COMPUTACIÓN HETEROGÉNEA: Un Viaje Hacia la Eficiencia y el Alto Rendimiento

Andrés Moyano, Facundo M. Cabral, y Federico N. Pagnotta Lemes,
Estudiantes, Facultad de Ingeniería, Universidad de Buenos Aires

***Abstract*—La computación heterogénea es el paradigma actual de la computación moderna, aún así sigue siendo un campo abierto a la innovación. En este documento, nos centraremos en el desarrollo de la Inteligencia Artificial, una de las áreas más prometedoras del presente, donde la computación heterogénea cumple un rol fundamental. Además, enumeraremos los tipos de procesadores que se utilizan y sus enfoques actuales, como así también las innovaciones necesarias para los usos especializados que requiere este campo de emergente demanda. Finalmente, profundizaremos en los desafíos asociados y las dificultades inherentes a este paradigma, con un enfoque en aspectos críticos como la complejidad en el desarrollo, la carencia de estándares en la industria, el mantenimiento de la coherencia de la caché y los costos de implementación y mantenimiento.**

***Índice de Términos*—Computación Heterogénea, ASIC, GPU, CPU, FPGA, Deep Learning, CNN, TPU.**

I. INTRODUCCIÓN

Con el paso de los años, el aumento de rendimiento en los sistemas se ha visto cada vez más limitado. La Ley de Moore[1], quedó obsoleta, pues el aumento de transistores ha alcanzado un tope, dado que a medida que el tamaño de los mismos se acercó a los límites atómicos, la disipación de calor y el consumo energético se han convertido en obstáculos críticos para el desarrollo de procesadores más potentes. Una alternativa para seguir aumentando la potencia de las máquinas fue la utilización de procesadores multinúcleo. Pero esto no es lo más idóneo, pues las restricciones energéticas hacen que no puedan funcionar a su máximo rendimiento en simultáneo. [1]

Es por esto que la computación heterogénea se posiciona como la mejor opción para aumentar el rendimiento a la vez que se mejora la eficiencia energética, utilizando aceleradores, donde cada uno está especializado en distintas tareas, pudiendo realizarlas de la mejor forma tanto en términos de rendimiento como energéticos. Es decir que el trabajo se divide entre distintos procesadores que están especializados en tareas específicas. [2]

Las CPU y GPU han demostrado ser herramientas poderosas para una gran cantidad de propósitos. Sin embargo ¿Qué ocurre cuando se necesita mejorar el rendimiento en una tarea específica que hace casi llegar a límites teóricos a estos procesadores? La respuesta son los aceleradores, como las ASIC (Application-Specific Integrated Circuits) y las FPGA (Field-Programmable Gate Arrays), diseñadas para realizar tareas específicas y para ser reprogramadas en tiempo real, respectivamente.

[1] La ley de Moore expresa que aproximadamente cada 2 años se duplica el número de transistores en un microprocesador.

II. ANÁLISIS DE COMPONENTES

A. CPU [3]

La CPU, como indica su nombre, es un procesador de uso general. Es el "cerebro" de la computadora, se encarga de la ejecución de las instrucciones de los programas.

Está formado por registros, la ALU, y una Unidad de Control que se encarga de coordinar a todos los componentes.

Para aumentar el rendimiento de la CPU se introdujeron los procesadores multinúcleo, para poder dividir la ejecución de tareas, así como distintos tipos de paralelismo:

1) **Paralelismo multi-chip**
   Consiste en tener varios procesadores físicos en una computadora, que comparten recursos, en especial memoria de sistema, a través de la cual se realiza una comunicación relativamente barata.
2) **Paralelismo multi-core**
   Los núcleos están en un mismo chip, por lo que estos pueden compartir recursos utilizando la caché del propio chip. Esto hace que el paralelismo sea aún menos costoso.
3) **Paralelismo multi-hilo**
   Ocurre en un mismo núcleo, cuando este puede cambiar entre múltiples contextos de ejecución con un costo casi nulo.
4) **Paralelismo de instrucción**
   Se da cuando un procesador puede ejecutar más de una instrucción en paralelo, utilizando varias unidades de instrucción.

Las CPUs multinúcleo actuales utilizan la mayor parte de sus transistores en lógica y caché, gastando mucha energía en unidades no computacionales. Las arquitecturas heterogéneas ofrecen una alternativa a esta estrategia, combinando arquitecturas multinúcleo con aceleradores, que maximizan el rendimiento utilizando un presupuesto de energía o de transistores fijo. Por lo general esto implica que los aceleradores utilicen menos transistores y corran a frecuencias menores que las CPUs tradicionales. También se sacrifica la funcionalidad compleja, lo que deshabilita su capacidad para ejecutar sistemas operativos y, por lo general, son administrados por núcleos tradicionales para descargar operaciones que consumen muchos recursos.

B. GPU

El acelerador más conocido es la GPU (Unidad de

Procesamiento Gráfico). Desde su inicio, ha sido utilizada para cómputos especializados en el dominio gráfico, pero ahora se utiliza en muchas aplicaciones diferentes.

Como vimos, hoy por hoy las CPUs pueden ser consideradas como procesadores multinúcleo (multicore), pues necesitan unos cuantos hilos para desarrollar su máximo potencial, dado que pueden servirse de la ejecución fuera de orden. En cambio, las GPUs son procesadores de muchos núcleos (manycore), ya que necesitan miles de hilos para alcanzar su plenitud. [7]

Esto causa que la CPU tenga poca latencia de memoria, ya que no necesita esperar a que lleguen los datos para seguir ejecutando un proceso. En cambio, la GPU depende de tener hilos liberados. Si esto no sucede, comienza a notarse el tiempo de acceso a memoria. [6]

Es por esto que el paralelismo es crucial para manejar las gigantescas cargas de trabajo que pueden entregarse a la GPU. Esta carga debe ser lo suficientemente grande para esconder la latencia del acceso a memoria. Por lo tanto, un alto throughput[2] es más importante que la latencia en estos procesadores. [7]

Un modelo de Inteligencia Artificial está compuesto de capas sobre capas de ecuaciones de álgebra lineal. Cada una de estas ecuaciones representa la probabilidad de que un dato esté relacionado con otro. Las GPUs, gracias a su paralelización masiva y la enorme cantidad de núcleos que tiene, puede realizar una gran cantidad de cómputos simultáneamente, reduciendo exponencialmente el tiempo necesario para entrenar a estos modelos. [8]

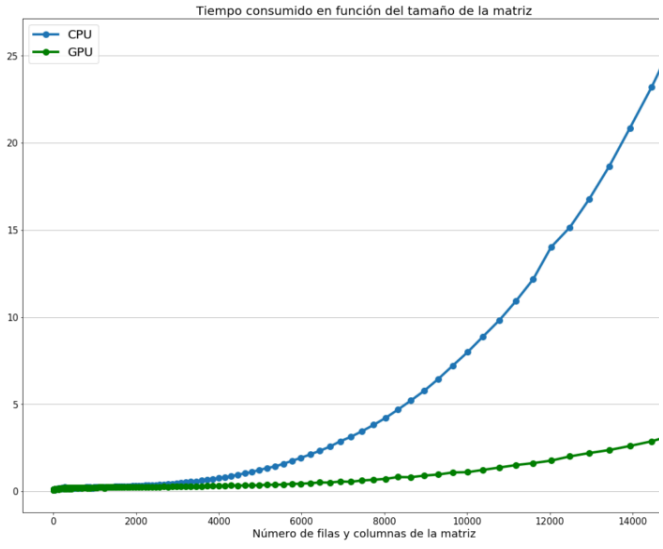

Fig. 1. Comparación entre el rendimiento de una GPU y una CPU en multiplicación de matrices [10]

Debido a la fuerte presencia de estas ecuaciones, en el entrenamiento de modelos de IA lo que más encontramos son operaciones matriciales, especialmente multiplicaciones. La Figura 1 muestra una comparación entre el rendimiento (en segundos) de una CPU AMD Ryzen 5 3600 y una GPU Asus Geforce GTX 1600 al momento de realizar multiplicaciones de matrices, en función del tamaño de las mismas [10]. Las llamadas al kernel de la GPU necesitan algunos pasos previos, por lo que hay un cierto costo involucrado en la ejecución de cada algoritmo de la GPU. Este tiempo prevalece en la ejecución total para matrices más pequeñas, mientras que se amortiza cuando son más grandes [7]. Además, se ve como la latencia de memoria se nota menos cuanto más grande es la carga de trabajo. Es por esto que, en matrices más pequeñas, el tiempo es casi igual, mientras que cuanto más grandes son el tiempo de la CPU aumenta exponencialmente, diferenciándose de la suave pendiente del de la GPU.

C. FPGA

Una FPGA, matriz de puerta programable en campo en español, es un tipo versátil de circuito integrado, que está diseñado para ser programado (y muchas veces reprogramado) para cumplir diferentes propósitos, especialmente computación de alto rendimiento (HPC)[11], que es una tecnología que usa clusters con procesadores muy potentes que trabajan en paralelo para procesar bases de datos masivas y multidimensionales, y resolver problemas complejos con velocidades extremadamente altas. [12]

Esto permite que las FPGA puedan ser adaptadas para muchos casos, sin la necesidad de modificar físicamente su hardware. Esta versatilidad reconfigurable se alcanza a través de un arreglo de bloques lógicos programables e interconexiones flexibles que pueden ser configuradas para resolver operaciones complejas o para servir como simples puertas lógicas. Además, las FPGAs incluyen elementos de memoria, desde flip-flops de un bit hasta arreglos de memoria muy densos.[11]

Aunque las FPGAs no tienen las ventajas de velocidad de operación, menor consumo de energía, y producción en masa más barata de las ASICs, de las que hablaremos a continuación, debido a su programabilidad, las FPGAs tienen ciclos de diseño más cortos, y pueden cubrir un espectro mucho más amplio de aplicaciones que las ASICs.

Las FPGAs pueden alcanzar un paralelismo alto y simplificar la lógica de los procesos de cálculo de una red neuronal modificando el diseño de hardware para modelos específicos. Muchas investigaciones muestran que un modelo de redes neuronales puede ser simplificado en una forma amigable con el hardware sin afectar la precisión del modelo. Es por esto que las FPGAs pueden lograr una mayor eficiencia energética que las CPUs o GPUs. [13]

Esta eficiencia energética frente a la GPU viene con una desventaja en términos de accesibilidad. El ecosistema de desarrollo para las GPUs está mucho más desarrollado que el de las FPGAs. Programar con paralelización para GPUs se ve facilitado por lenguajes de alto nivel como CUDA y OpenCL, o frameworks y bibliotecas como TensorFlow y Pytorch, que ya están optimizados para ejecutarse en GPUs. [13] En cambio, para FPGAs se utilizan lenguajes de descripción de hardware (HDL) como VHDL o Verilog, que requieren de un

[2] El throughput representa la cantidad total de trabajo que un procesador puede realizar por unidad de tiempo.

alto conocimiento de la arquitectura de la FPGA. Hoy por hoy, el diseño HDL se está cambiando a niveles de abstracción más altos, utilizando programación funcional o paradigmas imperativos, con el fin de hacer más accesible el ecosistema de las FPGAs. Todo esto se hace a través de herramientas de síntesis de alto nivel (HLS). Por ejemplo, hls4ml y Finn son dos paquetes basados en HLS que se utilizan para crear aceleradores de inteligencia artificial basados en FPGA. Aun así, los diseños generados con HLS no siempre están tan optimizados como aquellos desarrollados con HDL. [14]

Por último, es importante tener en cuenta que los recursos de memoria internos de las FPGAs son limitados y podrían ser insuficientes para modelos grandes de redes neuronales profundas (DNN). Por ejemplo, la red neuronal convolucional[3] (CNN) AlexNet [15] tiene alrededor de 60 millones de floats de 32 bits como parámetros, lo que produce 240 MB para almacenar los pesos, y requiere aproximadamente 1,5 GOPs (1,5 mil millones de operaciones) para procesar cada imagen de entrada. Estos recursos no pueden ser ofrecidos por la mayoría de FPGAs modernas. Por lo tanto, los pesos del modelo deberían ser almacenados en memoria externa y transferidos a la FPGA durante el entrenamiento del modelo, lo que agrega latencia disminuyendo el rendimiento. [14]

D. ASIC

En español, las siglas ASIC refieren a “circuito integrado para una aplicación específica”. El nombre en sí mismo es muy representativo de la idea de estos componentes: son circuitos integrados personalizados para un uso en particular, diferenciándose completamente de los procesadores que vimos hasta ahora, que en mayor o menor medida tenían propósitos más generales.

El diseño de las ASICs tiene 2 tipos: el personalizado completo (full-custom) y el semipersonalizado. En el primero se diseña todo, desde las celdas lógicas (bloques que implementan funciones lógicas básicas, como puertas AND, OR, flip-flops, multiplexores, etc.) hasta las capas de máscara. El plazo de fabricación de las ASICs full-custom es de aproximadamente 8 semanas, sin contar el tiempo de diseño. Utilizar este flujo de diseño tiene sentido cuando no hay una biblioteca existente adecuada a lo pretendido, o cuando las bibliotecas existentes traen problemas en términos de potencia o tamaño. Las ventajas de este tipo de diseño son el rendimiento y el costo, mientras que las desventajas se encuentran en el tiempo de diseño, el alto riesgo y la complejidad. [17]

El flujo de diseño semipersonalizado utiliza celdas estándar, bloques personalizados (con bloques nos referimos a unidades funcionales del diseño que realizan una tarea específica), otros bloques estándar, etc.

Hoy por hoy, las ASICs son diseñadas a través de la metodología “flujo de diseño basado en celdas”. Se reutilizan celdas lógicas previamente diseñadas, lo que ayuda a ahorrar tiempo y dinero. [17]

Como vemos, el proceso de fabricación de estos circuitos son caros y consumen tiempo, implica muchos pasos, como la creación de máscaras, la fabricación del wafer (semiconductor), y el empaquetamiento. De todas formas, una vez que la inversión inicial está hecha, el costo por unidad de las ASICs baja significativamente cuanto mayor es el volumen de producción. [18]

Con esto podemos ver que las ASICs son recomendables económicamente si se quieren utilizar en masa. El costo inicial no se llega a amortizar si no se pretende un alto volumen de producción.

La personalización tan específica que tienen las ASICs permite que estén extremadamente optimizadas para ciertas aplicaciones, en términos de potencia y eficiencia energética, que además pueden alcanzarse con un tamaño físico muy pequeño, el cual a su vez posibilita una disipación del calor más eficiente, resultando en un mejor rendimiento en términos de temperatura.

Pero no todo es bueno: además del alto costo inicial, el largo tiempo de desarrollo y la complejidad del diseño, las ASICs son altamente inflexibles, pues una vez que están diseñadas y fabricadas, su funcionalidad está fijada, y no puede ser cambiada sin diseñar y producir un nuevo chip. Si se quiere especificidad en aplicaciones cuyos requerimientos podrían cambiar en el tiempo, la mejor elección es una FPGA. Por otro lado, debido a la velocidad de los avances tecnológicos en la industria de los semiconductores, las ASICs pueden quedar obsoletas rápidamente. [18]

A continuación profundizaremos en las ASICs más utilizadas en inteligencia artificial: las TPUs.

La primera Unidad de Procesamiento de Tensores fue desarrollada por Google en 2015. El procesamiento tensorial es una técnica de aprendizaje automático que utiliza tensores, que son matrices multidimensionales, para representar datos de diversas fuentes, como imágenes, audio y texto. [19] Las TPUs están diseñadas para realizar operaciones matriciales rápidamente, lo que las hace ideales para cargas de trabajo de aprendizaje automático.

Un chip de TPU contiene uno o más núcleos tensoriales (TensorCores). La cantidad de éstos dependen de la versión del TPU. Cada TensorCore consiste de una o más unidades de multiplicación de matrices (MXUs), una unidad de vectores y una unidad de escalares.

Una MXU está compuesta de 128x128 multiplicadores-acumuladores[4] en un arreglo sistólico[5], y proporciona la mayor parte de la potencia de cómputo en un TensorCore. Cada MXU es capaz de realizar 16 mil operaciones de multiplicación-acumulación por ciclo. Todas las multiplicaciones tienen entradas de tipo bfloat16 [20], un formato personalizado de punto flotante de 16 bits para el aprendizaje automático, utilizados para disminuir el tiempo de convergencia sin perder la exactitud. Las operaciones de multiplicación de matrices se realizan en valores bfloat16,

[3] Las redes neuronales convolucionales utilizan datos tridimensionales para tareas de clasificación de imágenes y reconocimiento [16]

[4] La operación multiplicación-acumulación consiste en multiplicar dos números y sumar el producto a un acumulador. [21]

[5] Las matrices sistólicas son una familia de arquitecturas de ordenadores paralelos capaces de utilizar un gran número de procesadores simultáneamente para cálculos importantes en aplicaciones como la computación científica y el procesamiento de señales. [22]

pero las acumulaciones se hacen en floats de 32 bits. El rango dinámico de ambos tipos son equivalentes, pero bfloat16 ocupa la mitad de espacio de memoria. [23]

La unidad vectorial se utiliza para operaciones generales de redes neuronales y procesamiento de datos, como activaciones o softmax, y la unidad escalar se encarga de operaciones de control, cálculo de direcciones de memoria y otras operaciones de mantenimiento. [20]

Los modelos de Inteligencia Artificial más importantes de Google, como Gemini, Imagen 3 y Gemma 2, fueron entrenados y son entregados al público utilizando TPUs. Hoy por hoy, Google ya va por la sexta generación de TPUs, llamada Trillium. [24]

En la Figura 2 podemos ver una comparación entre el consumo de energía por pastilla o chip individual (watt/die) en relación a la carga de trabajo, en ejecución de una CNN. Esta comparación se realiza entre una TPU, una CPU Haswell y una GPU Nvidia K80. En el gráfico podemos observar que la TPU tiene el menor consumo, pero la peor proporcionalidad de energía: con una carga de 10%, la TPU consume el 88% de energía que utiliza al 100% de trabajo. [25]

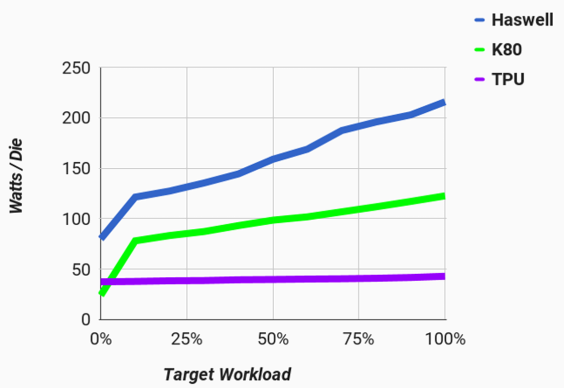

Fig. 2. Vatios/pastilla en una CNN con una variación en la utilización de 0% a 100% [25]

III. Computación Heterogénea En La Inteligencia Artificial

Por lo tratado anteriormente, podemos asegurar qué este paradigma no sólo aborda las limitaciones de los sistemas homogéneos, sino que también facilita la implementación y el entrenamiento de modelos IA complejos.

A continuación daremos dos ejemplos de aplicaciones reales de Deep Learning: un predictor de riesgo de cáncer de mama llamado Mirai, y una computadora de Nvidia que centraliza componentes utilizados para el manejo de vehículos inteligentes. Con estos ejemplos, veremos la potencia de la Inteligencia Artificial en un ámbito tan crucial como la salud, y un caso de uso de la computación heterogénea en el aprendizaje profundo, respectivamente.

A. Medicina

El análisis de imágenes que pueden realizar las inteligencias artificiales es muy prometedor para la detección de enfermedades. En 2021 se desarrolló un predictor de riesgo de cáncer de mama. Mirai es un modelo basado en el deep learning, más precisamente una Red Neuronal Convolucional Profunda (DCNN)[6]. Mirai utiliza más de 200.000 imágenes entre entrenamiento, validación y pruebas pero también considera factores de riesgo como la edad y el historial familiar, y puede predecir estos factores si no están disponibles. En la Figura 3, podemos ver que Mirai consta de 4 módulos [5]: un codificador de imágenes que recibe y procesa las mamografías, un módulo que combina la información de la imagen en todas las vistas para crear una representación completa de la mamografía, otro que predice los factores de riesgo del paciente a partir de la mamografía, y por último, un predictor del riesgo de un paciente cada año durante los próximos cinco años basándose en las etapas anteriores**.** [4]

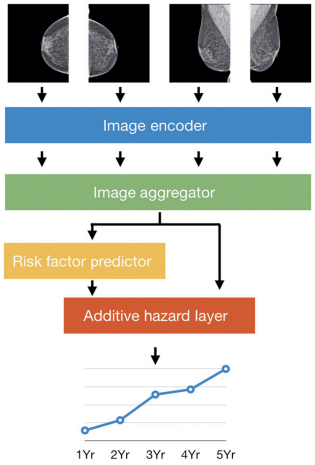

Fig. 3. Módulos de Mirai

B. Vehículos inteligentes

La conducción automática cuenta al día de hoy con un ritmo de desarrollo altamente competitivo. Diversas empresas producen en paralelo sus propios sistemas, en una puja por el mejor cómputo para el volumen de datos que conlleva esta tecnología.

Nos centraremos en un desarrollo que se ha anunciado recientemente, y que todavía no ha salido al mercado. Estamos hablando de DRIVE Thor [9], una arquitectura heterogénea de Nvidia, parte de su serie Drive AGX, que fue concebida para ser la computadora central de los futuros automóviles autónomos.

DRIVE Thor es una computadora heterogénea, lo que trae varias ventajas: reduce el espacio físico requerido y es más eficiente energéticamente. Además, y como podemos ver en la Figura 4, realiza todo el cómputo que otros automotores realizan en diferentes chips, lo que mejora notablemente la paralelización de tareas y se optimiza el rendimiento del sistema. En el sistema tradicional de computación distribuida,

[6] Las DCNN están especialmente diseñadas para procesar y analizar datos con una estructura de cuadrícula, como las imágenes. Esto representa una potencial revolución en la medicina. Por ejemplo, en el artículo[4], se citan otros modelos de DCNN, semejantes a Mirai, que ya están comparando sus resultados con los de un comité de radiólogos.

varias funciones del vehículo, como el estacionamiento, la seguridad activa, el monitoreo del conductor y el sistema de infotainment, son manejadas por diferentes SoCs (System on Chips). Cada uno de estos chips está localizado en distintas partes del automóvil, lo que puede aumentar la complejidad del diseño y la posibilidad de errores de sincronización y coherencia de la caché.

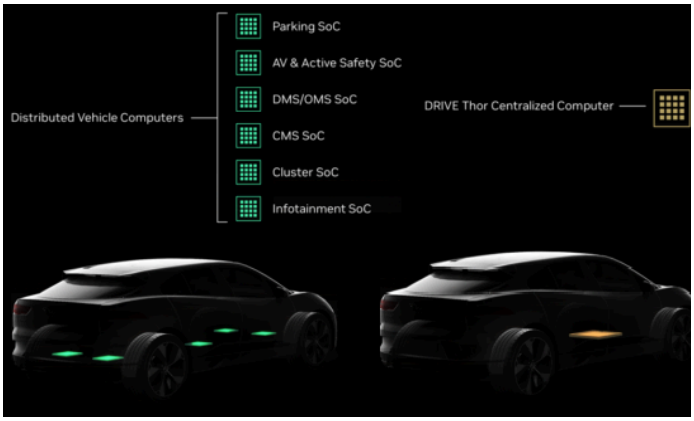

Fig. 4. Todas las pantallas, sensores, cámaras y radares se conectan a este único SoC.

Por último, la consolidación de múltiples componentes en un solo SoC no solo mejora la eficiencia, sino que también reduce los costos de fabricación y ensamblaje. Esto es crucial para los fabricantes de automóviles que buscan implementar tecnologías avanzadas de conducción autónoma optimizando los costos para lograr, en el futuro, que esta tecnología salga al gran público.

IV. Desafíos y Dificultades

Si bien la computación heterogénea parece ser la solución definitiva al dilema de la eficiencia energética y los problemas del rendimiento en comparación a otras arquitecturas, no está exenta de desafíos que dificultan su adopción generalizada en todos los ámbitos.

A. Complejidad en el Desarrollo

La creación de software para sistemas heterogéneos es increíblemente compleja. Los programadores deben enfrentarse a múltiples arquitecturas y diferentes lenguajes de programación, lo que aumenta considerablemente la dificultad del desarrollo y la depuración del código. Hemos visto anteriormente que, para GPUs, se utilizan lenguajes como CUDA y OpenCL, mientras que para FPGA tenemos lenguajes HDL, que agregan una dificultad mucho mayor por su bajo nivel, y HLS, que dan niveles mayores de abstracción a cambio de una disminución en la optimización.

B. Problemas de Estandarización [26]

En consecuencia, la falta de estándares universales también es una barrera importante. Actualmente, no existe un estándar universal para la computación heterogénea, lo que significa que a menudo las soluciones son propias y específicas de cada proveedor. Esto no sólo crea una clara fragmentación en el mercado, sino que puede crear dependencias de fabricantes específicos. Las organizaciones internacionales de estándares (cómo IEEE e ISO), deberán trabajar con la industria para lograr un consenso.

C. Coherencia de la caché [27]

¿Te imaginas mantener la coherencia de una memoria caché que está siendo accedida simultáneamente por una GPU que puede leer y escribir hasta 1 TB/s, a la vez que una FPGA realiza millones de operaciones por segundo y una CPU gestiona tareas adicionales? Pues, este es uno de los retos más formidables en la computación heterogénea. Las GPUs modernas, como las NVIDIA A100, pueden alcanzar anchos de banda de memoria de hasta 1.6 TB/s, permitiendo una cantidad masiva de operaciones de lectura y escritura por segundo. A su vez, debido a que las FPGAs pueden ser reprogramables en tiempo real, sus patrones de acceso a la memoria pueden cambiar dinámicamente, complejizando aún más la coherencia de la caché. Algunas empresas ya están desarrollando posibles soluciones; un claro ejemplo es NVIDIA, la cual dentro de su arquitectura CUDA, implementó la “UMA” (Unified Memory Architecture), una memoria que es accesible tanto por la CPU como por la GPU. Debido a estas implementaciones, es posible usar en C una variable de la función “Malloc” llamada “CudaMallocManaged”, la cual devuelve un puntero que puede ser accedido desde ambos procesadores.

D. Costos de Implementación y Mantenimiento [28]

Posiblemente la barrera más importante para que la computación heterogénea posea una adopción general, son los altos costos que conlleva implementar un sistema con la capacidad de trabajar eficientemente con modelos de Inteligencia Artificial. Tan solo el diseño de una ASIC puede costar millones de dólares. Eso, sin contar el requerimiento de un personal altamente capacitado para manejar estas tecnologías, lo cual aumenta exponencialmente el costo operativo y de mantenimiento. La supercomputadora Summit, ubicada en el Oak Ridge National Laboratory, es un ejemplo de un sistema heterogéneo de alto costo. Esta máquina combina CPUs IBM POWER9 con GPUs NVIDIA Volta, proporcionando un rendimiento nunca antes visto. Sin embargo, el costo de construcción de Summit fue de aproximadamente 200 millones de dólares, y los costos de operación y mantenimiento anual son demasiado elevados. ¿Pero, lo vale? Bueno, en operación, Summit llega a la cifra de 200 Petaflops de potencia de computación. Esto equivale a que, si todos los en la Tierra hiciéramos un cálculo por segundo, nos llevaría un año hacer el mismo número de cálculos que Summit puede hacer en un segundo.

V. Conclusión

La computación heterogénea se asienta como la solución idónea para enfrentar las limitaciones de potencia de los sistemas modernos a la vez que mejora la eficiencia energética, posibilitando avances significativos en muchas áreas tecnológicas cruciales, como la Inteligencia Artificial.

Como toda tecnología en crecimiento, estas arquitecturas traen consigo una gran cantidad de dificultades y desafíos, que buscan superarse para poder asentar la heterogeneidad como la regla en el ámbito informático.

Como ingenieros, diseñadores e investigadores, tenemos la tarea de seguir innovando. Hemos visto con el devenir de la Ley de Moore que el avance tecnológico no puede alcanzarse de una sola manera, se termina llegando a un límite que demanda nuevas tecnologías. La computación heterogénea es el ejemplo perfecto de los esfuerzos que se hacen desde la industria para empujar los límites de la tecnología, y cómo esto a su vez trae nuevas dificultades y problemas a solucionar, creando un ciclo virtuoso de innovación y desarrollo.

Referencias

[1] J. Herrera, Ene. 2024, "Computación heterogénea: qué es y qué puede mejorar", Guía Hardware. [En línea]. Disponible en: https://www.guiahardware.es/computacion-heterogenea

[2] A. B. Abdallah, "Heterogeneous Computing: An Emerging Paradigm of Embedded Systems Design" en "Computational Frameworks", 2017.

[3] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik y O. O. Storaasli, "State-of-the-art in heterogeneous computing" *SINTEF ICT, Departamento de matemática aplicada, Blindern, Oslo, Noruega,* doi: 10.3233/SPR-2009-0296.

[4] A. Yala, P. G. M. F. Strand, G. Lin, K. Smith, Y. L. Wan, L. Lamb, K. Hughes, C. Lehman y R. Barzilay, "Toward robust mammography-based models for breast cancer risk". *Sci. Transl. Med.* 13, eaba4373 (2021).

[5] E. L. Ridley, Ene. 2021, "Mirai AI model upgrades breast cancer risk assessment", AuntMinnie. [En línea]. Disponible en: https://www.auntminnie.com/clinical-news/womens-imaging /breast/article/15627702/mirai-ai-model-upgrades-breast-cancer-risk-assessment

[6] May. 2021, "GPU Memory Latency's Impact, and Updated Test", Chips And Cheese. [En línea]. Disponible en: https://chipsandcheese.com/2021/05/13/gpu-memory-latencys-impact-and-updated-test/

[7] M. J. Mišić, Đ. M. Đurđević y M. V. Tomašević, "Evolution and Trends in GPU Computing" *Universidad de Belgrado/Escuela de Ingeniería Eléctrica, Belgrado, Serbia,* May 2022.

[8] R. Merrit, Dic. 2023, "Why GPUs Are Great for AI", NVIDIA Blogs [En línea]. Disponible en: https://blogs.nvidia.com/blog/why-gpus-are-great-for-ai/

[9]A. Kani, Sep. 2022, "NVIDIA DRIVE Thor Strikes AI Performance Balance, Uniting AV and Cockpit on a Single Computer". [En línea]. Disponible en: https://blogs.nvidia.com/blog/drive-thor/

[10] A. Gonzalo, 2019, "Comparando CPUs y GPUs para inteligencia artificial". [En línea]. Disponible en:https://machinelearningparatodos.com/comparando-cpus-y-gpus-para-inteligencia-artificial/

[11] J. Schneider, I. Smalley, "What is a field programmable gate array (FPGA)?". [En línea]. Disponible en: https://www.ibm.com/think/topics/field-programmable-gate- arrays

[12] "What is HPC?". [En línea]. Disponible en: https://www.ibm.com/topics/hpc

[13] W. Tang, C. Wang, X. Zhou y H. Chen, "A Survey of FPGA Based Deep Learning Accelerators: Challenges and Opportunities".

[14] A. Nechi, L. Groth, S. Mulhelm, F. Merchant, R. Buchty y M. Berekovic, "FPGA-based Deep Learning Inference Accelerators: Where Are We Standing?". *ACM Trans. Reconfig. Technol. Syst. 16, 4, Artículo 60 (Octubre 2023)*, *32 páginas,* doi: 10.1145/3613963.

[15] A. Krizhevsky, I. Sutskever y G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks".

[16] "What are convolutional neural networks?", IBM. [En línea]. Disponible en: https://www.ibm.com/topics/convolutional-neural-networks

[17] A. A. Shetty, "ASIC Design Flow And Methodology - An Overview". *SSRG International Journal of Electrical and Electronics Engineering (SSRG - IJEEE), Volume 6 Issue 7, 2019*, doi: 10.14445/23488379.

[18] R. Rao, "ASIC vs FPGA: A Comprehensive Comparison". [En línea]. Disponible en: https://www.wevolver.com/article/asic-vs-fpga

[19] "¿Qué es la Unidad de Procesamiento Tensorial y por qué es importante en la actualidad?". [En línea]. Disponible en: https://tensorprocessingunit.com/que-es-la-unidad-de-procesamiento-tensorial-y-por-que-es-importante-en-la-actualidad/

[20] Documentación de TPUs de Google Cloud, apartado "Arquitectura del sistema". [En línea]. Disponible en: https://cloud.google.com/tpu/docs/system-architecture-tpu-vm

[21] "Multiply-Accumulate Operation (MAC)". [En línea]. Disponible en: https://www.allaboutcircuits.com/ip-cores/arithmetic-core/multiply-accumulate

[22] H. T. Kung. 2003. Systolic array. Encyclopedia of Computer Science. John Wiley and Sons Ltd., GBR, 1741–1743. (nota: cita realizada en formato ACM, pues el texto fue extraído del capítulo "Systolic array" del libro "Encyclopedia of Computer Science", de múltiples autores. H. T. Kung es el autor del capítulo).

[23] Documentación de TPUs de Google Cloud, apartado "Formato de número bfloat16". [En línea]. Disponible en: https://cloud.google.com/tpu/docs/bfloat16?hl=es-419

[24] "Announcing Trillium, the sixth generation of Google Cloud TPU". [En línea]. Disponible en: https://cloud.google.com/blog/products/compute/introducing-trillium-6th-gen-tpus

[25] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit". *Google, Inc., Mountain View, CA USA 2017. In Proceedings of ISCA '17, Toronto, ON, Canada, June 24-28, 2017, 12 pages.* https://doi.org/10.1145/3079856.3080246

[26] "A Unified Programming Model for Heterogeneous Computing with CPU and Accelerator Technologies". [En línea]. Disponible en: https://cs.paperswithcode.com/paper/a-unified-programming-model-for-heterogeneous

[27] "Unified Memory for CUDA Beginners". [En línea]. Disponible en: https://developer.nvidia.com/blog/unified-memory-cuda-beginners/

[28] "Summit, el superordenador para la Inteligencia Artificial". [En línea]. Disponible en: https://hardwaresfera.com/noticias/ciencia/summit-el-superodenador-para-la-inteligencia-artificial-cuenta-con-27-648-gpu-nvidia-volta-v100/