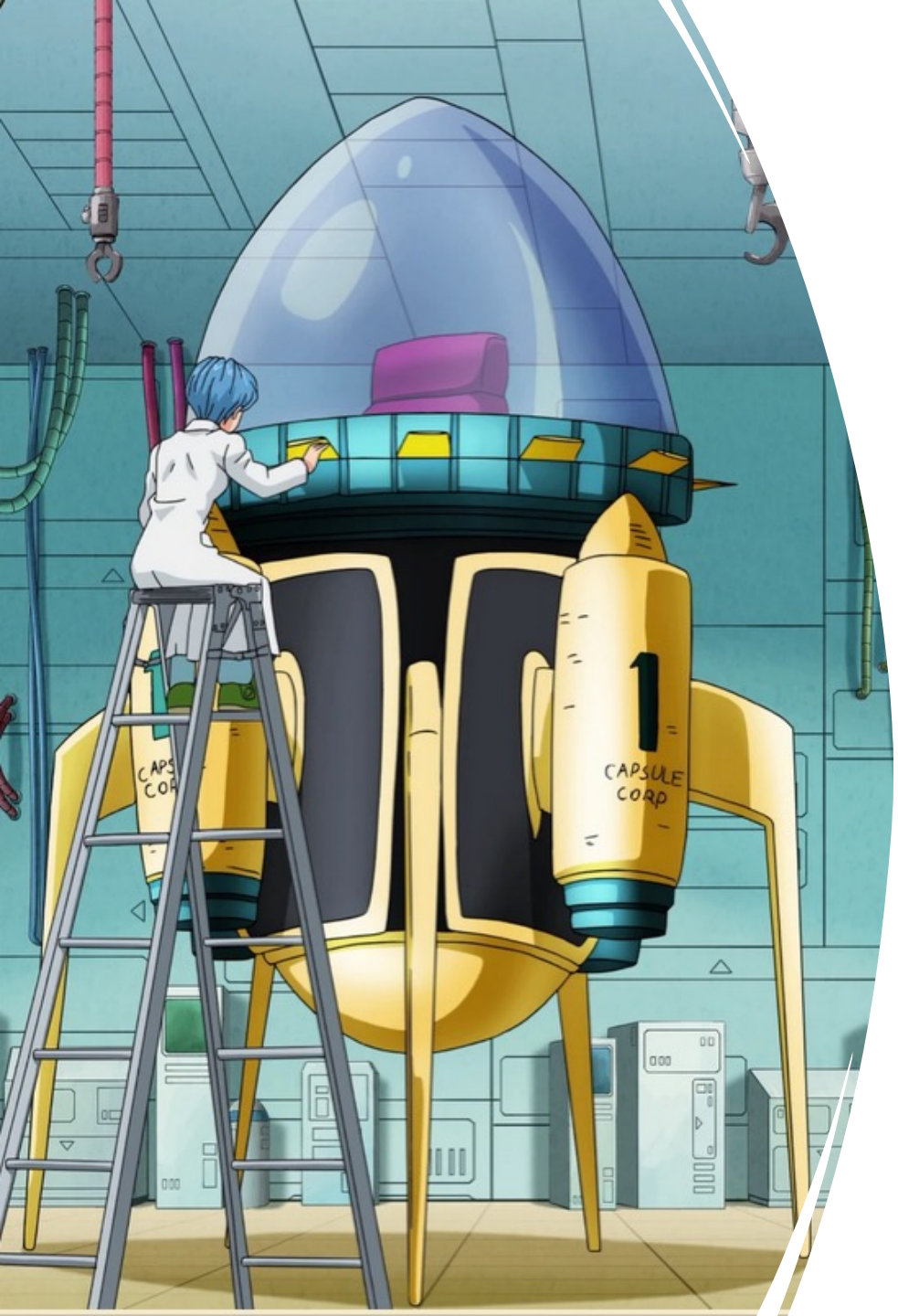
Introduction to GPU programming (part 1)

By Kevin Fotso



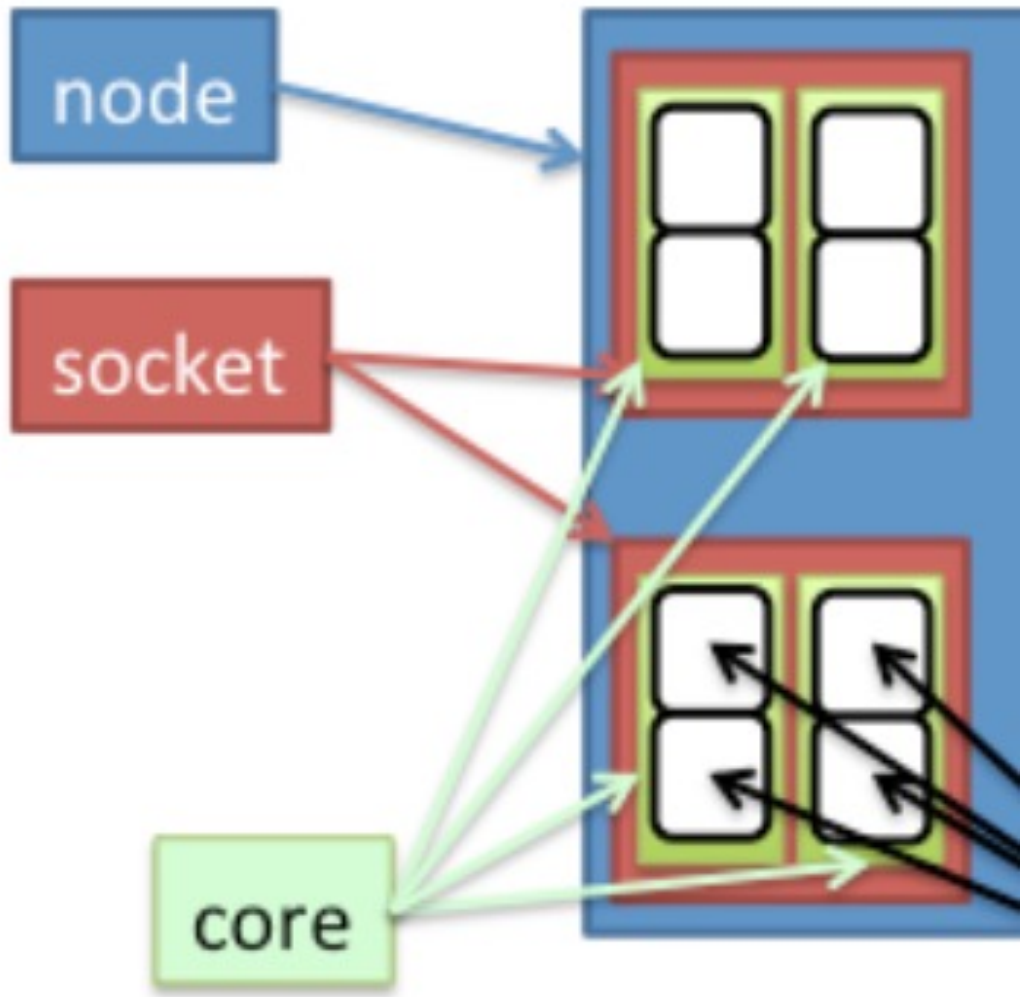
I- Part 1) Elements of computing

Parts of the processing chip (1)

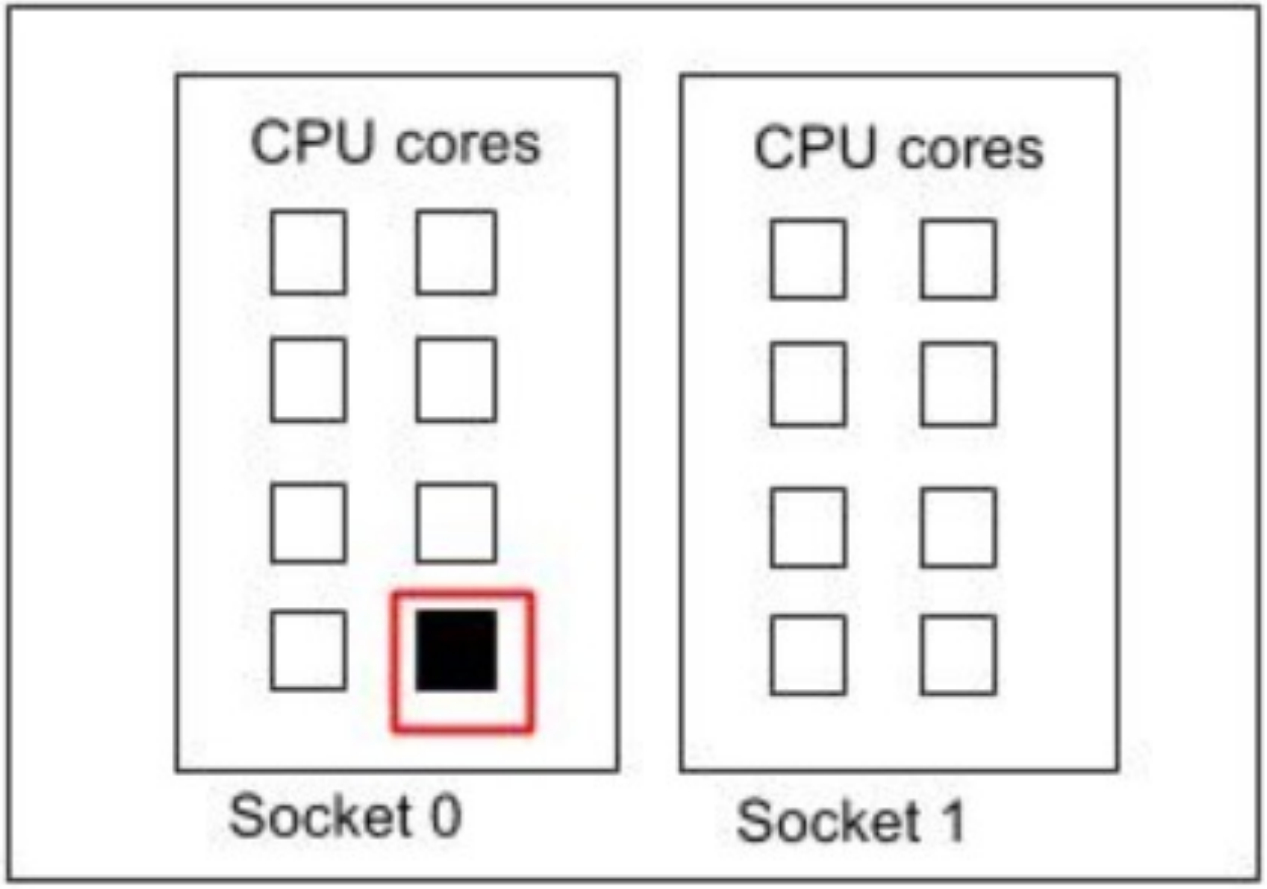
- A socket can be considered an entire computing chip in modern computers.
- Contains more than 1 core.
- On supercomputers a node usually contains more than 1 socket.

A core contains an Arithmetic and Logic Unit **(ALU)** + Registers

Parts of the processing chip (2)

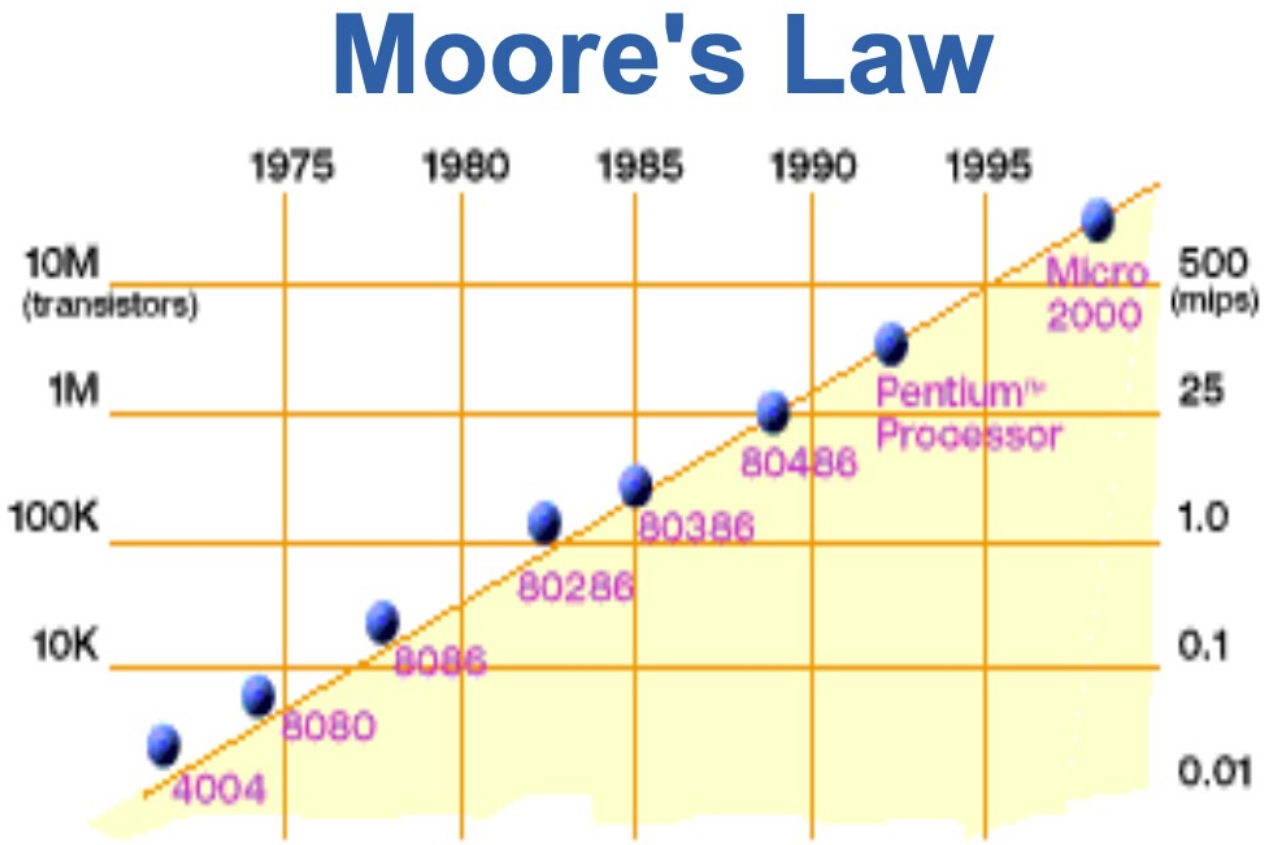


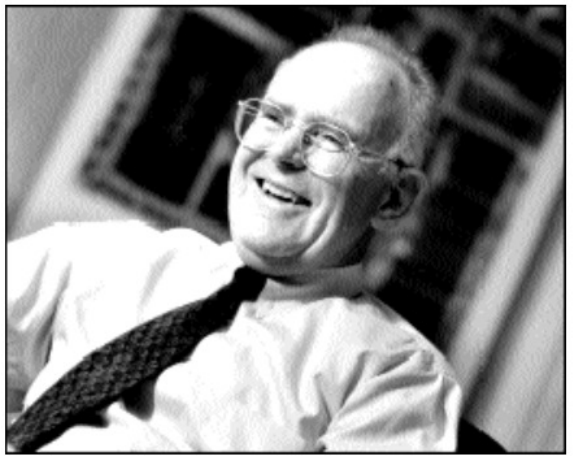
Parts of the processing chip (2)



Moore’s law (1)

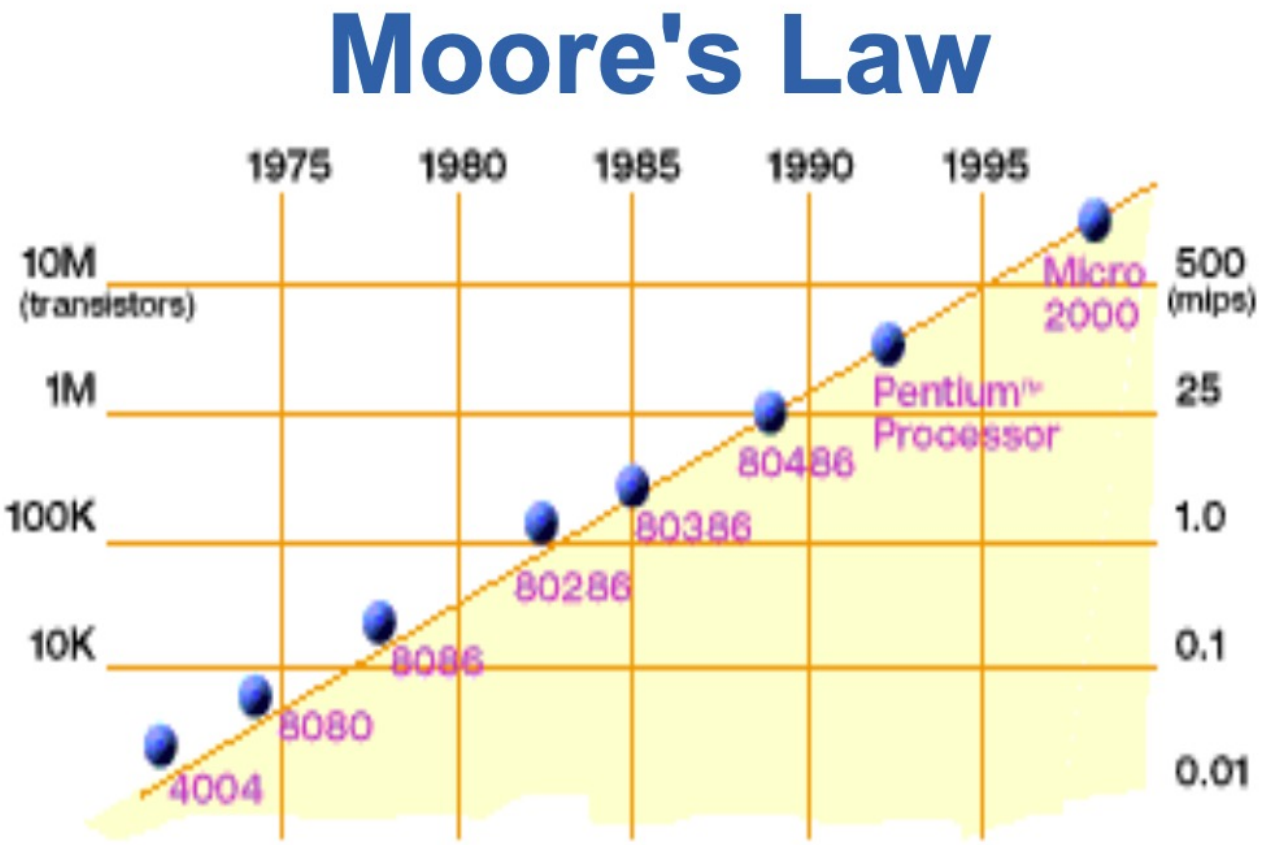
- Transistor count doubles in a chip every 18 months.

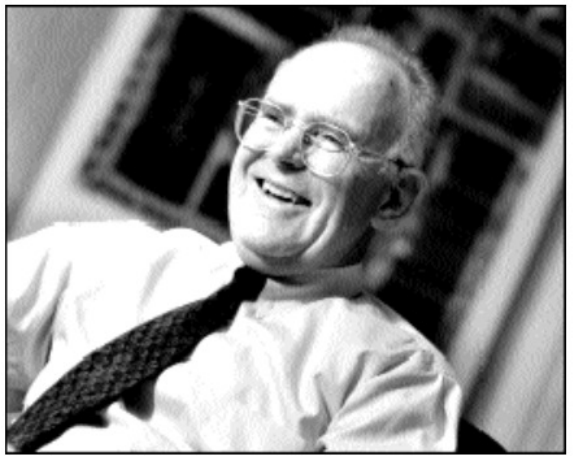




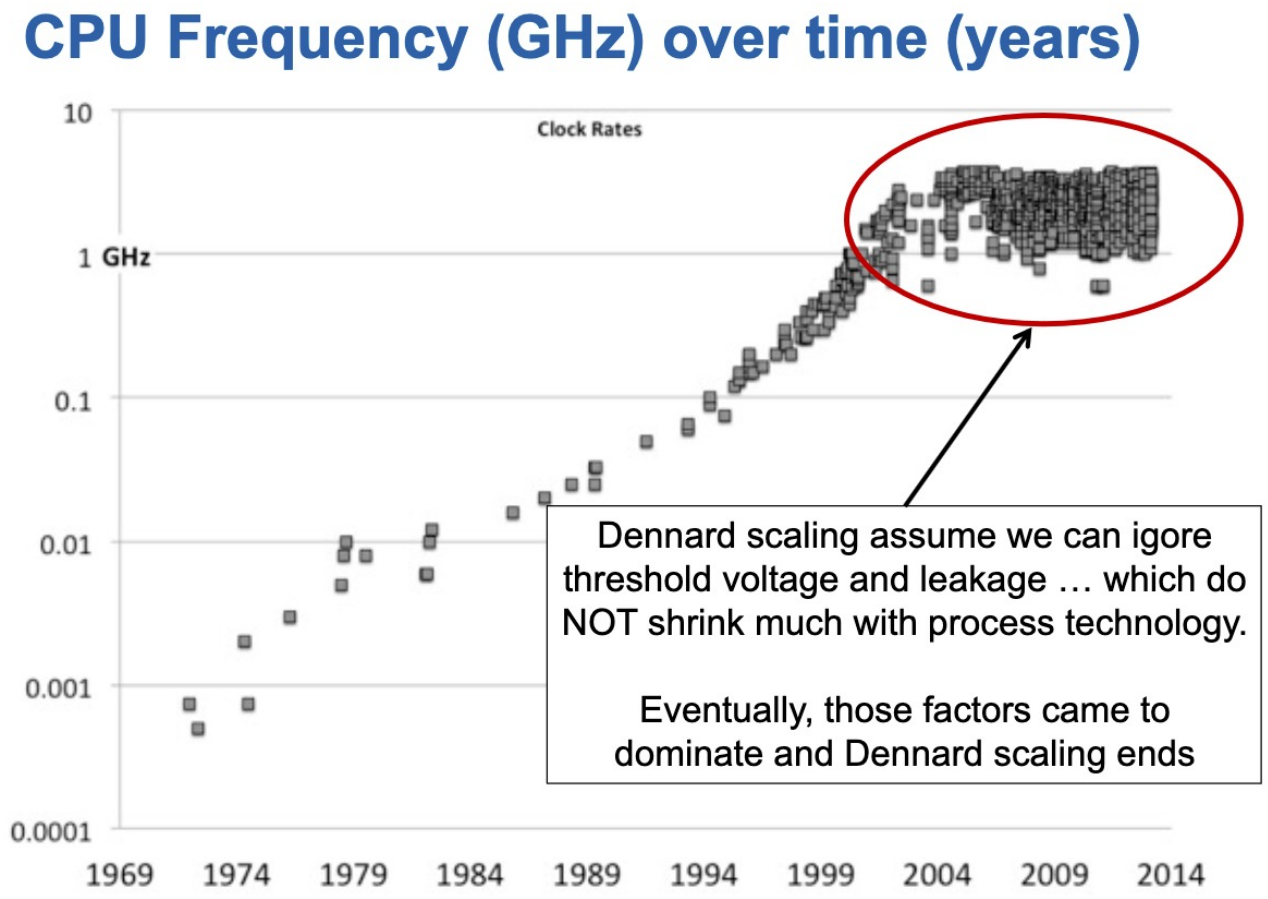
Moore’s law (2)

- Worked until around 2010 where it started to plateau



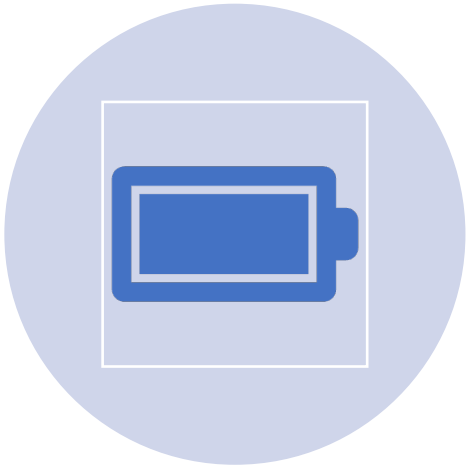


Moore's law plateau



- Number of cycles per seconds plateaued in the 2010s
- Why?

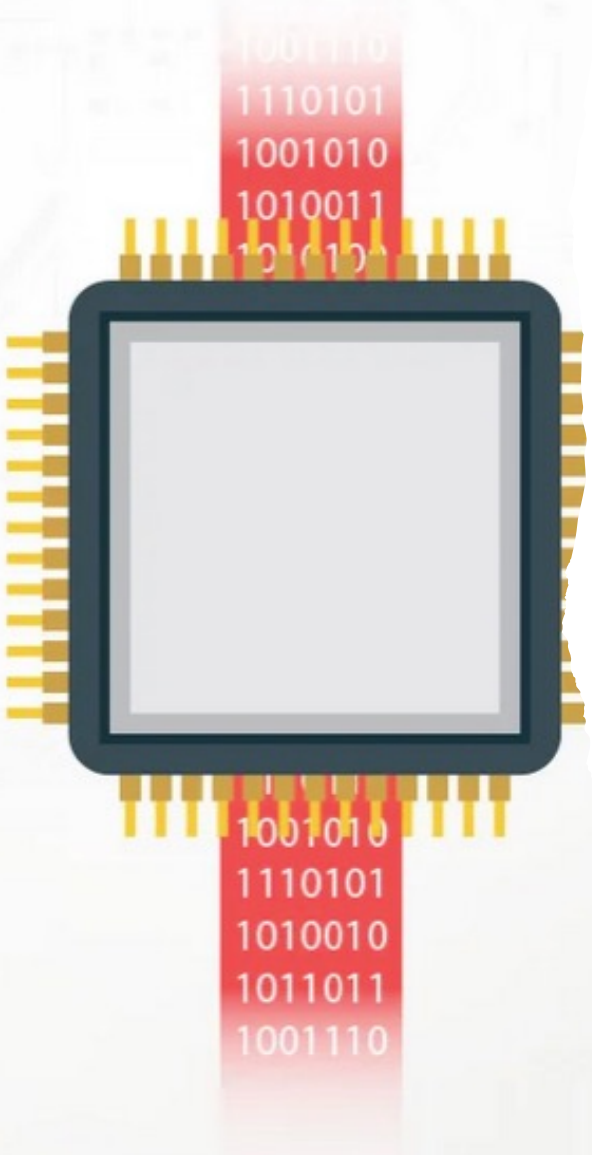
Why?



TRADITIONALLY, POWER DENSITY STAYS CONSTANT AS CHIPS FEATURE GOT SMALLER.

HOWEVER AROUND 2010, CLOCK FREQUENCY COULD NOT BE RAISED FURTHER BECAUSE THE HEAT DISSIPATION OR POWER WOULD HAVE GONE TOO FAR.

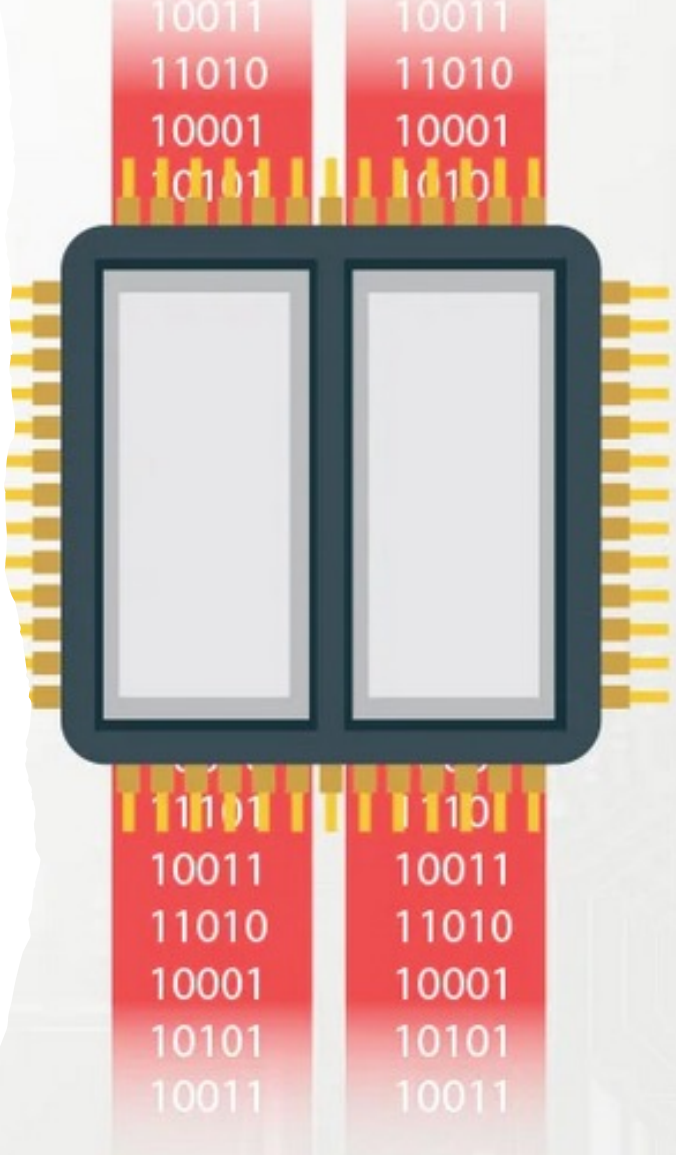
Increase need of parallelism



Single-core

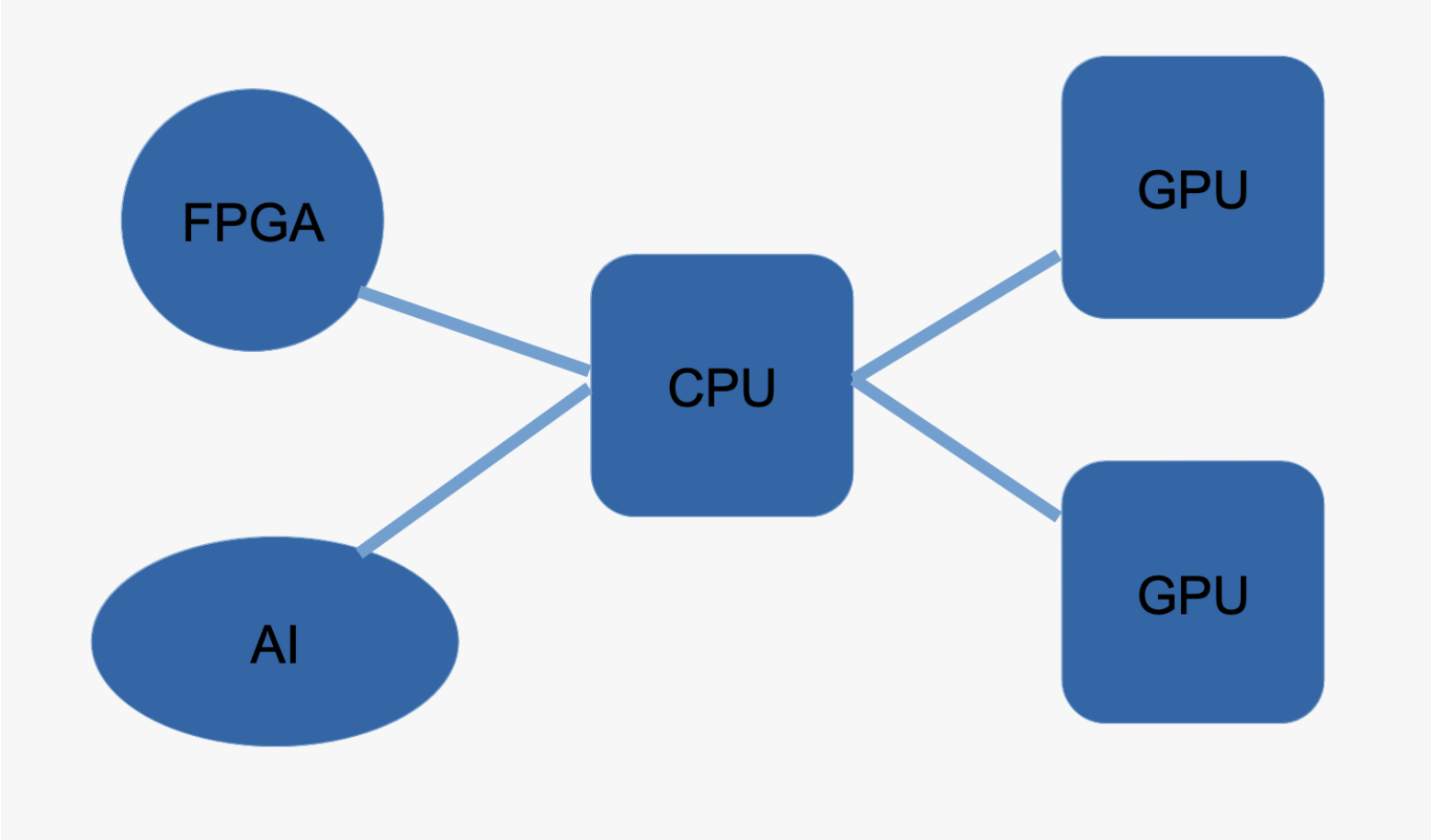
Reduce power by adding cores of small frequency

2 cores at half frequency f can achieve the same throuput as a single core of frequency f



Dual Core

Hence the rise of heterogeneous computing

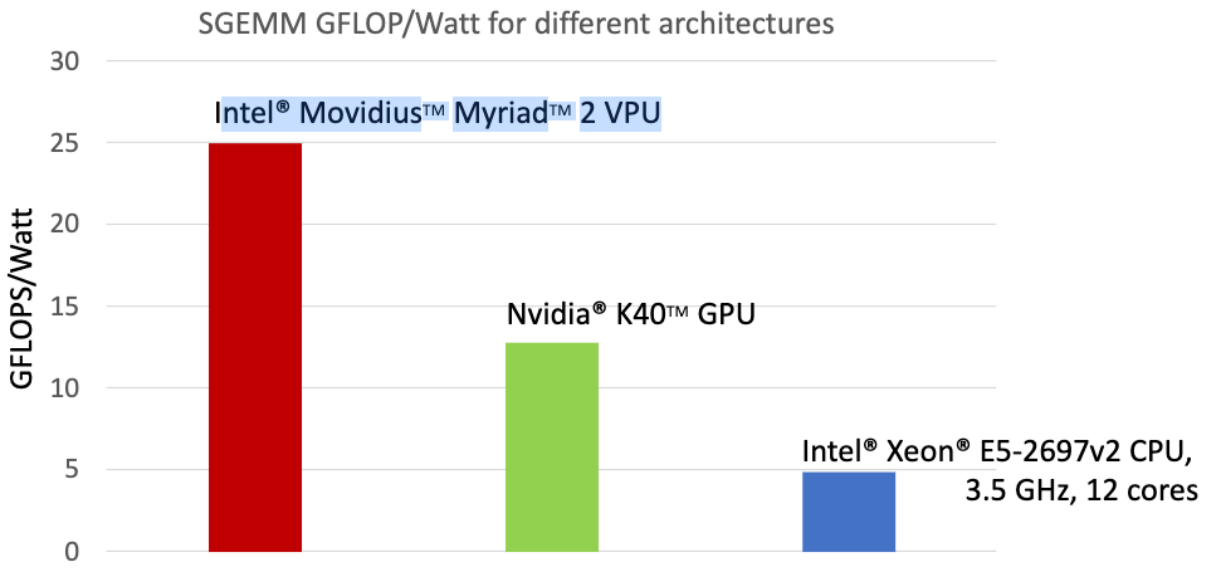


Source: https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2023/08/ATPESC-2023-Track-2D-Talk-1-Bagusetty-SYCL-1.pdf

Why care about accelerators?

If you care about power, the world is heterogeneous?

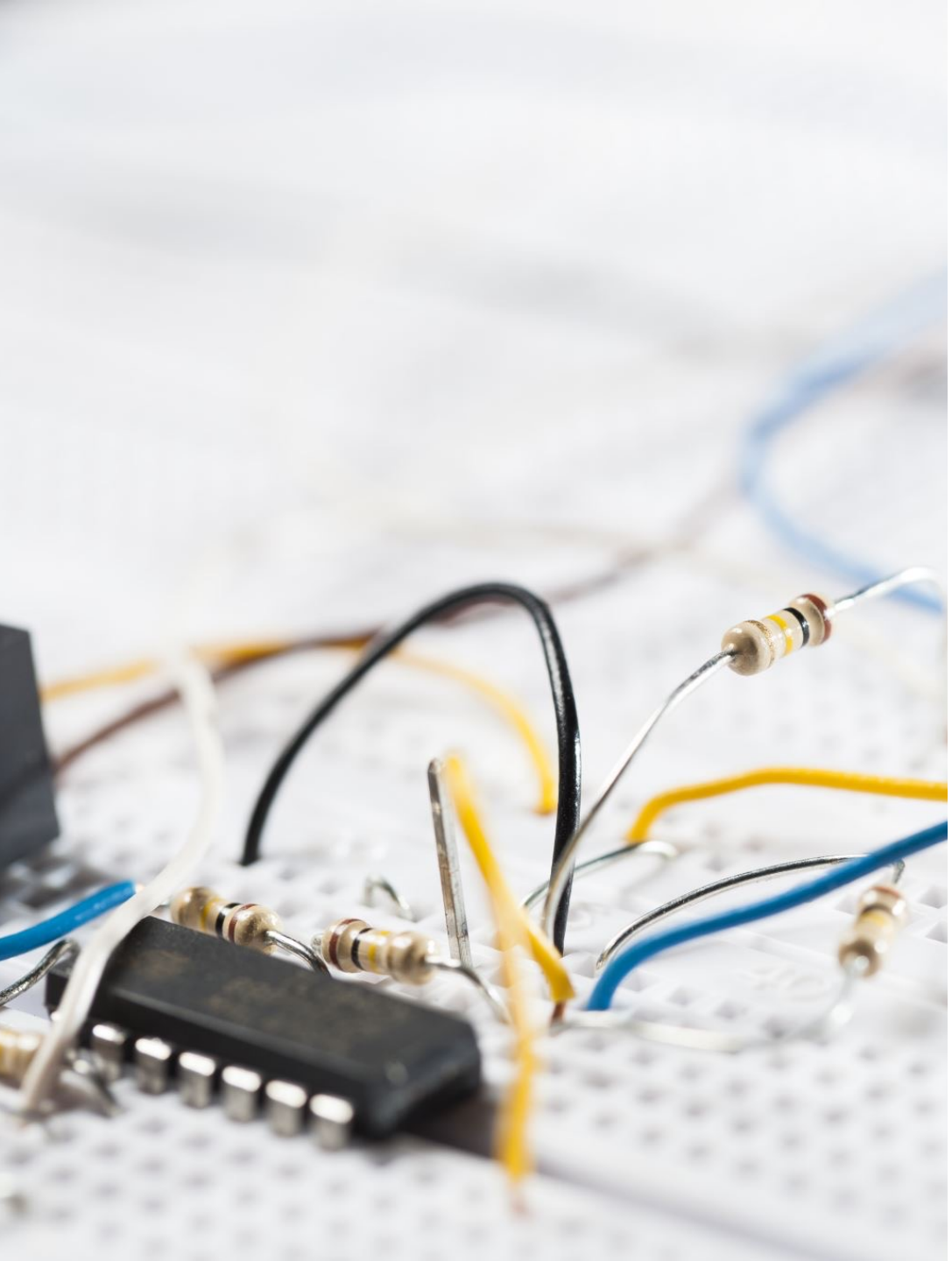
Specialized processors doing operations suited to their architecture are more efficient than general purpose processors.



Hence, future systems will be increasingly heterogeneous … GPUs, CPUs, FPGAs, and a wide range of accelerators

Source: Suyash Bakshi and Lennart Johnsson, "A Highly Efficient SGEMM Implementation using DMA on the Intel/Movidius Myriad-2. IEEE International Symposium on Computer Architecture and High Performance Computing, 2020

Source: https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2023/08/ATPESC-2023-Track-2b-Matteson-Openmp.pdf



Important note!!

- If you are looking for Deep learning acceleration on GPU please refer to the Horovod workshop given on 10/24/23.
- It covered Parallelization across multiple GPUs/nodes using Horovod Tensorflow/Pytorch
- https://github.com/kf-cuanschutz/CU-Anschutz-HPC-documentation/blob/main/Workshops/Introduction_to_Horovod_102423_part1_official_v2.pdf

Historical context

- GPU are graphics processing units.

Historical context

- GPU are graphics processing units.
- Originally came from the video game and movie industry.

Historical context

- GPU are graphics processing units.
- Originally came from the video game and movie industry.

Jurassic Park

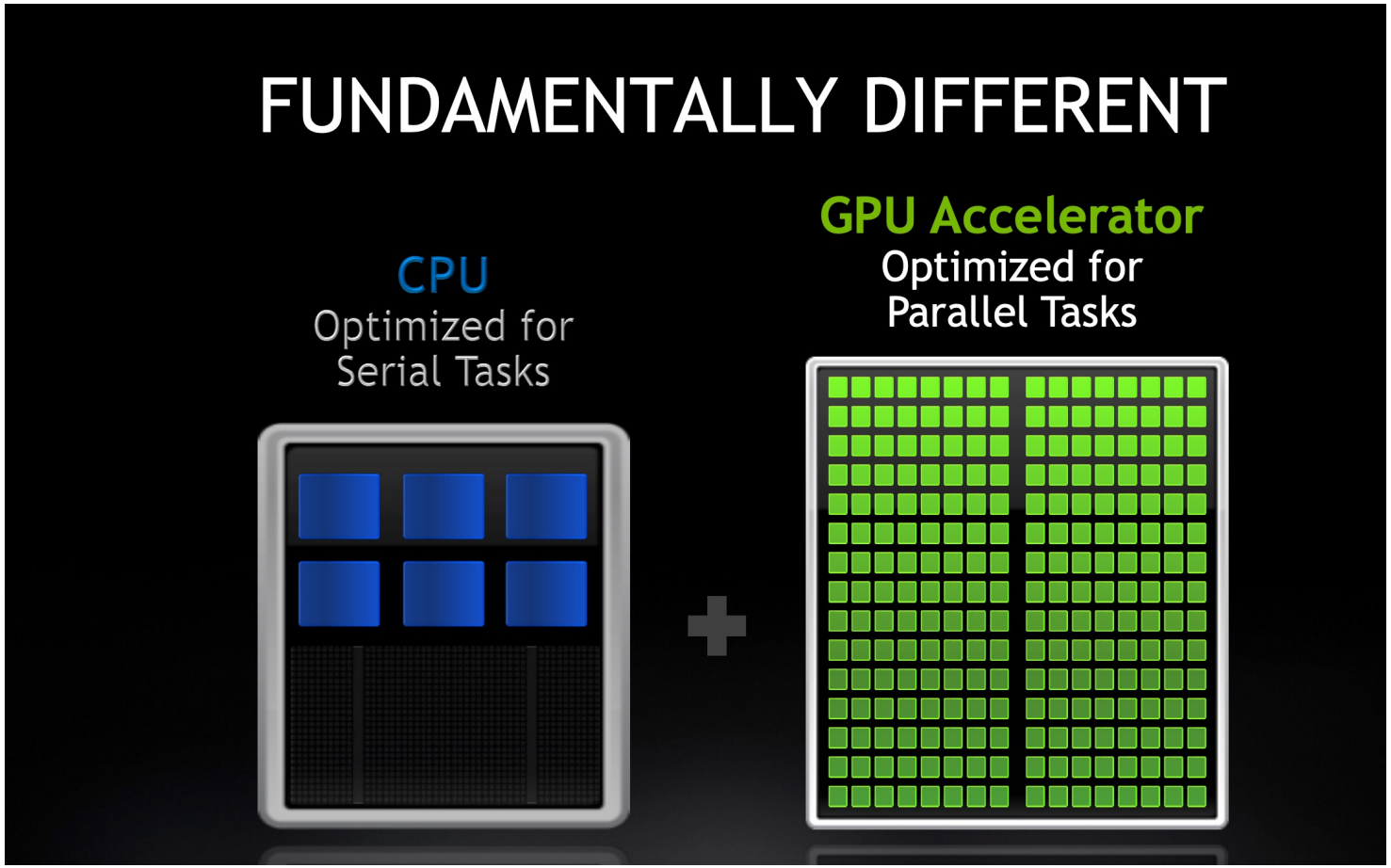
Historical context

- GPU are graphics processing units.
- Originally came from the video game and movie industry.

Jurassic Park

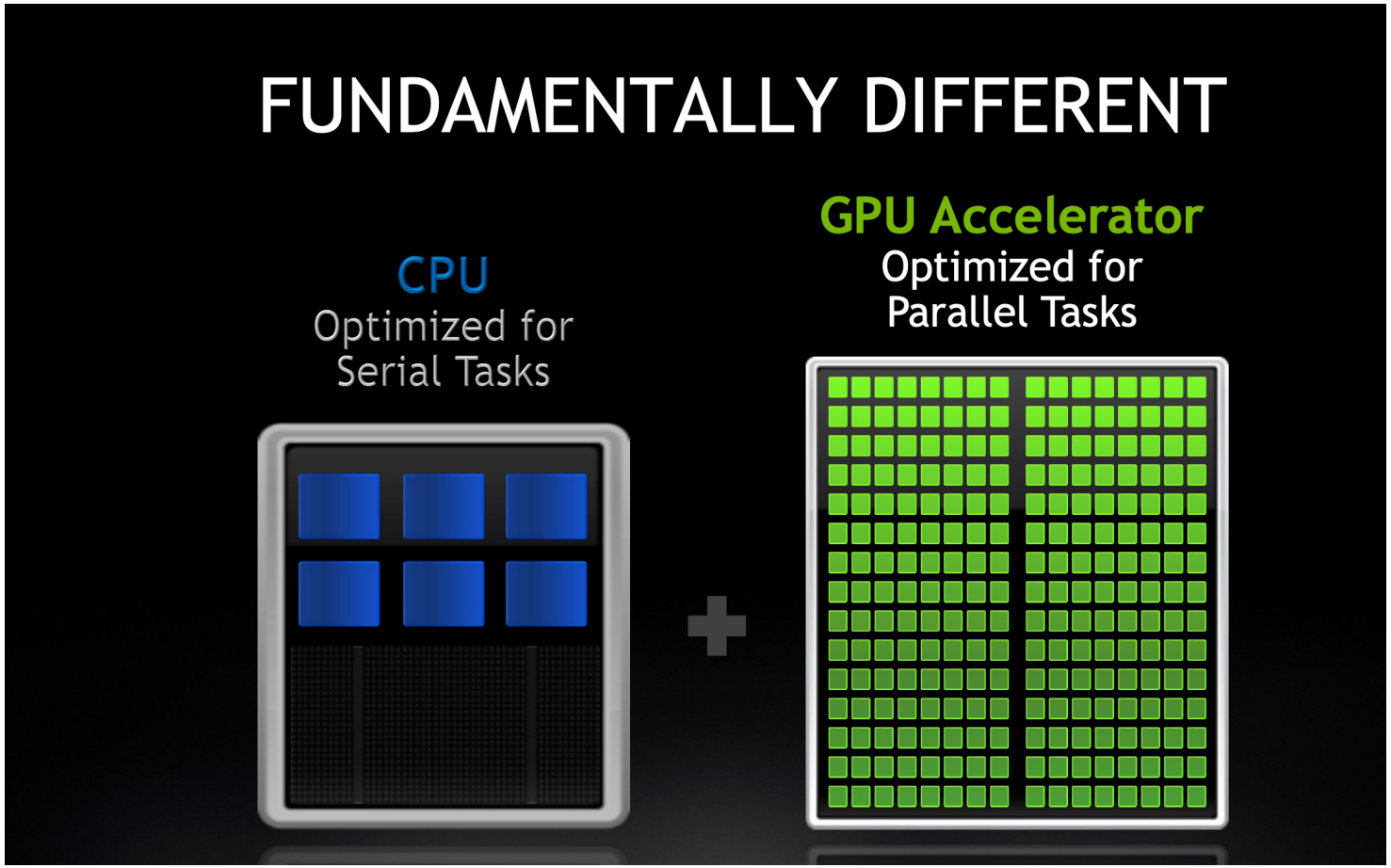
Zelda Ocarina of Time

CPUs vs GPUs comparison



Source: Openhackaton @ Princeton 2023 "Introduction to GPU computing

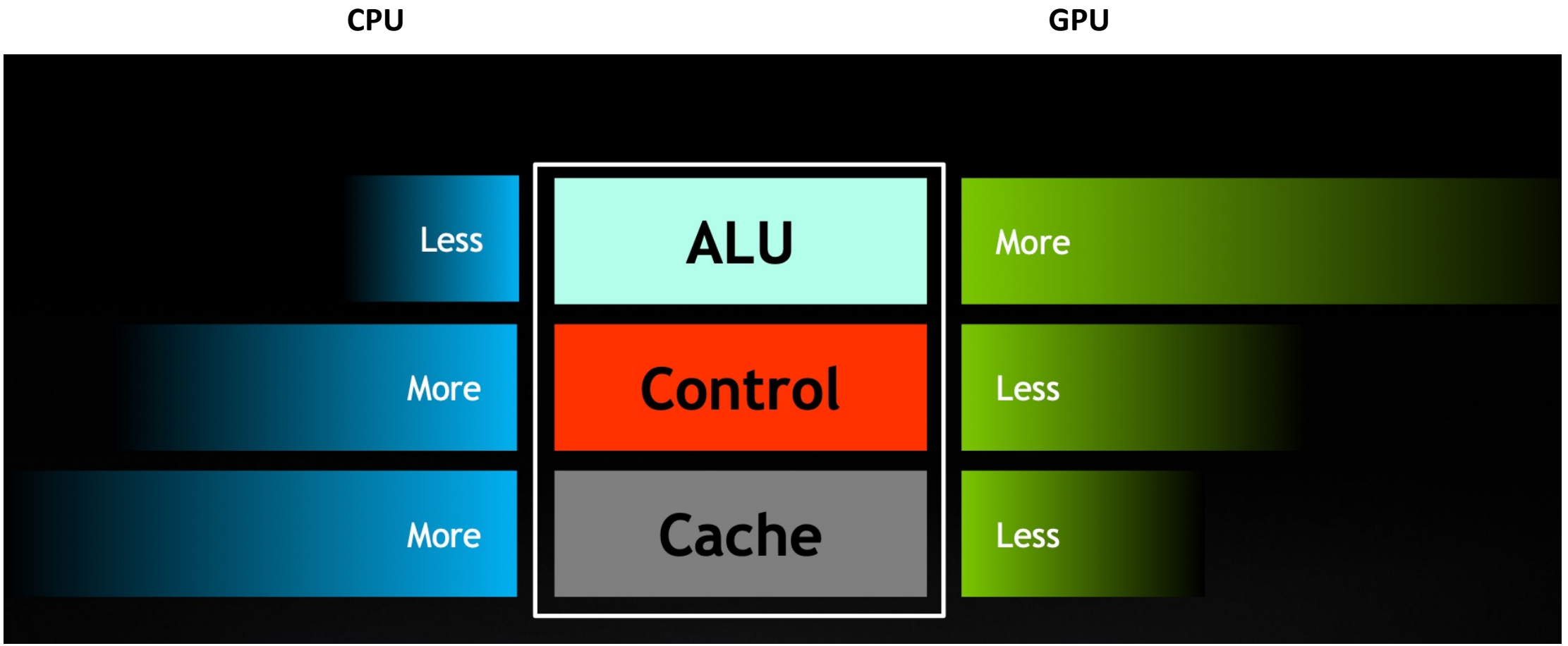
CPUs vs GPUs comparison



GPU follow SIMT pattern (Single Instruction multiple threads)

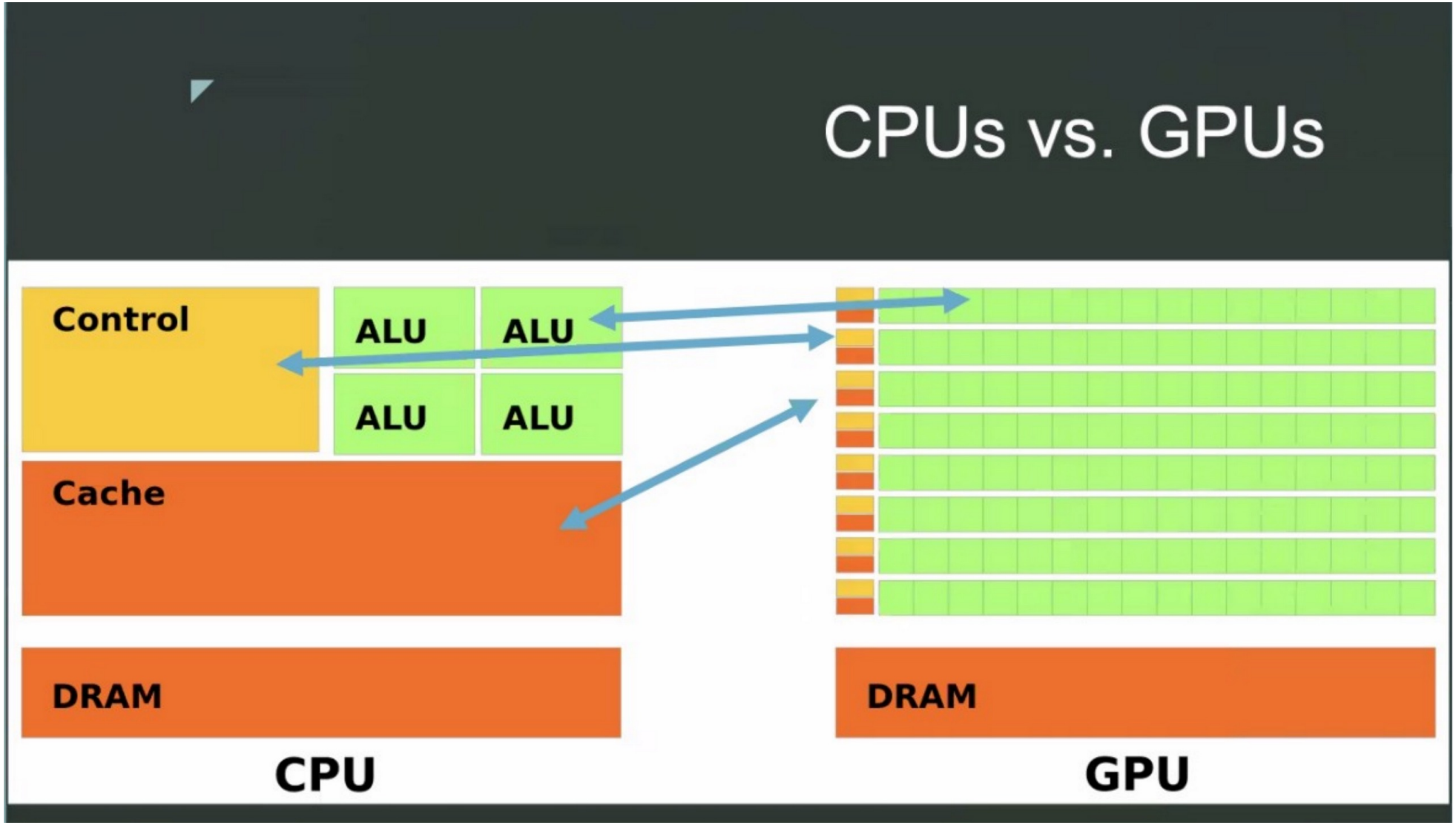
Source: Openhackaton @ Princeton 2023 "Introduction to GPU computing

CPUs vs GPUs comparison



Source: Openhackaton @ Princeton 2023 "Introduction to GPU computing

CPU vs GPU comparison



Source: Jayce Dowell Presentation on GPU University of New Mexico, Fall 2021

CPU vs GPU comparison

- DRAM is the global memory
- CPUs have few cores (ALU) when GPUs have a large number of cores.

Source: https://thedatafrog.com/en/articles/cuda-kernel-python/

CPU vs GPU comparison

- DRAM is the global memory
- CPUs have few cores (ALU) when GPUs have a large number of cores.

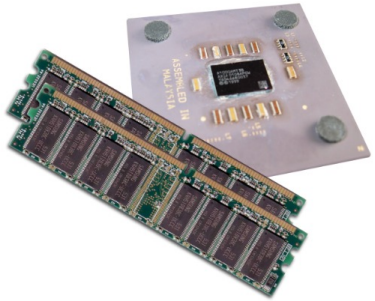
	Base Clock speed	Number of cores
Tesla V100 GPU	1.45 GHz	5120
Intel Core i9-9900K	3.6 GHz	8

Source: https://thedatafrog.com/en/articles/cuda-kernel-python/

CPU vs GPU programming model

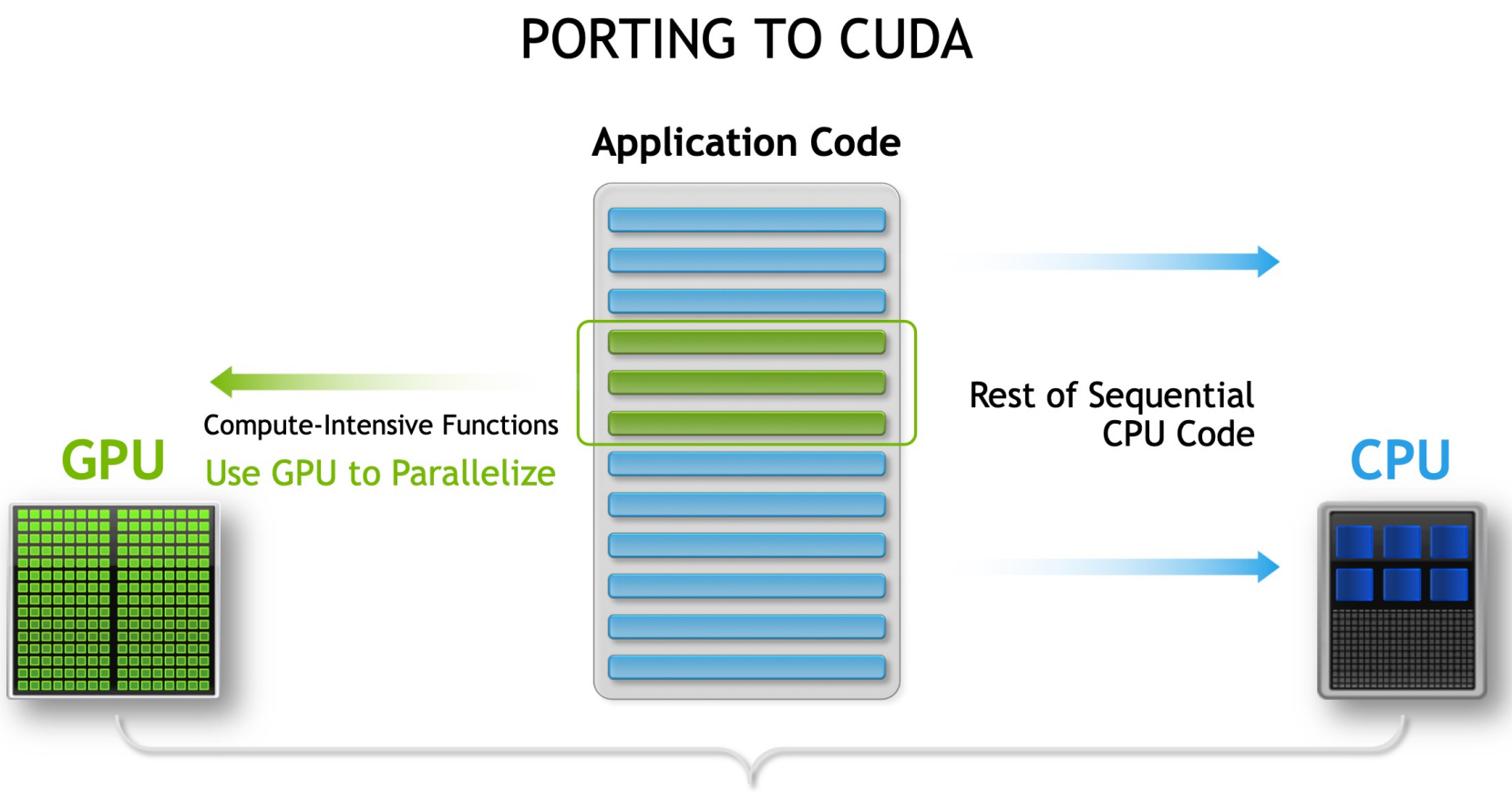
HETEROGENEOUS COMPUTING

- *Host* The CPU and its memory (host memory)
- *Device* The GPU and its memory (device memory)



Source: https://www.olcf.ornl.gov/cuda-training-series/ ; CUDA C++

CPU vs GPU programming model



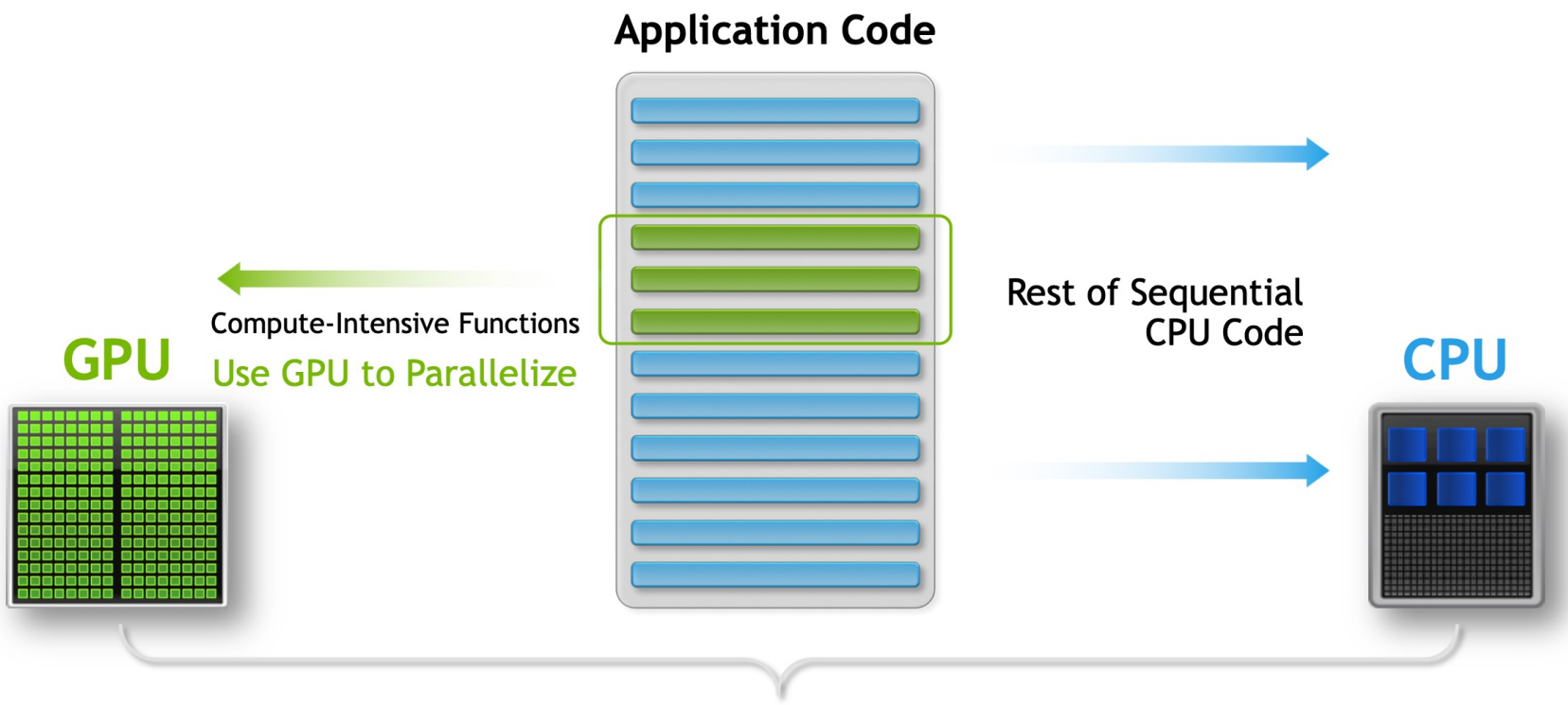
Source: https://www.olcf.ornl.gov/cuda-training-series/ ; CUDA C++

CPU vs GPU programming model

GPU: Matrix operations. (e.g. FFT)

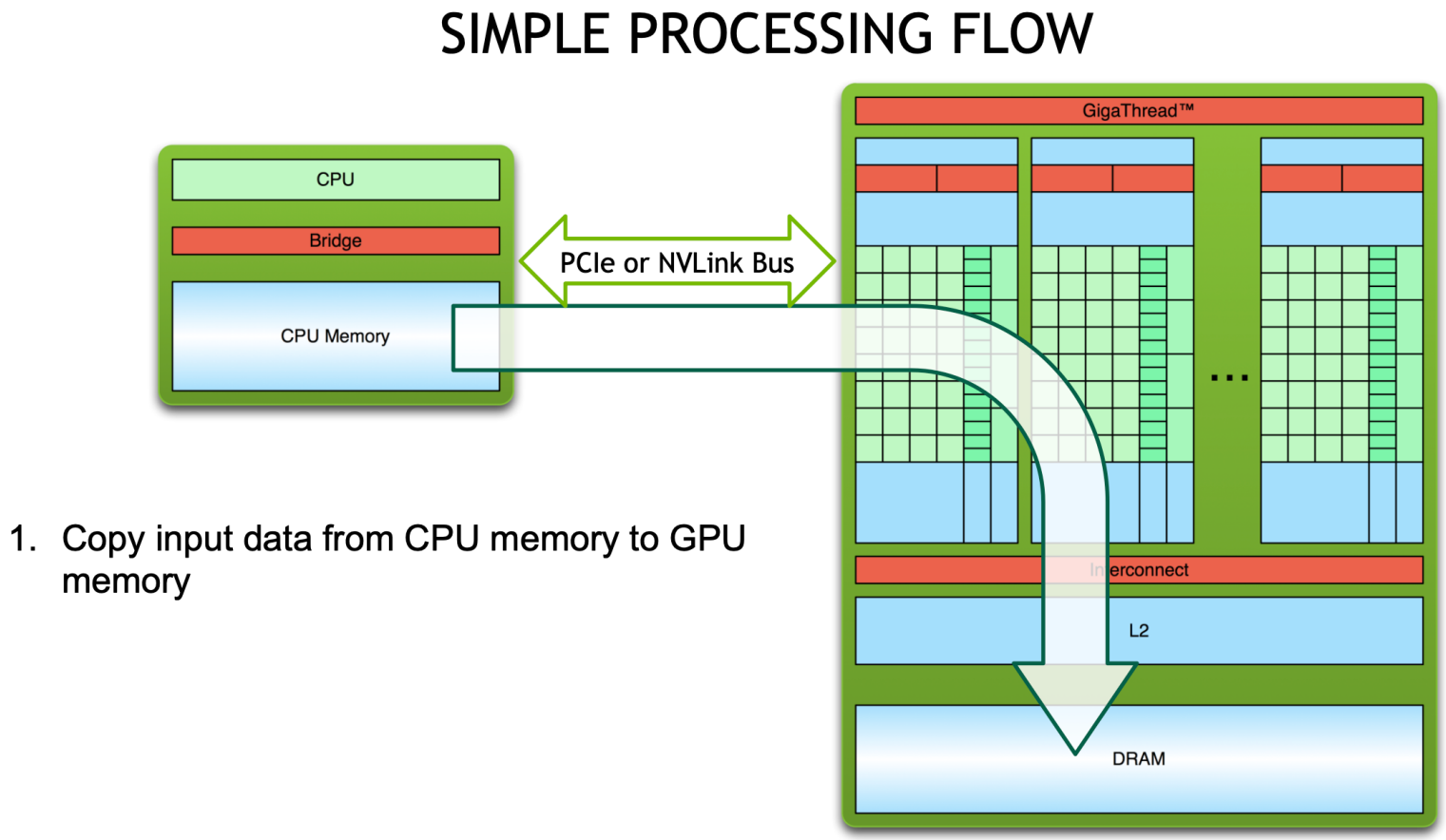
CPU:

OS, security etc …



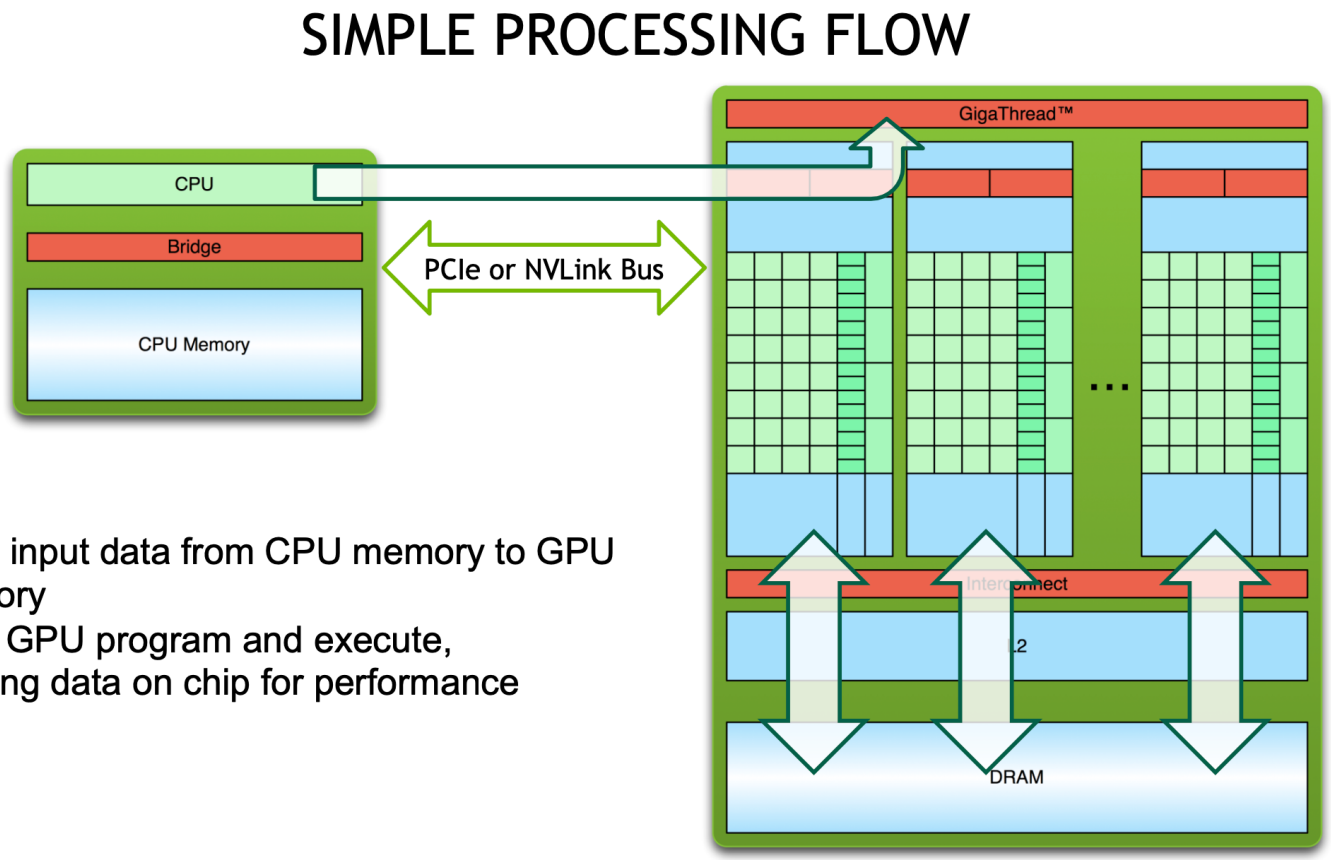
Source: https://www.olcf.ornl.gov/cuda-training-series/ ; CUDA C++

CPU vs GPU programming model



Source: https://www.olcf.ornl.gov/cuda-training-series/ ; CUDA C++

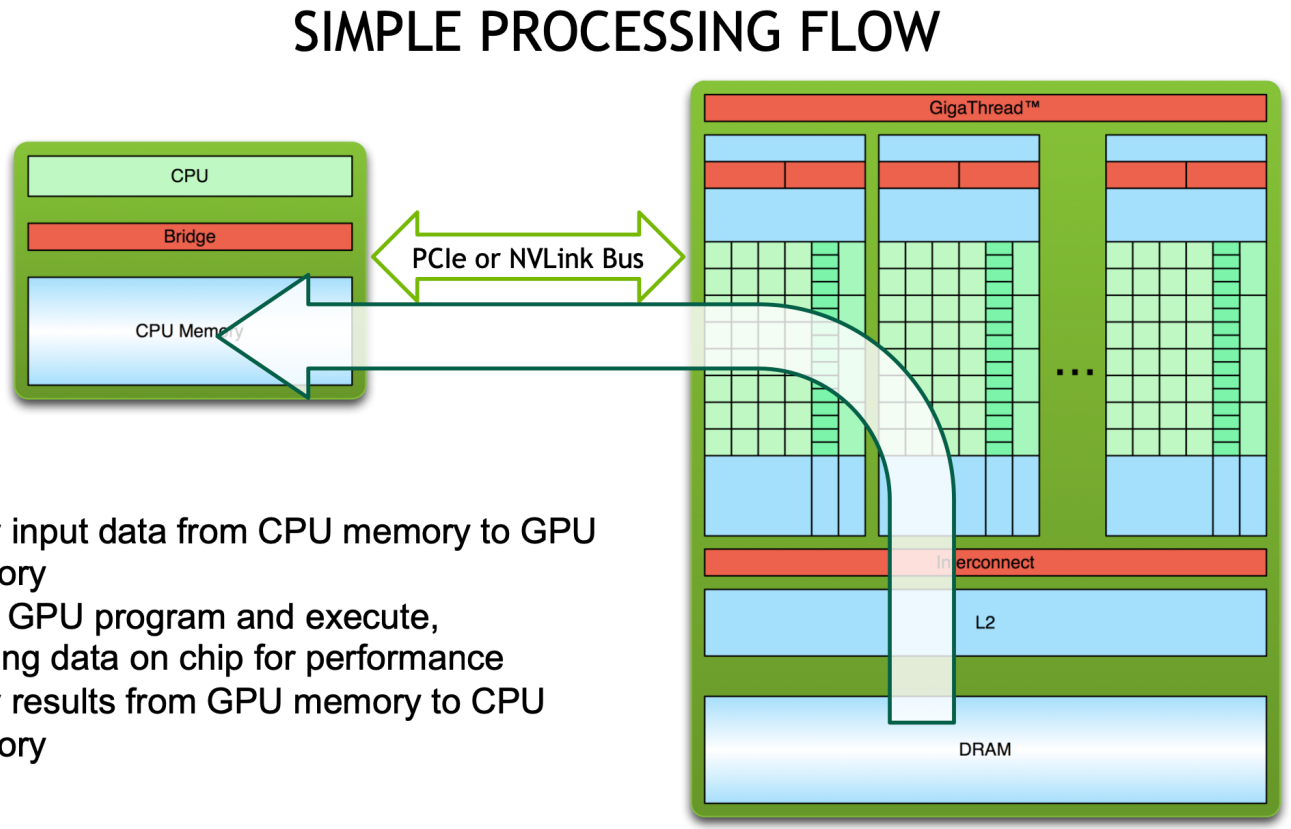
CPU vs GPU programming model



1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance

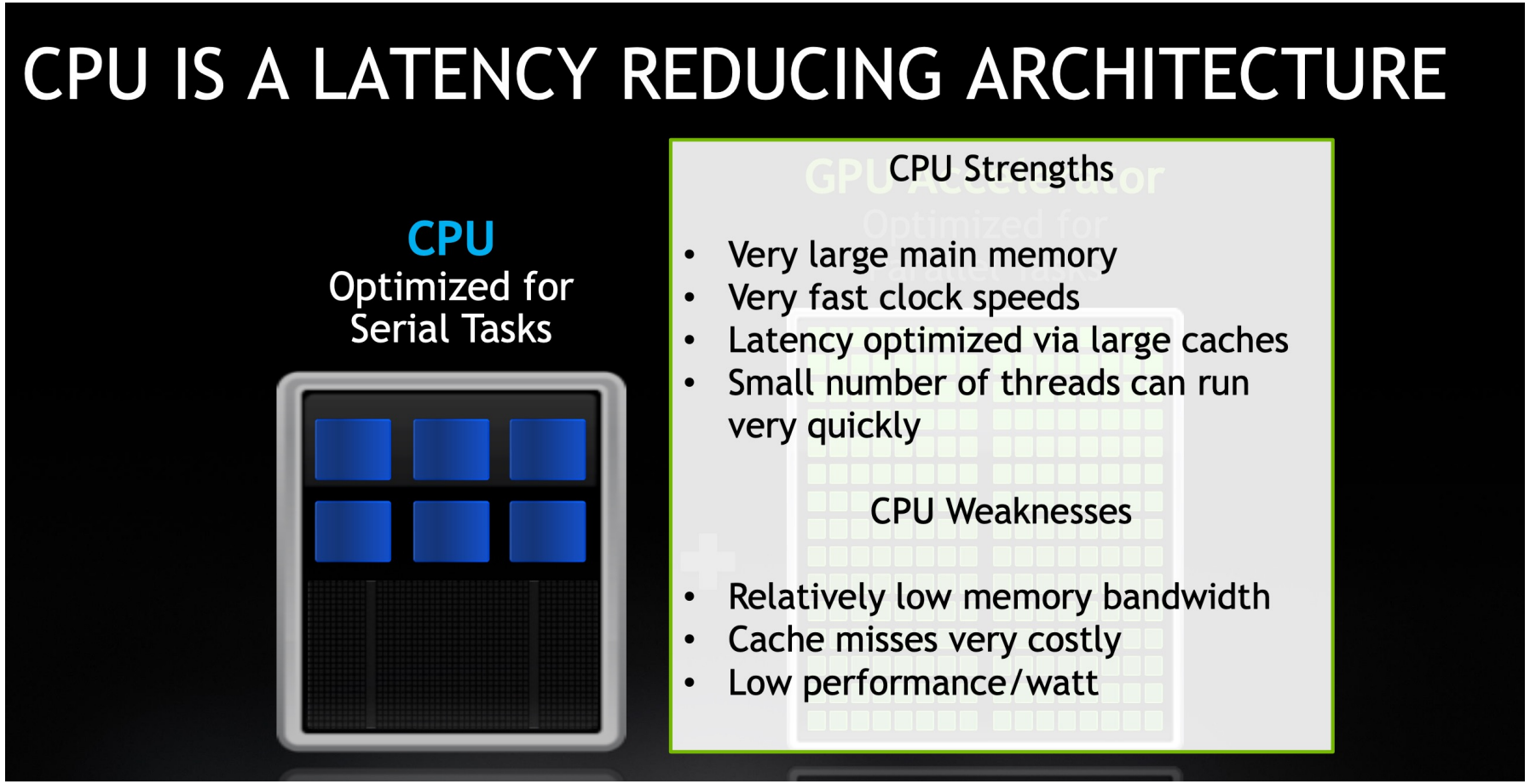
Source: https://www.olcf.ornl.gov/cuda-training-series/ ; CUDA C++

CPU vs GPU programming model



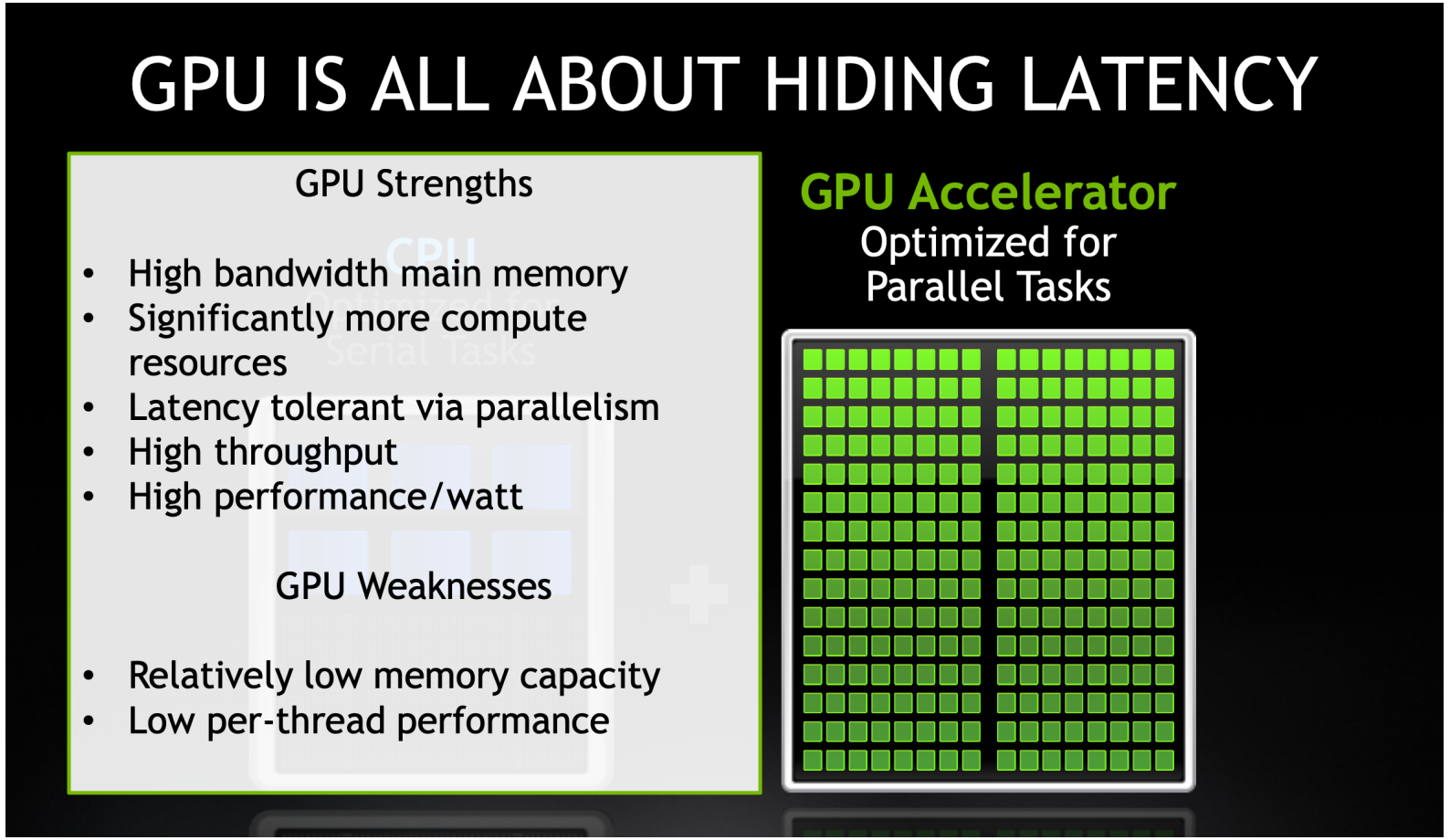
Source: https://www.olcf.ornl.gov/cuda-training-series/ ; CUDA C++

CPU vs GPU programming model



Source: Openhackaton @ Princeton 2023 "Introduction to GPU computing

CPU vs GPU programming model



Source: Openhackaton @ Princeton 2023 "Introduction to GPU computing

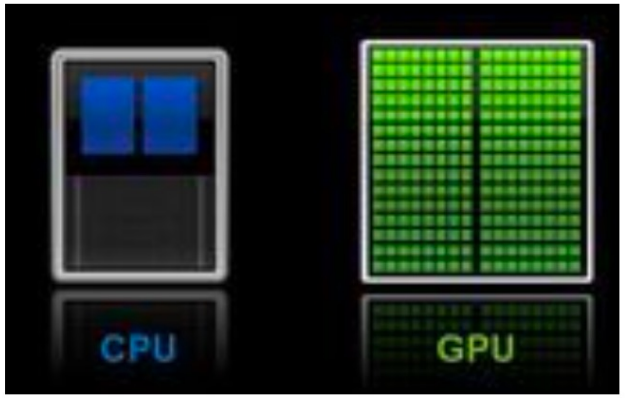
CPU vs GPU programming model

- You need lots of parallelism with GPU cores to optimize computation.
- GPU memory is very fast but moving data from CPU to GPU is very costly.

CPU vs GPU programming model

CPU (Haswell)

- 128GB DDR
- ~120 GB/Sec Memory Bandwidth



GPU (A100)

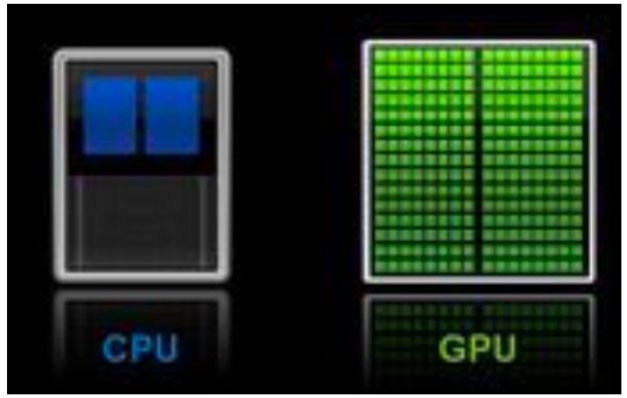
- 40GB HBM
- 1,500 GB/Sec Memory Bandwidth

Source: https://www.olcf.ornl.gov/cuda-training-series/ : CUDA C++

CPU vs GPU programming model

CPU (Haswell)

- 128GB DDR
- ~120 GB/Sec Memory Bandwidth



GPU (A100)

- 40GB HBM
- 1,500 GB/Sec Memory Bandwidth

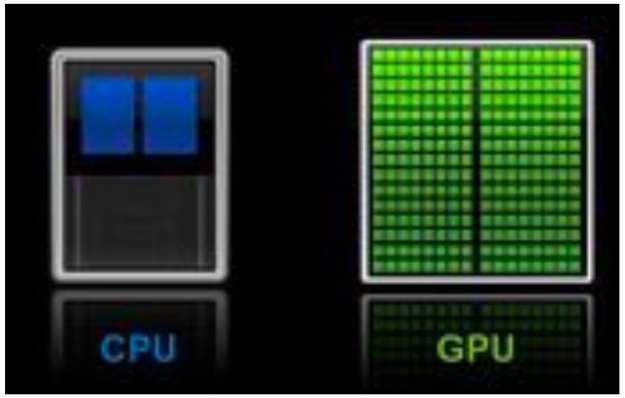
PCIe ~ 32 GB/Sec

Source: https://www.olcf.ornl.gov/cuda-training-series/ : CUDA C++

CPU vs GPU programming model

CPU (Haswell)

- 128GB DDR
- ~120 GB/Sec Memory Bandwidth



GPU (A100)

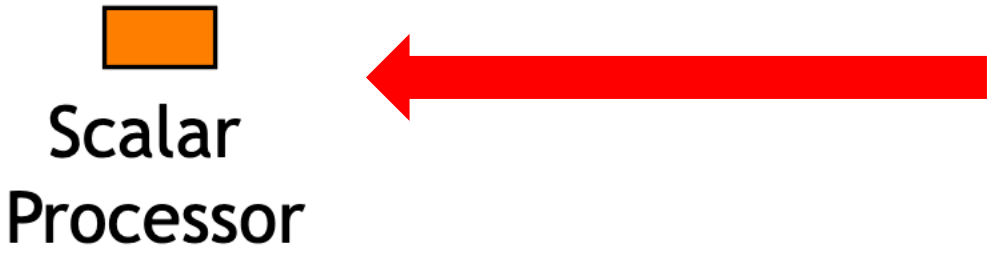
- 40GB HBM
- 1,500 GB/Sec Memory Bandwidth

CPU - Speed

GPU - Throughput

Source: https://www.olcf.ornl.gov/cuda-training-series/ : CUDA C++

NVIDIA A100 specs



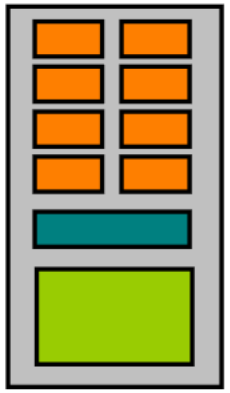
cuda core

Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

NVIDIA A100 specs

Scalar Processor

cuda core.



Multiprocessor

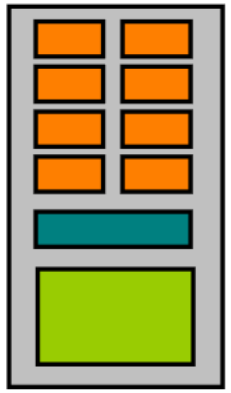
streaming multiprocessor (SMs).

Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

NVIDIA A100 specs

Scalar Processor

cuda core.



Multiprocessor

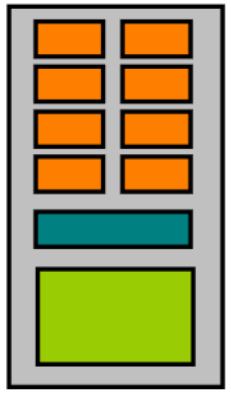
streaming multiprocessor
: **64 cuda cores per SM**

Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

NVIDIA A100 specs

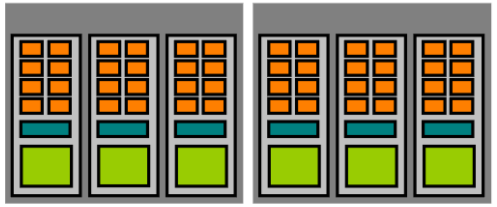
Scalar Processor

cuda core. **64 per SM**



Multiprocessor

streaming multiprocessor
: **108 SMs**



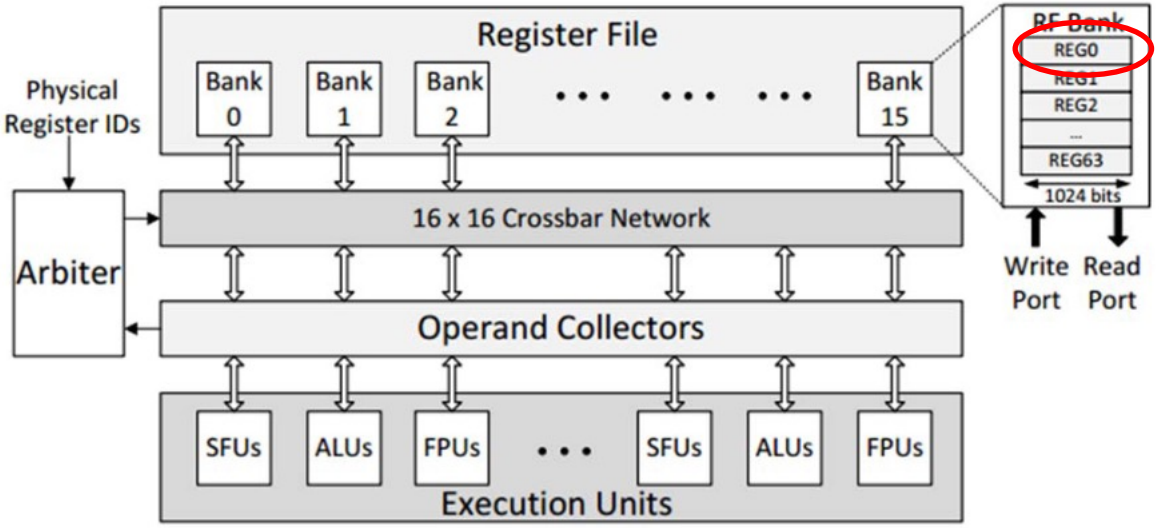
Device

108 SMs per device

Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

NVIDIA GPU memory hierarchy

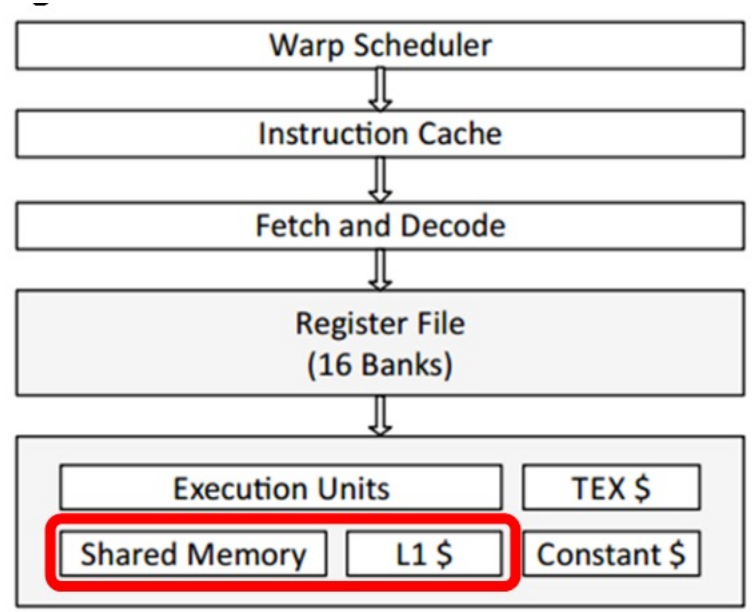
- Streaming multiprocessors registers
- Very fast **(every 1 clock cycle)**
- Only usable by 1 thread

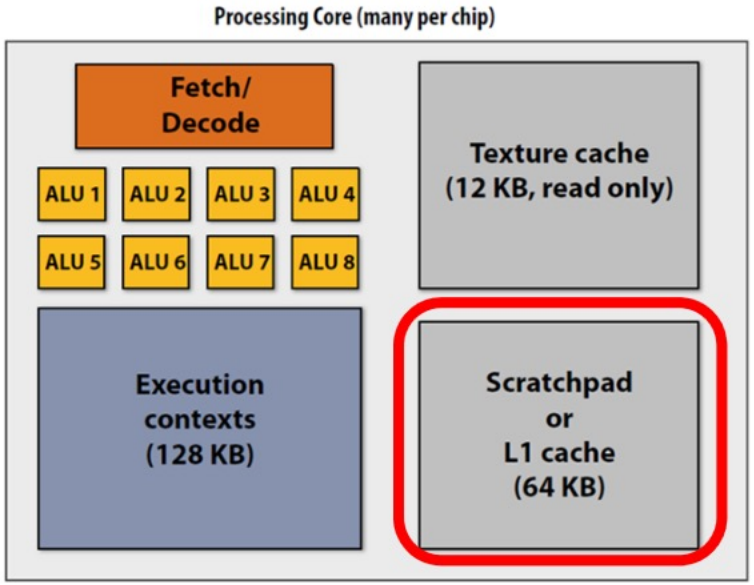


Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy

- Shared/L1 Memory
- Organized into banks that can be accessed simultaneously

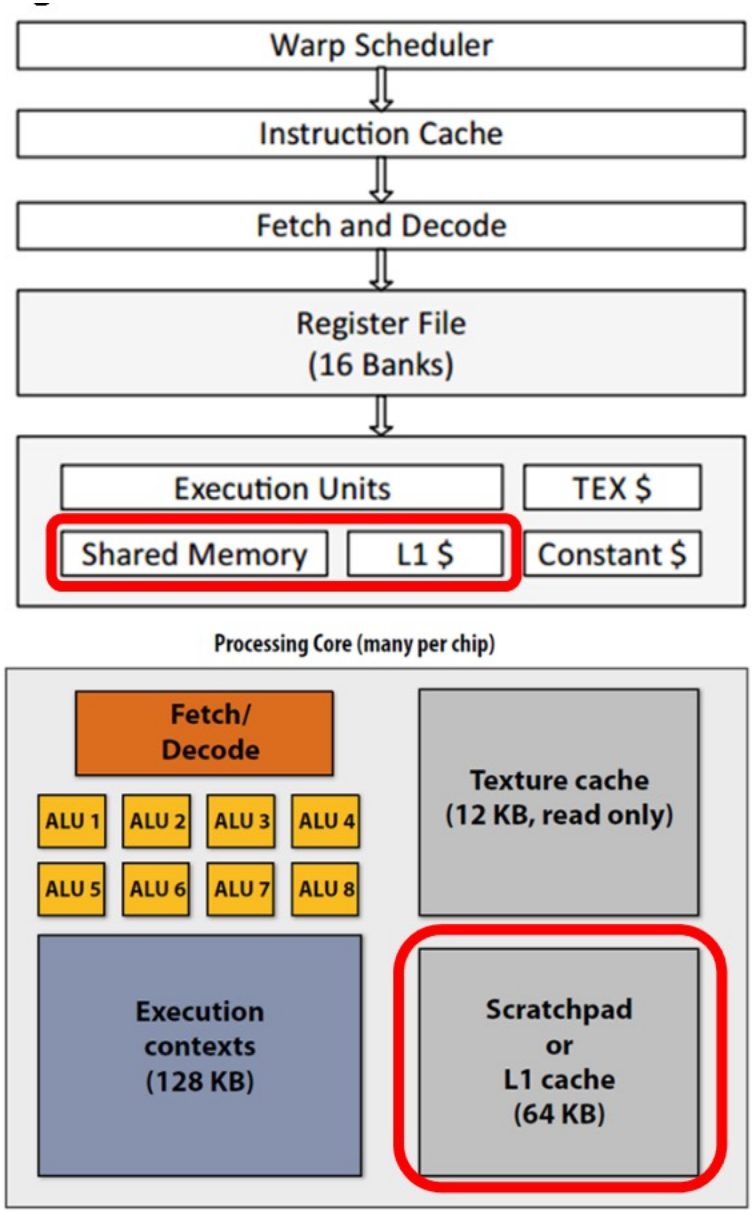




Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy

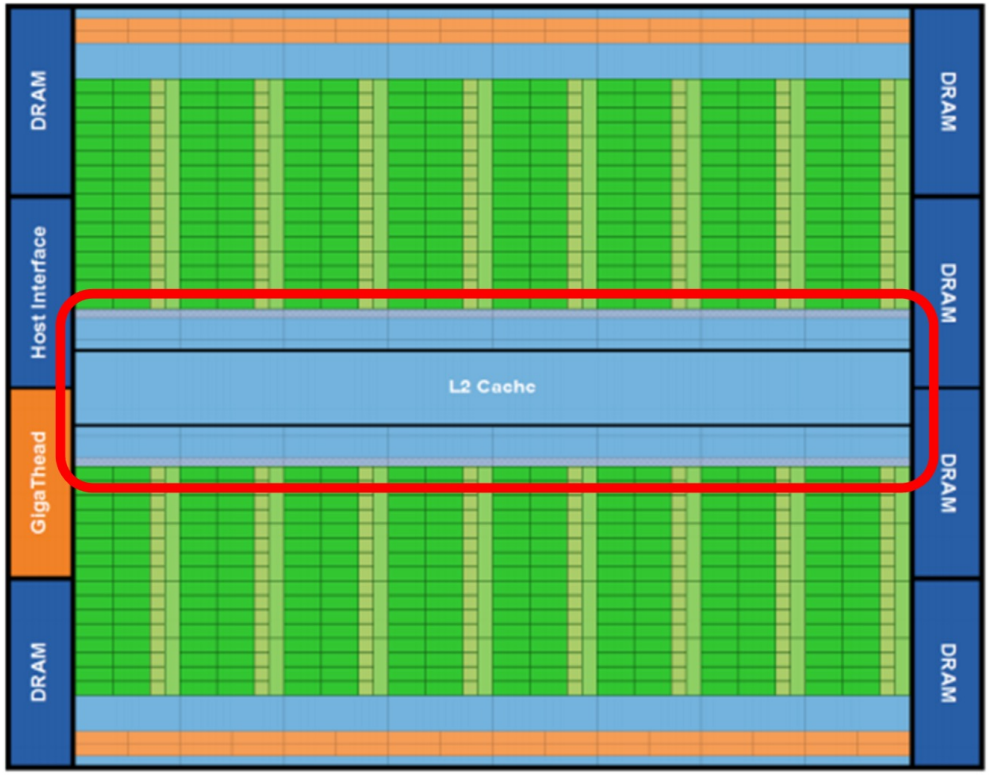
- Shared/L1 Memory
- Organized into banks that can be accessed simultaneously
- Located in an SM



Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy

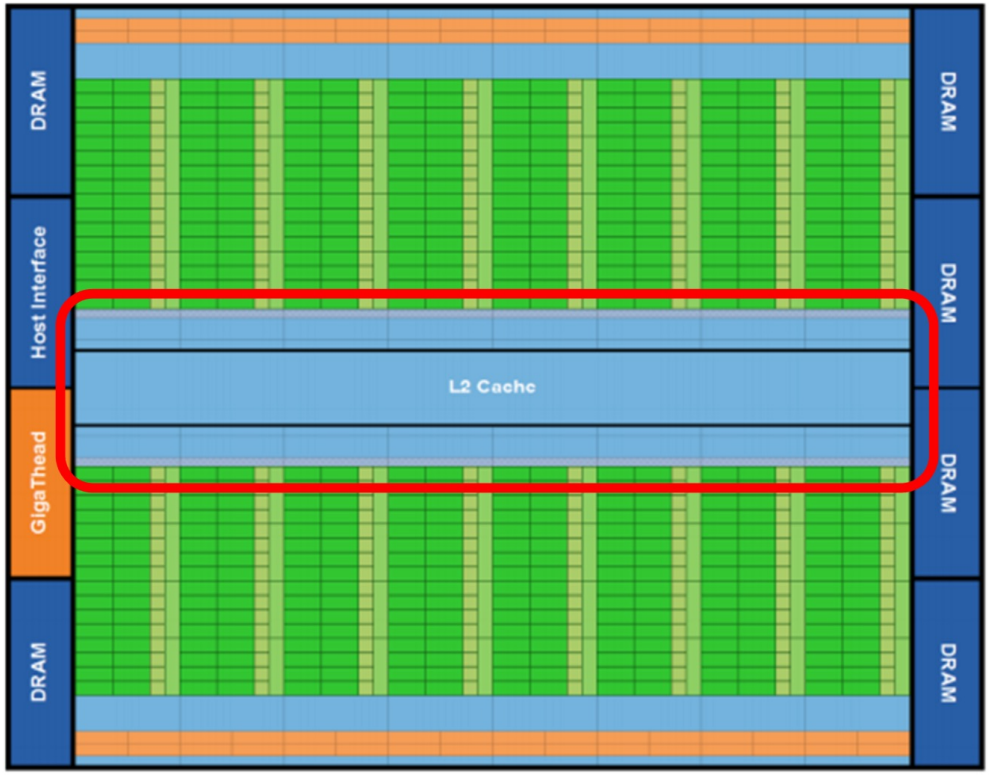
- L2 cache
- Shared among SMs



Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy

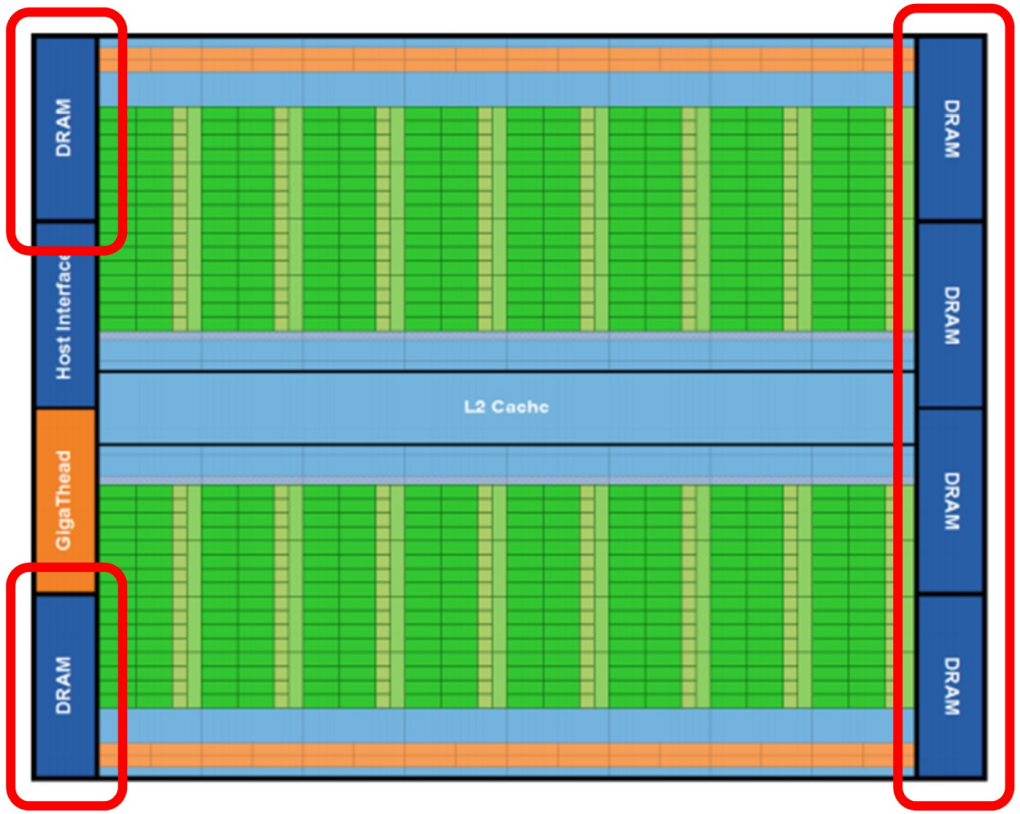
- L2 cache
- Shared among SMs
- Slower bandwidth than L1



Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy

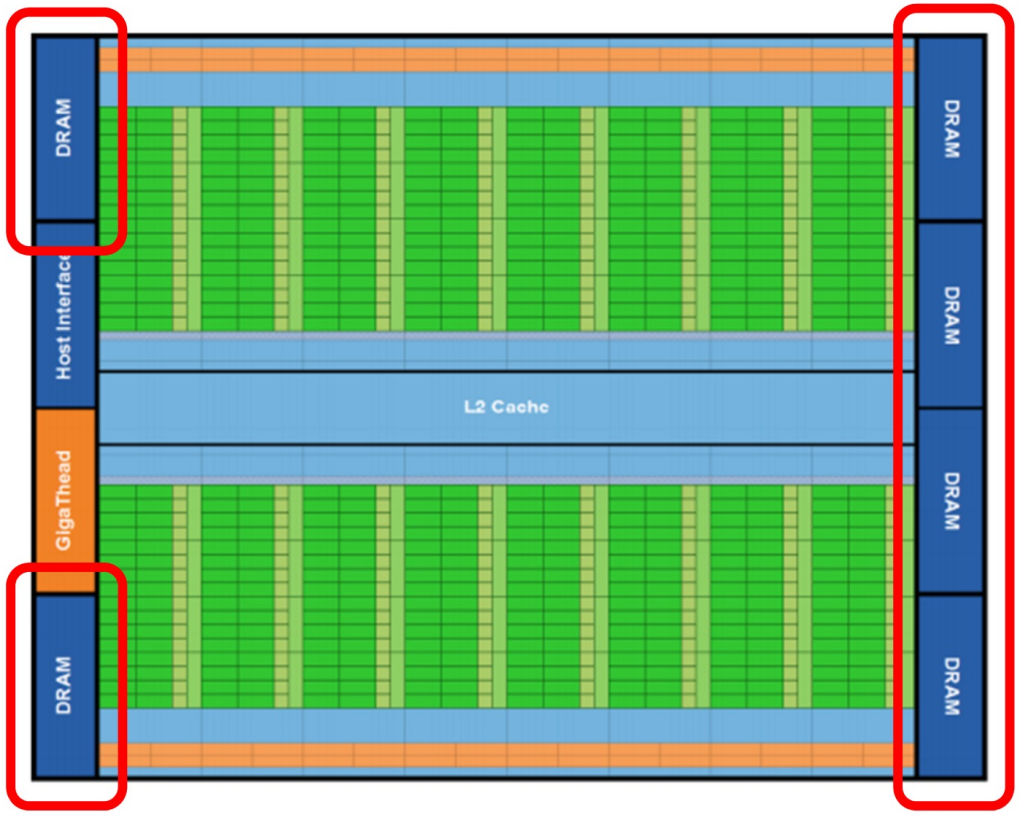
- DRAM
- Much slower than L2 (hundreds of clock cycles)



Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy

- DRAM
- Much slower than L2 (hundreds of clock cycles)
- Can be accessed by both CPU and GPU



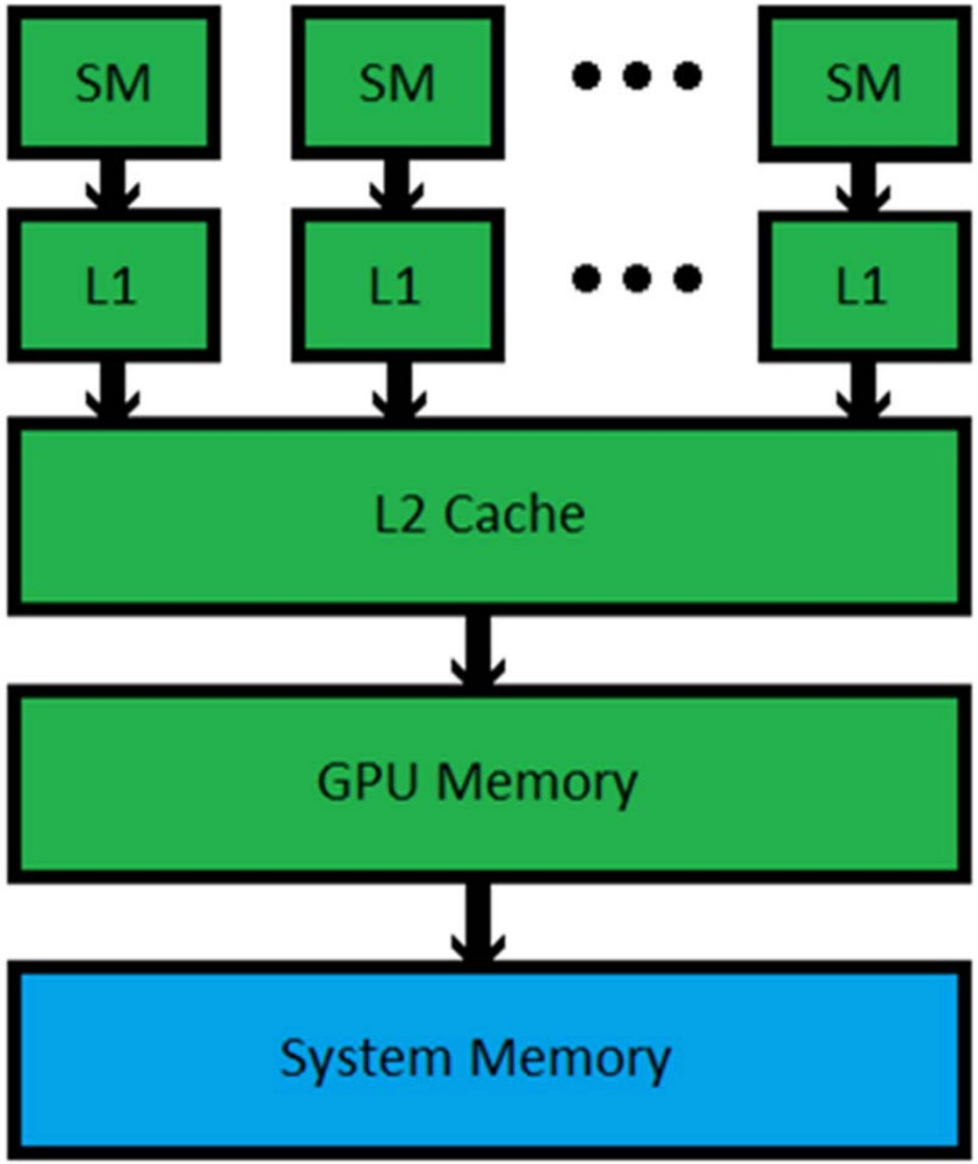
Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy

- PCIe (10-100 GB/s)

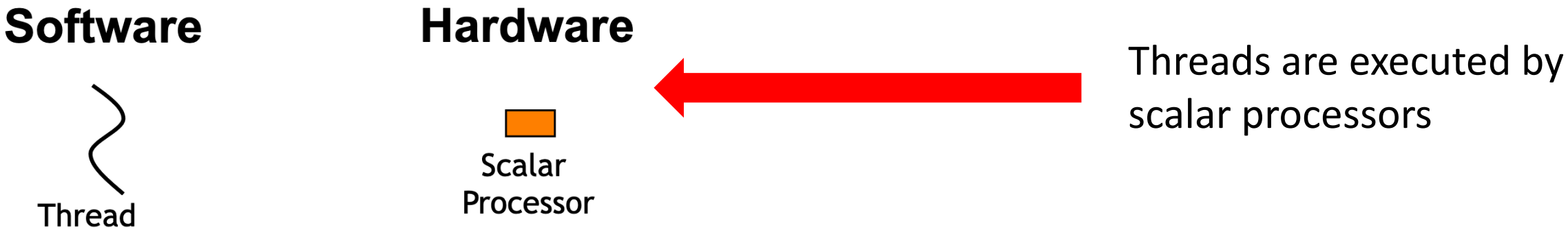
Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

NVIDIA GPU memory hierarchy



Source: http://meseec.ce.rit.edu/551-projects/spring2015/3-2.pdf; https://people.duke.edu/~ccc14/sta-663/CUDAPython.html

GPU software programming



Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPU software programming

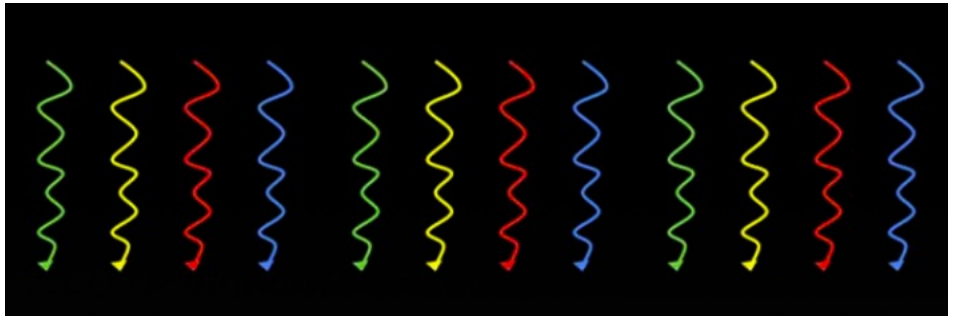
Software

Thread

Hardware

Scalar Processor

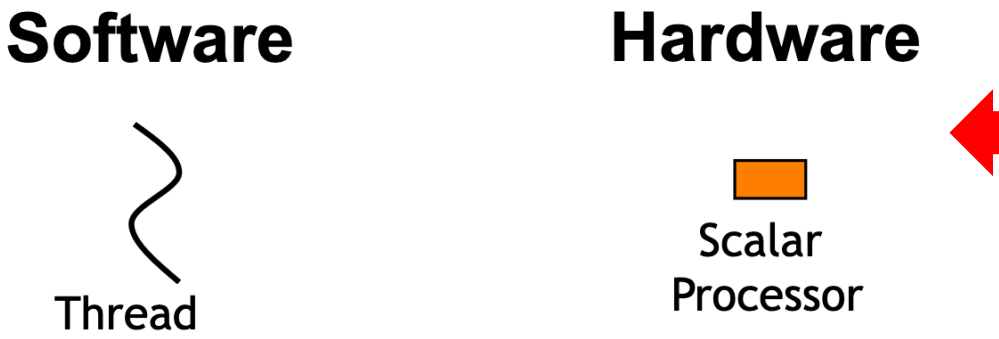
Threads are executed by scalar processors



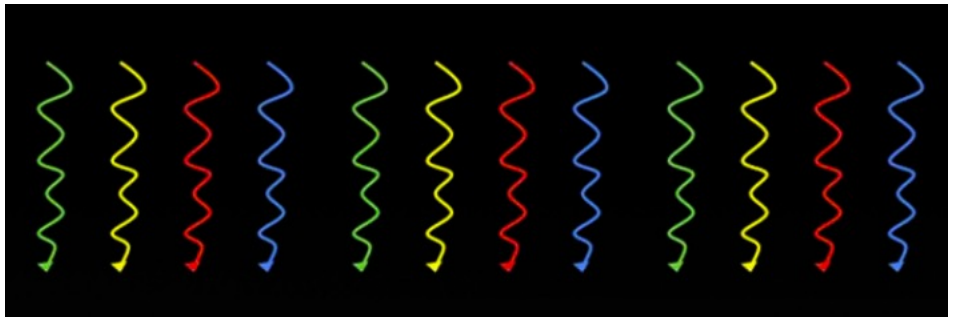
GPUs can handle thousands of concurrent threads

Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPU software programming



Threads are executed by scalar processors

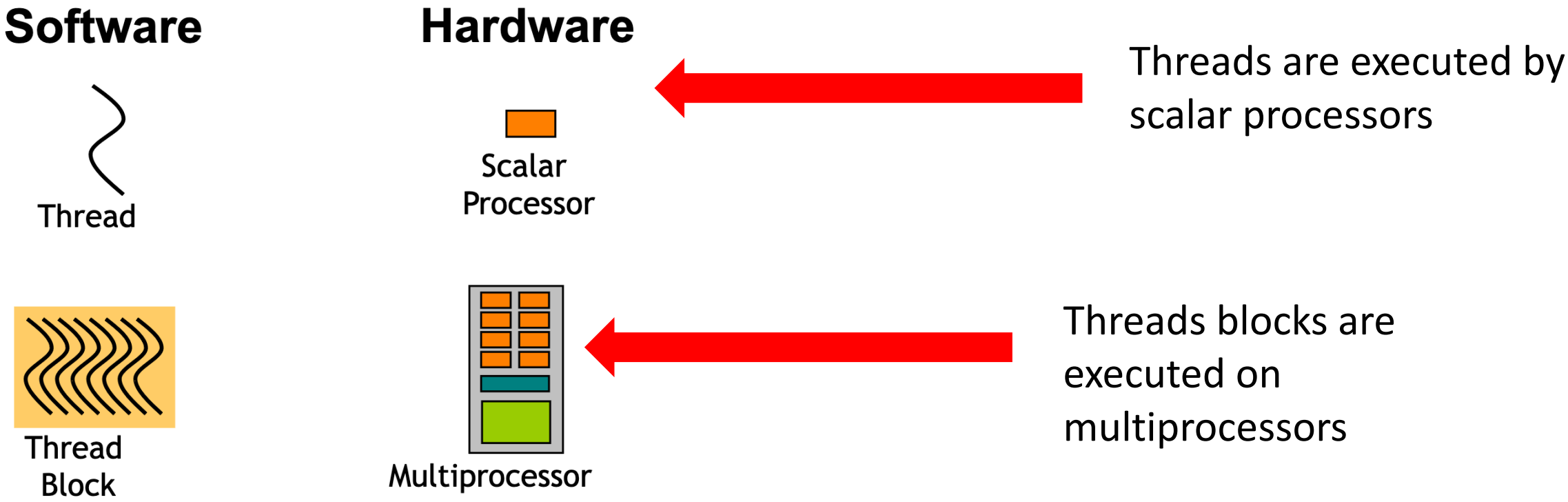


GPUs can handle thousands of concurrent threads

With CUDA, more threads than the GPU can handle concurrently can be launched.

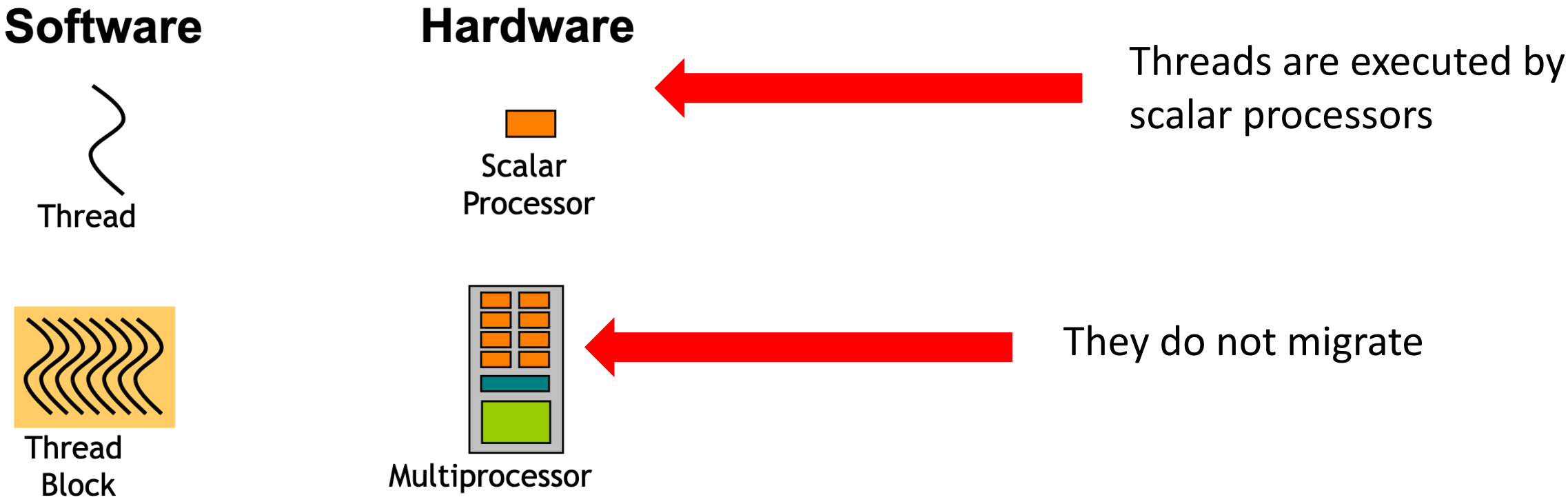
Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPU software programming



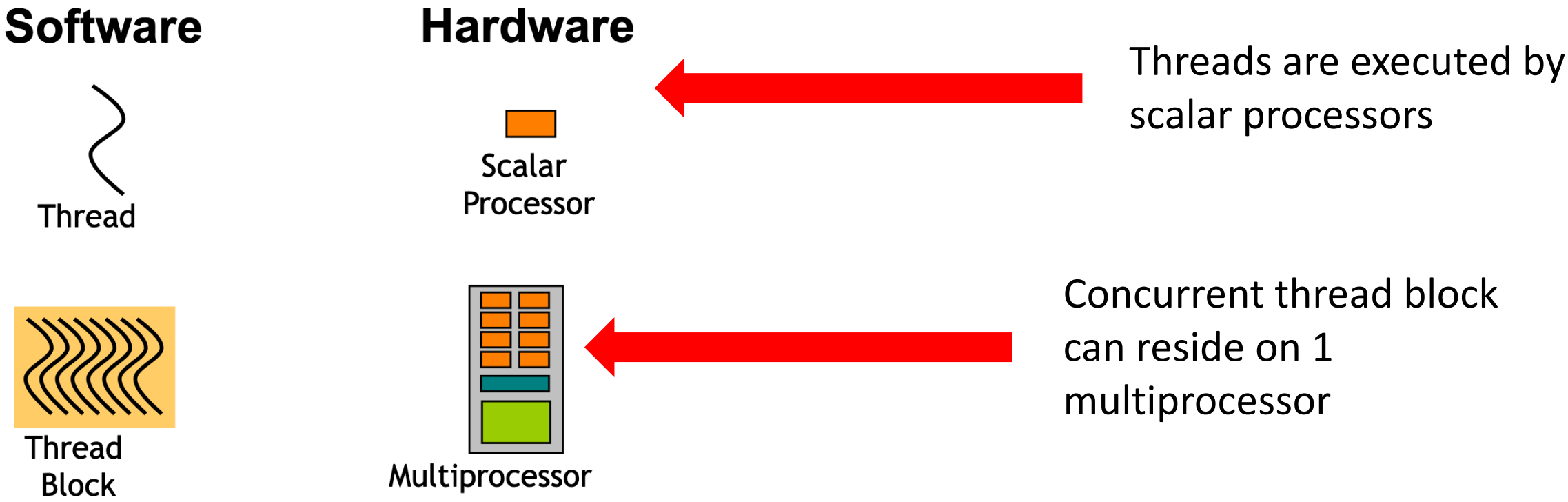
Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPU software programming



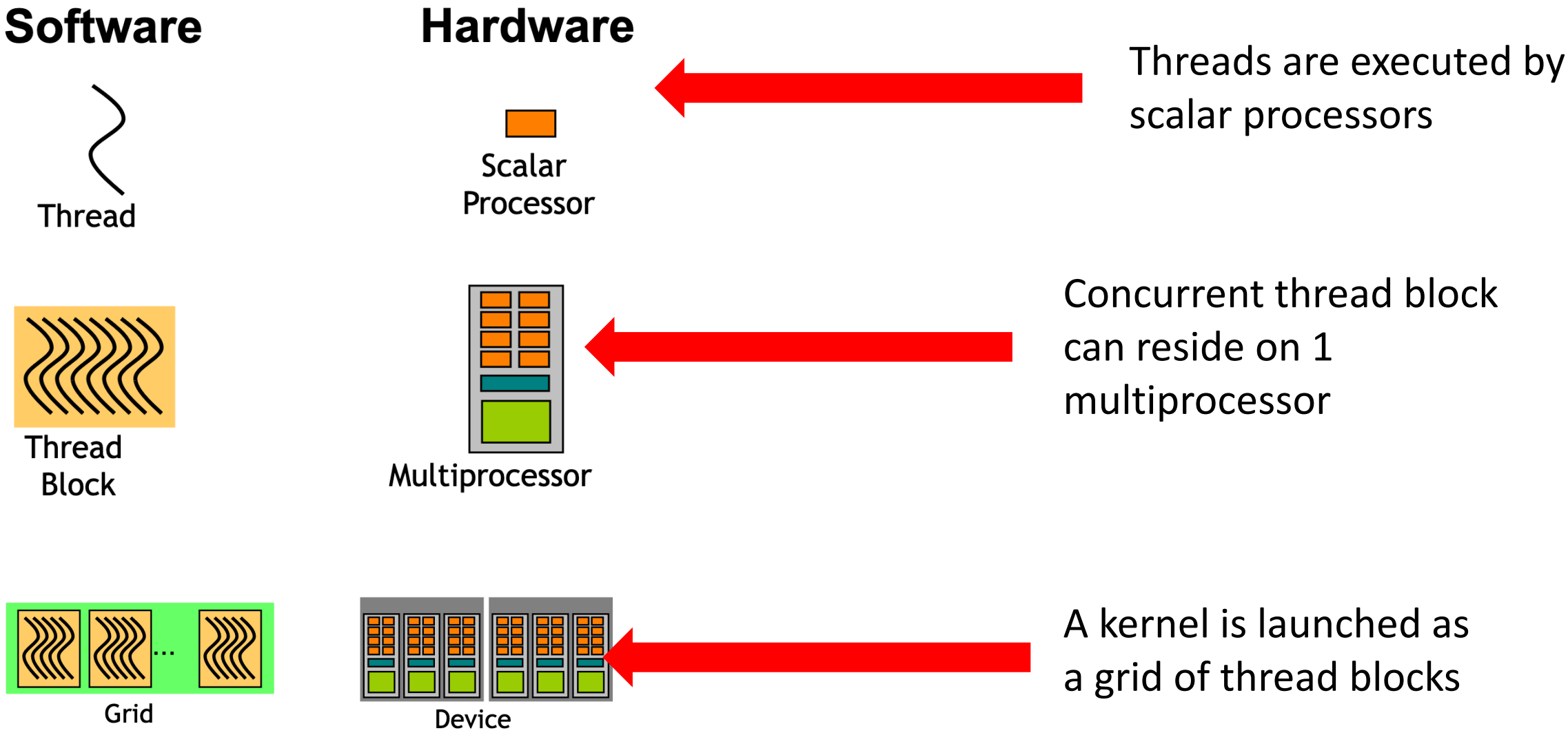
Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPU software programming



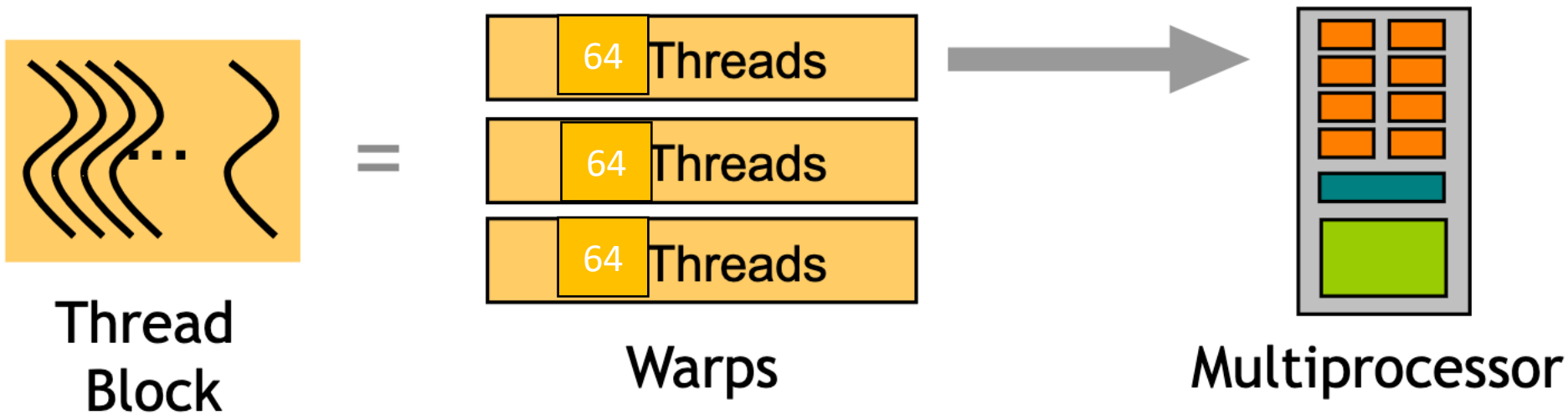
Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPU software programming



Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

Warps

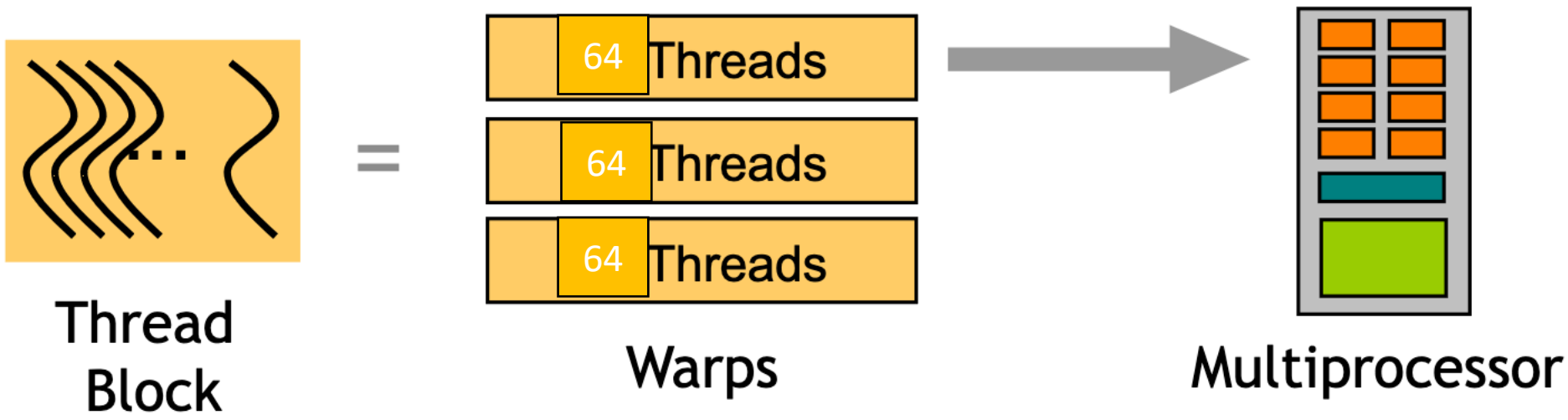


* A thread block consists of 64 thread warps

* A warp is executed in parallel on a multiprocessor

Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

Warps



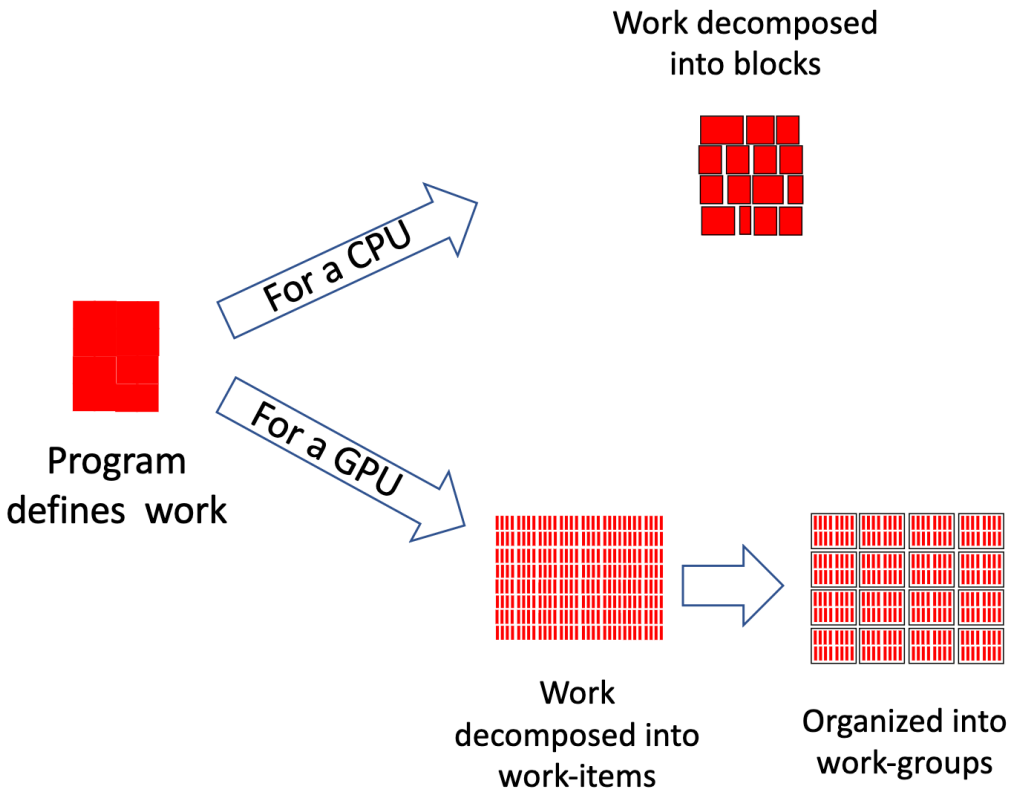
* A thread block consists of 64 thread warps

* A warp is executed in parallel on a multiprocessor

* Thread within the same warp execute the same operation at a given clock cycle.

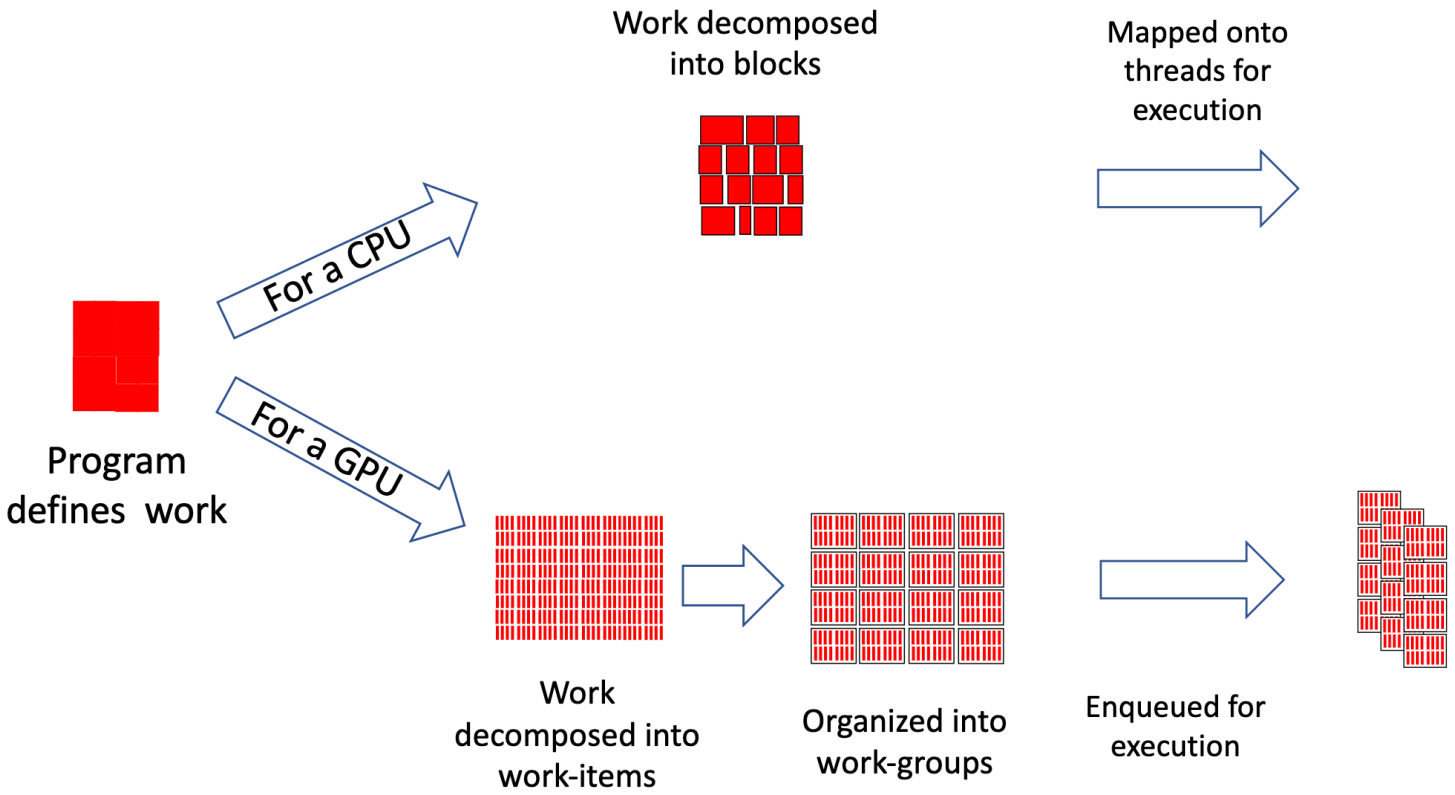
Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

Summary CPU vs GPU computation



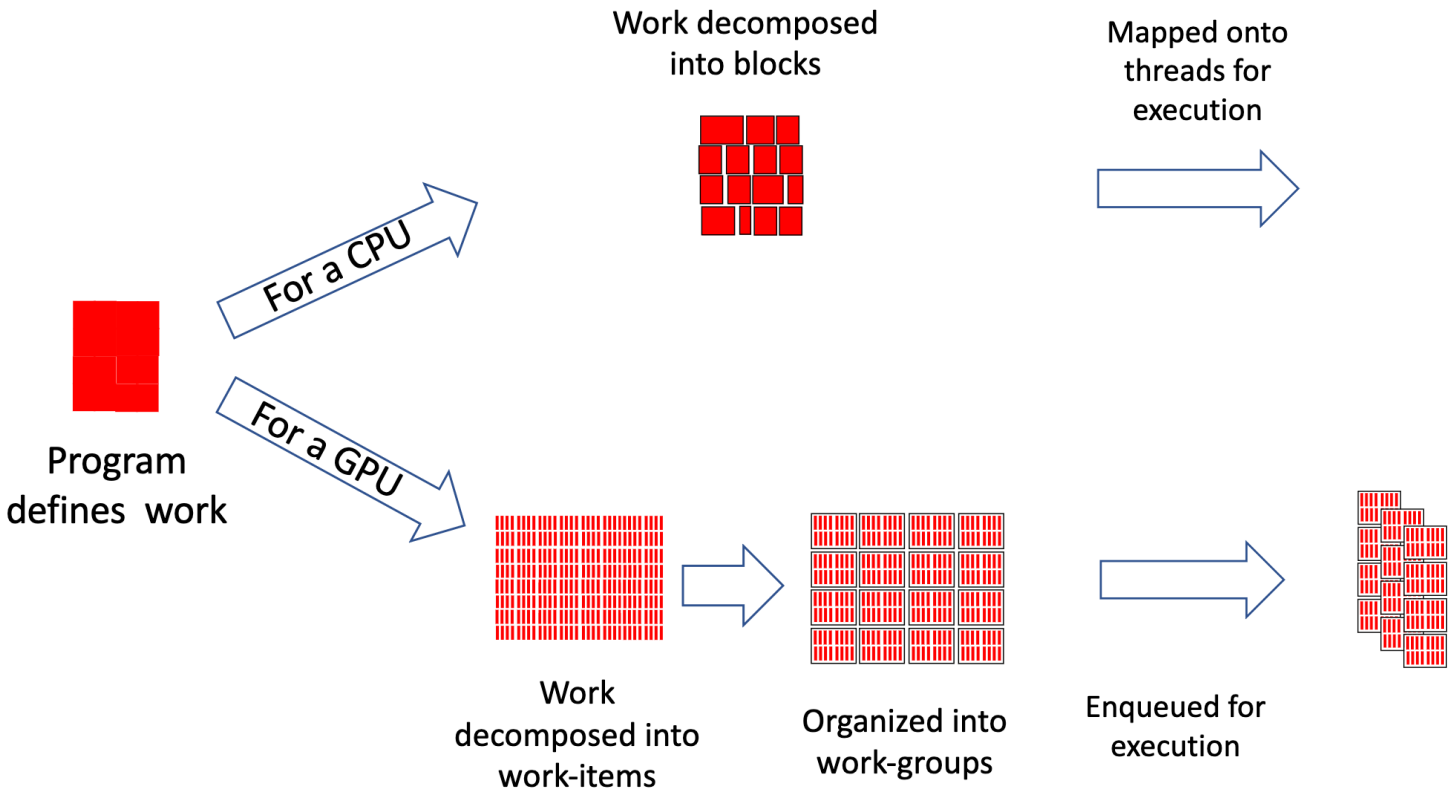
Source: https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2023/08/ATPESC-2023-Track-2b-Matteson-Openmp.pdf; ATPESC 2023

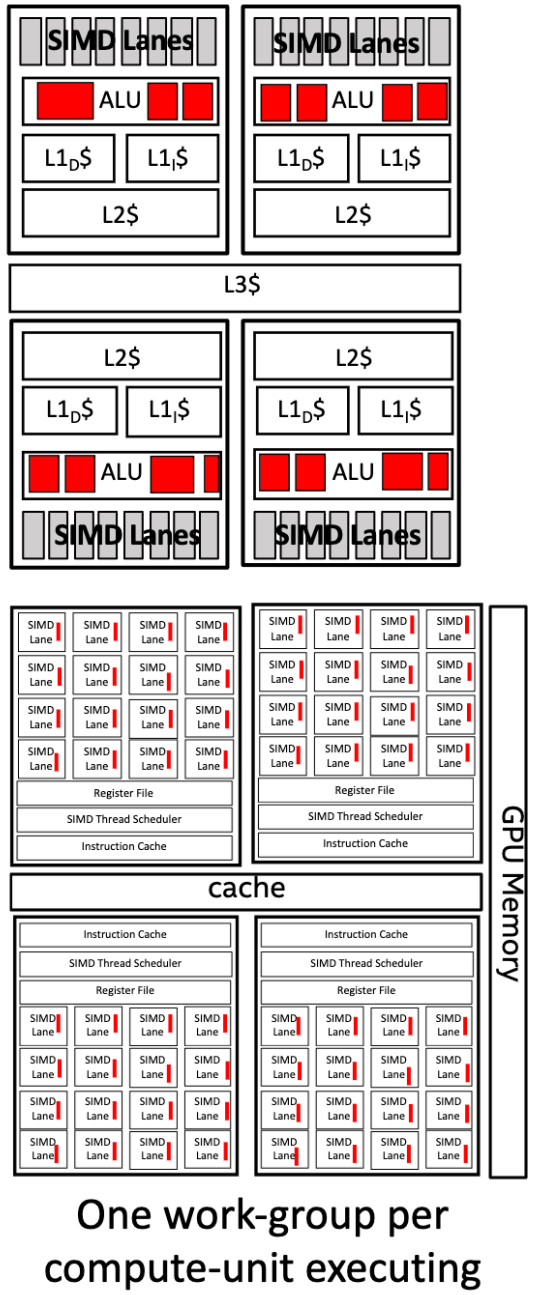
Summary CPU vs GPU computation



Source: https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2023/08/ATPESC-2023-Track-2b-Matteson-Openmp.pdf; ATPESC 2023

Summary CPU vs GPU computation



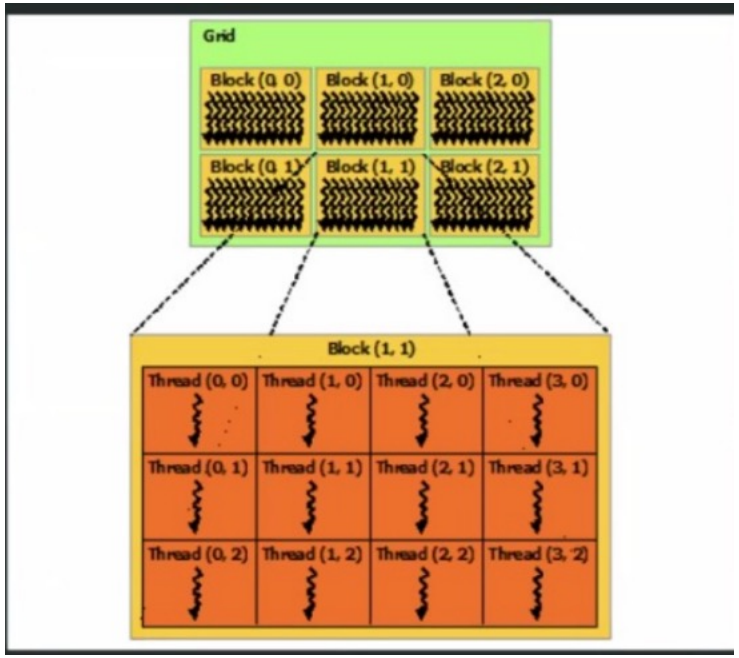


One work-group per compute-unit executing

Source: https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2023/08/ATPESC-2023-Track-2b-Matteson-Openmp.pdf; ATPESC 2023

CUDA terminology

* A kernel is the operation you want to run on your data which will be shipped to the GPU

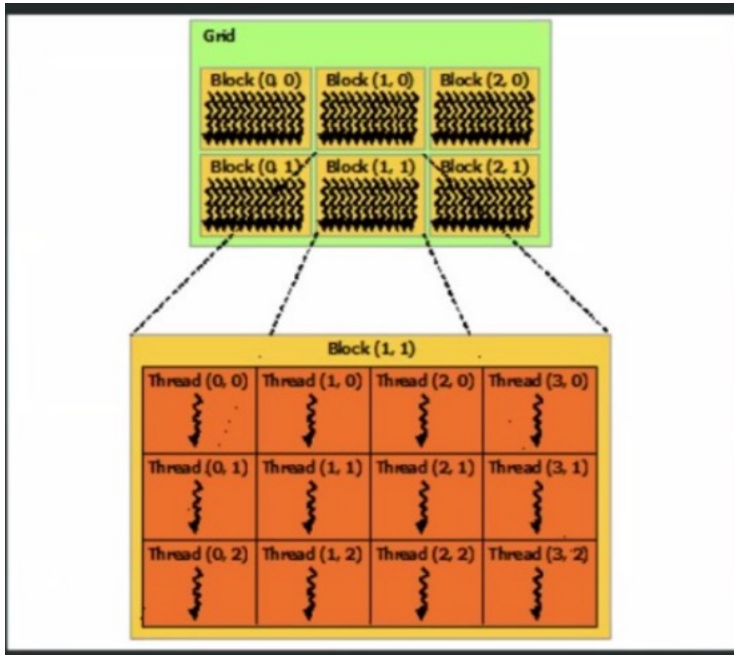


Source: Jayce Dowell Presentation on GPU University of New Mexico, Fall 2021

CUDA terminology

* A kernel is the operation you want to run on your data which will be shipped to the GPU

* Kernels are launched in grid of blocks



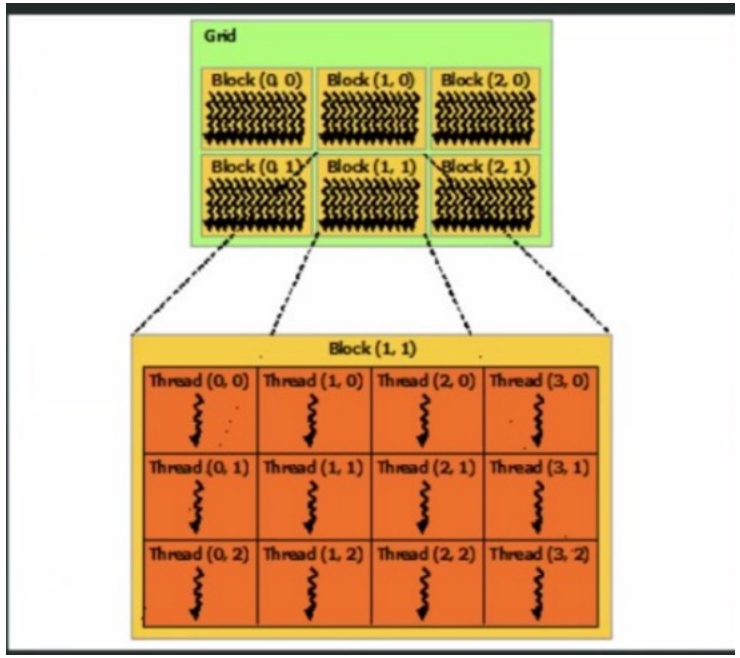
Source: Jayce Dowell Presentation on GPU University of New Mexico, Fall 2021

CUDA terminology

* A kernel is the operation you want to run on your data which will be shipped to the GPU

* Kernels are launched in grid of blocks

* Grids and blocks can be 1D, 2D or 3D



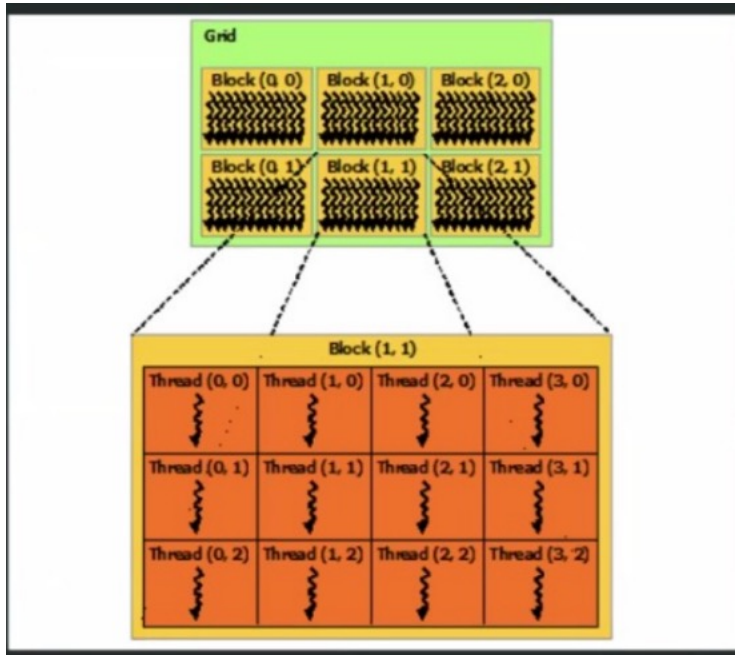
Source: Jayce Dowell Presentation on GPU University of New Mexico, Fall 2021

CUDA terminology

* A kernel is the operation you want to run on your data which will be shipped to the GPU

* Kernels are launched in grid of blocks

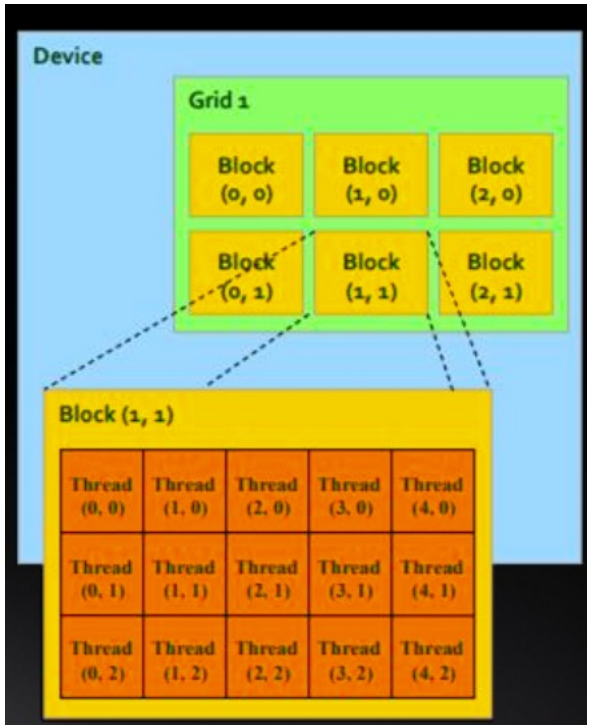
* Grids and blocks can be 1D, 2D or 3D



Source: Jayce Dowell Presentation on GPU University of New Mexico, Fall 2021

Mapping Software to Hardware

* Threads have 3D IDs unique within block

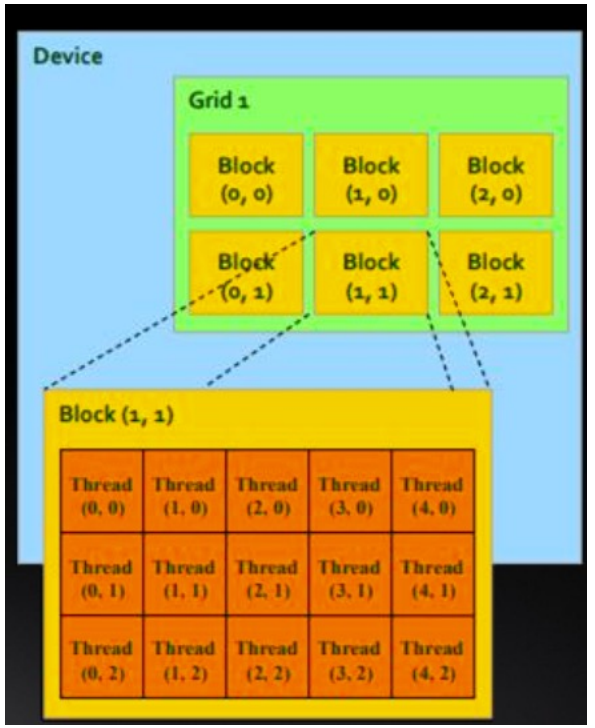


Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

Mapping Software to Hardware

* Threads have 3D IDs unique within block

* Thread blocks have 2D ids unique within grid



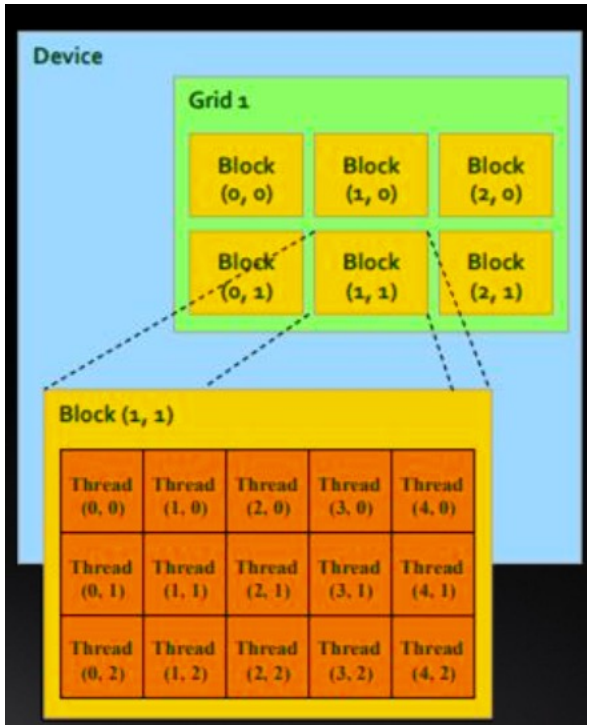
Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

Mapping Software to Hardware

* Threads have 3D IDs unique within block

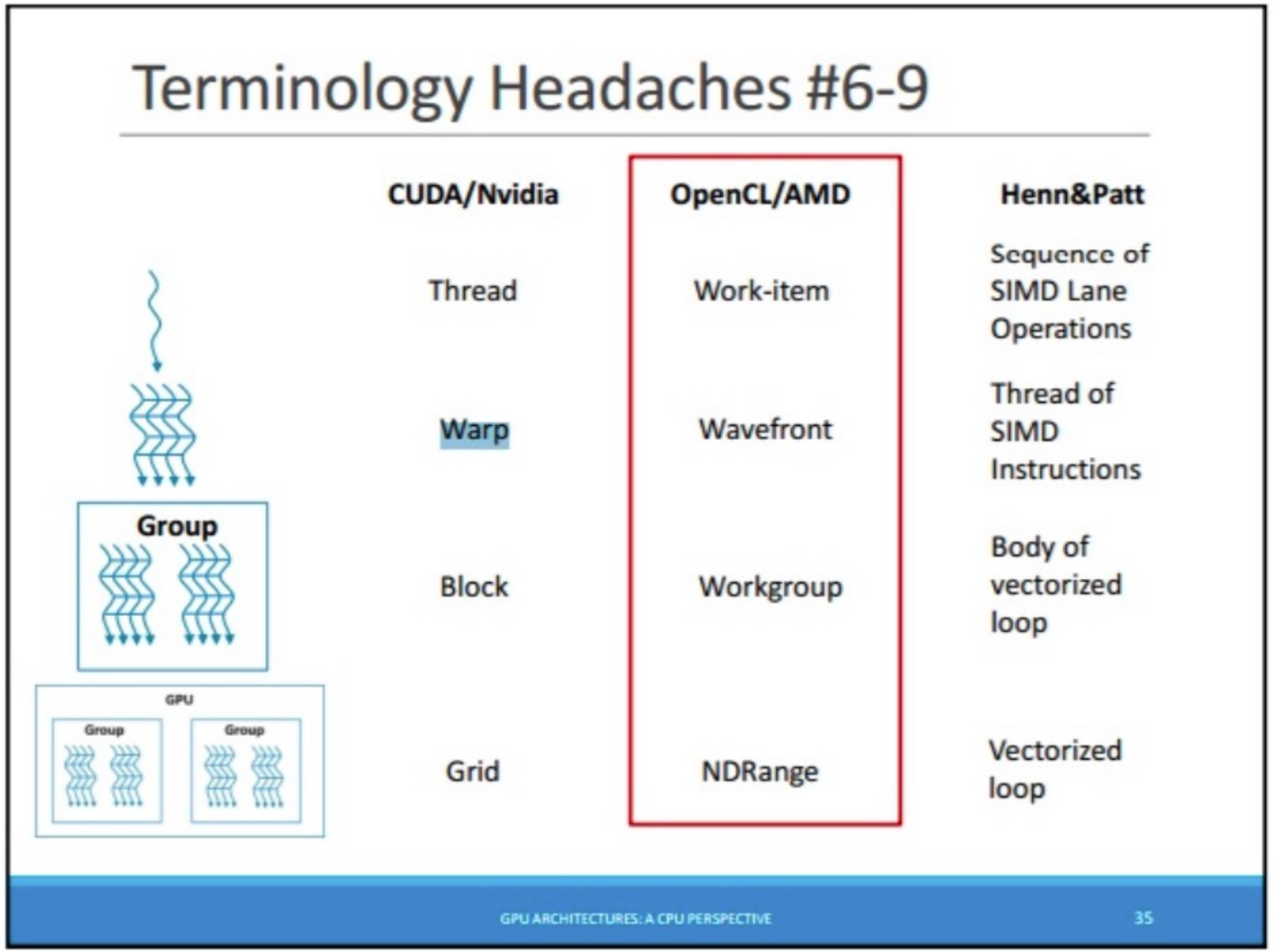
* Thread blocks have 2D ids unique within grid

* Built-in variables”
threadIdx, blockIdx
blockDim, gridDim



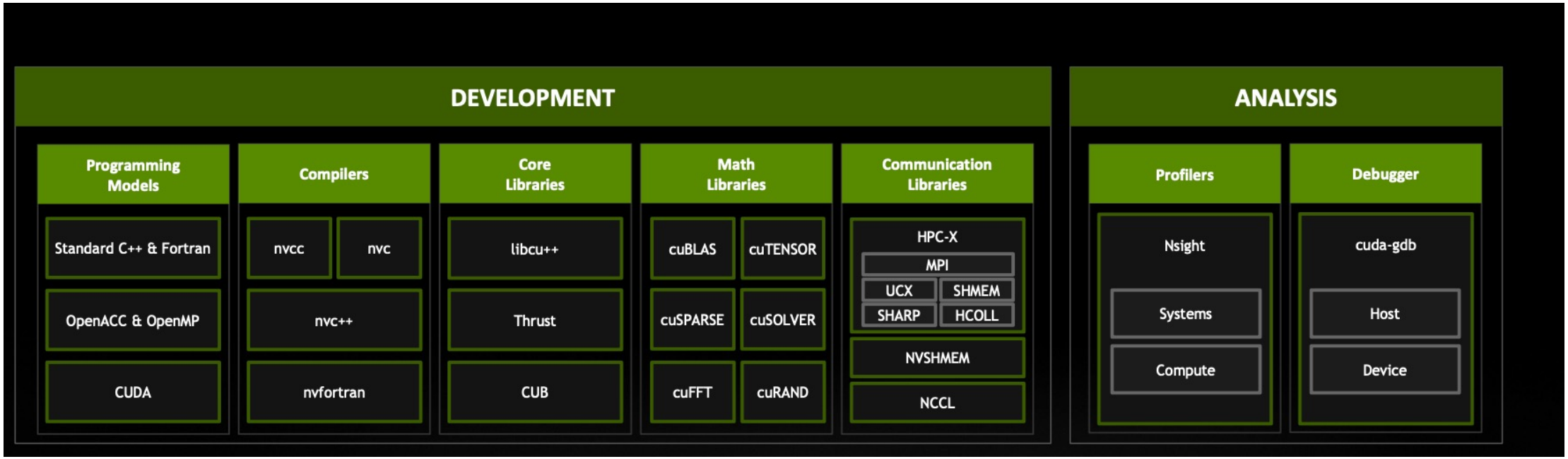
Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPU terminology varies a lot!!!



Source:https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2023/08/ATPESC-2023-Track-2b-Matteson-Openmp.pdf ; ATPESC 2023

NVIDIA HPC SDK



Source: https://www.olcf.ornl.gov/wp-content/uploads/2019/12/03-CUDA-Fundamental-Optimization-Part-1.pdf

GPUs in Python

- NumPy and SciPy do not utilize GPUs out of the box

Source: Using Python at NERSC, September 28th 2022

GPUs in Python

- NumPy and SciPy do not utilize GPUs out of the box

- There are many Python GPU frameworks out there:
  - “drop in” replacements for numpy, scipy, pandas, scikit-learn, etc
    - **CuPy**, **RAPIDS**

Source: Using Python at NERSC, September 28th 2022

GPUs in Python

- NumPy and SciPy do not utilize GPUs out of the box

- There are many Python GPU frameworks out there:
  - "drop in" replacements for numpy, scipy, pandas, scikit-learn, etc
    - **CuPy**, **RAPIDS**
  - "machine learning" libraries that also support general GPU computing
    - **PyTorch**, **TensorFlow, JAX**

Source: Using Python at NERSC, September 28th 2022

GPUs in Python

- NumPy and SciPy do not utilize GPUs out of the box

- There are many Python GPU frameworks out there:
  - "drop in" replacements for numpy, scipy, pandas, scikit-learn, etc
    - **CuPy**, **RAPIDS**
  - "machine learning" libraries that also support general GPU computing
    - **PyTorch**, **TensorFlow, JAX**
  - "I want to write my own GPU kernels"
    - **Numba, PyOpenCL, PyCUDA, CUDA Python**
  - multi-node / distributed memory:
    - **mpi4py+X, dask, cuNumeric**

Source: Using Python at NERSC, September 28th 2022

GPUs in Python

- NumPy and SciPy do not utilize GPUs out of the box
- There are many Python GPU frameworks out there:
  - "drop in" replacements for numpy, scipy, pandas, scikit-learn, etc
    - **CuPy**, **RAPIDS**
  - "machine learning" libraries that also support general GPU computing
    - **PyTorch**, **TensorFlow, JAX**
  - "I want to write my own GPU kernels"
    - **Numba, PyOpenCL, PyCUDA, CUDA Python**
  - multi-node / distributed memory:
    - **mpi4py+X, dask, cuNumeric**

Source: Using Python at NERSC, September 28th 2022

Next time

- We will learn about GPU programming with Python (Numba, Cupy)
- Learn basic kernel programming and operations
- Things to be aware of (how to locate the threadID, determine the correct number of threads to launch etc …)