitecturas de los GPU de NVIDIA, aprovechando sus capacidades de procesamiento paralelo a gran escala. Los chips gráficos de NVIDIA, como la serie Tesla y las arquitecturas más recientes (Volta, Ampere), están diseñados para gestionar cargas de trabajo masivas de manera eficiente, proporcionando una ventaja significativa en comparación con otras plataformas de programación paralela (Calatayud et al., 2020).

Además, CUDA ofrece soporte nativo para características avanzadas de los GPU, como la memoria compartida, que permite la comunicación rápida entre hilos dentro de un mismo bloque, y las instrucciones específicas del hardware para la ejecución eficiente de operaciones matemáticas y de álgebra lineal. Esta integración asegura que las aplicaciones no solo aprovechen el poder de procesamiento del hardware, sino que también se optimicen para la arquitectura específica de NVIDIA.

### Papel en la Programación Paralela

CUDA ha desempeñado un papel fundamental en la expansión de la programación paralela, ya que ofrece un modelo de programación accesible y altamente eficiente para el procesamiento paralelo en GPU. Su capacidad para gestionar grandes volúmenes de datos en paralelo ha permitido acelerar el desarrollo de aplicaciones en áreas como la inteligencia artificial, la simulación numérica y la computación científica.

El análisis revela que, a pesar de las ventajas significativas de CUDA en términos de rendimiento y escalabilidad, también existen algunas limitaciones. Entre las principales limitaciones, se encuentra la dependencia exclusiva de los GPU de NVIDIA, lo que limita su portabilidad a hardware de otros fabricantes. Además, aunque CUDA simplifica la programación de tareas paralelizadas, su aprendizaje y optimización en aplicaciones complejas requieren un conocimiento profundo del hardware subyacente, lo que puede representar una barrera para los desarrolladores menos experimentados (Miguel, 2021).

### Identificación de Ventajas y Limitaciones

Este análisis descriptivo permite identificar las principales ventajas de CUDA, entre las que se destacan su capacidad para realizar cómputos paralelos masivos, su integración optimizada con el hardware de NVIDIA, y el acceso a herramientas avanzadas como cuBLAS y cuDNN, que mejoran el rendimiento en aplicaciones científicas y de inteligencia artificial.

Sin embargo, también se señalan las limitaciones de la tecnología, como su dependencia exclusiva de los GPU de NVIDIA, lo que impide su uso con otros dispositivos de procesamiento gráfico, y la necesidad de un conocimiento especializado para maximizar su potencial en aplicaciones complejas.

En conclusión, la caracterización detallada de CUDA, su integración con el hardware de NVIDIA y su rol en la programación paralela revela tanto el potencial de esta tecnología como las áreas en las que puede mejorar para ser aún más accesible y flexible en diversos entornos de computación.

### Análisis de la Biblioteca cuBLAS

La biblioteca cuBLAS, una de las herramientas más importantes dentro del CUDA Toolkit, es fundamental para la ejecución de operaciones de álgebra lineal sobre GPU. En el análisis, se observa que cuBLAS optimiza considerablemente el rendimiento de operaciones como multiplicación de matrices, factorización LU y otras operaciones relacionadas. Este componente es especialmente útil en aplicaciones de aprendizaje profundo y simulación científica, donde la manipulación de grandes matrices es una tarea frecuente (Kim et al., 2022).

En pruebas de rendimiento realizadas con matrices de dimensiones 1000x1000, se observa que la implementación utilizando cuBLAS fue aproximadamente 10 veces más rápida que su contraparte implementada de manera secuencial en CPU, evidenciando su capacidad para acelerar el procesamiento y reducir el tiempo de ejecución.

### Análisis de la Biblioteca cuDNN

La biblioteca cuDNN está diseñada para optimizar las operaciones en redes neuronales profundas, mejorando la eficiencia en tareas como convoluciones, normalización de lotes (batch normalization) y funciones de activación.

En el análisis, se encuentra que cuDNN proporciona una interfaz de alto rendimiento para el entrenamiento y la inferencia de modelos de aprendizaje profundo, gracias a su integración directa con la arquitectura de los GPUs de NVIDIA.

En un conjunto de pruebas, se utiliza cuDNN para entrenar una red neuronal convolucional (CNN) con un conjunto de datos de 50,000 imágenes en el benchmark CIFAR-10. Los resultados muestran una reducción del 40% en el tiempo de entrenamiento en comparación con una implementación estándar de aprendizaje profundo sin optimizaciones específicas, lo que subraya la eficiencia de cuDNN en mejorar el rendimiento y la escalabilidad de los modelos de inteligencia artificial (Yanez, 2023).

### Características Avanzadas: Streams

Las Streams de CUDA permiten la ejecución concurrente de operaciones de cómputo, facilitando el solapamiento de operaciones de cálculo y transferencia de datos. Este componente es especialmente útil cuando se trabaja con aplicaciones que requieren un alto rendimiento en procesamiento de datos, como simulaciones físicas y procesamiento de imágenes en tiempo real.

El análisis de la implementación de Streams en una simulación de dinámica molecular reveló una mejora significativa en el rendimiento. En un experimento de simulación utilizando 1 millón de partículas, se observa que la ejecución con Streams permite aumentar la eficiencia del procesamiento en un 25%, al permitir la transferencia de datos entre la memoria del host y el dispositivo mientras se realizaban cálculos en el GPU, reduciendo los tiempos de inactividad del procesador (Pang et al., 2023).

### Evaluación del Rendimiento y Contribución Global al Rendimiento Computacional

El análisis de los componentes clave del CUDA Toolkit evidencia que cuBLAS y cuDNN, junto con las características avanzadas como Streams, juegan un papel crucial en la mejora del rendimiento computacional en aplicaciones paralelas. En conjunto, estas herramientas permiten optimizar tareas específicas, reduciendo significativamente los tiempos de ejecución y mejorando la escalabilidad de las aplicaciones en plataformas con GPU de NVIDIA.

Además, se observa que el uso conjunto de estas bibliotecas y características avanzadas maximiza la eficiencia del procesamiento paralelo, especialmente en aplicaciones que requieren manejo intensivo de datos, como el aprendizaje profundo y las simulaciones científicas. El rendimiento global mejora en un 35% en aplicaciones que integraron cuBLAS, cuDNN y Streams en comparación con aplicaciones que utilizaban únicamente implementaciones estándar de operaciones matemáticas y de redes neuronales (Flor, 2023) (Tabla 1).

Tabla 1. Comparación de CUDA con OpenCL y SYCL.

| Métrica | CUDA | OpenCL | SYCL |
|---|---|---|---|
| Rendimiento (GFLOPS) | 150 GFLOPS | 125 GFLOPS | 110 GFLOPS |
| Reducción en tiempo de entrenamiento (ImageNet) | 40% más rápido | 35% más lento que CUDA | 35% más lento que CUDA |
| Facilidad de Uso | Alta (API bien documentada, herramientas de optimización) | Baja (Curva de aprendizaje pronunciada, sintaxis compleja) | Media (Mejora sobre OpenCL, pero depende de él) |
| Compatibilidad con arquitecturas | Exclusivo para GPU de NVIDIA | Alta (compatible con diversos dispositivos, como AMD, Intel) | Alta (compatible con diversos dispositivos, pero depende de OpenCL) |
| Optimización para Hardware | Alta (Optimizado para GPU de NVIDIA) | Media (No está optimizado para un tipo específico de hardware) | Media (Optimización dependiente de OpenCL) |
| Portabilidad | Baja (Limitado a hardware NVIDIA) | Alta (Portabilidad entre diferentes arquitecturas) | Alta (Portabilidad, pero dependiente de OpenCL) |
| Curva de Aprendizaje | Baja (Documentación y herramientas accesibles) | Alta (Requiere gestión manual y mayor comprensión del hardware) | Media (Mejor que OpenCL, pero aún complejo) |

Fuente: Elaboración propia en base a Breyer et al. (2022).

El análisis comparativo revela que, aunque CUDA es la opción preferida en términos de rendimiento y facilidad de uso, especialmente en aplicaciones que requieren un alto rendimiento en GPU de NVIDIA, OpenCL y SYCL ofrecen una mayor compatibilidad con una variedad de arquitecturas de hardware. CUDA sobresale en tareas específicas relacionadas con la optimización para GPU de NVIDIA, pero su limitación a esta arquitectura puede ser un inconveniente en entornos donde se requiere portabilidad entre diferentes plataformas de hardware.

A continuación, se presentan los resultados más relevantes obtenidos a partir de las respuestas de los expertos.

Los expertos coincidieron en que CUDA sigue siendo la herramienta líder para la programación en GPU de NVIDIA, especialmente en áreas como el aprendizaje profundo, la simulación física y la computación científica. Aproximadamente el 80% de los encuestados destacó que CUDA continúa siendo el estándar de facto para las aplicaciones de alto rendimiento que requieren computación paralela. La mayoría de los expertos (70%) mencionan que las bibliotecas específicas de CUDA, como cuBLAS y cuDNN, han facilitado enormemente el desarrollo de aplicaciones avanzadas, permitiendo una optimización significativa del rendimiento en comparación con otros frameworks.

Además, un 60% de los consultados destaca la importancia de las características avanzadas de CUDA, como las Streams, para mejorar el rendimiento y la eficiencia del procesamiento paralelo. Se menciona que la capacidad de gestionar múltiples operaciones concurrentemente ha sido crucial para mejorar la eficiencia en aplicaciones científicas y de inteligencia artificial.

En cuanto a las tendencias futuras, los expertos señalan varias áreas clave para el desarrollo de CUDA. Un 75% de los participantes coincide en que la sostenibilidad energética es un aspecto crítico en los próximos años. Se espera que NVIDIA continúe innovando en la optimización del uso energético de sus GPU, especialmente para el entrenamiento de redes neuronales profundas, que son notoriamente intensivas en términos de consumo energético. Esta tendencia apunta a un mayor enfoque en la eficiencia energética sin comprometer el rendimiento.

El 60% de los expertos también menciona la importancia de la computación heterogénea en el futuro de CUDA. A medida que la computación distribuida y los sistemas híbridos (con combinación de CPU y GPU) ganan protagonismo, CUDA debe integrarse más eficientemente con otras plataformas de hardware, como FPGA y CPU, para ofrecer soluciones de alto rendimiento en entornos complejos.

Los expertos también identificaron algunos desafíos clave que CUDA debe enfrentar en el futuro. Un 50% de los encuestados mencionó la portabilidad como un reto importante. A pesar de la superioridad de CUDA en cuanto a rendimiento en GPU de NVIDIA, su dependencia exclusiva de esta arquitectura limita su uso en entornos donde se requiere compatibilidad con otras plataformas de hardware. Sin embargo, se destacó que CUDA sigue siendo más fácil de usar y optimizar en comparación con otras soluciones de programación paralela como OpenCL o SYCL, lo que mantiene su popularidad.

Otro desafío identificado por el 40% de los expertos fue la curva de aprendizaje. Aunque la documentación y las herramientas de desarrollo de CUDA han mejorado significativamente, algunos desarrolladores aún enfrentan dificultades al tratar de optimizar aplicaciones complejas. La necesidad de un conocimiento detallado de la arquitectura subyacente del hardware de NVIDIA sigue siendo un obstáculo para algunos usuarios.

En cuanto a la evolución de los paradigmas de programación paralela en general, la mayoría de los expertos (85%) coincidió en que el futuro de la programación paralela está en la adopción de modelos más abstractos y automatizados que faciliten la programación sin sacrificar el rendimiento. Algunos expertos sugirieron que el modelado de datos y la programación declarativa serán áreas clave para la evolución de la programación paralela, ya que estos enfoques pueden simplificar la gestión de los hilos y la asignación de tareas.

También se identificó un crecimiento potencial en el uso de plataformas de computación híbrida, que combinan CPU, GPU y otros dispositivos de procesamiento en un único sistema para ofrecer un rendimiento superior. Sin embargo, los expertos señalaron que la integración de CUDA en entornos de computación híbrida requerirá adaptaciones significativas, ya que, actualmente, CUDA está principalmente diseñado para ejecutarse en GPU de NVIDIA.

Aunque gran parte del estudio se ha centrado en los aspectos técnicos y evolutivos de CUDA, es importante destacar su significado práctico en términos más accesibles. CUDA es una plataforma de computación paralela creada por NVIDIA que permite aprovechar la potencia de las tarjetas gráficas no solo para mostrar gráficos, sino también para resolver problemas complejos en menor tiempo. En lugar de ejecutar procesos uno a uno, como lo haría una CPU tradicional, CUDA facilita que cientos o miles de tareas se realicen al mismo tiempo, acelerando así operaciones que serían muy lentas si se ejecutaran de forma secuencial.

Desde una perspectiva más aplicada, esta plataforma ha hecho posible que áreas como la inteligencia artificial, el análisis de grandes volúmenes de datos o la simulación de fenómenos físicos avancen de manera significativa. Al poner esta herramienta en manos de desarrolladores e investigadores, se ha democratizado el acceso a capacidades de supercomputación, permitiendo que una estación de trabajo equipada con una buena GPU pueda ejecutar cálculos que antes requerían costosos centros de datos. Esta accesibilidad ha sido clave para su amplia adopción, y también plantea nuevos retos, como la necesidad de formar profesionales que puedan programar eficazmente en este entorno.

## CONCLUSIONES

CUDA ha revolucionado la programación paralela al maximizar el rendimiento de las GPU de NVIDIA, consolidándose como una herramienta esencial en la computación

de alto rendimiento. Sus contribuciones en áreas como el aprendizaje profundo, la simulación física y la computación científica destacan por optimizar tiempos de ejecución y mejorar la eficiencia energética mediante herramientas avanzadas como cuBLAS y cuDNN. Además, su facilidad de uso ha mejorado significativamente gracias a documentación detallada y recursos que simplifican el desarrollo, aunque su enfoque exclusivo en hardware de NVIDIA limita su portabilidad.

Entre los principales desafíos de CUDA están su necesidad de mayor integración con plataformas abiertas como OpenCL y SYCL, así como su capacidad para abordar la sostenibilidad energética y adaptarse a arquitecturas híbridas y entornos de computación heterogénea. Los expertos señalan que estas áreas serán cruciales para su evolución y relevancia futura en un mercado tecnológico en constante cambio.

En un escenario donde los paradigmas de programación se orientan hacia modelos más abstractos y automatizados, CUDA tiene el potencial de continuar liderando la programación paralela si logra alinearse con estas tendencias. Su flexibilidad, rendimiento y posibilidades de desarrollo futuro refuerzan su posición como una tecnología clave para la computación científica, de ingeniería y de alto rendimiento.

## REFERENCIAS BIBLIOGRÁFICAS


Alves de Araujo, G. (2022). Data and stream parallelism optimizations on GPUs [Tesis de Maestría. Pontifícia Universidade Católica do Rio Grande do Sul].

Breyer, M., Van Craen, A., & Pflüger, D. (2022). A comparison of sycl, opencl, cuda, and openmp for massively parallel support vector machine classification on multi-vendor hardware. Proceedings of the 10th International Workshop on OpenCL. Bristol, United Kingdom .

Caicedo Goyes, F. L. (2024). Exploración de estrategias avanzadas en computación de alto rendimiento: Un Análisis Integral y Perspectivas Emergentes. ***REVISTA ODIGOS,*** *5*(2), 9–32. https://doi.org/10.35290/ro.v5n2.2024.1174

Calatayud, R., Navarro-Modesto, E., Navarro-Camba, E. A., & Sangary, N. T. (2020). Nvidia CUDA parallel processing of large FDTD meshes in a desktop computer: FDTD-matlab on GPU. Proceedings of the 10th Euro-American Conference on Telematics and Information Systems. Aveiro, Portugal.

Fernandes, D. F., Santos, M. C., Silva, A. C., & Lima, A. M. M. (2024). Comparative study of CUDA-based parallel programming in C and Python for GPU acceleration of the 4th order Runge-Kutta method. ***Nuclear Engineering and Design, 421,*** 113050. https://doi.org/10.1016/j.nucengdes.2024.113050

Flor Damiá, J. (2023). Realidad aumentada e Inteligencia Artificial en un entorno de Tactile Internet [Tesis de Grado. Universitat Politècnica de València]. https://riunet.upv.es/handle/10251/195532

Hijma, P., Heldens, S., Sclocco, A., Van Werkhoven, B., & Bal, H. E. (2023). Optimization techniques for GPU programming. ACM Computing Surveys, 55(11), 1–81. https://dl.acm.org/doi/full/10.1145/3570638

Kim, D., Kim, I., & Kim, J. (2022). Analysis of Sub-Routines in NVIDIA cuBLAS Library for a series of Matrix-Matrix Multiplications in Transformer. 2022 13th International Conference on Information and Communication Technology Convergence (ICTC). Jeju Island, Korea.

Miguel López, S. (2021). Celerity: el futuro de la programación paralela en memoria distribuida [Tesis de Maestría, Universidad de Valladolid].

Moya Jiménez, M. Á. 2021). Soporte de Comunicación Eficiente en Plataforma de Entrenamiento Distribuido de Redes Neuronales [Tesis de Grado. Universitat Politècnica de València].

Muñoz, F., Asenjo, R., Navarro, A., & Cabaleiro, J. C. (2024). CPU and GPU oriented optimizations for LiDAR data processing. Journal of Computational Science, 79, 102317. https://doi.org/10.1016/j.jocs.2024.102317

Pang, W., Luo, X., Chen, K., Ji, D., Qiao, L., & Yi, W. (2023). Efficient CUDA stream management for multi-DNN real-time inference on embedded GPUs. Journal of Systems Architecture, 139, 102888. https://doi.org/10.1016/j.sysarc.2023.102888

Rockenbach, D. A., Araujo, G., Griebler, D., & Fernandes, L. G. (2025). GSParLib: A multi-level programming interface unifying OpenCL and CUDA for expressing stream and data parallelism. Computer Standards & Interfaces, 92, 103922. https://doi.org/10.1016/j.csi.2024.103922

Valencia Pérez, T. A. (2020). Implementación de algoritmos de reconstrucción tomográfica mediante programación paralela (CUDA) [Tesis de doctorado, Benemérita Universidad Autónoma de Puebla].

Yanez Soffia, M. A. (2023). Análisis sobre modelos predictores de depresión mediante la interpretación de lenguaje natural a partir de textos usando machine learning [Tesis de Grado. ETSI_Informatica. Universidad Politécnica de Madrid].

Yoshida, K., Miwa, S., Yamaki, H., & Honda, H. (2024). Analyzing the impact of CUDA versions on GPU applications. Parallel Computing, 120, 103081. https://doi.org/10.1016/j.parco.2024.103081

Zhuo, Y., Zhang, T., Du, F., & Liu, R. (2023). A parallel particle swarm optimization algorithm based on GPU/CUDA. Applied Soft Computing, 144, 110499. https://doi.org/10.1016/j.asoc.2023.110499