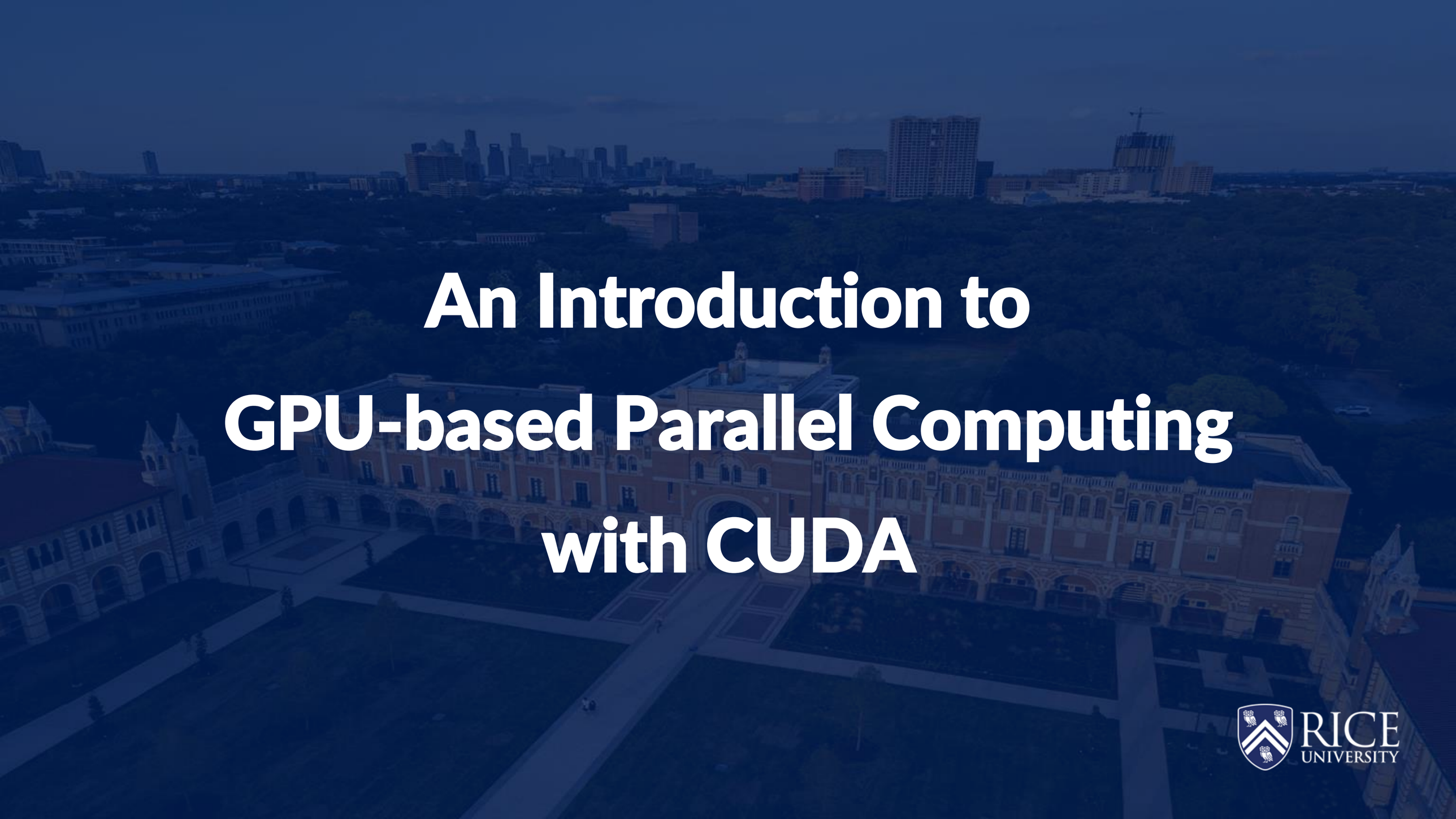


An Introduction to GPU-based Parallel Computing with CUDA

RICE UNIVERSITY

The Rise of Nvidia

The key moments leading to its rise from a market cap of $600M to $3.5T.

Nvidia's Quarterly Revenue (since IPO)

Nvidia's RIVA 128 is released, its first commercially successful chip. 1997

Nvidia's first GPU, the GeForce 256, is released.

Nvidia releases Tesla, their first HPC-focused GPU architecture. 2007

AlexNext won the ImageNet Visual Recognition Challenge, using CUDA and Nvidia GPUs. 2012

Nvidia acquires Mellanox, the leader in InfiniBand networking. 2020

ChatGPT released. 2022

DGX Cloud released. 2023

Nvidia surpasses $35B in quarterly revenue, up 5x in just 6 quarters. 2024

Nvidia is Founded

Nvidia's first chip, the NV1, is released. 1995

Nvidia IPOs at a $600M valuation. 1999

Programmable shaders are released with GeForce 3. 2001

CUDA is released, introducing GPU computing. 2006

The Kepler architecture is released.

The Fermi architecture is released. 2010

Maxwell architecture released. 2014

OpenAI founded. 2015

Pascal architecture released. 2016

Volta architecture released. 2017

Turing architecture released. 2018

Ampere architecture released. 2019

Hopper architecture released, Grace CPU chip announced, and DGX server released.

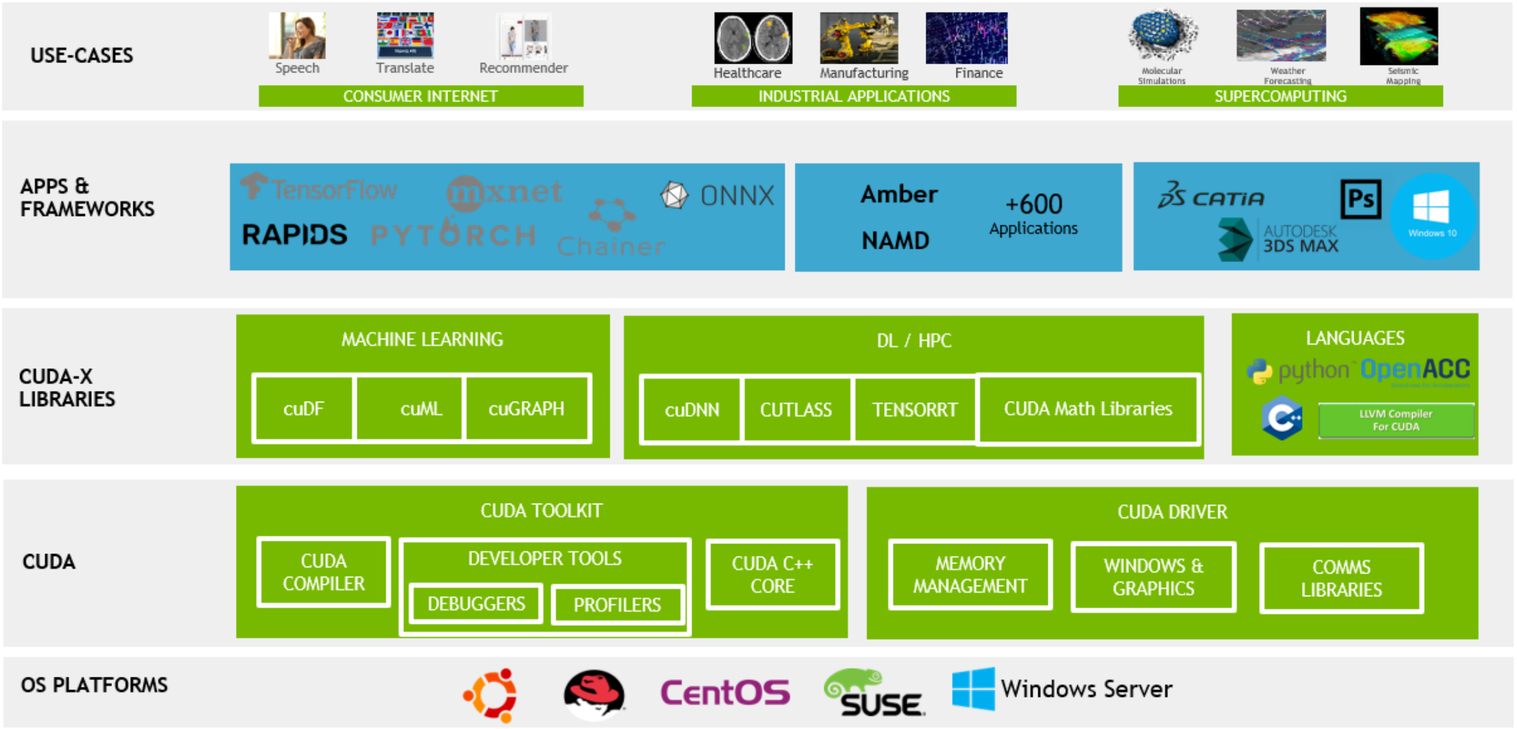
1993 1997 2001 2005 2009 2013 2017 2021 2024

Passed $1B in annual revenue

Passed $10B in annual revenue

What is CUDA?

- Compute Unified Device Architecture.
- NVIDIA's parallel computing platform and programming model.
- Allows using a GPU for general-purpose computing (GPGPU).
- Provides C/C++ language extensions to write scalable parallel programs.



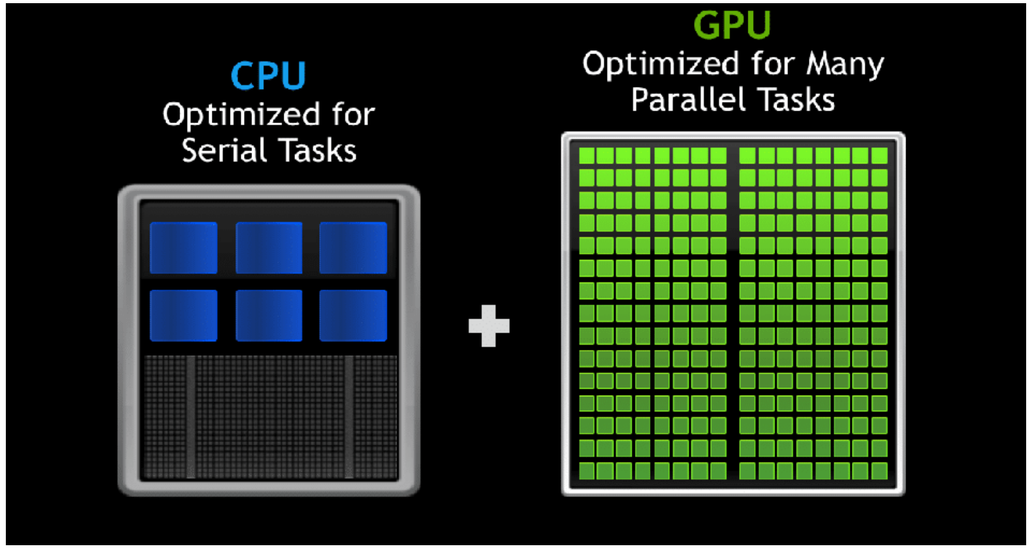
CPU vs. GPU: A Tale of Two Processors

Host (CPU) - The "CEO"

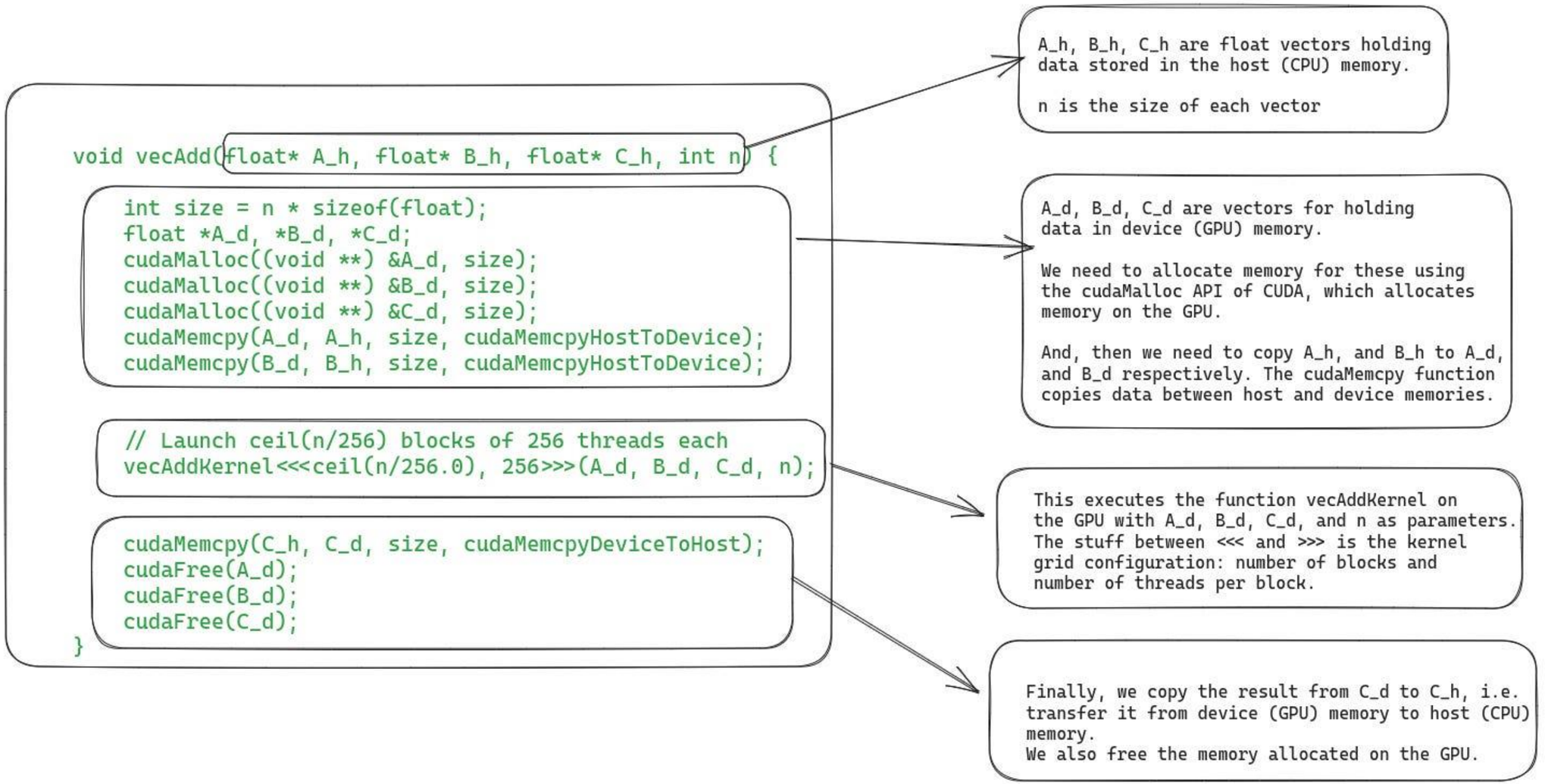
- Fewer, more powerful cores.
- Optimized for sequential task latency.
- Manages the system, runs the OS, and orchestrates tasks.

Device (GPU) - The "Workforce"

- Thousands of smaller, efficient cores.
- Optimized for parallel task throughput.
- Executes highly parallel "kernels" at the direction of the Host.



How to run GPU kernel function on Host



The Parallel Hierarchy

Thread

The smallest unit of execution. Executes one instance of the kernel. Identified by threadIdx.

Block

A group of threads. Threads in a block can cooperate (share memory, synchronize). Identified by blockIdx.

Grid

A group of blocks. Represents the full kernel launch. Identified by gridDim.

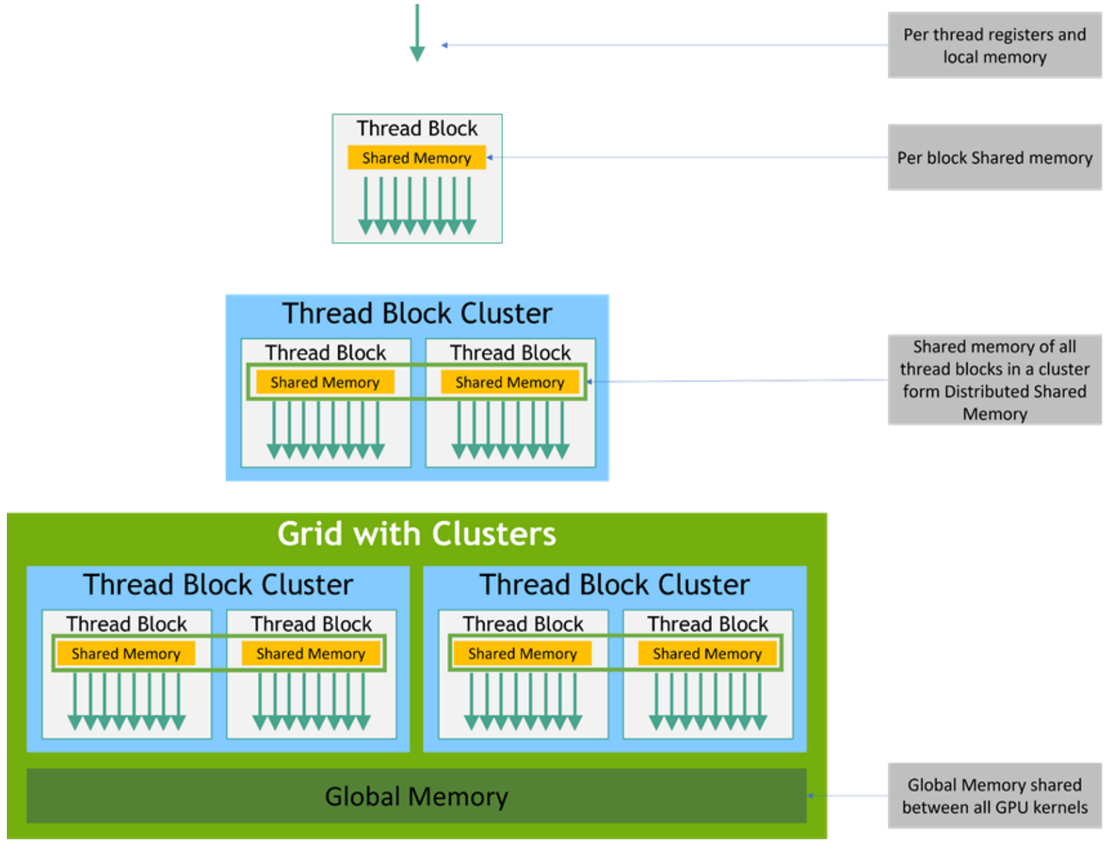
Visualizing the GPU Computing Hierarchy

The kernel launch parameters << >> define the execution geometry.

- gridSize: Number of blocks in the grid.
- blockSize: Number of threads in each block.

Total Threads = gridSize * blockSize

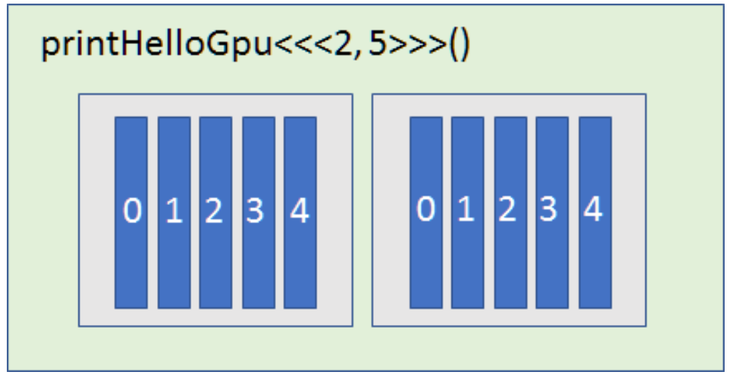
This structure maps threads to data, allowing for massive parallelism.

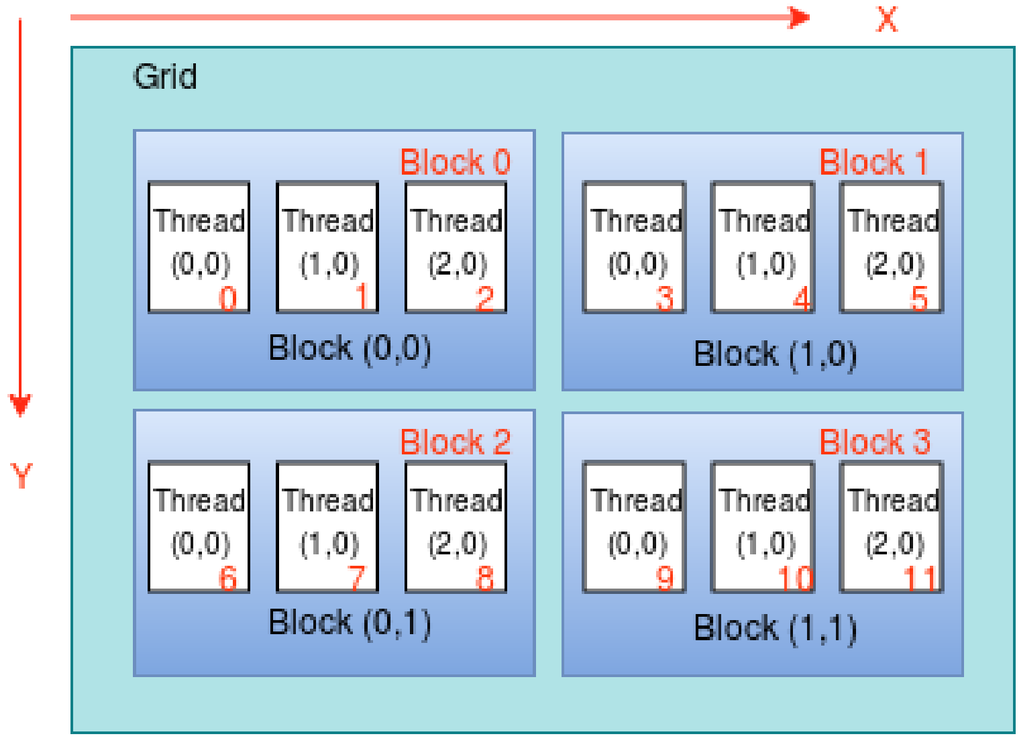


Identifying the Thread (Thread Indexing)

How does each thread know what data to process? We must calculate a unique global index.

- threadIdx.x: Thread's ID **within its block.**
- blockIdx.x: Block's ID **within the grid.**
- blockDim.x: Number of threads **in one block.**





Host vs. Device Memory

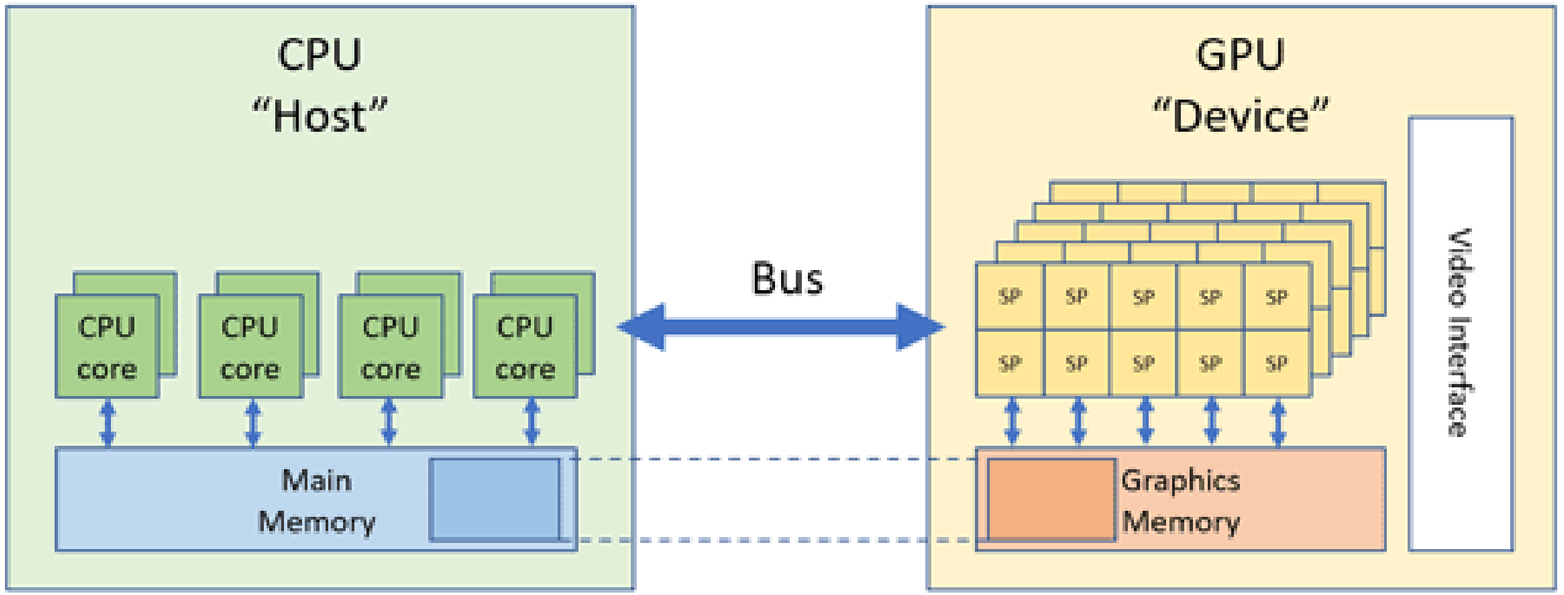
Host (CPU) Memory

- System RAM.
- Managed by malloc(), new.
- Accessible by the CPU.
- (Host Pointers: int *h_data)

Device (GPU) Memory

- GPU's onboard VRAM (High-Bandwidth).
- Managed by cudaMalloc().
- Accessible by the GPU.
- (Device Pointers: int *d_data)

Host vs. Device Memory



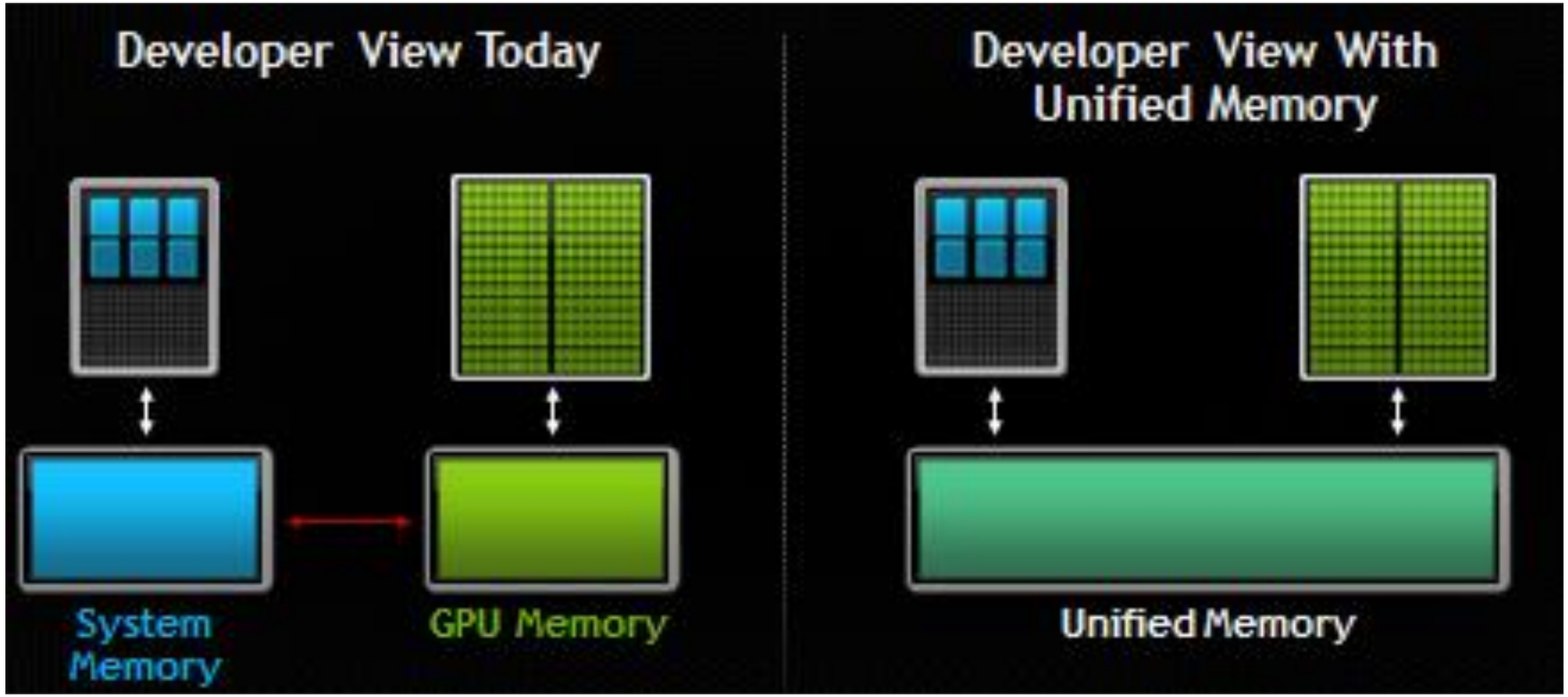
Supported by all current Nvidia GPUs

cudaMallocHost()
Host and Device can access memory at same address

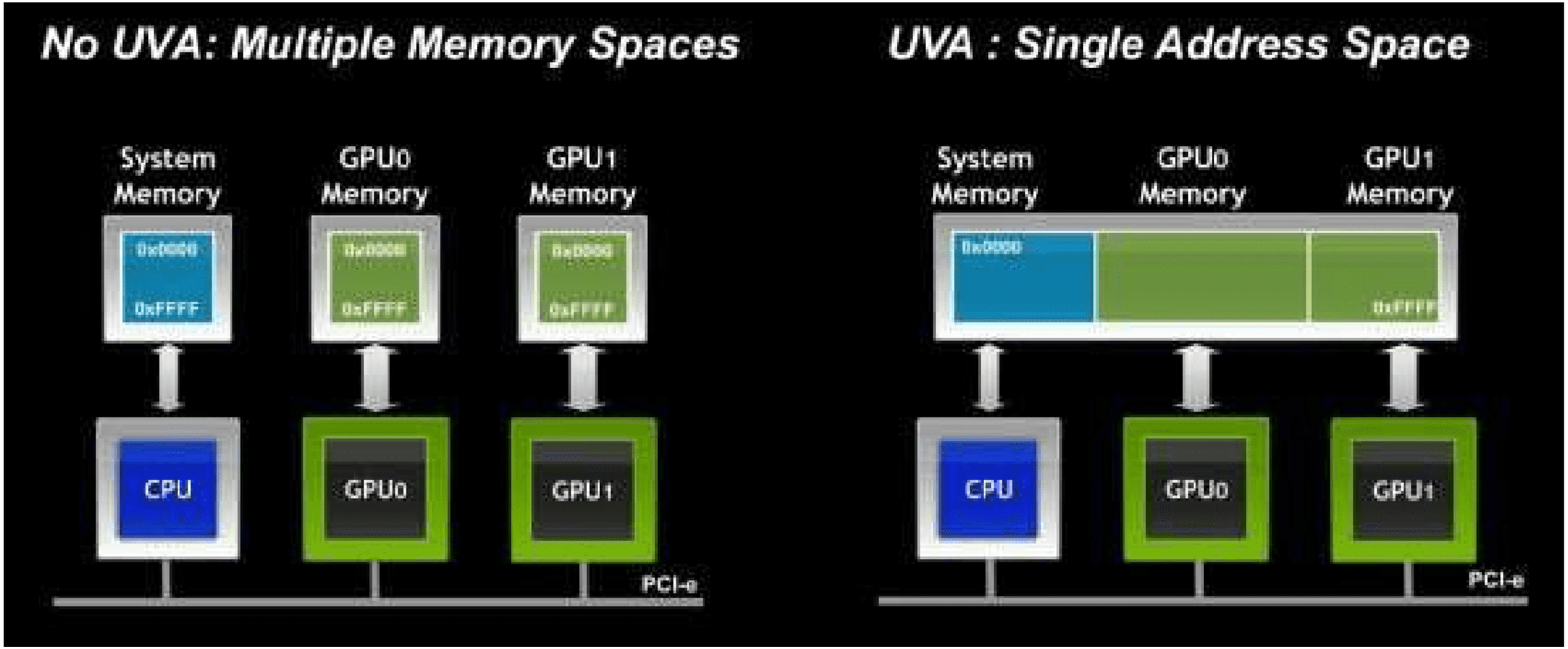
cudaFreeHost()
memory freed on both Host and Device

Host memory pinned in CPU (cannot be swapped),
copied to Device on access from Device
copied back to Host on access from CPU

Program with Unified Memory



Program with Unified Virtual Address



Verification is Critical!!!

- How do we know the GPU result is correct?
- **Never trust, always verify.**
    1. Run the computation on the CPU (the "gold standard" sequential version).
    2. Run the computation on the GPU.
    3. Copy the GPU result back to the host.
    4. Compare the two result arrays element-by-element.
- Allow for small floating-point error (epsilon) due to different operation ordering.

Questions?

RICE UNIVERSITY