FUN@UA:/$ CUDA

3DPL 2023 - UNIVERSIDAD DE ALICANTE

David Mulero-Pérez <dmulero@dtic.ua.es>
Manuel Benavent-Lledó <mbenavent@dtic.ua.es>
José García-Rodríguez <jgarcia@dtic.ua.es>

DAVID

MULERO PÉREZ

- **Grado en Ingeniería Multimedia**
  - Universidad de Alicante
  - 2017 – 2021

- **Máster en Ciencia de Datos**
  - Universidad de Alicante
  - 2021 – 2022

- **Doctorado en Informática (Virtual Reality and Deep Learning)**
  - Univerisdad de Alicante
  - 2022 – Actualidad

MAIL: **DMULERO @ DTIC.UA.ES**

MANUEL

BENAVENT LLEDÓ

- **Grado en Ingeniería Informática**
  - Universidad de Alicante
  - 2017 – 2021

- **Máster en Automática y Robótica**
  - Universidad de Alicante
  - 2021 – 2022

- **Doctorado en Informática (Deep Learning and Computer Vision)**
  - Univerisdad de Alicante
  - 2022 – Actualidad

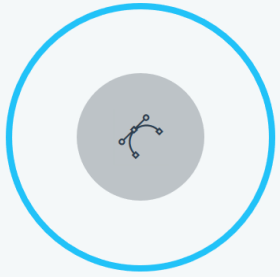
MAIL: **MBENAVENT @ DTIC.UA.ES**

DEDICACIÓN

INVESTIGACIÓN

3DPLAB

The 3D Perception Lab at the University of Alicante is a group of researchers interested in the intersection of machine learning and computer vision. Our research mission focuses on various aspects of perception often related with mobile robotics in which we exploit 3D data as the main source of information. Some of our research lines include object recognition, semantic segmentation, rigid and non-rigid registration, visual localization and mapping, behavior analysis, and depth estimation. Apart from those general lines we are also highly interested in making those solutions run efficiently by leveraging GPU acceleration using CUDA. Aside from 3D data as our backbone, we are also tied together by our shared vision in the great potential of artificial intelligence, mainly deep learning, which we try to apply and push its limits in every project we work on.



DEEP LEARNING

Artificial intelligence applied to computer vision and robotics (semantic segmentation, depth estimation, scene understanding, object recognition, localization and maping).

3D COMPUTER VISION

Traditional computer vision methods and new challenges for 3D data (rigid registration, non-rigid registration, reconstruction).

GPU COMPUTING

Acceleration of computer vision methods and artificial intelligence pipelines for real-time execution and maximum efficiency.

INICIOS Y EVOLUCIÓN DE LOS PROCESADORES GRÁFICOS (GPUs)

MÁSTER EN INTELIGENCIA ARTIFICIAL

UNIVERSIDAD DE ALICANTE

Manuel Benavent-Lledó <mbenavent@dtic.ua.es>

David Mulero-Pérez <dmulero@dtic.ua.es>

José García-Rodríguez <jgarcia@dtic.ua.es>

CONTENIDO

LA LEY DE MOORE

GORDON MOORE

Cramming More Components onto Integrated Circuits. Gordon E. Moore, 1965

“ THE NUMBER OF TRANSISTORS ON A CHIP DOUBLES EVERY 12 MONTHS”

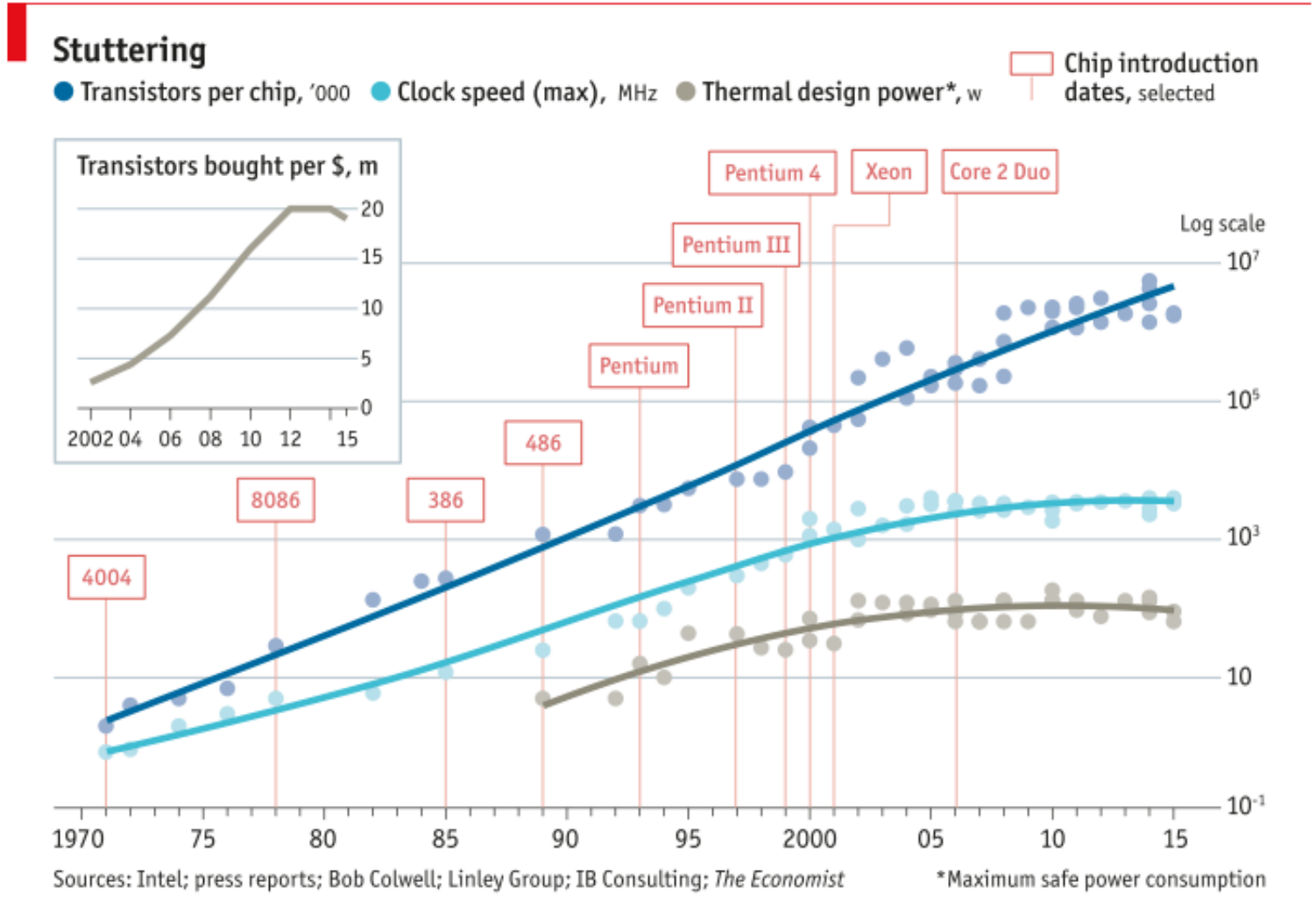
– GORDON MOORE, COFUNDADOR DE INTEL, 1965

“ THE NUMBER OF TRANSISTORS ON A CHIP DOUBLES EVERY 24 MONTHS”

– GORDON MOORE, COFUNDADOR DE INTEL, 1975

LA LEY DE MOORE

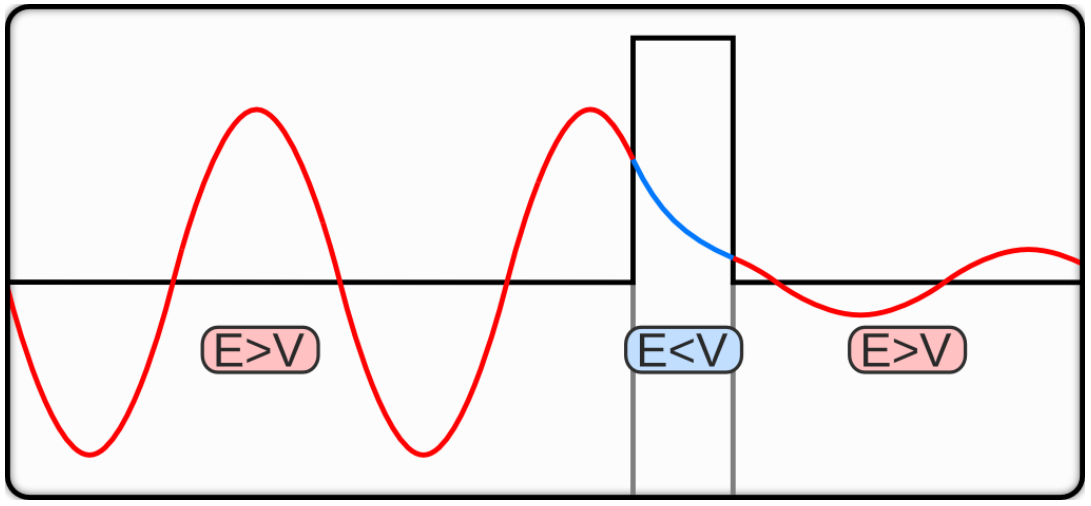
CADA DOS AÑOS, APROXIMADAMENTE, SE DUPLICA EL NÚMERO DE TRANSISTORES

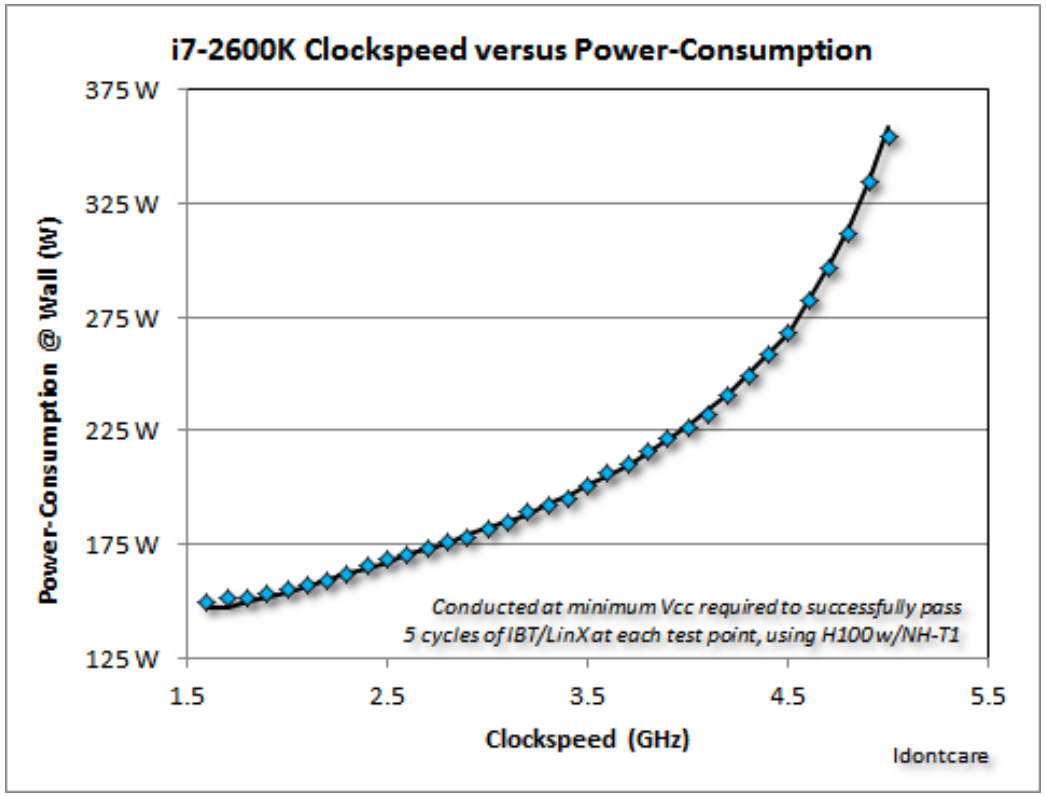


LA LEY DE MOORE

PROBLEMAS: TAMAÑO DEL TRANSISTOR Y EFECTO TÚNEL
CONSUMO ENERGÉTICO Y DISIPACIÓN DE CALOR

Los transistores están llegando a sus límites de tamaño, lo que puede afectar negativamente la eficiencia y la fiabilidad de los circuitos integrados.



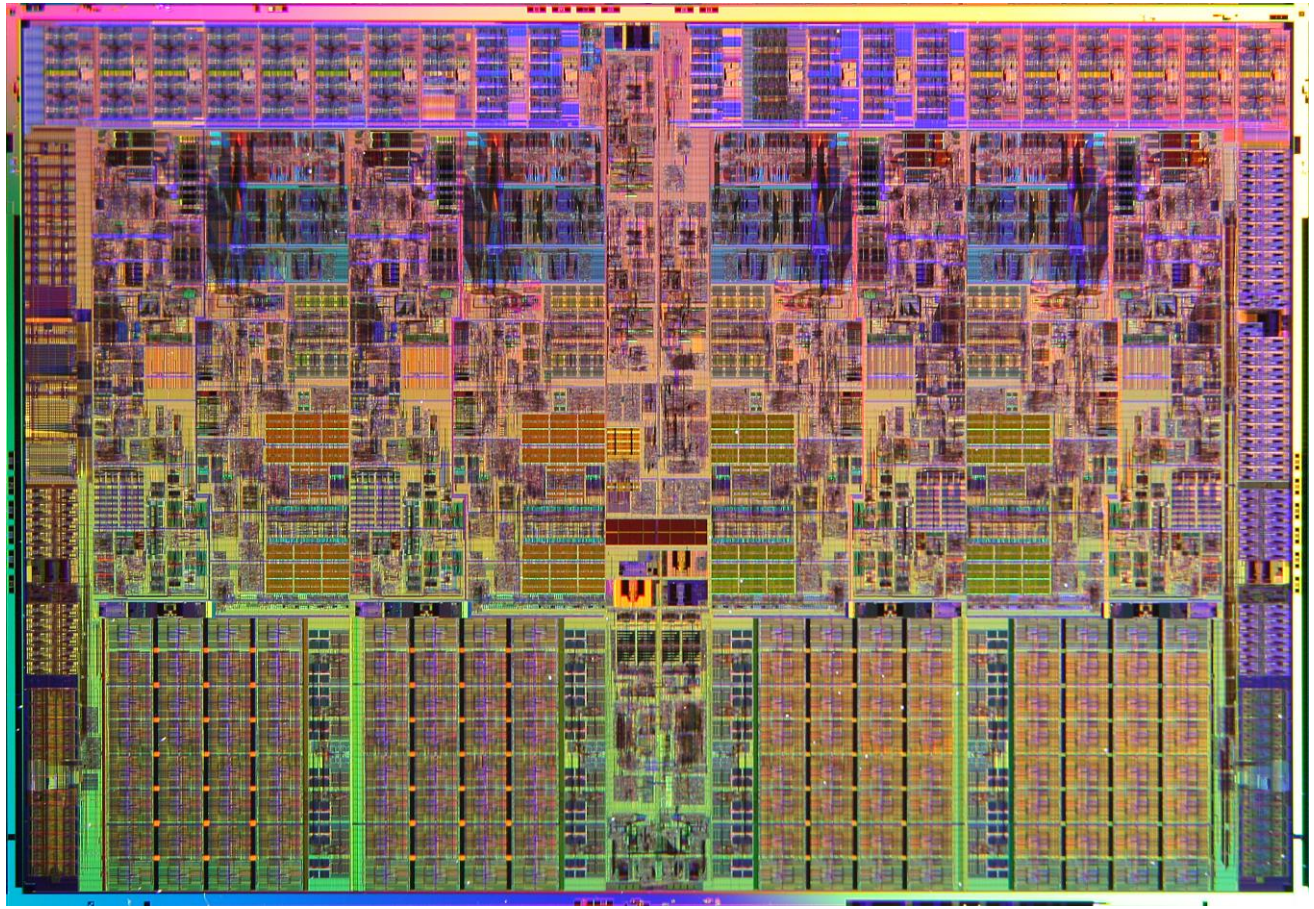


LA LEY DE MOORE

¿CÓMO CONTINUAR ESCALANDO? MULTICORE

La necesidad de continuar escalando la capacidad de procesamiento ha llevado a la adopción de un enfoque **multinúcleo** para la arquitectura de los procesadores.

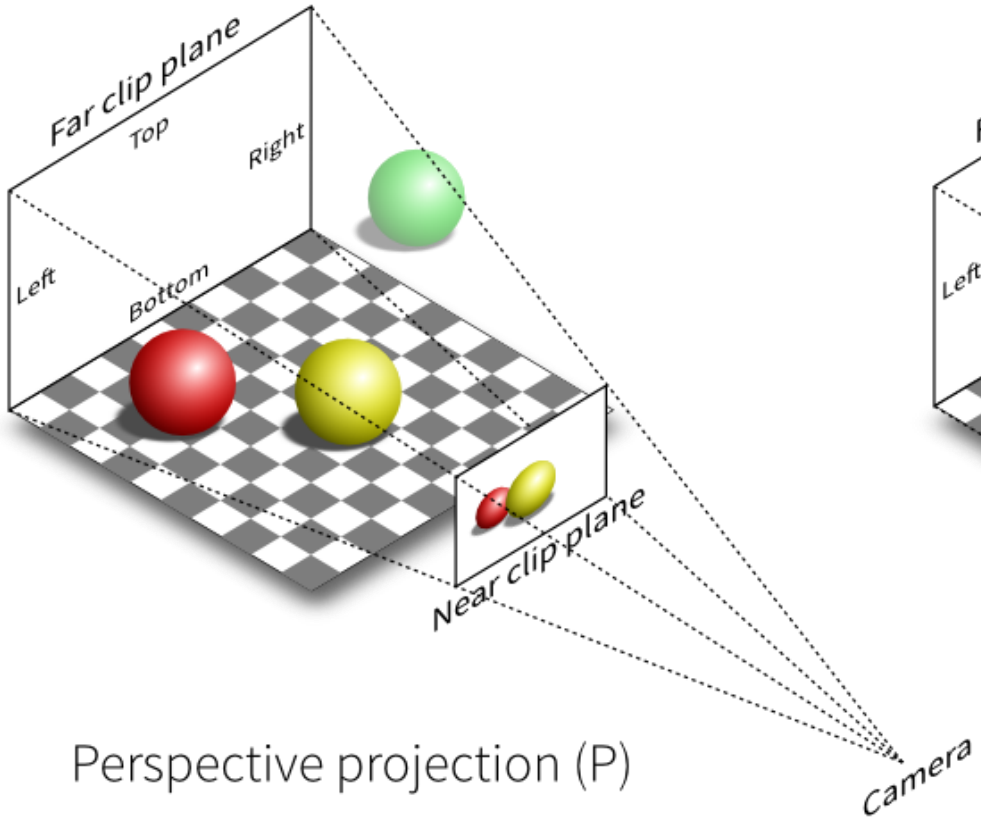
Los procesadores multicore tienen múltiples núcleos de procesamiento en un solo chip, lo que permite un mayor rendimiento y una mejor utilización del consumo energético.



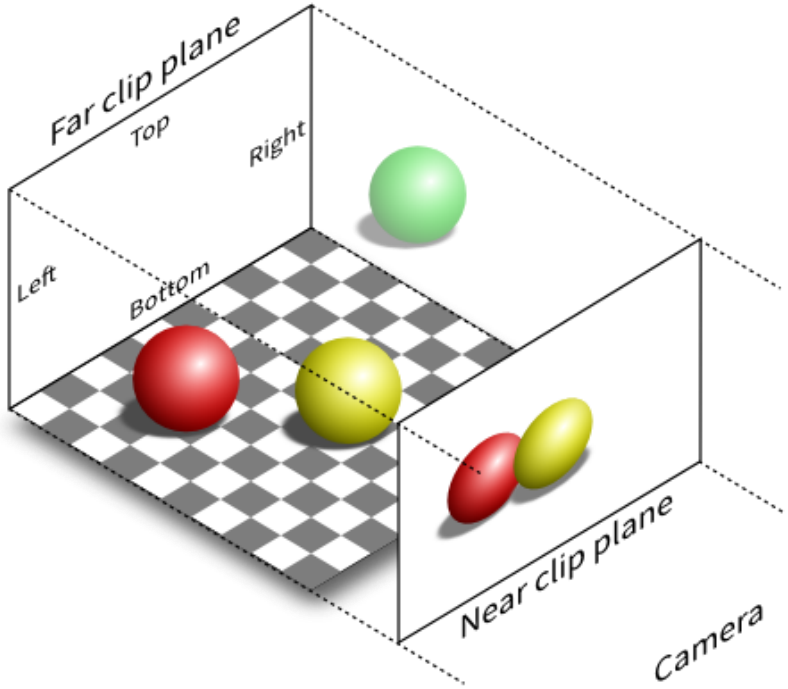
LA UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

EL PIPELINE GRÁFICO: RENDERING

El pipeline gráfico es el conjunto operaciones que se realizan para generar una imagen o un gráfico en un sistema informático.



Perspective projection (P)

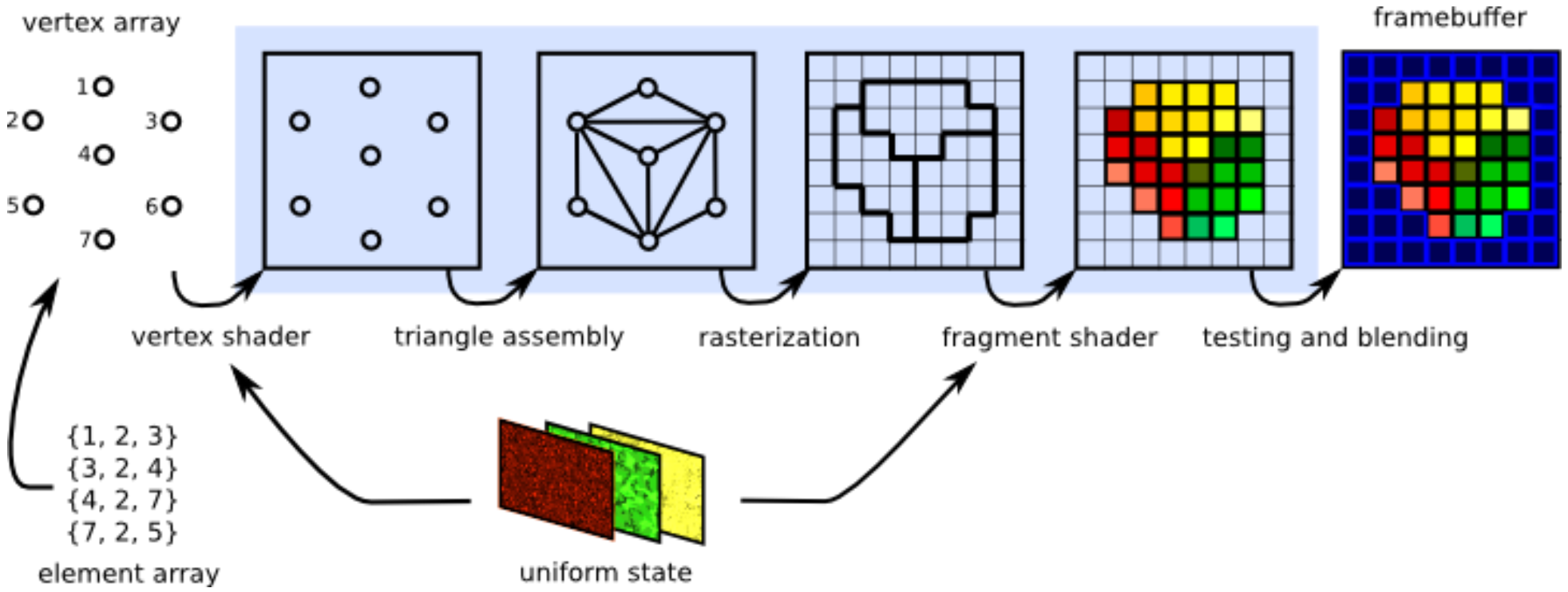


Orthographic projection (O)

LA UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

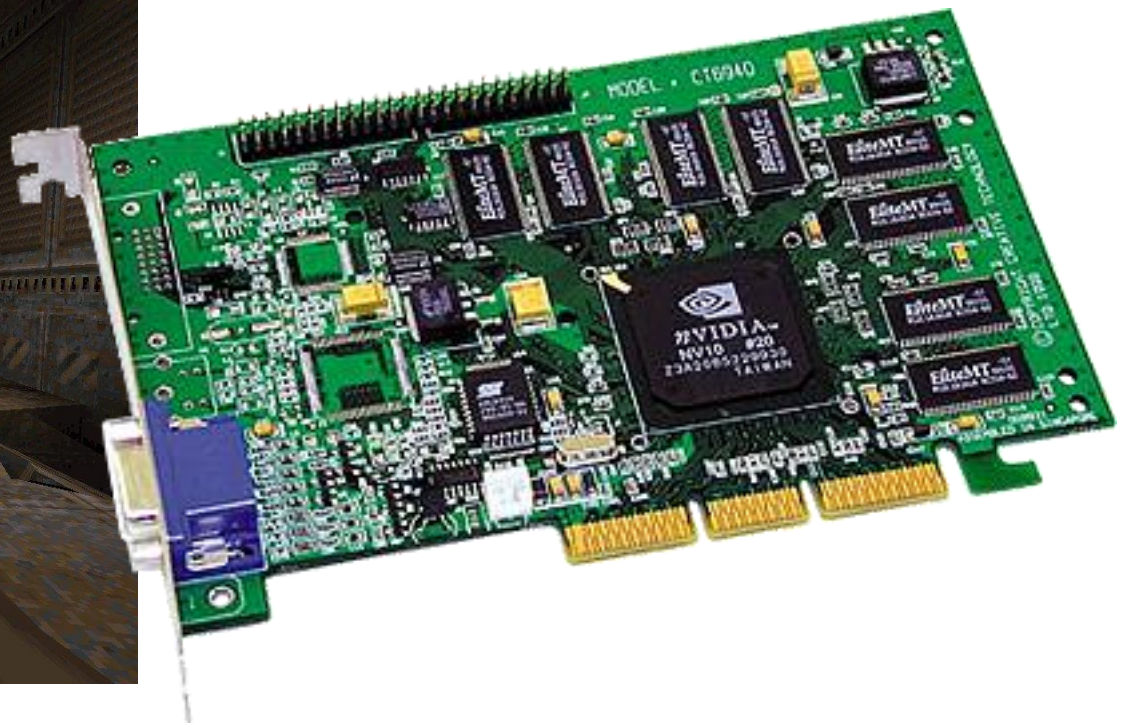
EL PIPELINE GRÁFICO: RENDERING

La etapa de rendering es una parte crucial del pipeline gráfico, ya que se encarga de **calcular y procesar la imagen en tiempo real**.



LA UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

GEFORCE 256 (1999)



GeForce 256 se comercializó como "la primera GPU del mundo"

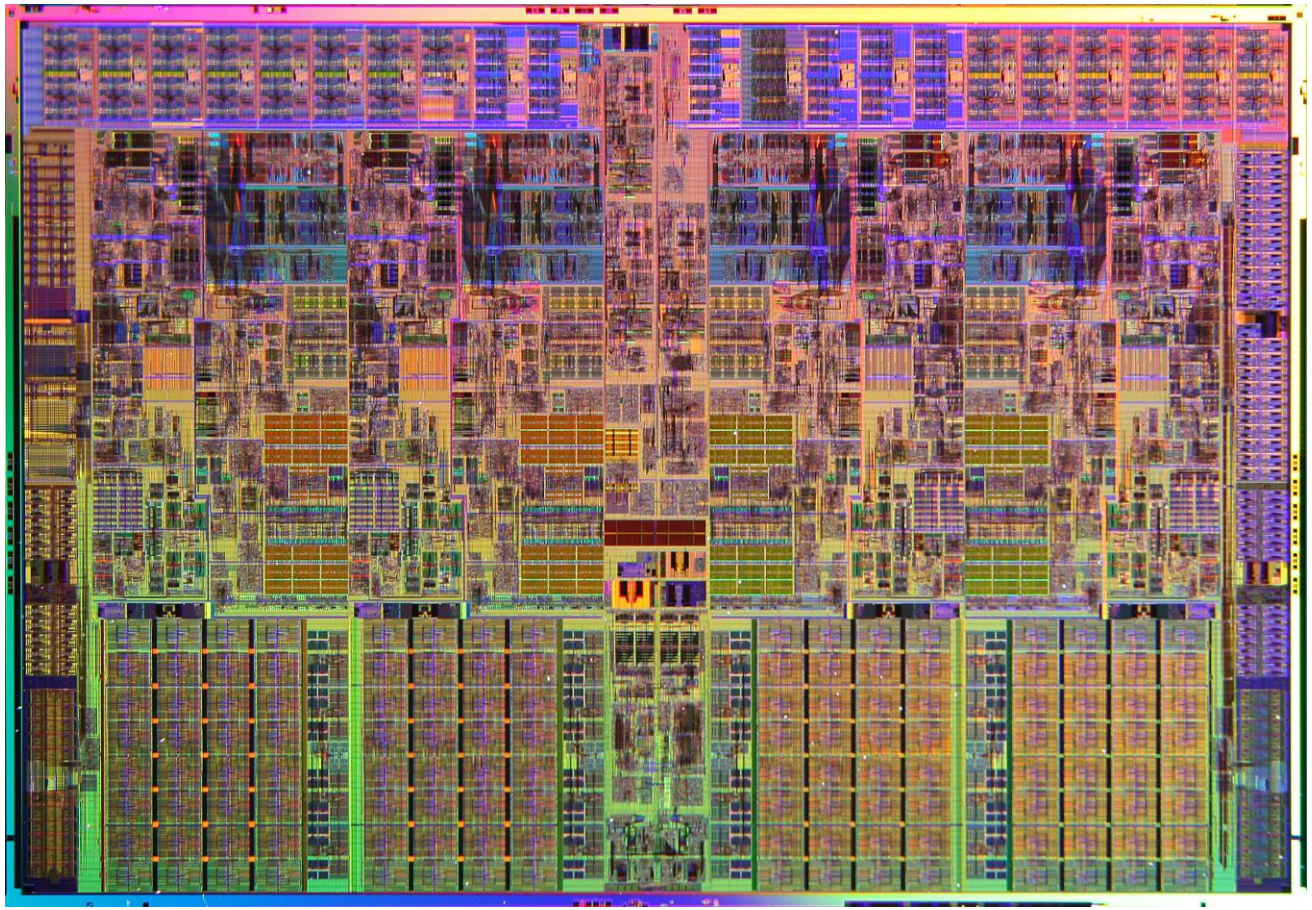
LA UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

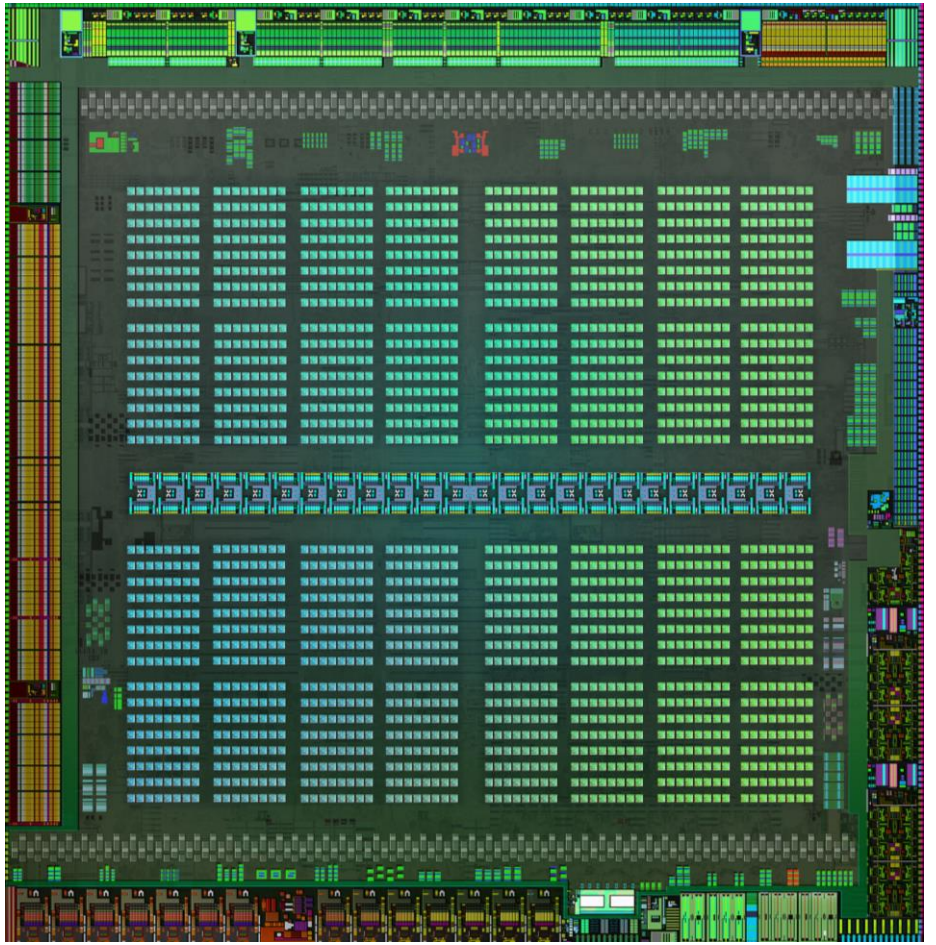
GEFORCE 256 (1999)

"A SINGLE-CHIP PROCESSOR WITH INTEGRATED TRANSFORM, LIGHTING, TRIANGLE SETUP/CLIPPING, AND RENDERING ENGINES THAT IS CAPABLE OF PROCESSING A MINIMUM OF 10 MILLION POLYGONS PER SECOND."

LA UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

LA ESENCIA DE LA GPU



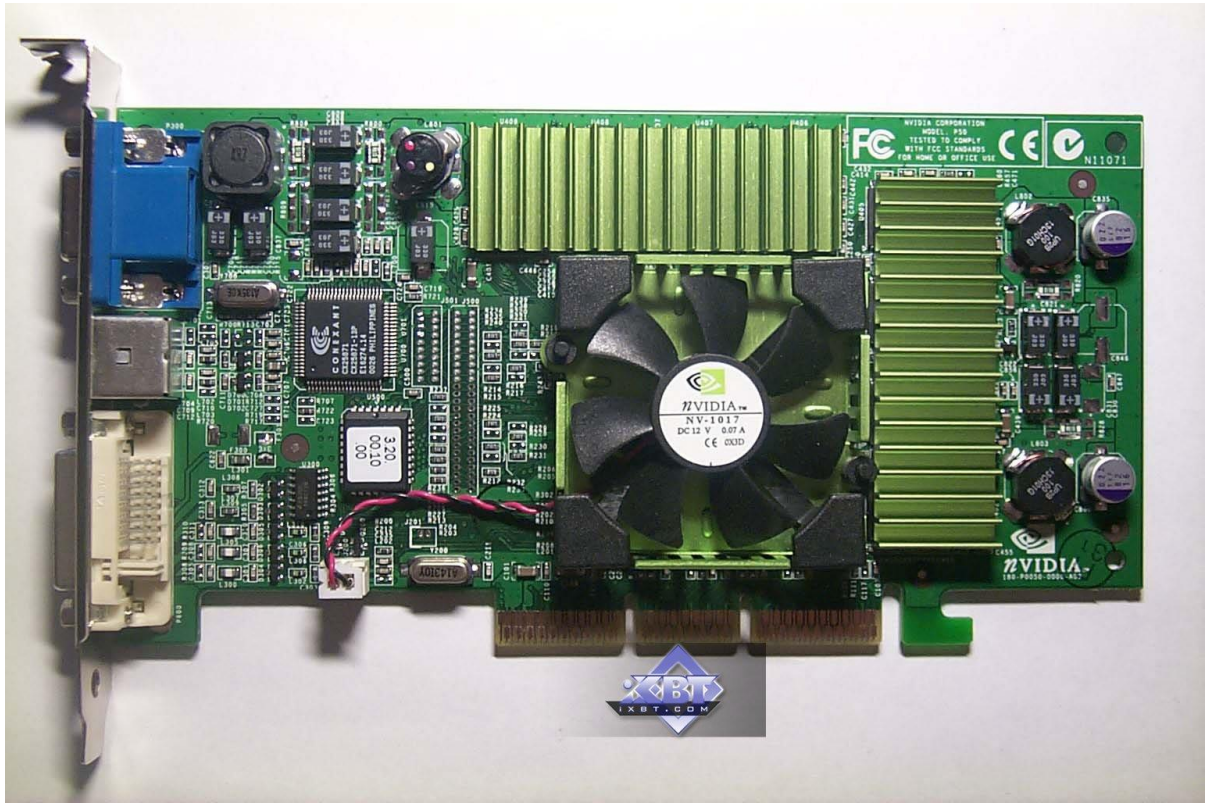


PRIMEROS PASOS EN COMPUTACIÓN SOBRE GPUs

GEFORCE 3 CON VERTEX Y PIXEL SHADERS PROGRAMABLES (2001)

La inclusión de shaders programables en las GeForce 3 allanó el camino para una amplia gama de aplicaciones más allá de los videojuegos, desde la **simulación científica** hasta la criptografía y el **aprendizaje profundo**.

Esto marcó el comienzo de la computación en paralelo accesible a través de GPUs.



PRIMEROS PASOS EN COMPUTACIÓN SOBRE GPUS

ENGAÑAR A LA GPU EMPLEANDO APIs GRÁFICAS (OPENGL O DIRECTX)

```
float saxpy (
        float2 coords : TEXCOORD0,
    uniform sampler2D textureY,
    uniform sampler2D textureX,
    uniform float alpha ) : COLOR
{
        float result;
        float yval=y_old[i];
        float y = tex2D(textureY,coords);
        float xval=x[i];
        float x = tex2D(textureX,coords);
        y_new[i]=yval+alpha*xval;
        result = y + alpha * x;
        return result;
}
```

Y = Y + alpha * X

PRIMEROS PASOS EN COMPUTACIÓN SOBRE GPUS

LIMITACIONES QUE IMPIDIERON EL PROGRESO

CURVA DE APRENDIZAJE DE OPENGL/DIRECTX Y ESFUERZO DE TRADUCCIÓN

NECESIDAD DE APRENDER LENGUAJES DE SHADING (CG, GLSL)

SOPORTE DE FLOAT O DOUBLE NO GARANTIZADO

LIMITACIONES EN LOS PATRONES DE ESCRITURA Y LECTURA DE MEMORIA

CARENCIA DE HERRAMIENTAS DE DEPURACIÓN Y CONTROL DE ERRORES

RECURSOS LIMITADOS: MEMORIA, VELOCIDAD, FLEXIBILIDAD...

ARQUITECTURA CUDA

2007

CUDA (Compute Unified Device Architecture) es una plataforma de computación paralela en GPUs desarrollada por Nvidia

NVIDIA GEFORCE 8800 GTX (2007)

ARQUITECTURA CUDA

ECOSISTEMA

ARQUITECTURA HARDWARE PROPIA

DRIVER ESPECIALIZADO PARA LA GPU

LENGUAJE DE PROGRAMACIÓN FLEXIBLE (BASADO EN C++ INICIALMENTE)

COMPILADOR Y ENTORNO DE DESARROLLO Y DEPURACIÓN

DOCUMENTACIÓN, TUTORIALES, DONACIONES

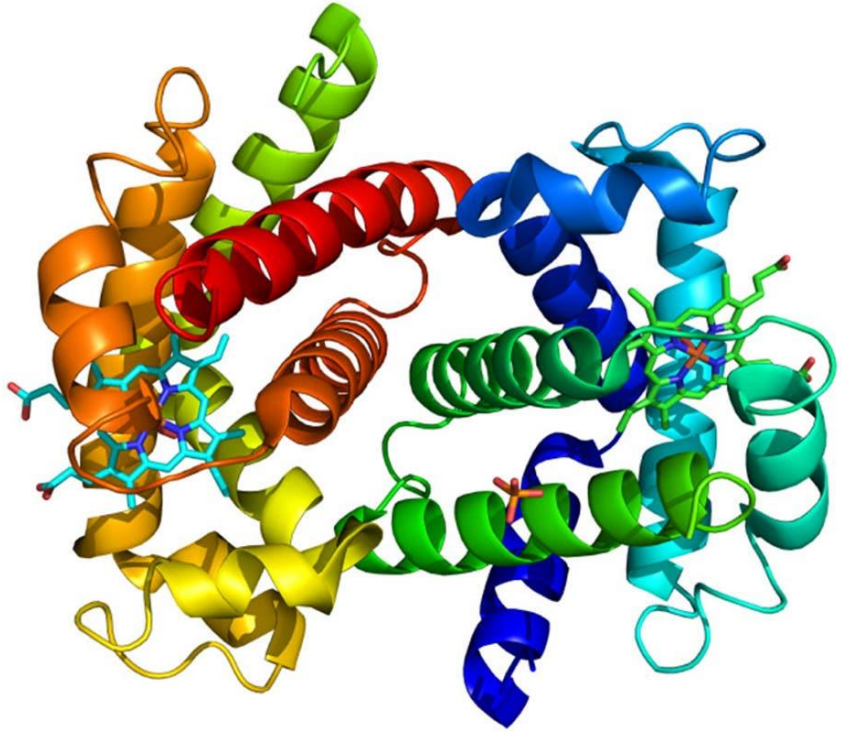
INTRODUCCIÓN A CUDA

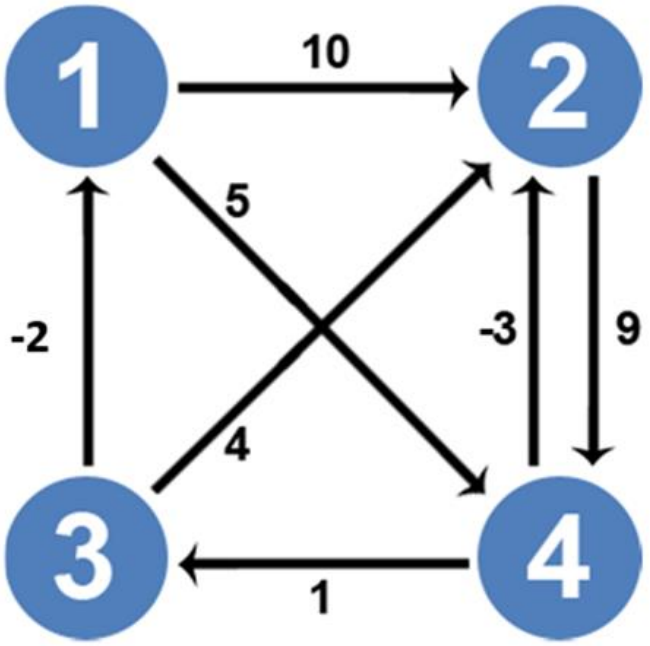
¿QUÉ ES CUDA?

NVIDIA RTX 4080 (2022)

¿QUÉ ES CUDA?

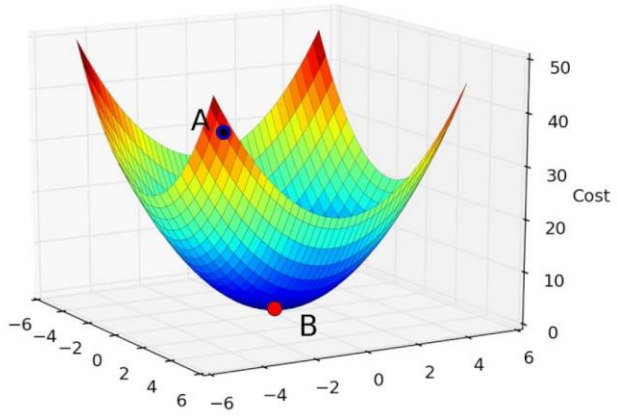
PROTEIN FOLDING, FLOYD WARSHALL ALGORITHM

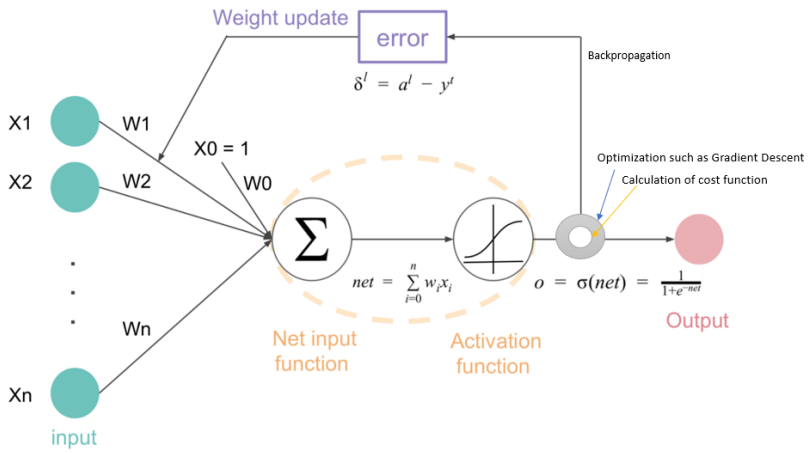




¿QUÉ ES CUDA?

GRADIENT DESCENT Y BACKPROPAGATION





Machine Learning

Chat GPT
Whisper
DALL-E 2

AlphaGo
AlphaFold 2
AlphaTensor

BARD
Imagen
PaLM 2

StableDifussion

Claude 2

¿QUÉ ES CUDA?

DEJEMOS QUE LOS CAZADORES DE MITOS LO EXPLIQUEN

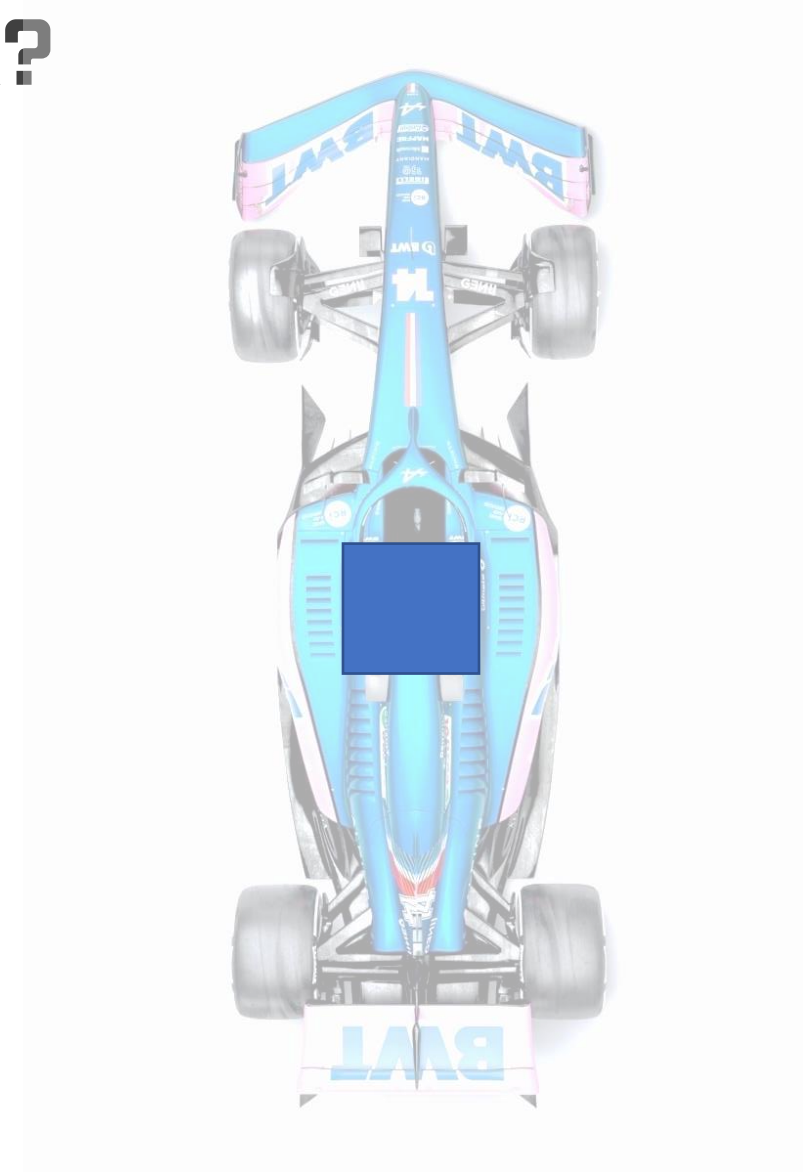
https://youtu.be/-P28LKWTzrI

¿QUÉ ES CUDA?

ANALOGÍA

¿QUÉ ES CUDA?

ANALOGÍA



CPU Mononúcleo

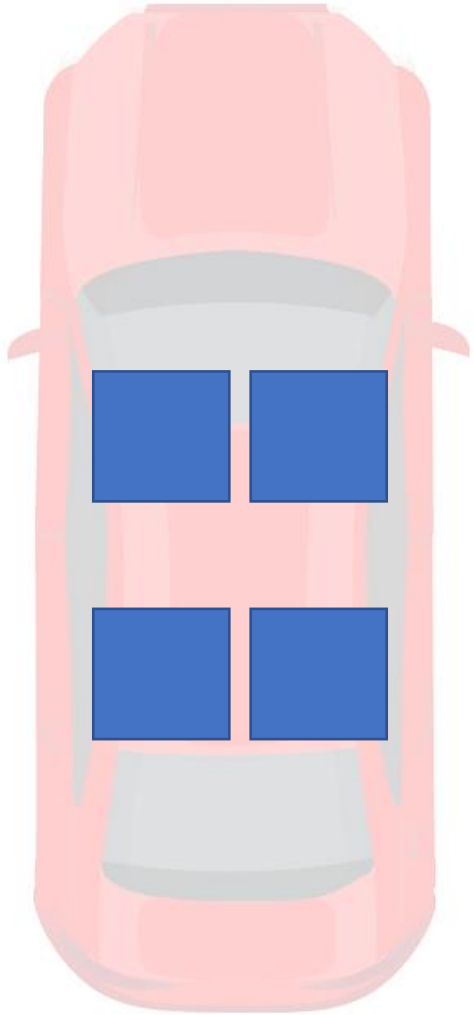
"Rápido"
Solo una plaza

¿QUÉ ES CUDA?

ANALOGÍA

¿QUÉ ES CUDA?

ANALOGÍA



CPU Multinúcleo

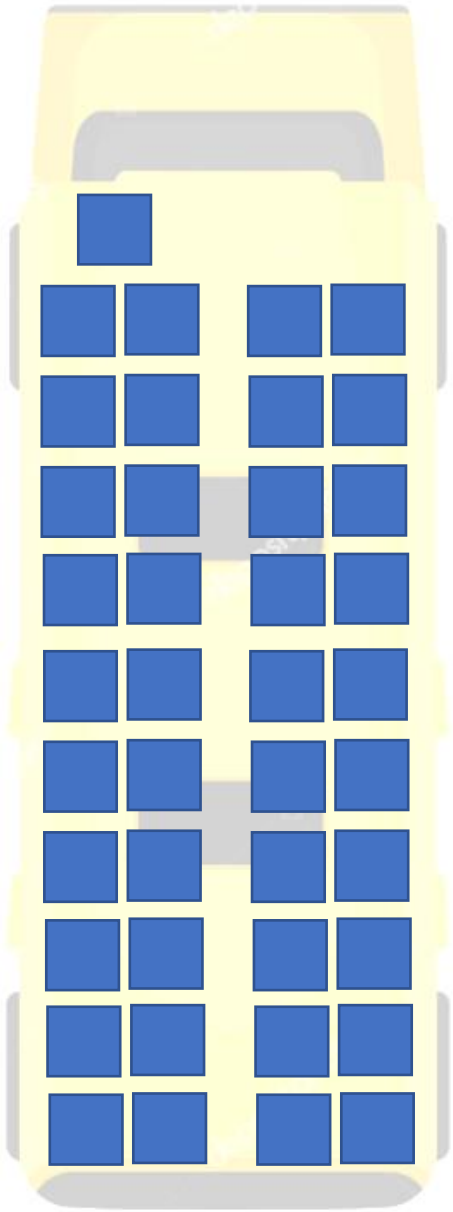
"Menos rápido que un F1"
Cuatro plazas

¿QUÉ ES CUDA?

ANALOGÍA

¿QUÉ ES CUDA?

ANALOGÍA



GPU

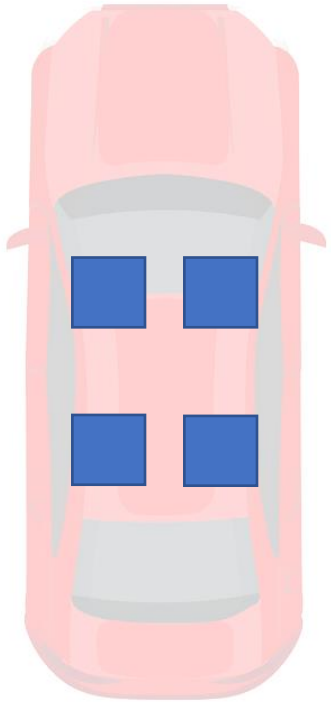
Lento
Muchas plazas

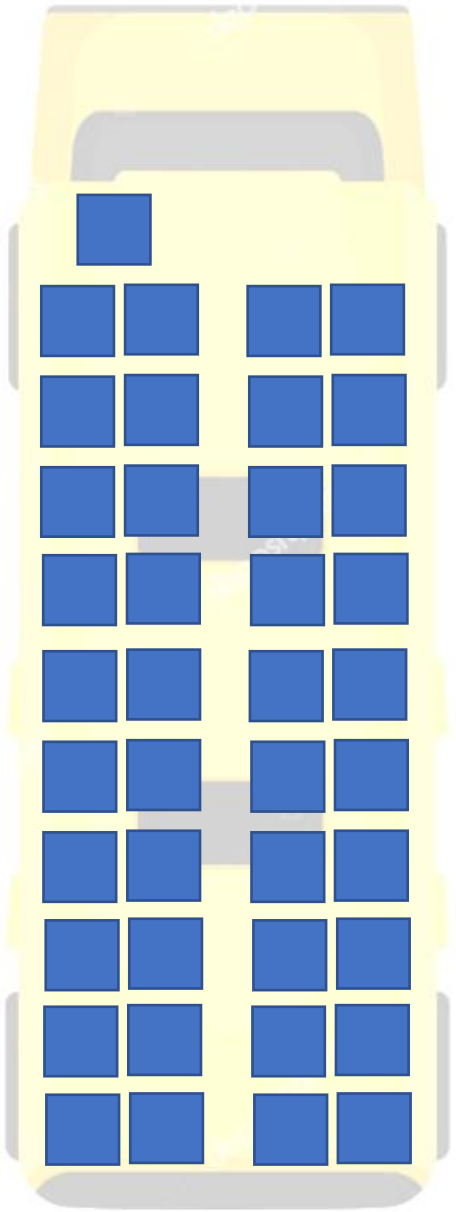
¿QUÉ ES CUDA?

ANALOGÍA

CPU

Rápida
Varios núcleos



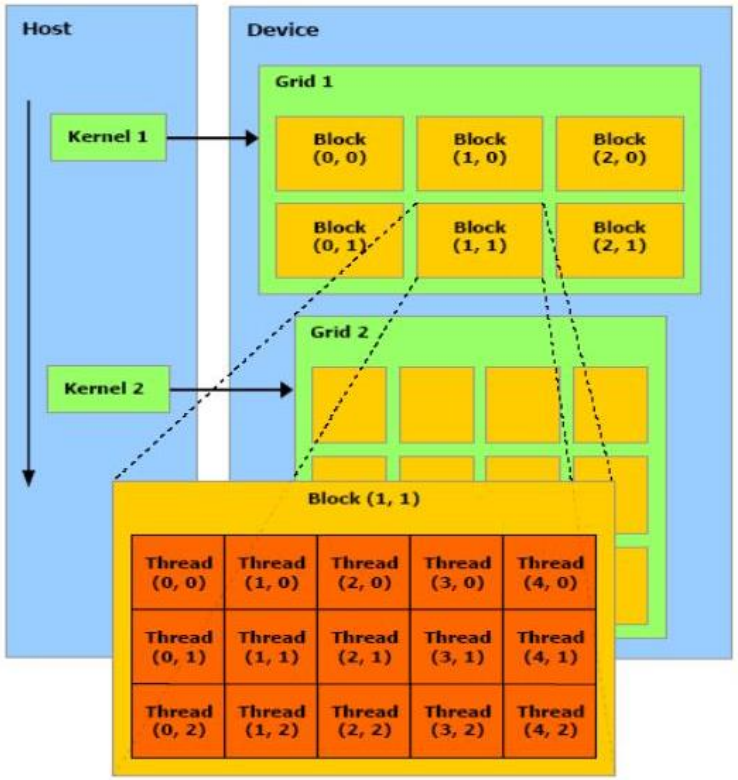


GPU

Lento
Muchos núcleos

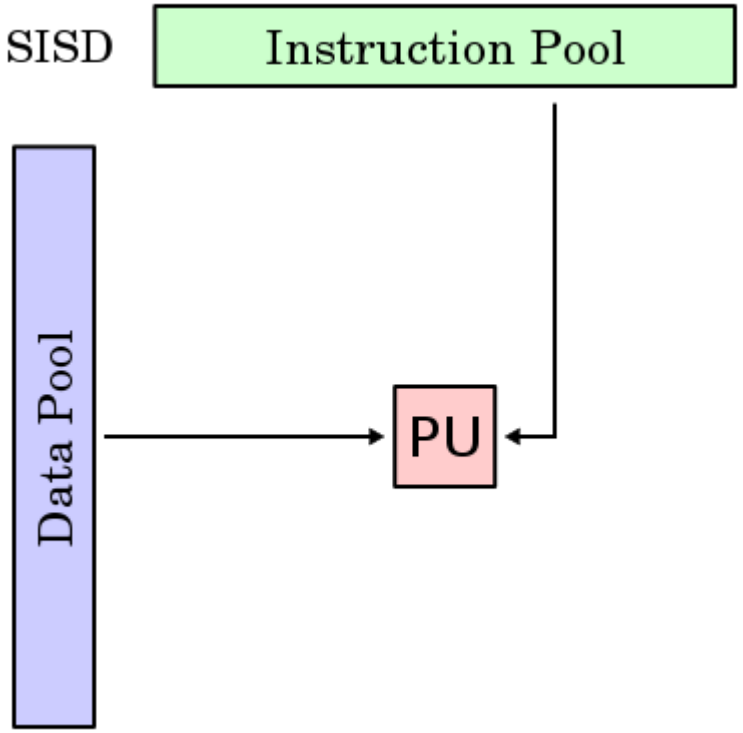
ARQUITECTURA HARDWARE CUDA

SUBTÍTULO QUE NADIE LEE



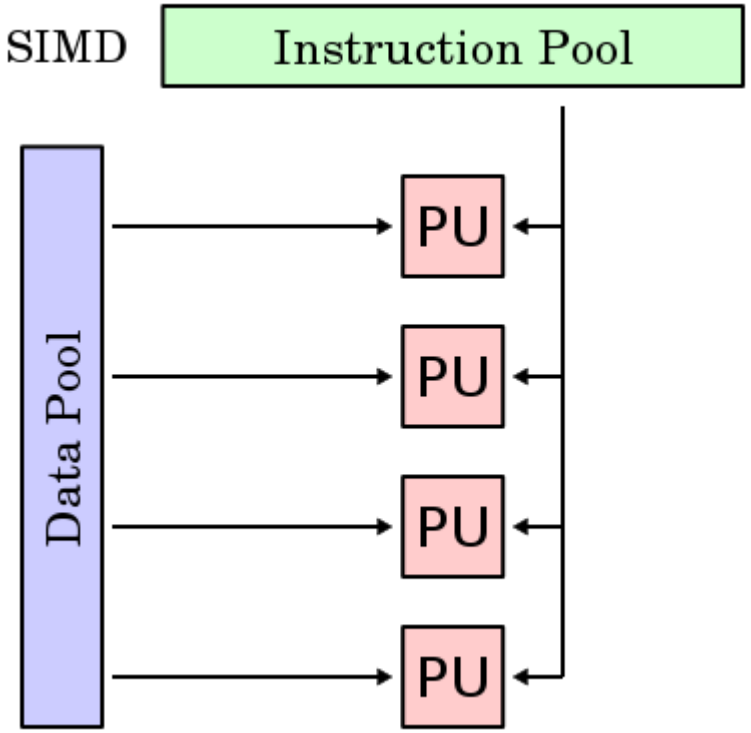
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (SISD)



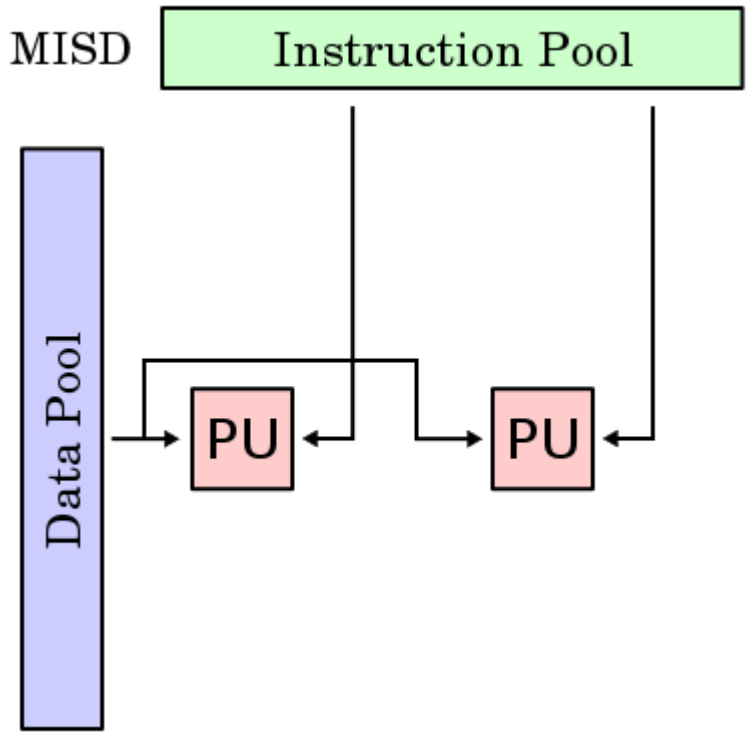
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (SIMD)



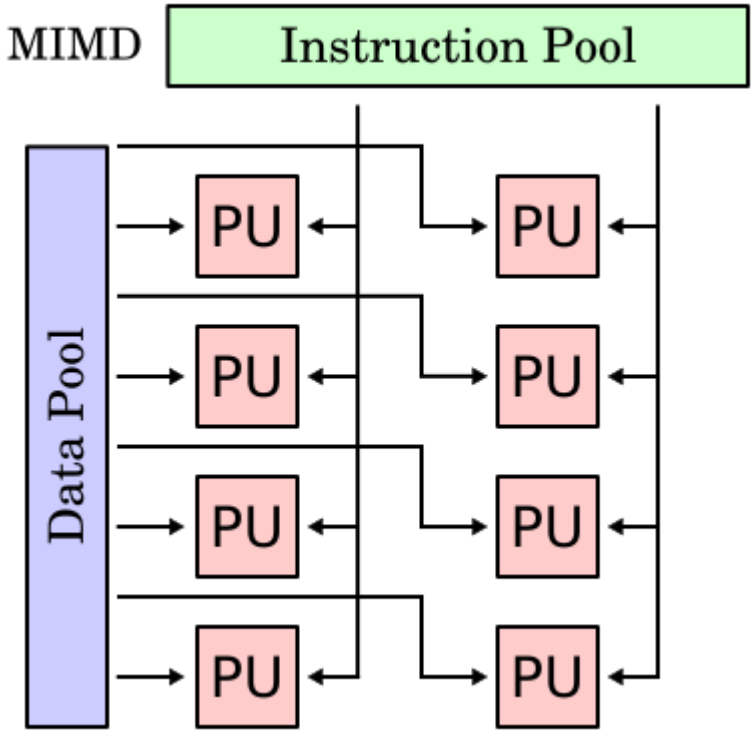
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (MISD)



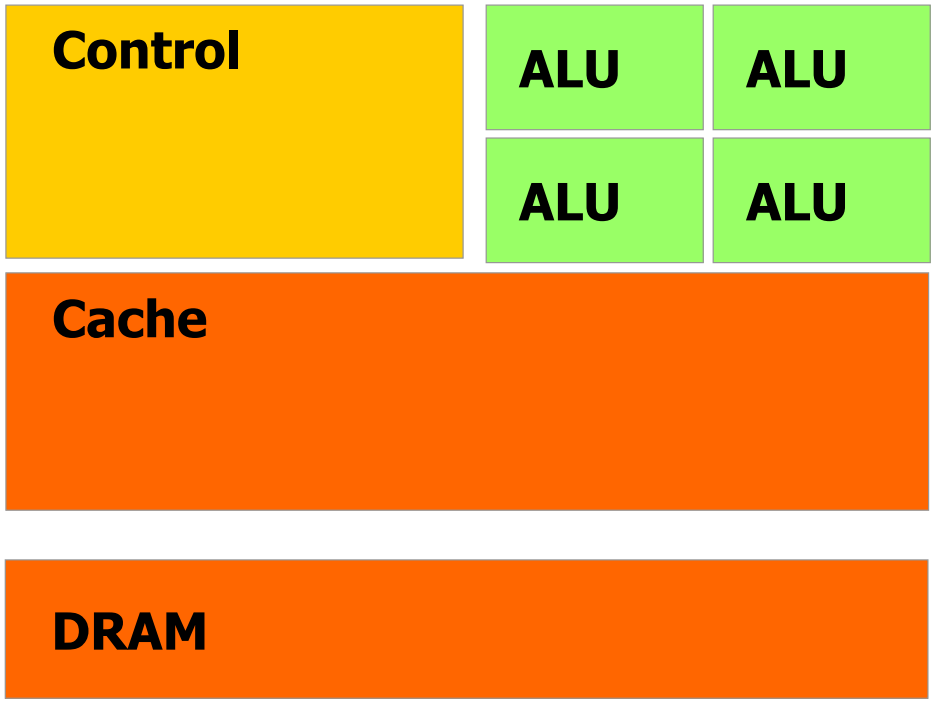
ARQUITECTURA HARDWARE

TAXONOMÍA DE FLYNN (MIMD)

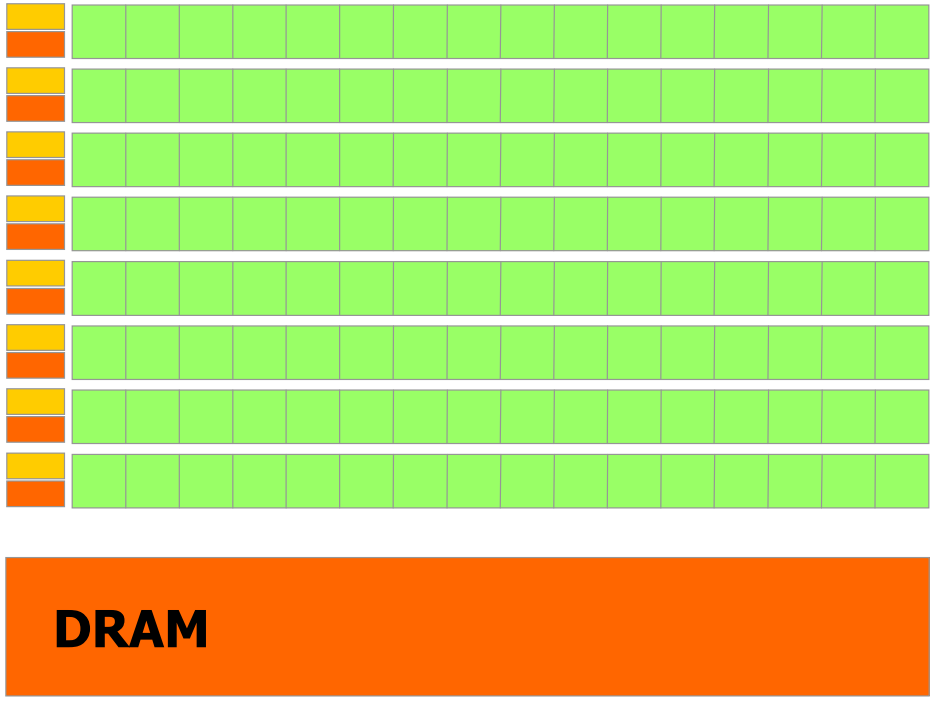


ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU



CPU



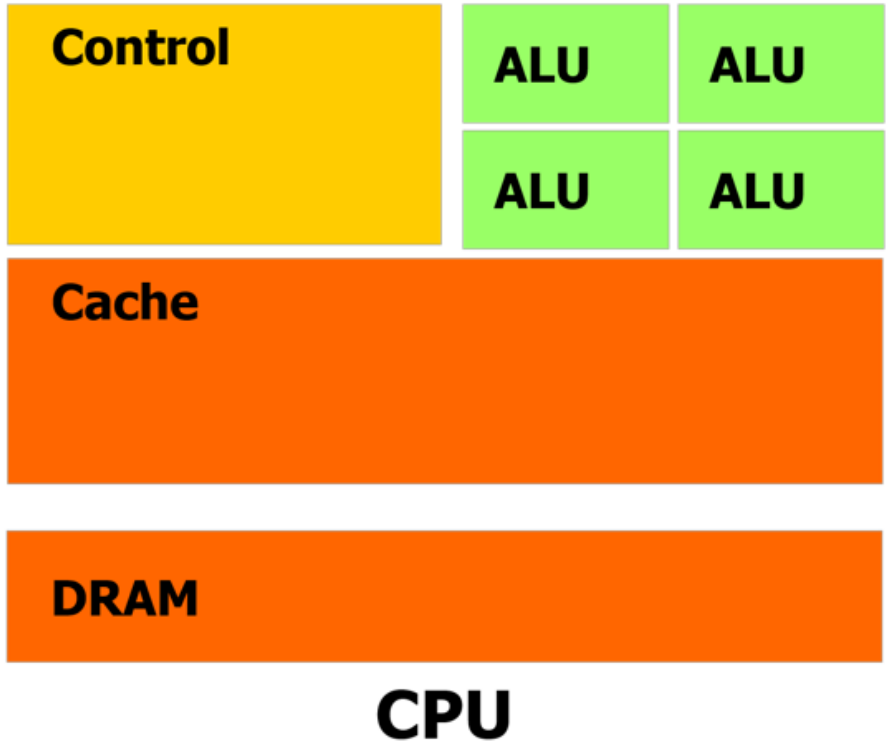
GPU

ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU

CPU

- **Caches grandes**
  - Permiten bajar la latencia en los accesos a memoria.
- **Unidad de control compleja**
  - Branch prediction
  - Data forwarding
- **ALUs complejas**
  - Reducen latencia de las operaciones

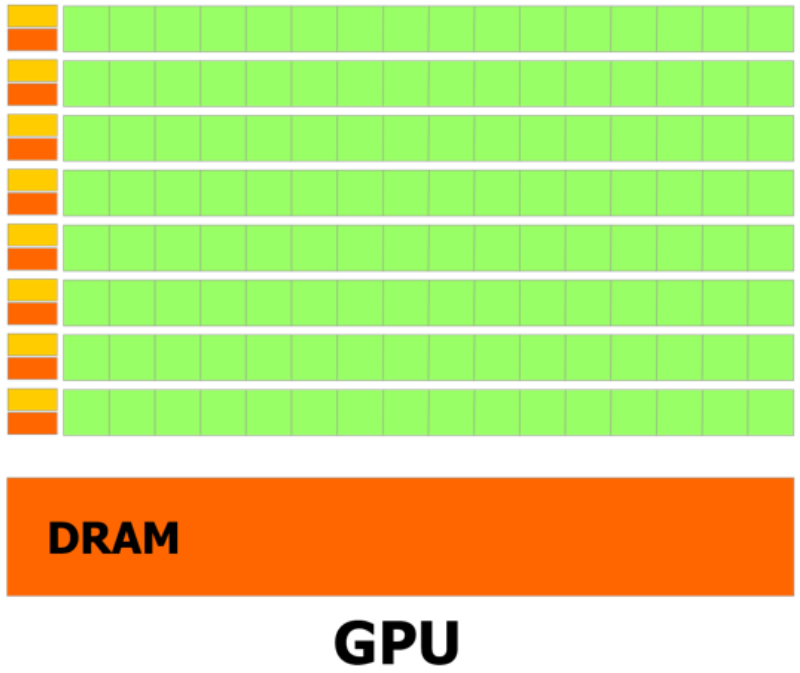


ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU

GPU

- **Caches pequeñas**
  - Potenciar ancho banda memoria.
- **Unidad de control simple**
  - No Branch prediction
  - No Data forwarding
- **Muchas ALUs simples**
  - Altamente segmentadas para aumentar el ancho de banda de la memoria
- **Requiere un número muy elevado de hilos para ocultar las latencias**



ARQUITECTURA HARDWARE

DIFERENCIAS ENTRE CPU Y GPU

- **Los hilos en la GPU son muy ligeros.**
  - Se necesita muy poco tiempo para crear/destruir hilos.
- **La GPU necesita muchos hilos para ser eficiente.**
  - Una CPU multi-núcleo solo necesita unos pocos hilos.
- **El tiempo de acceso a memoria en la GPU es alto.**
  - En la CPU el ancho de banda es menor.

ARQUITECTURA HARDWARE

COMPARACIÓN ENTRE CPU Y GPU

- **Intel Core i7-12700**
  - 12 núcleos MIMD
  - Pocos registros, cache multi-nivel
  - **70 GB/s** ancho de banda hacia la memoria principal

- **NVIDIA GTX4080**
  - 9728 núcleos, organizados en 76 unidades SM cada una con 128 núcleos.
  - Muchos registros, inclusión cachés nivel 1 y 2.
  - 5 GB/s ancho de banda hacia el procesador HOST.
  - **700 GB/s** ancho de banda memoria tarjeta gráfica.

ARQUITECTURA HARDWARE

STREAMING MULTIPROCESSORS

INSTRUCTION CACHE/DECODER

SCHEDULER

CUDA CORES

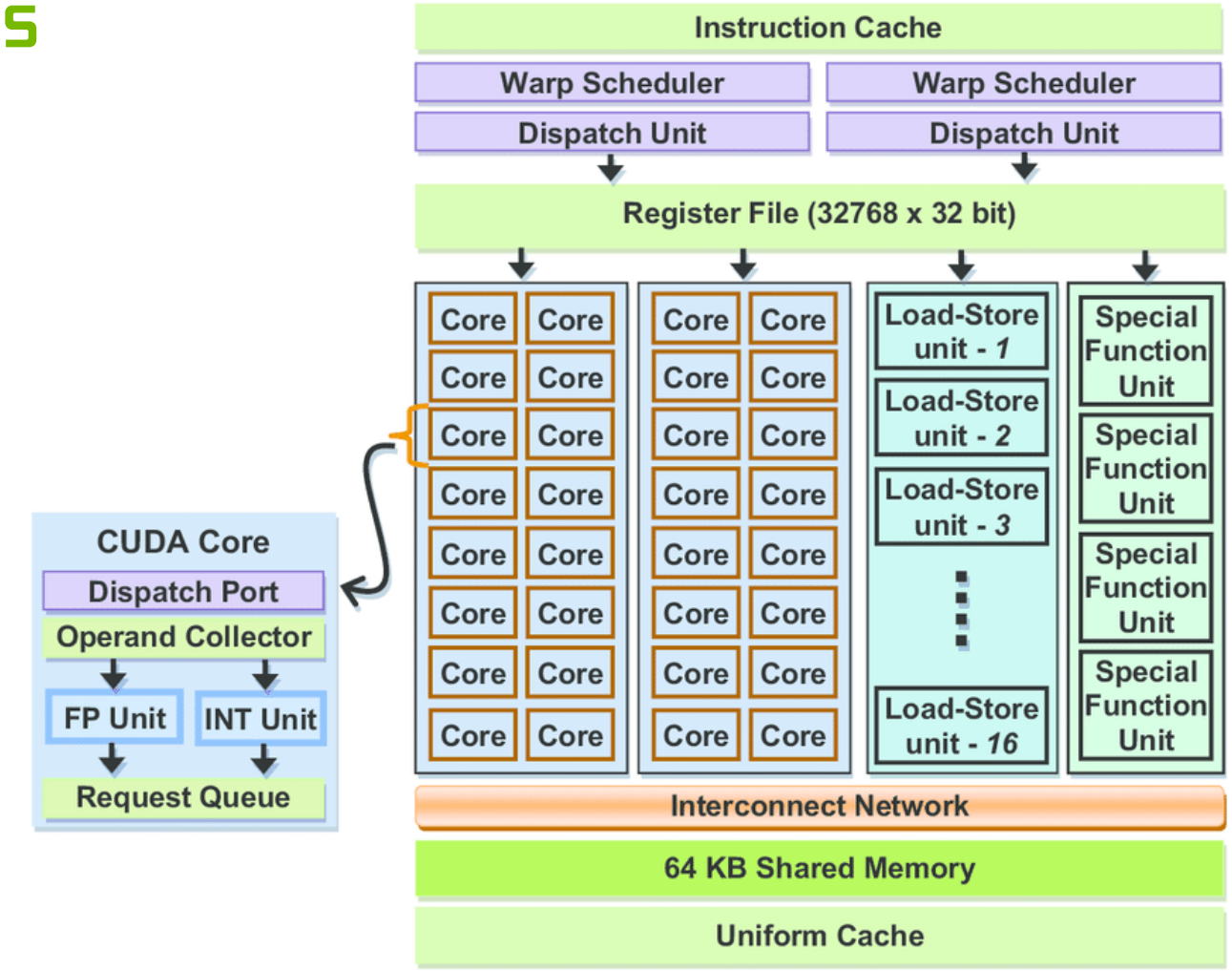
UNIDADES LOAD-STORE

UNIDADES SFU

MEMORIA COMPARTIDA

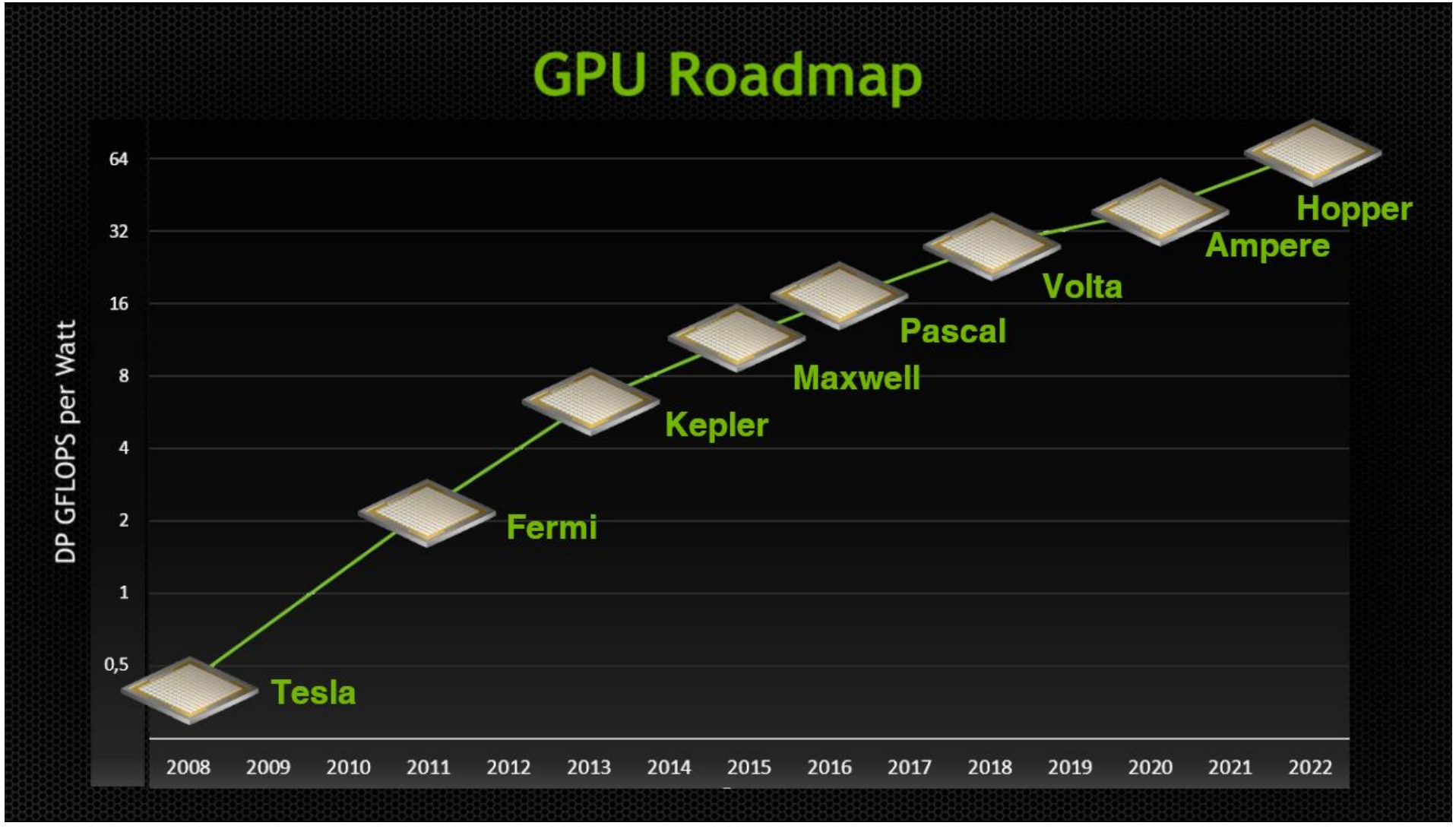
CACHÉS DE TEXTURAS

ARCHIVO DE REGISTROS



ARQUITECTURA HARDWARE

EVOLUCIÓN DE MICROARQUITECTURAS



ARQUITECTURA HARDWARE

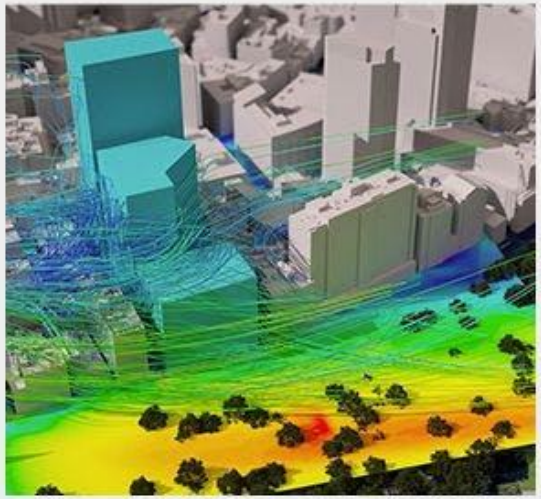
GPU TECHNOLOGY CONFERENCE 2024

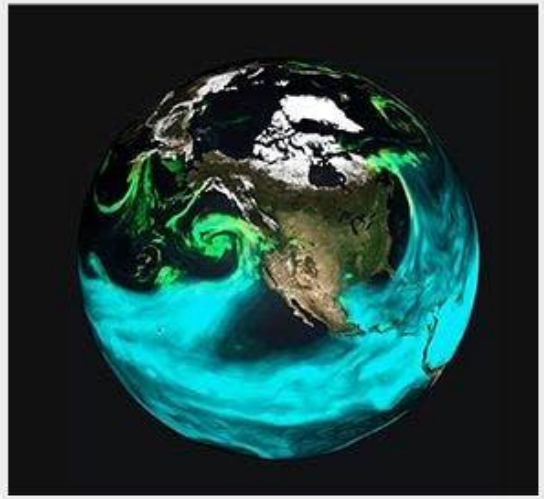
March 18, 2024

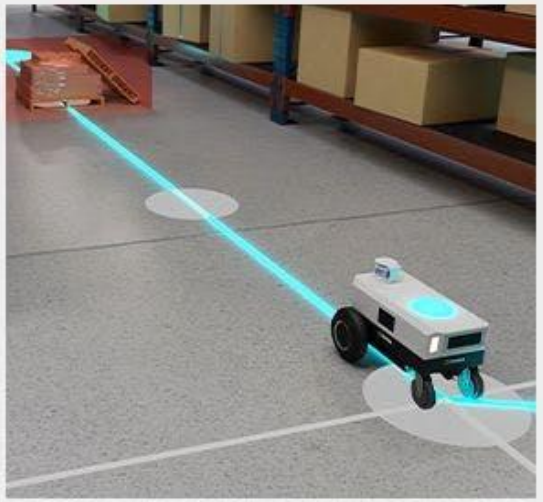
March 20-23

NVIDIA GTC

The Conference for the Era of AI and the Metaverse







ARQUITECTURA SOFTWARE

CUDA

- **CUDA (Compute Unified Device Architecture)**
  - Hace referencia tanto a un compilador como a un conjunto de herramientas de desarrollo creadas por NVIDIA.

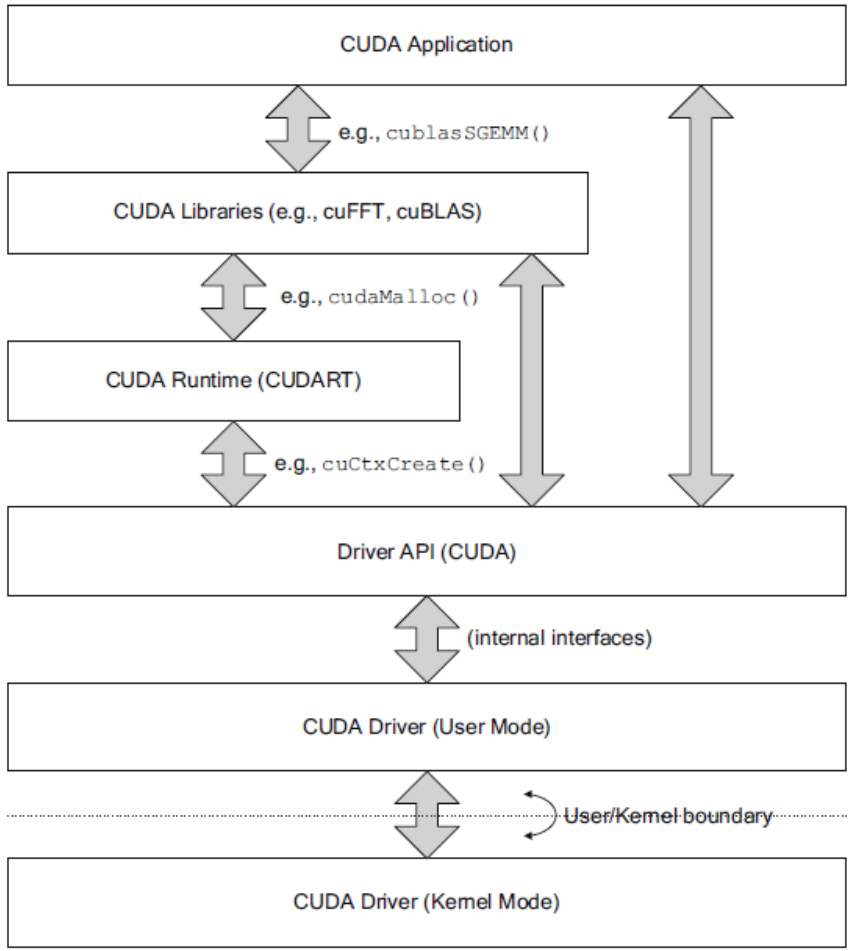
- **Basado en C/C++** con algunas extensiones

- Soporte C++ , Fortran. Wrappers para otros lenguajes: .NET, Python, Java…

- **Gran cantidad de ejemplos** y buena documentación, lo cual reduce la curva de aprendizaje para aquellos con experiencia en lenguajes como OpenMPI y MPI.
- Extensa comunidad de usuarios en los **foros de NVIDIA**.

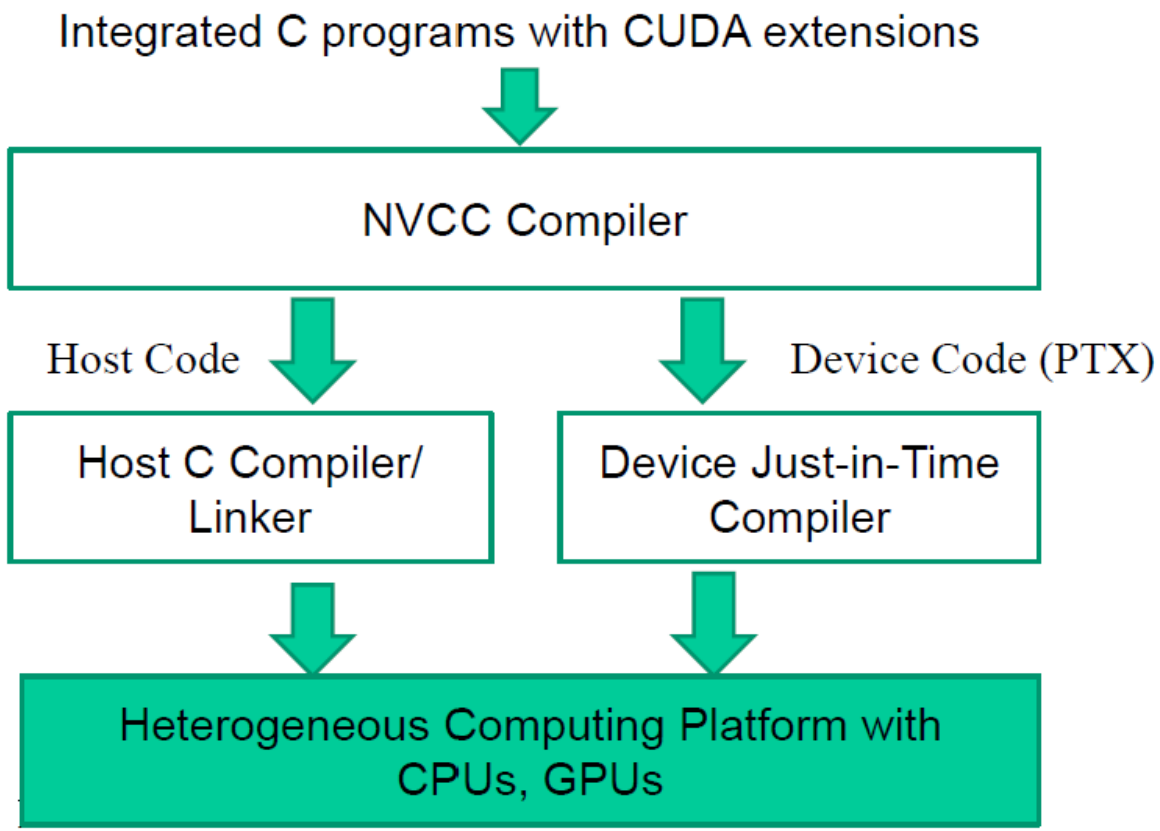
ARQUITECTURA SOFTWARE

STACK CUDA



ARQUITECTURA SOFTWARE

COMPILACIÓN



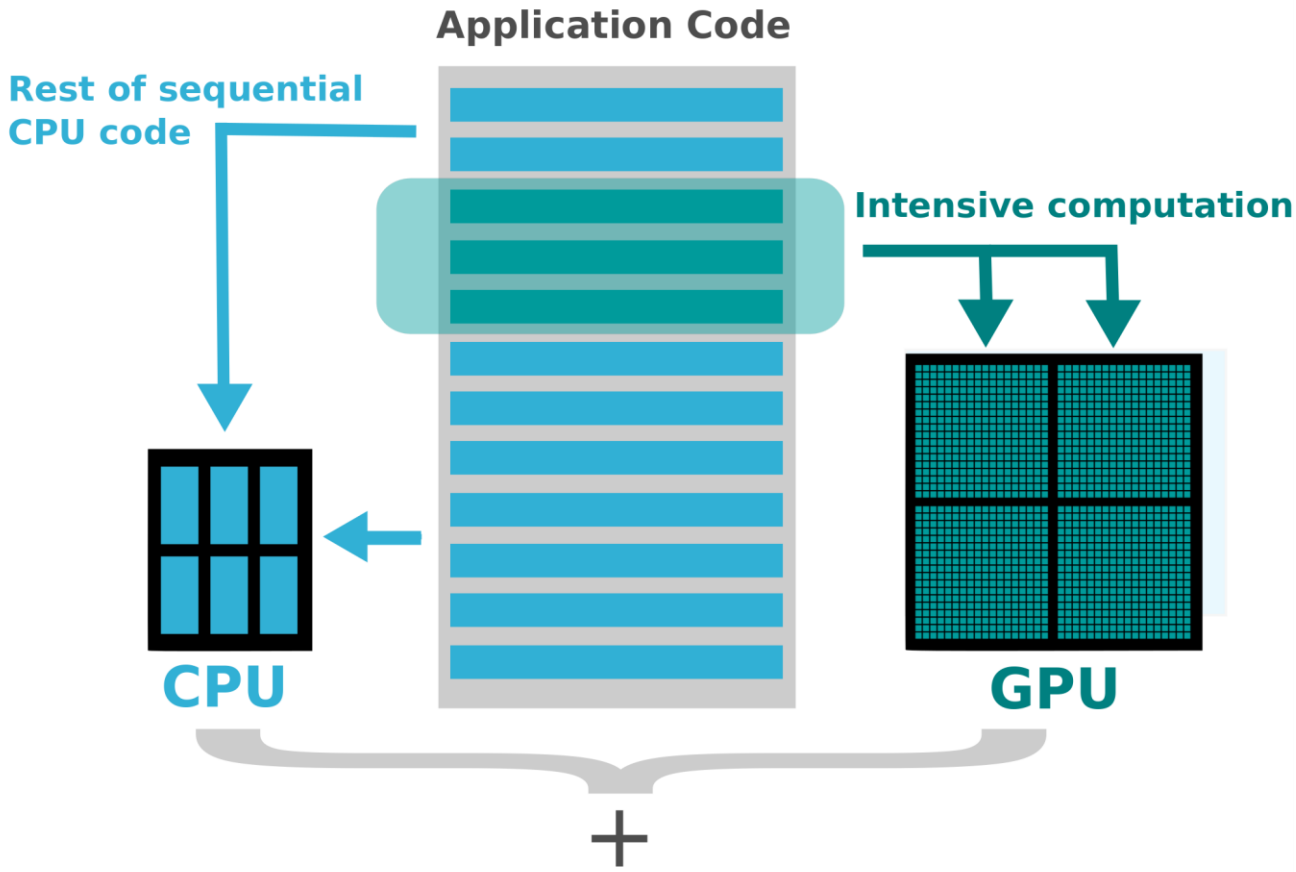
ARQUITECTURA SOFTWARE

COMPUTACIÓN HETEROGÉNEA

Enfoque que utiliza **múltiples tipos de procesadores especializados**, incluyendo GPUs, para realizar tareas informáticas.

Permite la **asignación de tareas específicas a los procesadores más adecuados** para su ejecución, lo que mejora el rendimiento y la eficiencia.

La GPU es clave, ya que es muy eficiente en la resolución de problemas que involucran cálculos intensivos.



ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

HILO: es la unidad de ejecución en la GPU, que puede realizar cálculos y acceder a la memoria.

KERNEL: es una función que al ejecutarse lo hará en una gran cantidad de hilos y en la GPU.

BLOCK: es una agrupación de hilos en 1D, 2D o 3D. Cada bloque se ejecuta sobre un único SM, pero un SM puede tener asignados varios bloques para ejecución.

GRID: es una forma de estructurar los bloques, bien en 1D, 2D o 3D

ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

INVOCACIÓN DE KERNEL:

```
kernel_routine<<<grid_dim, block_dim>>> (args...);
```

- **gridDim :** número de bloques. Tamaño del grid.
- **blockDim:** número de hilos que se ejecutan dentro de un bloque.
- **Args:** número limitado de argumentos, normalmente punteros a memoria de la GPU.

TAMAÑOS DE BLOQUE Y MALLA DEFINIDOS CON DIM3:

```
dim3 block_dim (32, 32, 1); // 1024 hilos en bloque de 32 x 32 x 1 (2D)
dim3 grid_dim (4, 4, 1); // 16 bloques en malla de 4 x 4 x 1 (2D)
```

ARQUITECTURA SOFTWARE

CONCEPTOS BÁSICOS

CADA HILO EJECUTA UNA COPIA DEL KERNEL, Y DISPONE DE LA SIGUIENTE INFORMACIÓN:

VARIABLES PASADAS COMO ARGUMENTO

- PUNTEROS A MEMORIA GPU
- VARIABLES SIMPLES POR VALOR (INT, FLOAT, CHAR, LÍMITE DE TAMAÑO)

CONSTANTES GLOBALES EN MEMORIA GPU

VARIABLES ESPECIALES (DIM3) PARA IDENTIFICAR AL HILO:

gridDim (tamaño de la malla)
blockDim (tamaño de los bloques)
blockIdx (identificador de bloque) LOCAL PARA CADA BLOQUE
threadIdx (identificador de hilo) LOCAL PARA CADA BLOQUE

ARQUITECTURA SOFTWARE

FLUJO DE EJECUCIÓN

En el nivel más alto encontramos un proceso sobre la CPU (Host) que realiza los siguientes pasos:

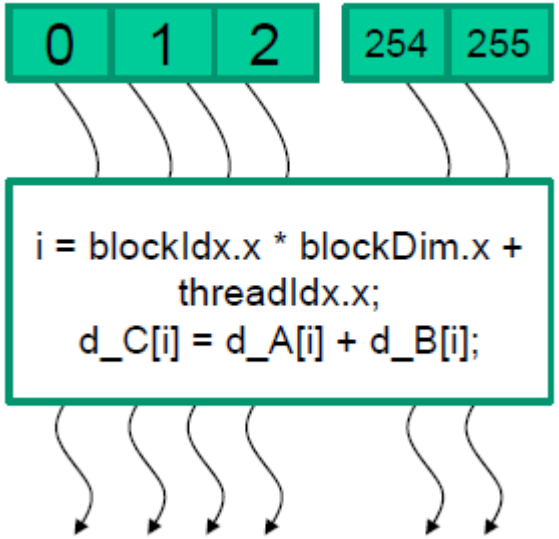
1. Inicializa GPU
2. Reserva memoria en la parte host y device
3. Copia datos desde el **host** hacia la memoria **device**
4. Lanza la ejecución de múltiples copias del **kernel**
5. Copia datos desde la memoria **device** al **host**
6. Se repiten los pasos 3-5 tantas veces como sea necesario
7. Libera memoria y finaliza la ejecución proceso maestro

ARQUITECTURA SOFTWARE

EJECUCIÓN DE UN KERNEL

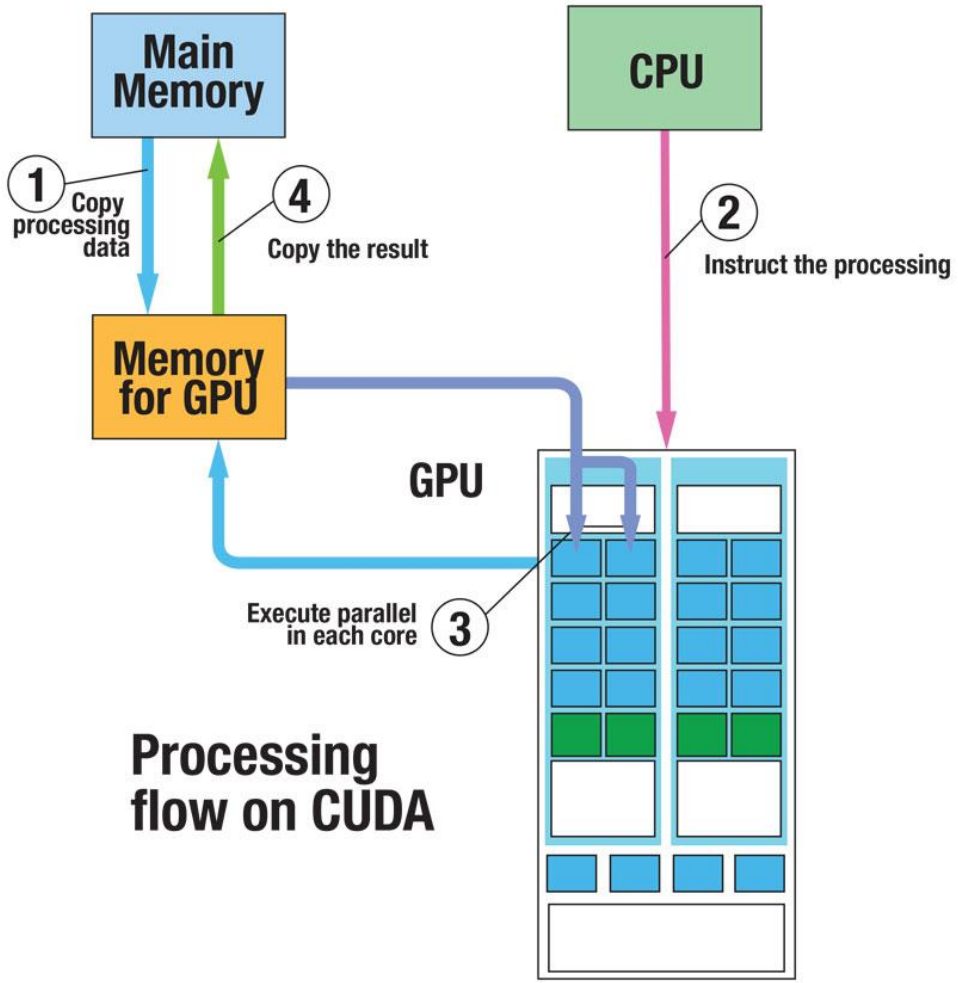
Un kernel CUDA se ejecuta sobre un grid de hilos:

- Todos los threads del grid ejecutan el mismo código
- Cada hilo tiene sus índices y lo utiliza para direccionar la memoria y realizar su lógica.



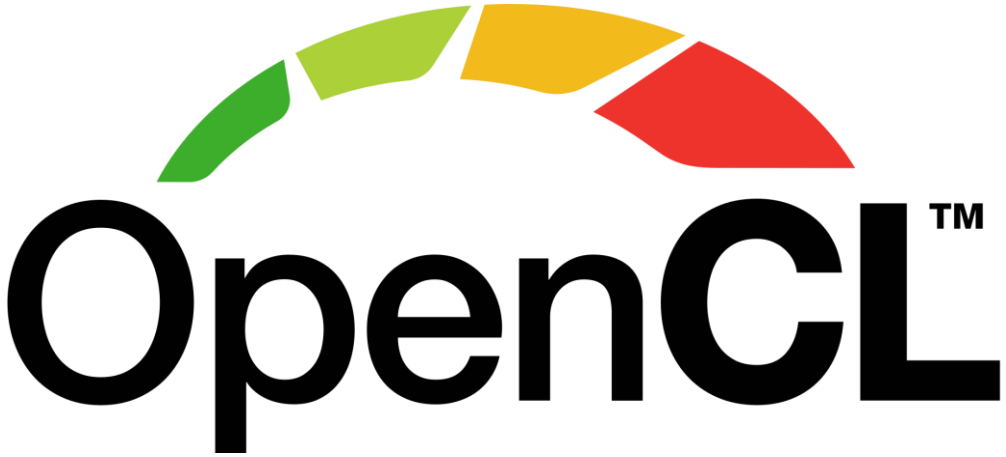
ARQUITECTURA SOFTWARE

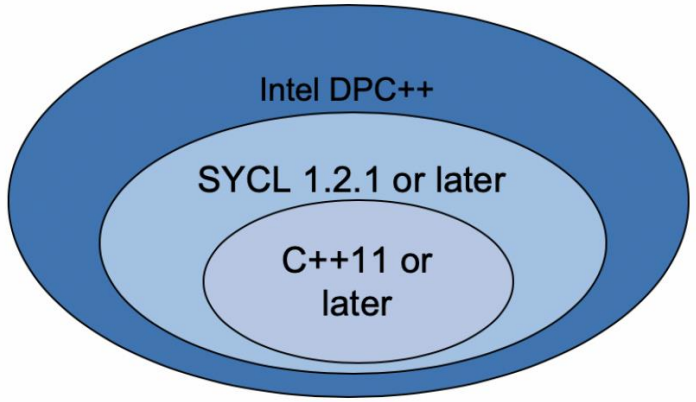
FLUJO DE EJECUCIÓN



ALTERNATIVAS

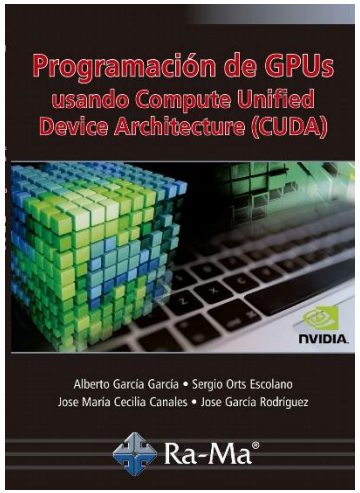
OTROS MODELOS DE COMPUTACIÓN HETEROGÉNEA





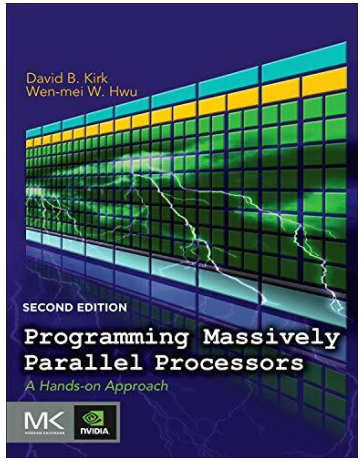
BIBLIOGRAFÍA RECOMENDADA

PARA APRENDER UN POQUITO MÁS



Programación de GPUs Usando Compute Unified Device Architecture (CUDA)

- Alberto García García, Sergio Orts Escolano, José Celilia Canales, José García Rodríguez

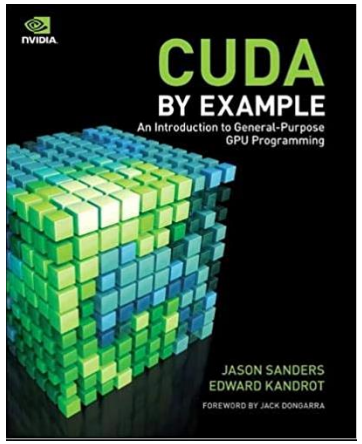


Programming Massively Parallel Processors (A Hands-on Approach)

- David B. Kirk, Wen-mei W. Hwu

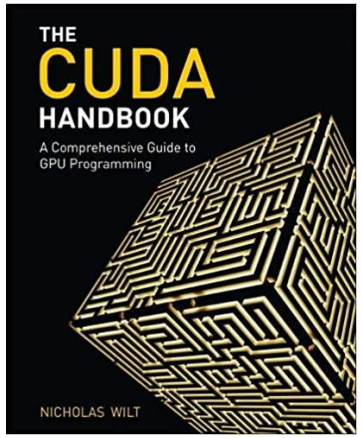
BIBLIOGRAFÍA RECOMENDADA

PARA APRENDER UN POQUITO MÁS



CUDA BY EXAMPLE (An Introduction to General-Purpose GPU Programming)

- Jason Sanders, Edward Kandrot



THE CUDA HANDBOOK (A Comprehensive Guide to GPU Programming)

- Nicholas Wilt

ALGUNA PREGUNTA?

MÁSTER EN INTELIGENCIA ARTIFICIAL

UNIVERSIDAD DE ALICANTE

David Mulero-Pérez <dmulero@dtic.ua.es>

Manuel Benavent-Lledó <mbenavent@dtic.ua.es>

José García-Rodríguez <jgarcia@dtic.ua.es>