Deep Learning

A course about theory & practice

UNIVERSITÀ DI PAVIA

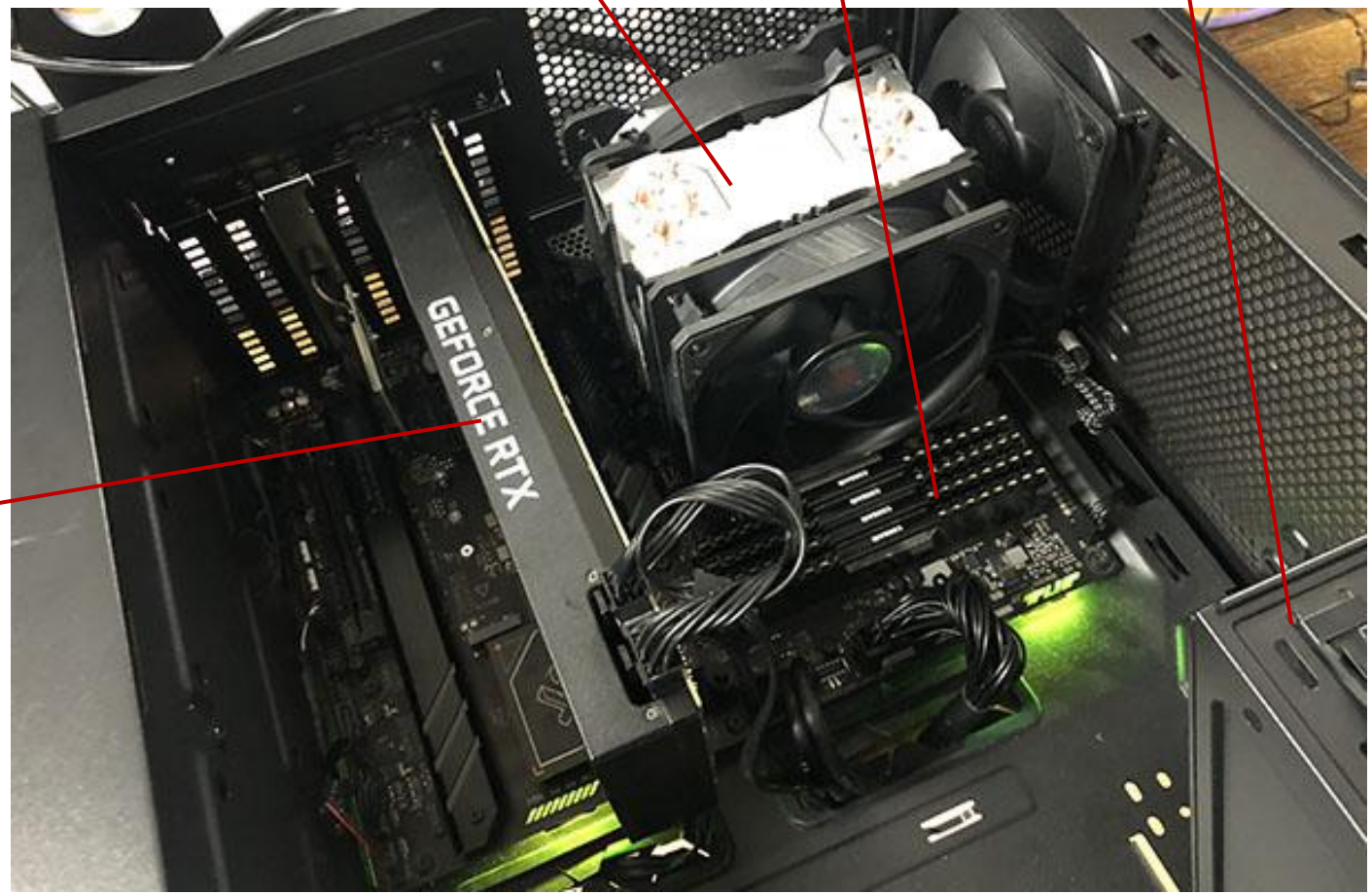
Hardware for Deep Learning

Marco Piastra

GPU vs. CPU

- ***The GPU resides on a separate board***

  An almost independent computer



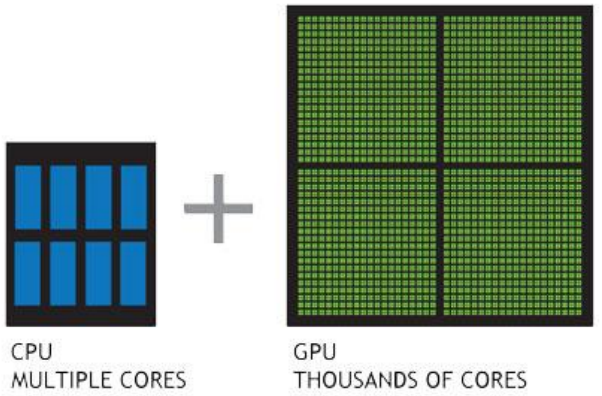
[image from https://www.researchgate.net/publication/322525660]

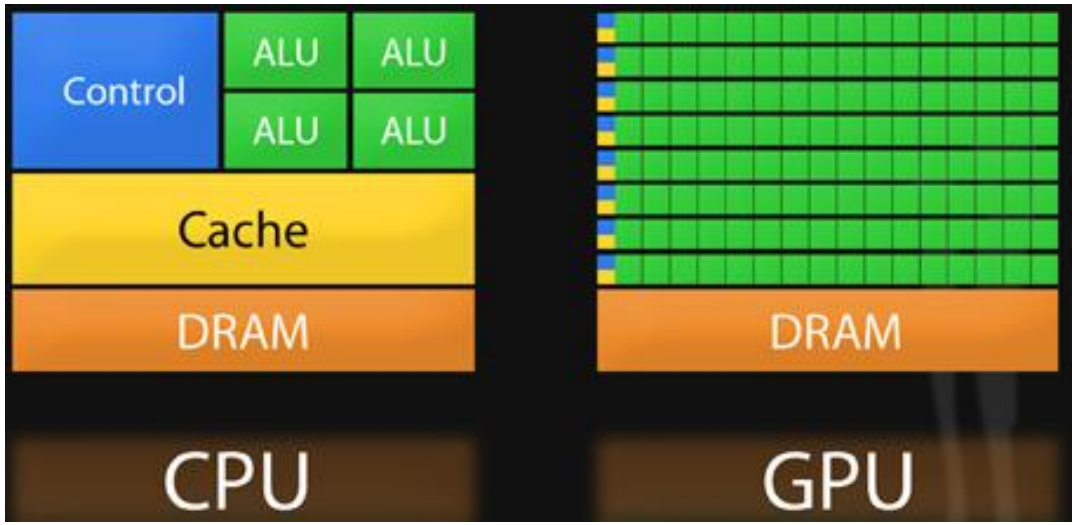
GPU vs. CPU

- ***Different hardware architectures***

  Different computing paradigms

  A trade-off between
  - fully independent cores (CPU)
  - interdependent cores (GPU)
    with some (limited) degrees of independence





[images from http://www.nvidia.com/docs/]

GPU vs. CPU

- ***Different hardware architectures***

Device Type	Device Model	Core Count / Architecture	Memory	TFLOPS (FP32)	Key Advantage
CPU	**Intel Xeon 6 (Granite Rapids)**	128 Cores (P-cores)	12-channel DDR5 / MCR DIMM	~4.5 TFLOPS (via AMX)	Extreme versatile branching & high-speed system IO.
GPU (Consumer)	**NVIDIA RTX 5090**	21,760 CUDA Cores	32 GB GDDR7	~100+ TFLOPS	Leading edge for local LLM inference and rendering.
GPU (Data Center)	**NVIDIA B200 (Blackwell)**	160 SMs / Transformer Engine Gen 2 (approx. 20,480 CUDA Cores)	192 GB HBM3e (8 TB/s)	~40 TFLOPS (FP32) / 20 PFLOPS (FP4)	The standard for trillion-parameter model training.
GPU (Data Center)	**AMD Instinct MI325X**	304 CUs (CDNA 3) (approx. 19,456 Stream Processors)	288 GB HBM3e (6 TB/s)	~100+ TFLOPS	Massive VRAM capacity for large-model inference without offloading.
TPU	**Google Cloud TPU v5p**	8,960 chips per pod	95 GB HBM per chip	~450+ TFLOPS	Superior "Performance-per-Dollar" for large-scale Pod training.

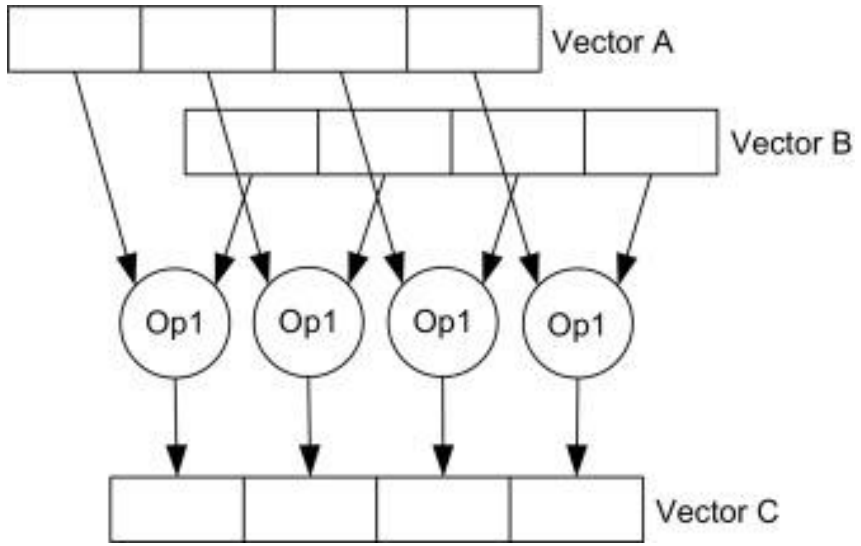
SIMT Parallelism

- ***Single Instruction, Multiple Data*** (SIMD)

  Execution is parallel

  All cores are executing the same instruction, in sync

  Each core works on specific data



[images from https://www.sciencedirect.com/topics/computer-science/single-instruction-multiple-data]

SIMT Parallelism

- ***Single Instruction, Multiple Threads*** (SIMT)

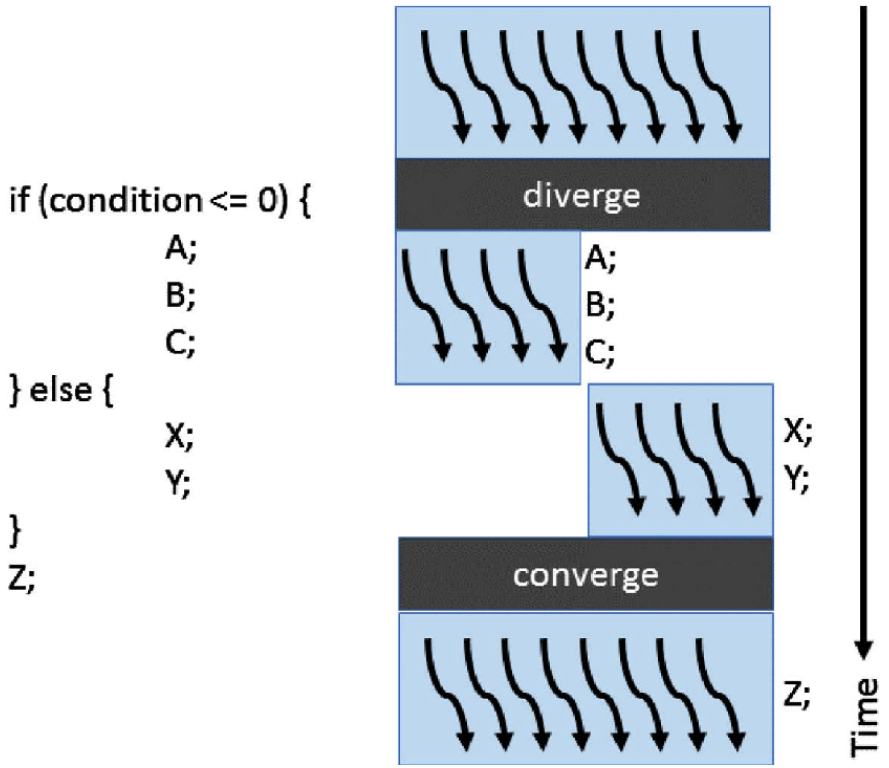
  Execution is parallel

  All ***active*** cores are executing the same instruction, in sync

  Each core works on specific data

  The control system activates and deactivates cores on each *execution branch*

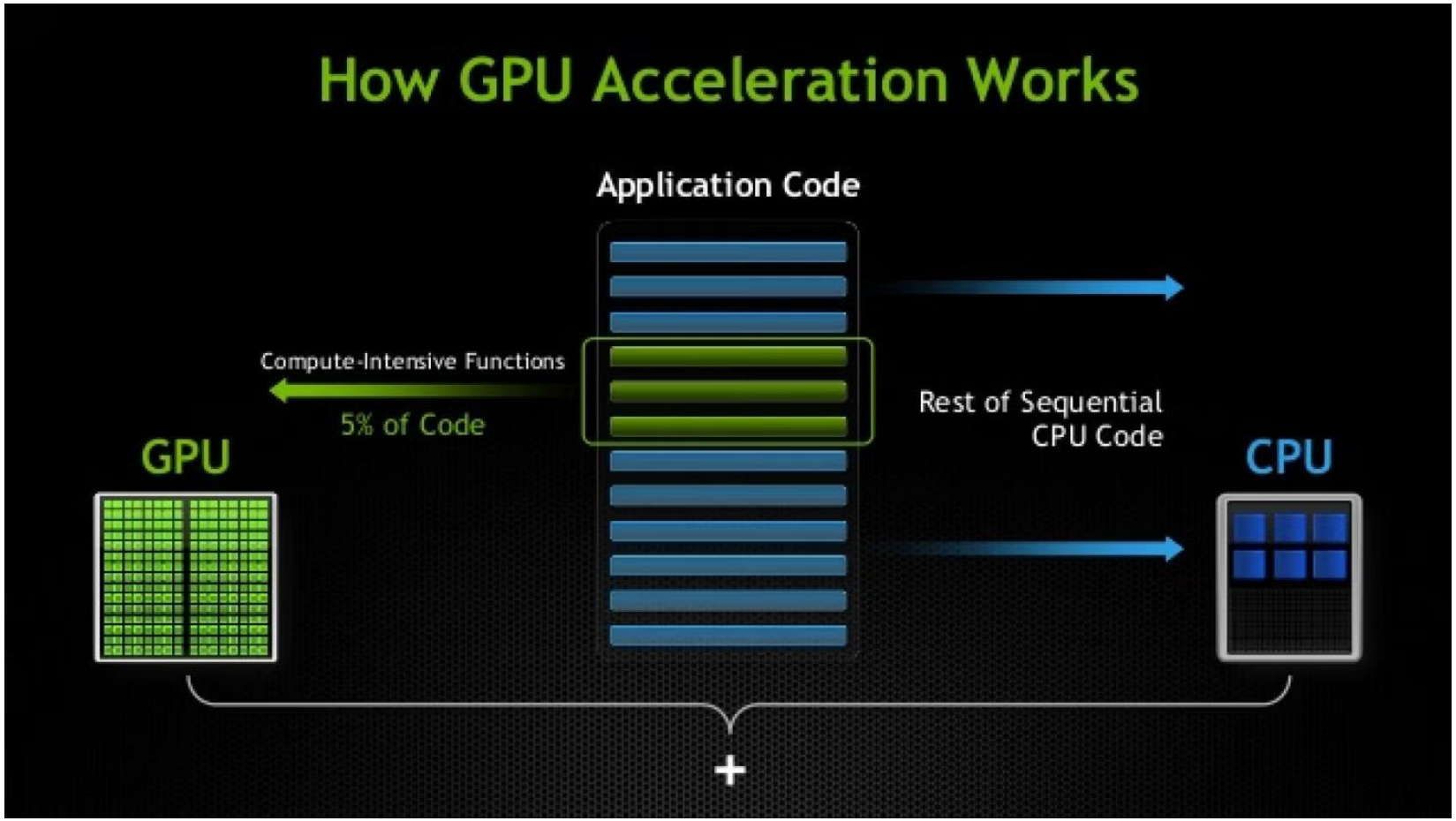
  Moral: any computation might be performed, but divergent threads will be sequentialized



[images from https://www.sciencedirect.com/topics/computer-science/single-instruction-multiple-data]

Selective parallelization

Not all parts of a program are worth executing in parallel...



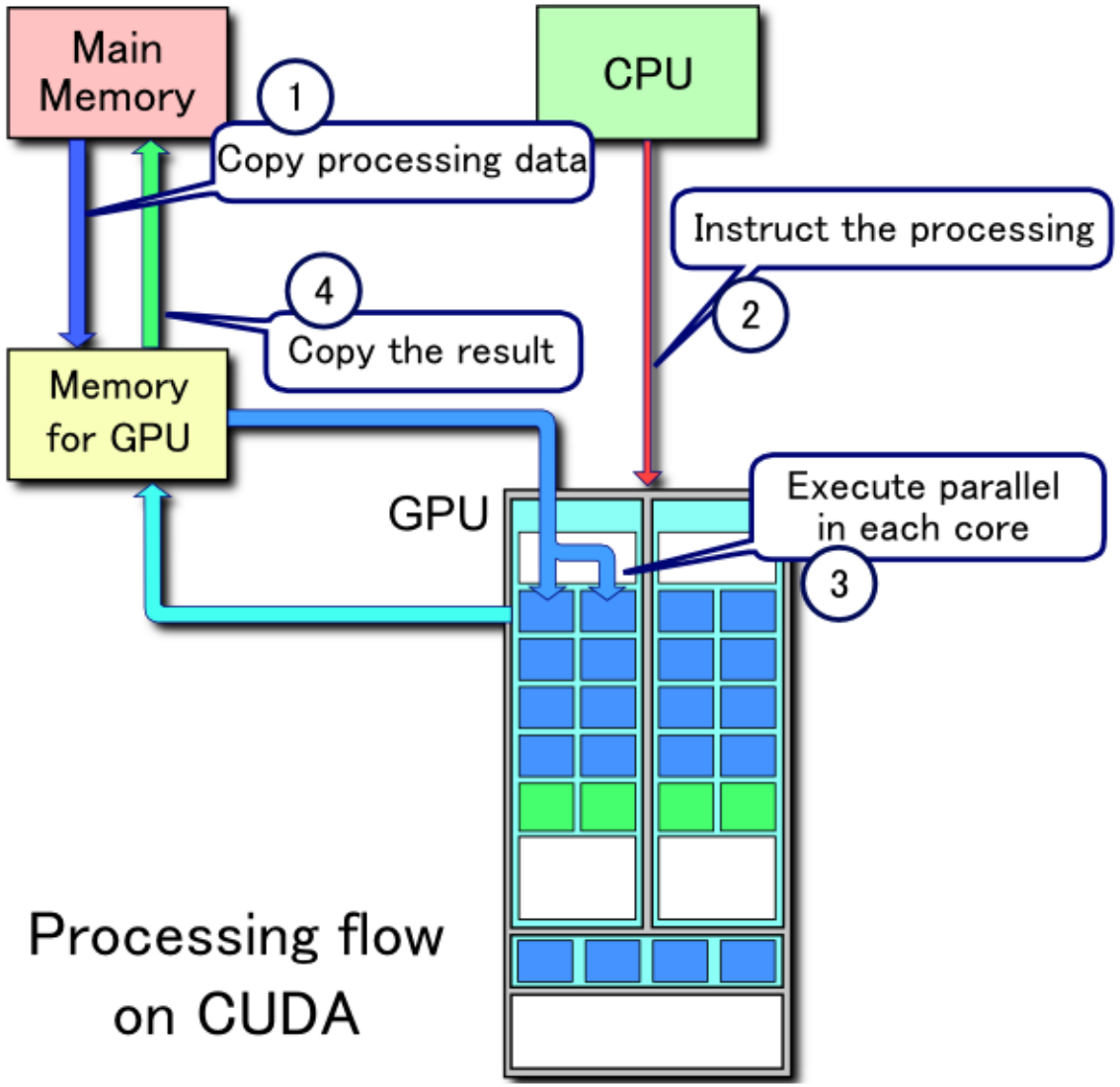
[images from http://www.nvidia.com/docs/]

GPU Processing Cycle

- ***CPU > Memory Transfer > GPU and back***

  The program on the CPU drives the execution:

  1. All data (program + actual data) are transferred from main memory to GPU DRAM
  2. The GPU kernel is launched
  3. The Kernel is executed in parallel onto GPU cores, using GPU DRAM
  4. Results are copied back from GPU DRAM to main memory



[image from https://commons.wikimedia.org/wiki/File:CUDA_processing_flow_(En).PNG]

PyTorch and GPUs

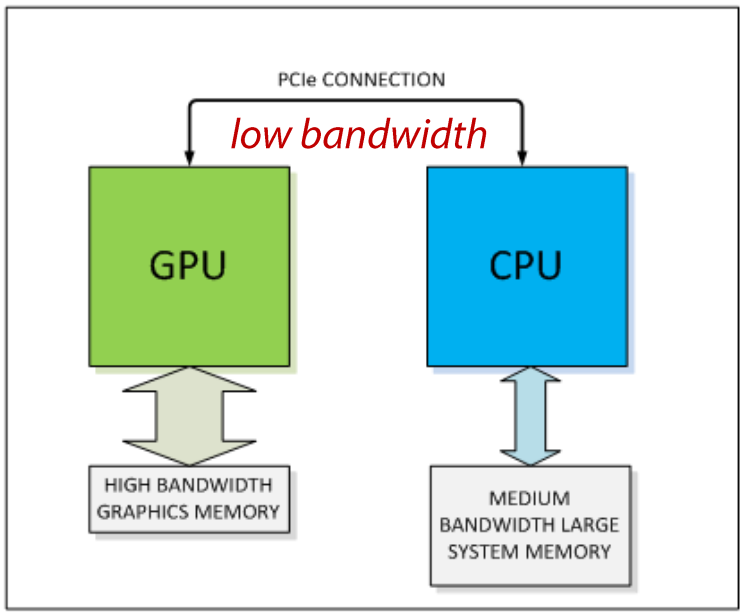
- PyTorch computations are optimized to be run on **GPUs**

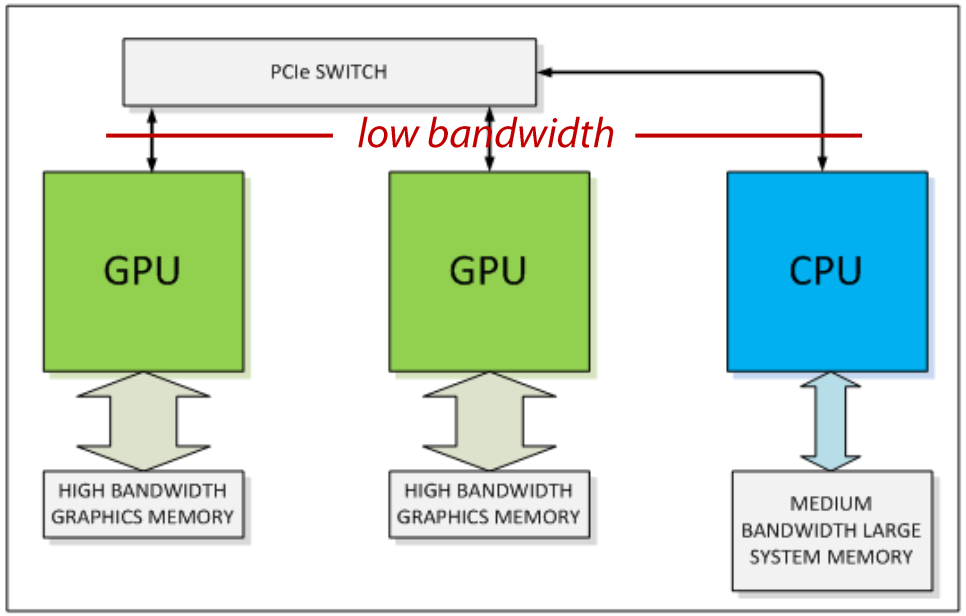
  For the programmer, these implementation details are (mostly) **transparent**

  TF can also run on the CPU only, but with lower performance.

- PyTorch automatically manages **memory transfers** to/from GPUs

  Memory transfers are very costly, due to low bandwidth PCIe



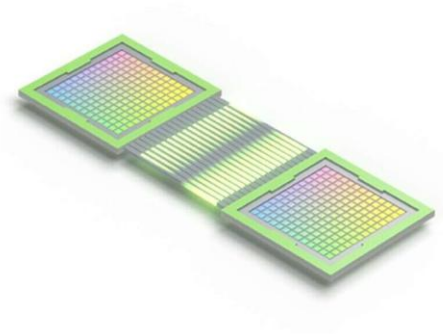


High Speed Interconnect

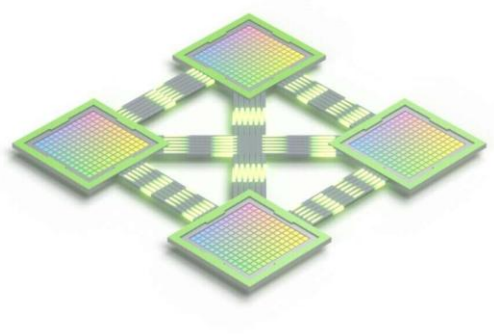
- Available for large GPUs (data center)

  Dedicate direct link between GPUs:

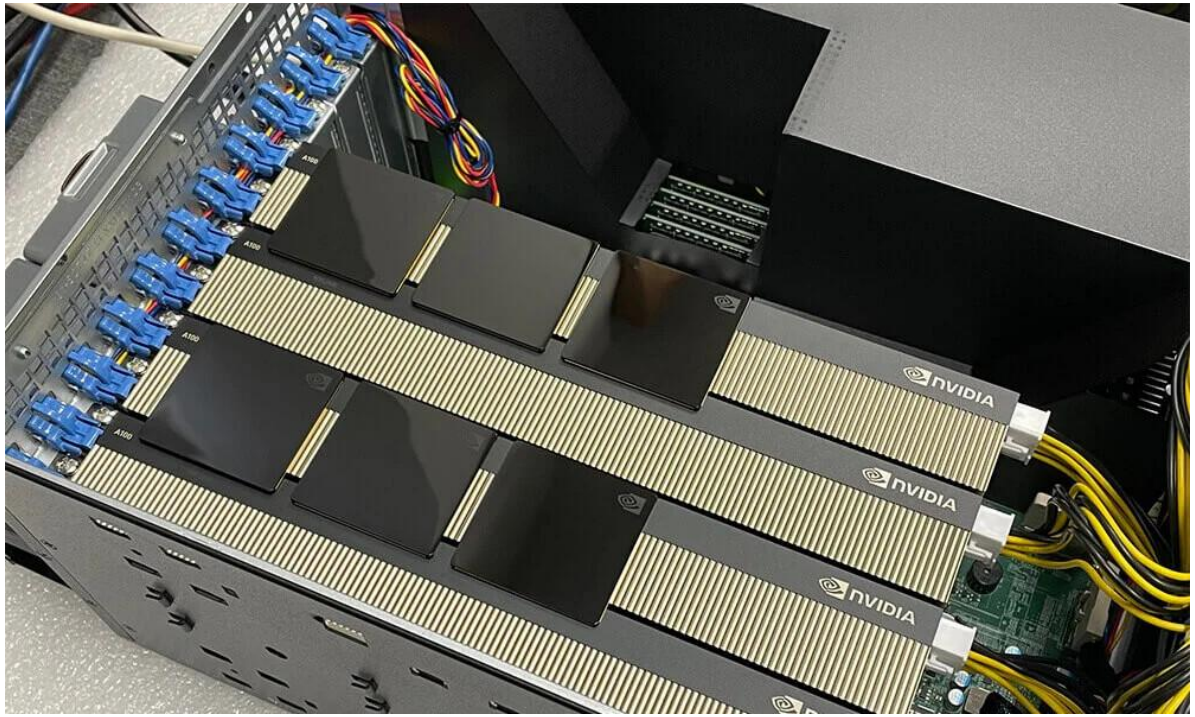
  - High bandwidth, for faster data communication
  - Low latency
  - Scalability
  - Energy efficiency



NVIDIA H100 PCIe with NVLink GPU-to-GPU connection



NVIDIA H100 with NVLink GPU-to-GPU connections

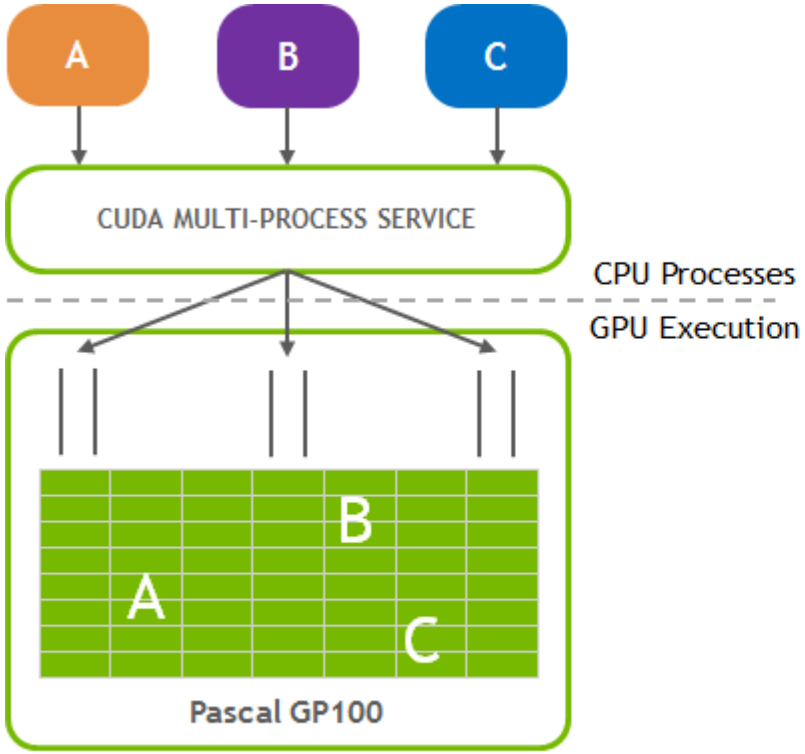


[image from https://www.cudocompute.com/blog/a-beginners-guide-to-nvidia-gpus]

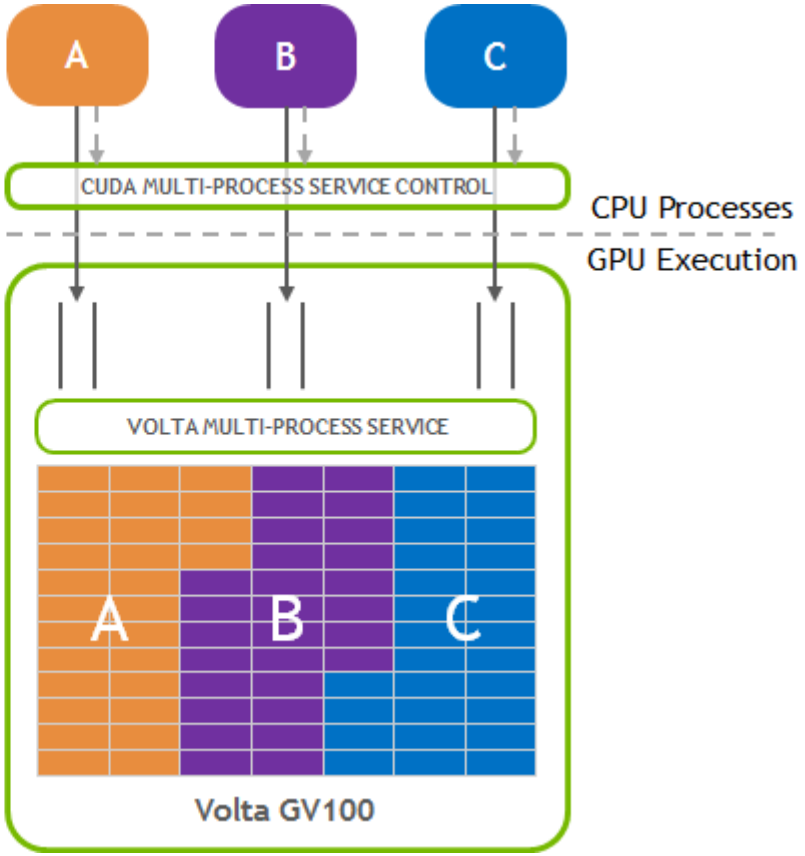
GPU Multiprocessing

Until recently, GPUs could only serve one process at time

Now they can be partitioned among several processes



2016



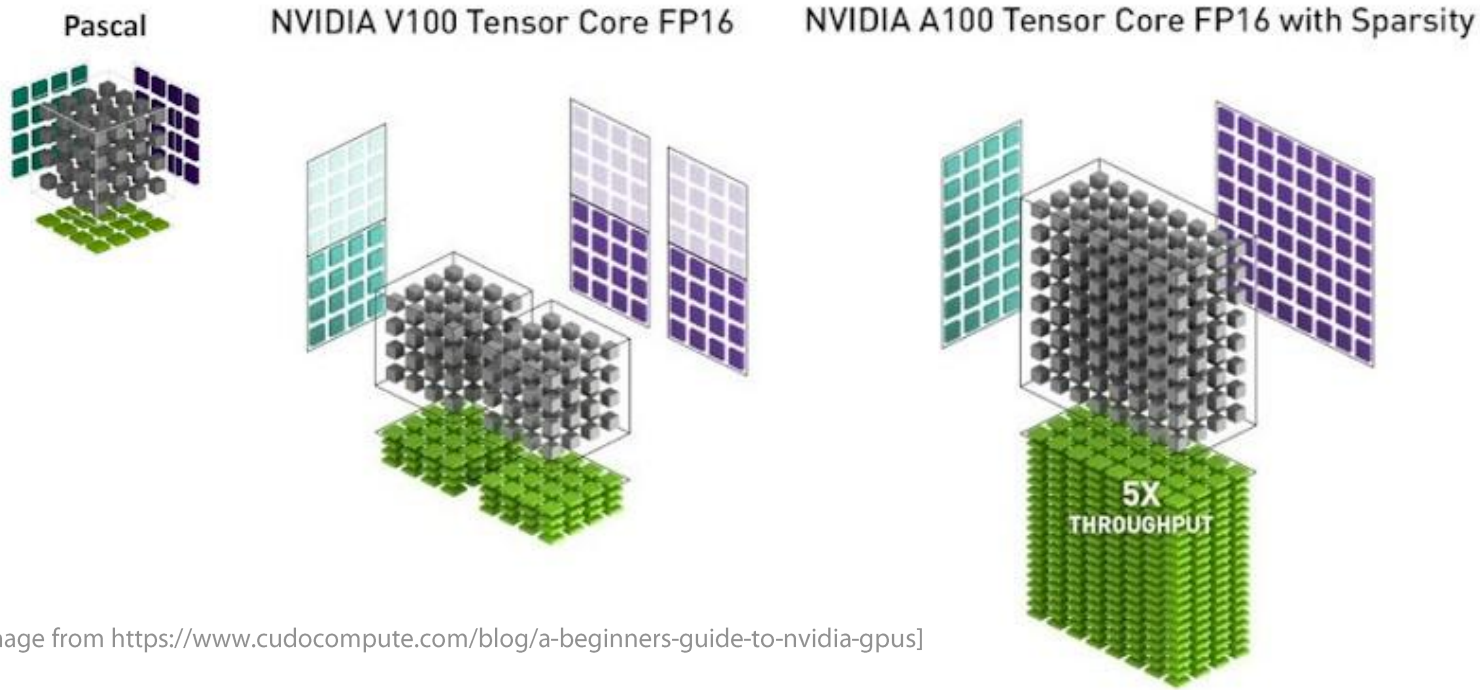
2017

[images from https://docs.nvidia.com/deploy/mps/]

Tensor Cores

Specialized processing units to accelerate tensor algebra operations

- **Matrix Multiply-Accumulate (MMA) units**
  Each MMA unit can perform a 4x4 matrix multiply-accumulate operation in a single clock cycle
- **Warp schedulers**
  MMA units are kept busy and the data flow is optimized
- **High-speed registers and shared memory**
  For storing and sharing intermediate among threads



[image from https://www.cudocompute.com/blog/a-beginners-guide-to-nvidia-gpus]

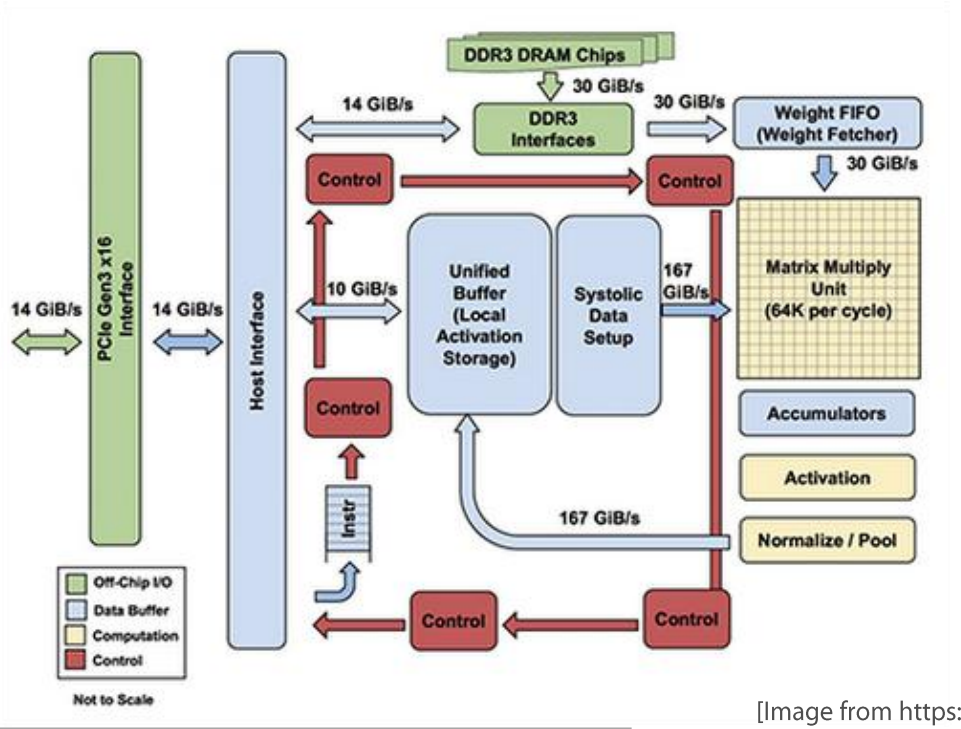
In-Cloud TPUs

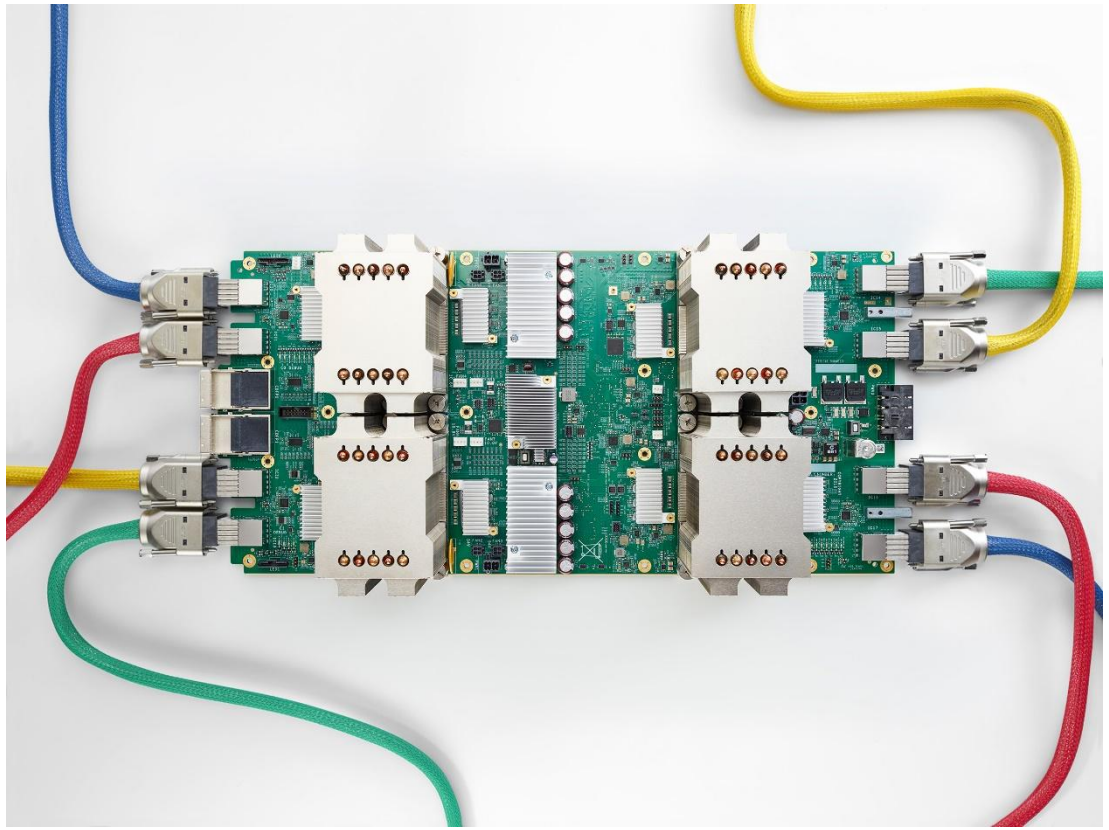
- Tensor Processing Units (TPUs)

  They are ASICs (Application-Specific Integrated Circuits) and are not on sale

  As computation resources, they are only available in cloud (at Google)

  TPUs are mounted on separate boards, much like GPUs





[Image from https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu]

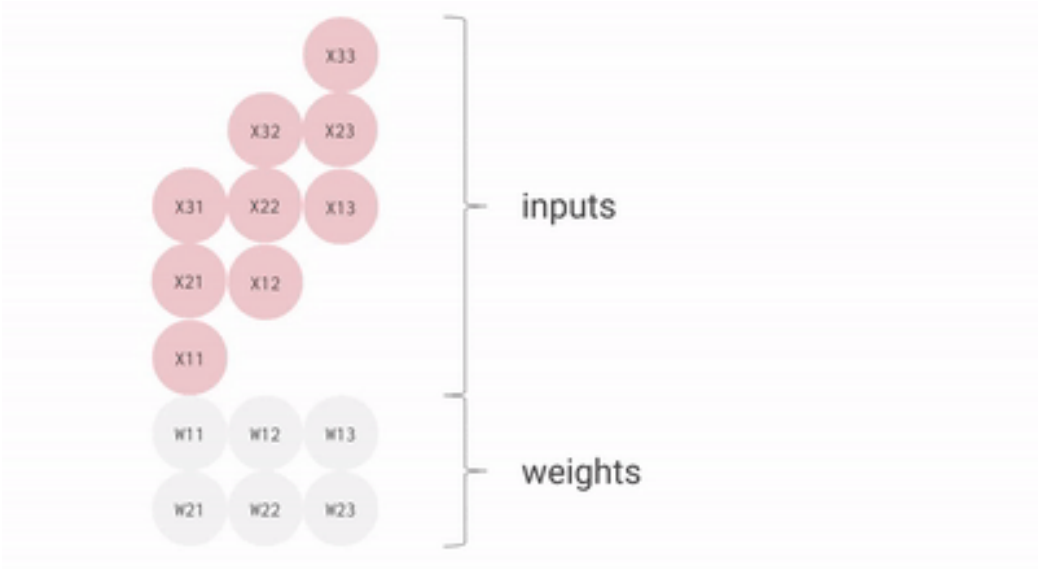
Systolic Parallel Processing

- Data flow through cores

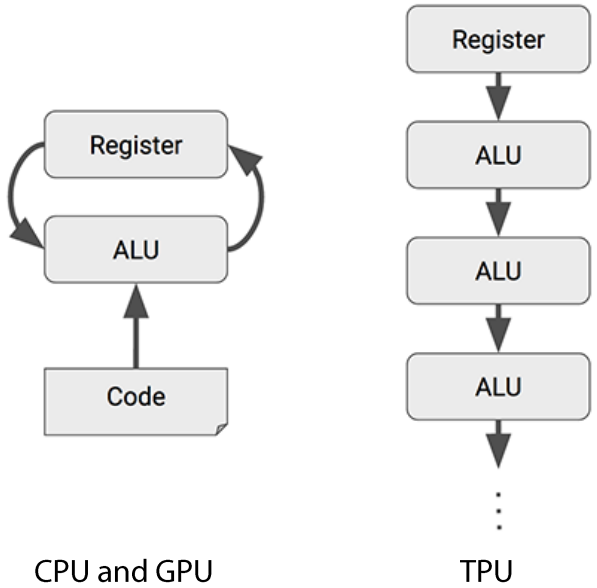
  TPU architecture is optimized natively for tensor processing and not for graphics

  Arithmetic Logic Units are organized in a pipeline

  Tensor data are made to 'flow through' the pipeline



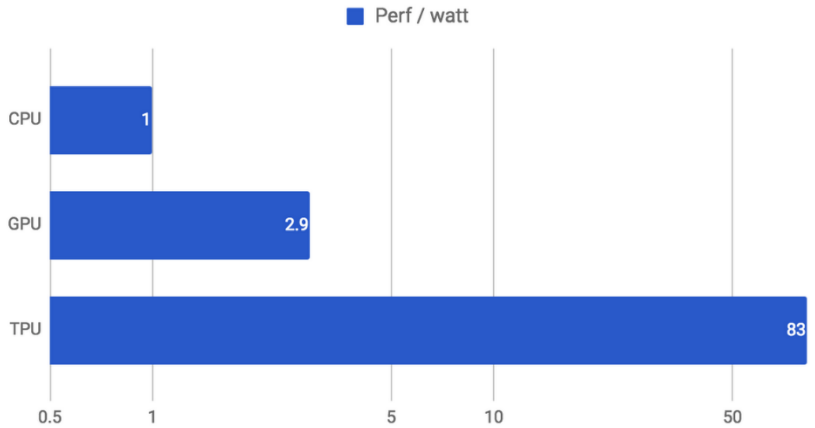
Matrix multiplication



CPU and GPU

TPU

TPUs can be much more efficient for tensor computations



[Image from https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu]