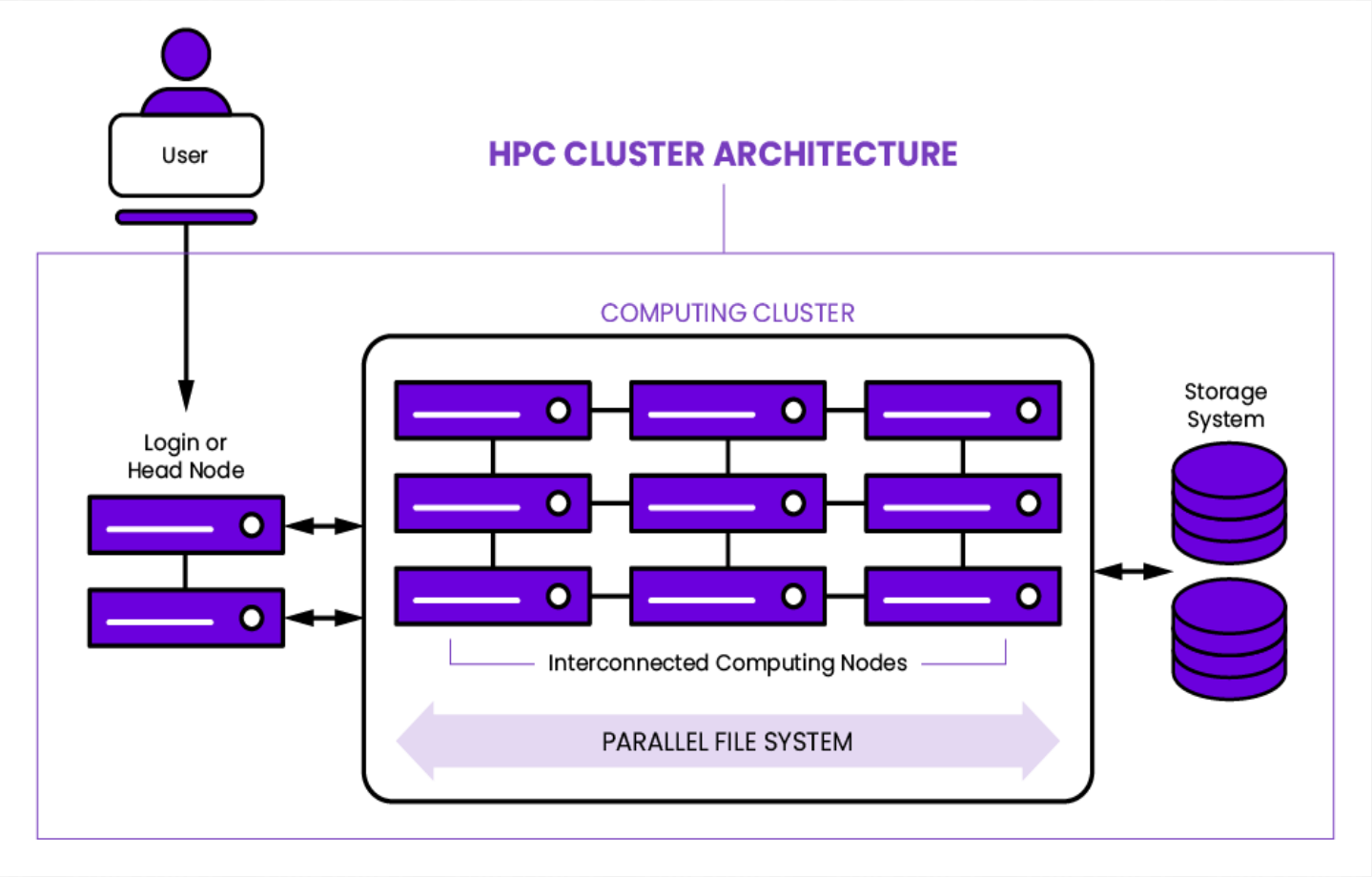
Overview of Accelerated Computer Architectures

3rd Latin American Introductory School on Parallel Programming and Parallel Architecture for HPC

Dr. Fernando Posada
Assoc. Research Professor
Temple University

HPC Cluster



- Compute Power
- Network
- File system

- Deployment
- Scheduler
- Software Stack

- The HPC Cluster design should match research's needs (not researcher's).

HPC Cluster Trends

Past

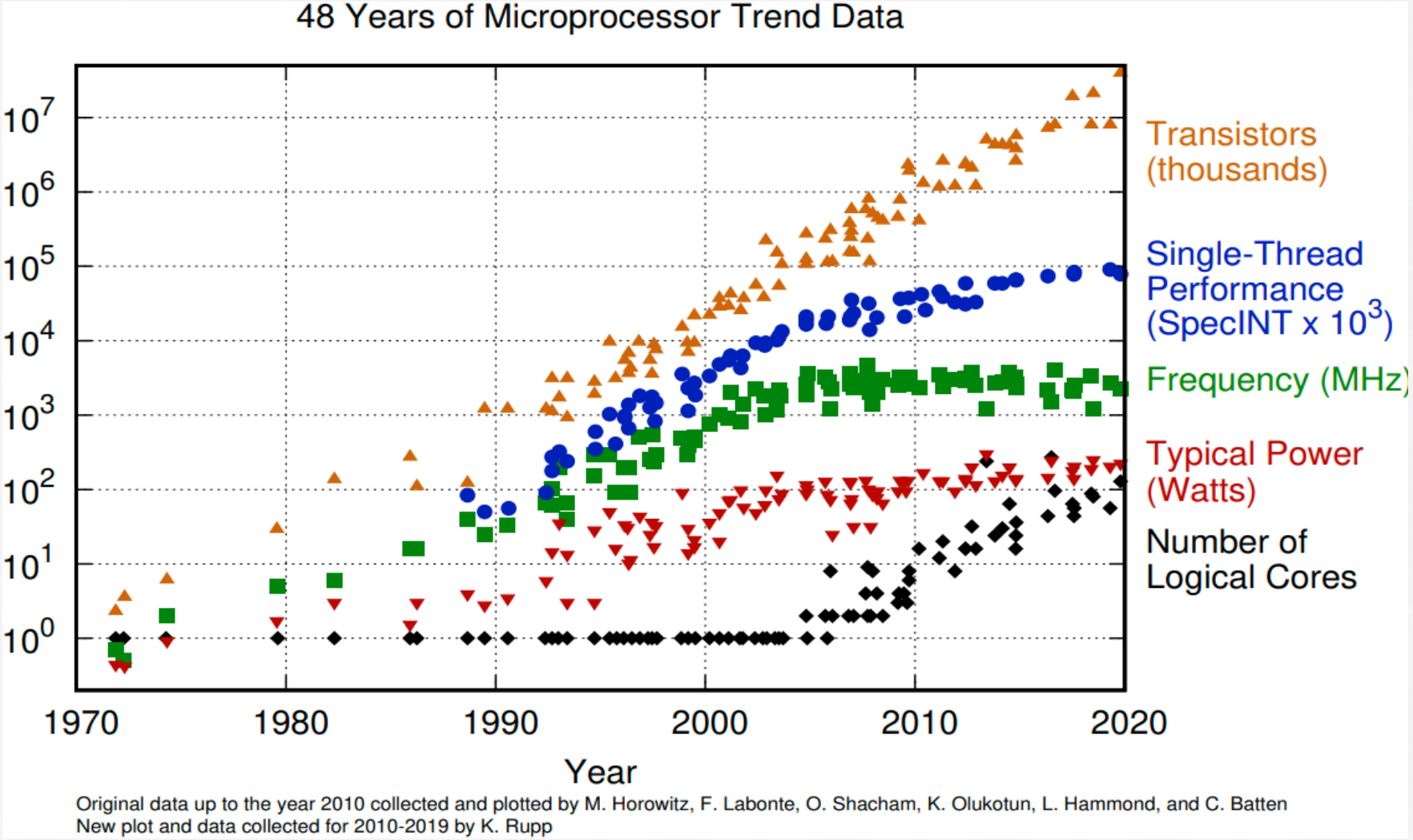
- CPU based
- HW-bounded
  - Memory
  - Storage
  - Interconnect
- Critical code optimization

Present

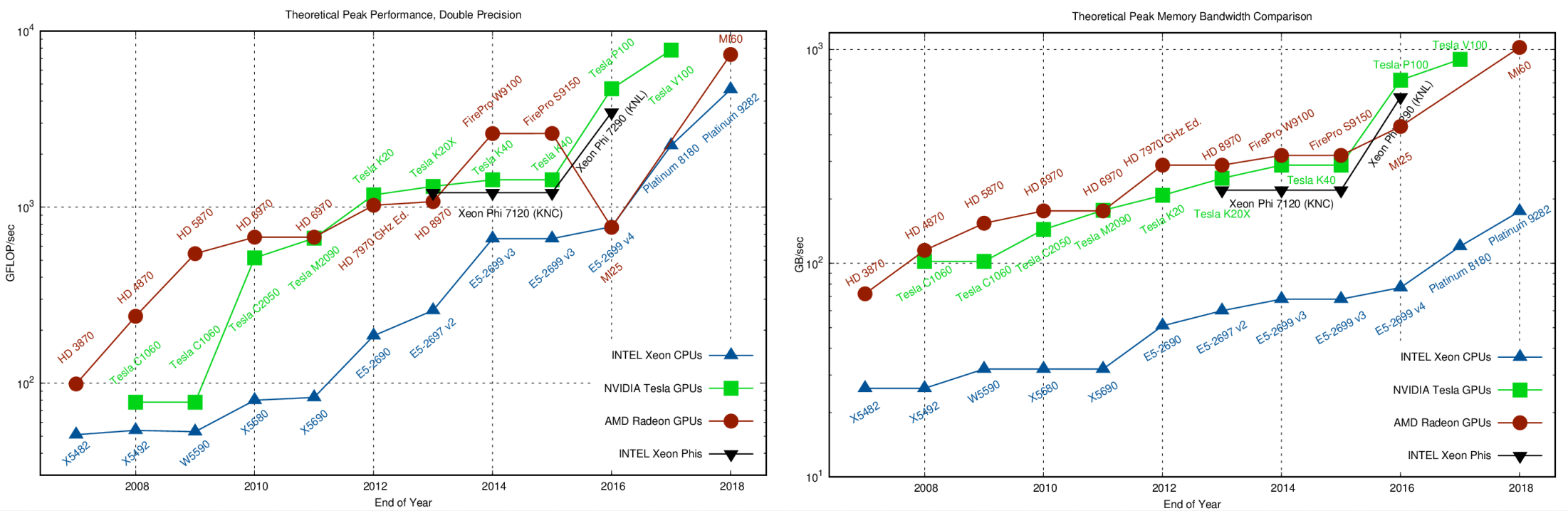
- Heterogenous Architecture
  - Accelerators / Coprocessors
- AI / ML driven
- Data-intensive
- Framework Oriented
- Interconnect-bounded

Moore's Law

The number of transistors per chip increase every 2 years or so



GPU vs CPU



A growth in accelerator performance over the years in comparison to Intel CPU performance. (Image taken from ENCCS)

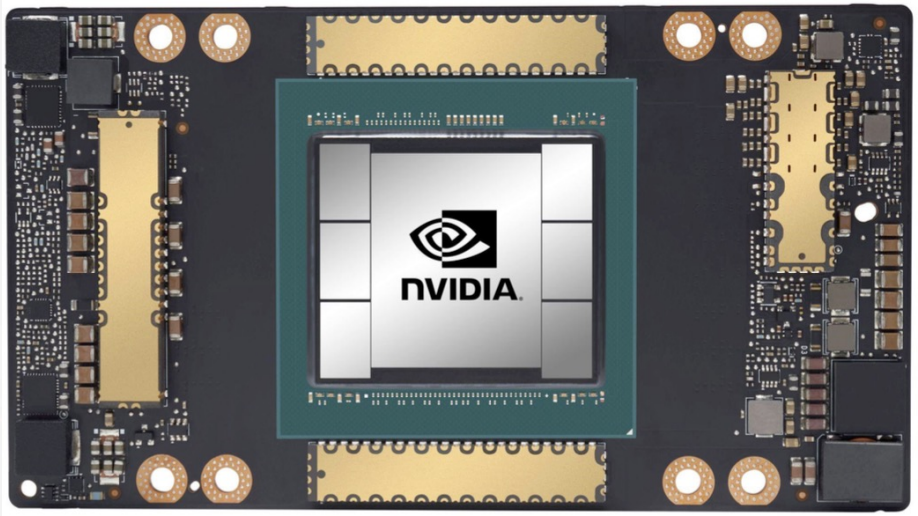
TOP500

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

- >1 Exaflop/s!
- Only possible with accelerators

Accelerators = GPGPU

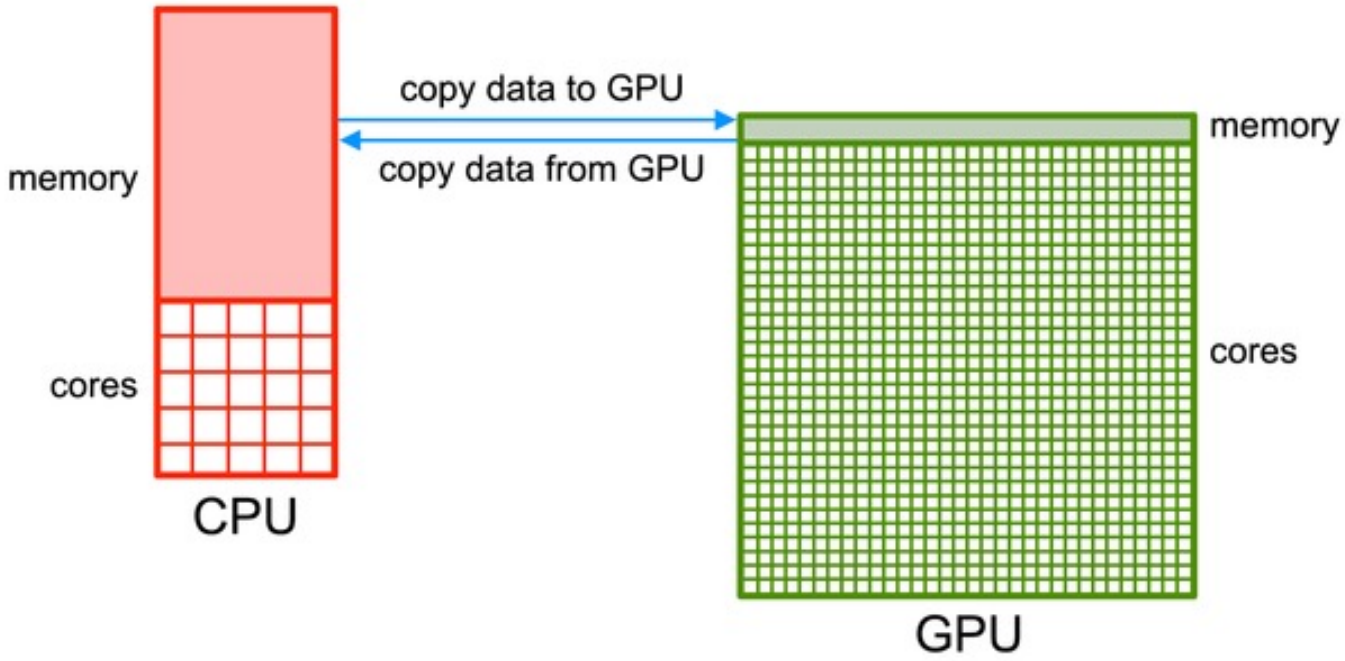
- General-Purpose Graphics Processing Unit
- Using GPUs not only for graphics!
- GPGPUs are widely used in finance, healthcare, and scientific research.
- NVIDIA and AMD commodity hardware.



NVIDIA A100 GPU SMX4 Module

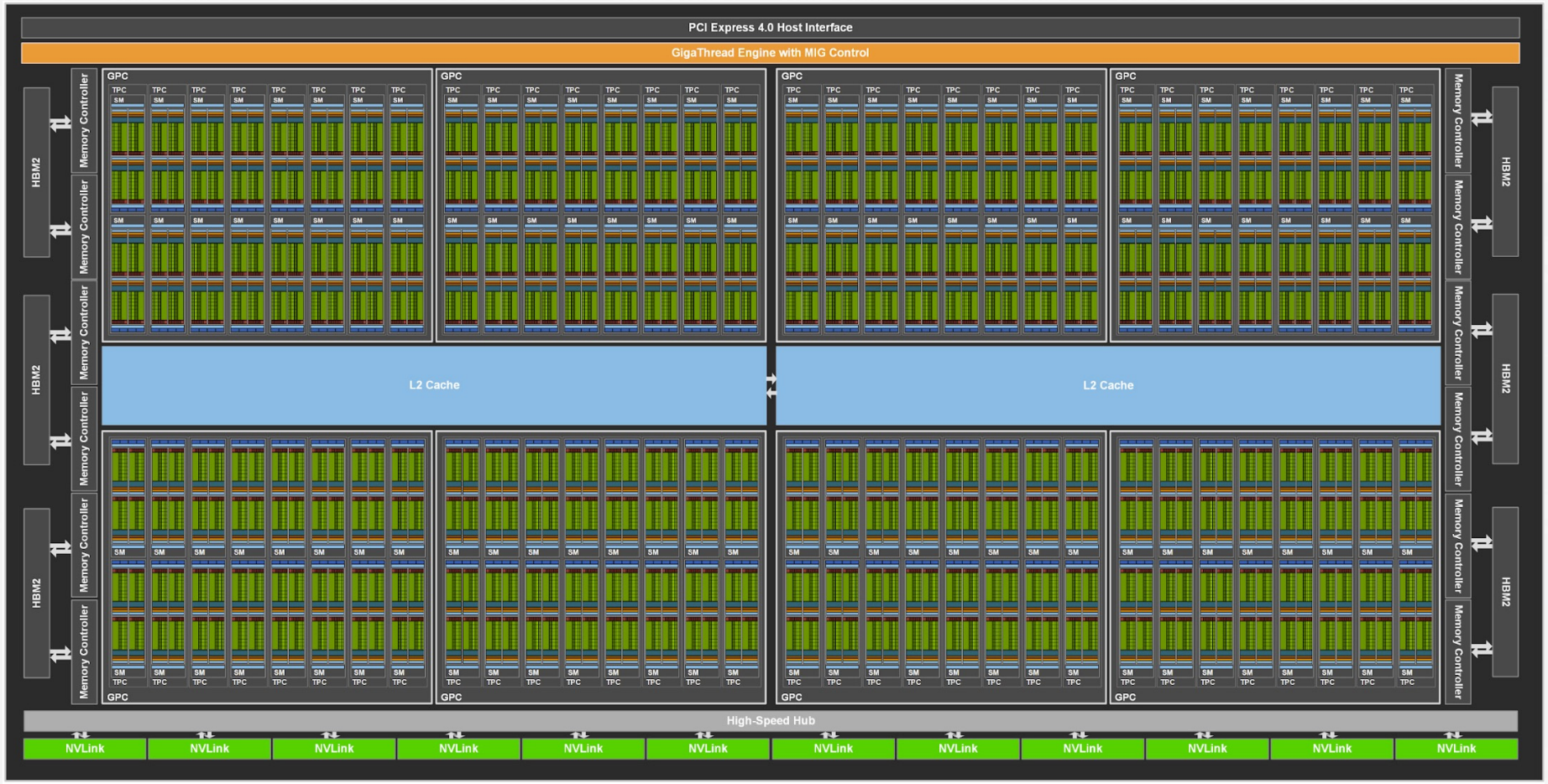
CPU + GPU

(co-processor)




```
data = open("input.dat");          # read the data on the CPU
copyToGPU(data);                   # copy the data to the GPU
matrix_inverse(data.gpu);          # perform a matrix operation on the GPU
copyFromGPU(data);                 # copy the resulting output to the CPU
write(data, "output.dat");         # write the output to file on the CPU
```

GPU Hardware Components (NVIDIA A100)



A100 128 SMs

- 8 GPCs
- 16 SM/GPC
- 80GB HBM2 1.5TB/s
- 40 MB L2
- PCIe V4 31.5 GB/s
- MGI (NVLINK)

DDR5-8000 ~ 64GB/s

GPU Streaming Processor

(GRID)

Each SM (Grid) Contains

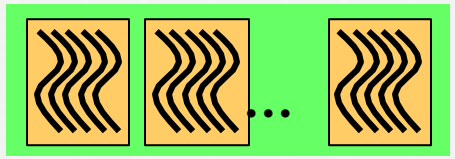
- 4 Processing Blocks
- 1 Warp scheduler/Block
- 16 FP32 CUDA Cores/Block
- 64 INT32 CUDA Cores/Block
- 8 FP64 CUDA Cores/Block
- 1 Tensor Core/Block

Execution Model

Software

Thread

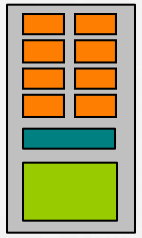
Thread Block



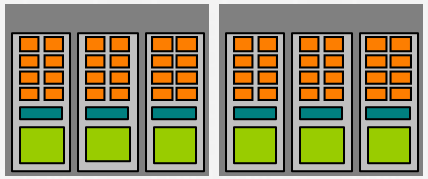
Grid

Hardware

Scalar Processor



Multiprocessor



Device

Threads are executed by scalar processors

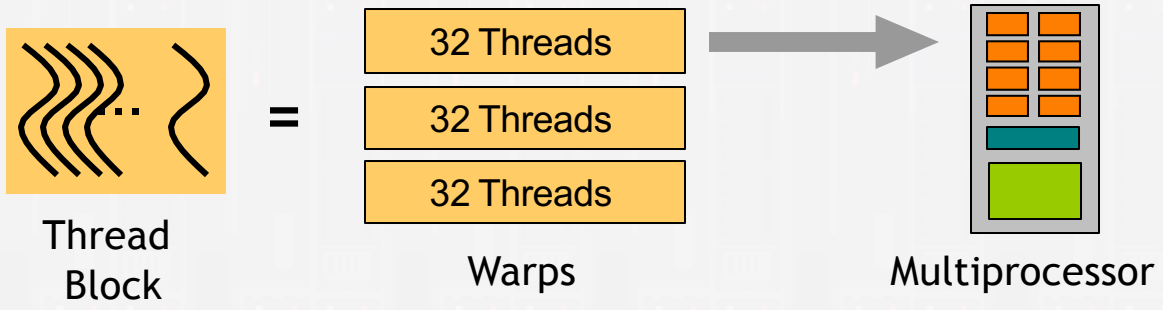
Thread blocks are executed on multiprocessors

Thread blocks do not migrate

Several concurrent thread blocks can reside on one multiprocessor - limited by multiprocessor resources (shared memory and register file)

A kernel is launched as a grid of thread blocks

Warps

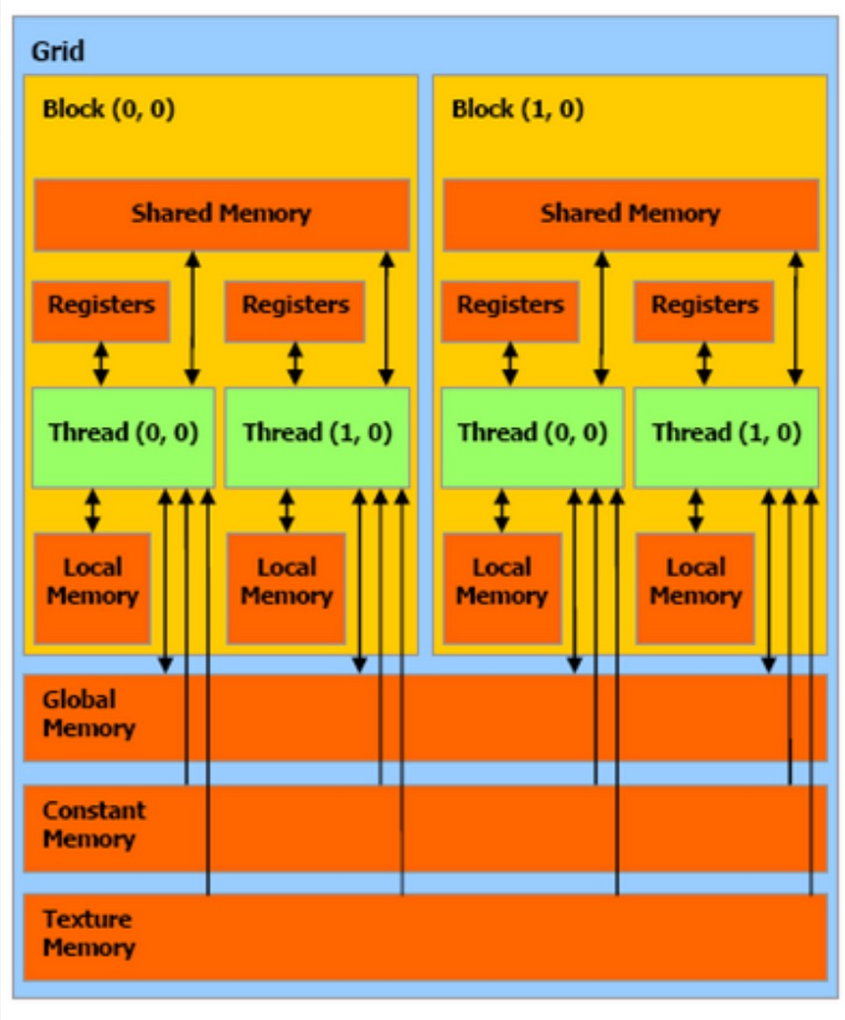


A thread block consists of 32-thread warps

A warp is executed physically in parallel (SIMT) on a multiprocessor

GPU Memory Hierarchy

- Global Memory (off-chip) is accessed by all SMs (Grids)
- Each thread has (on-chip):
  - Registers (fast)
  - Local memory (spillover for registers)
- On-chip shared memory (for threads in a block)
- L1, shared, and texture memory are combined (192 KB).
- L2 is part of the Global Memory (40 MB).



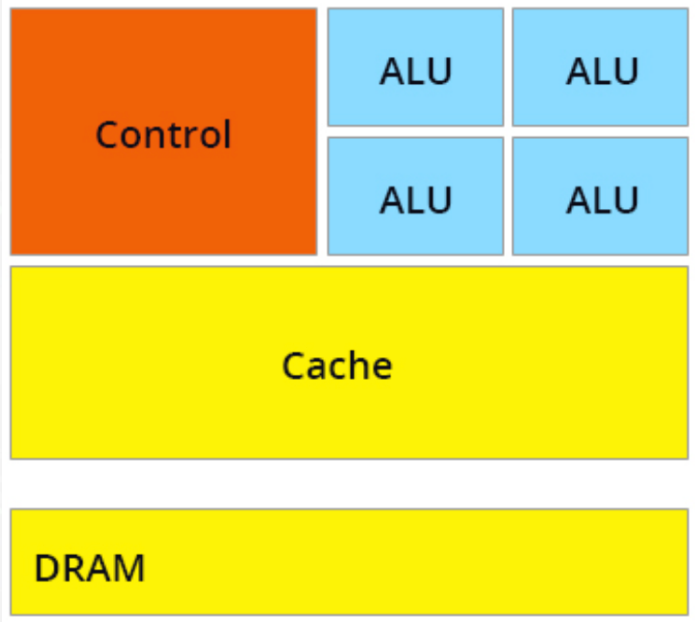
GPU Memory

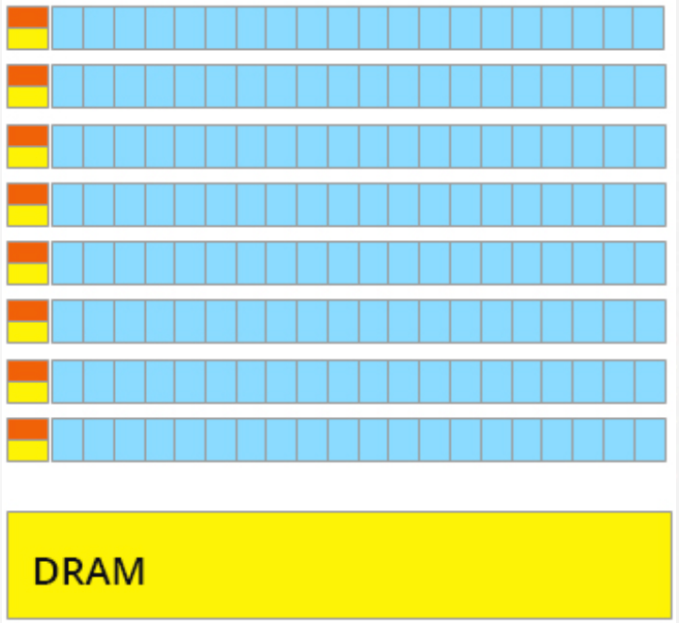
Memory	Location	Cached	Access	Scope
Local	off-chip	No	R/W	thread
Shared	on-chip	N/A	R/W	all threads in a block
Global	off-chip	No	R/W	all threads + host
Constant	off-chip	Yes	RO	all threads + host
Texture	off-chip	Yes	RO	all threads + host

CPU

Vs

GPU





CPU

Low latency

Low compute density

Optimized for serial operations

GPU

High throughput

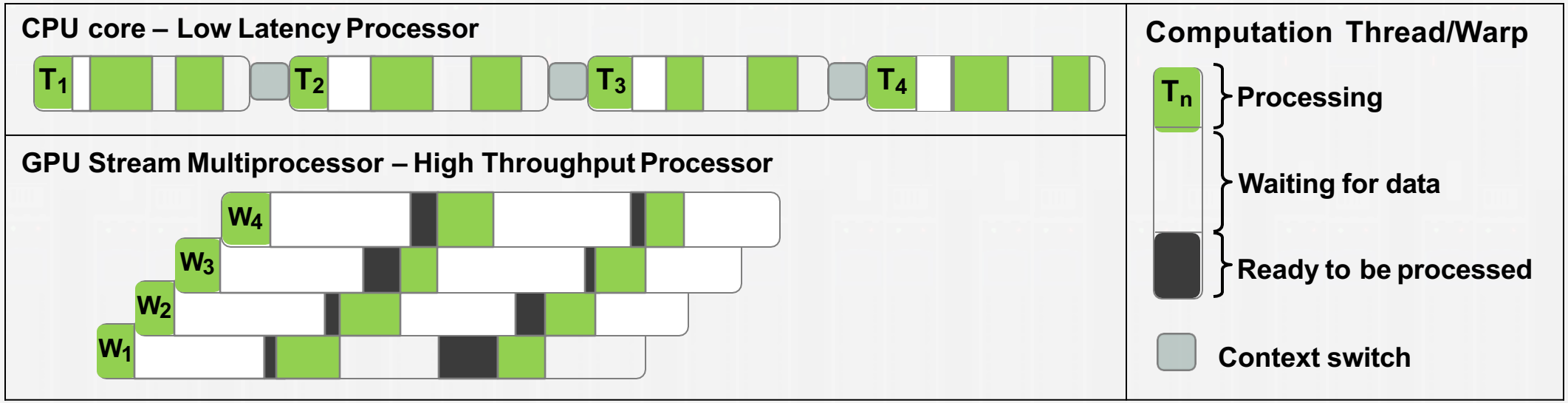
High compute density

Built for parallel operations

Low Latency or High Throughput?

CPU architecture must **minimize latency** within each thread

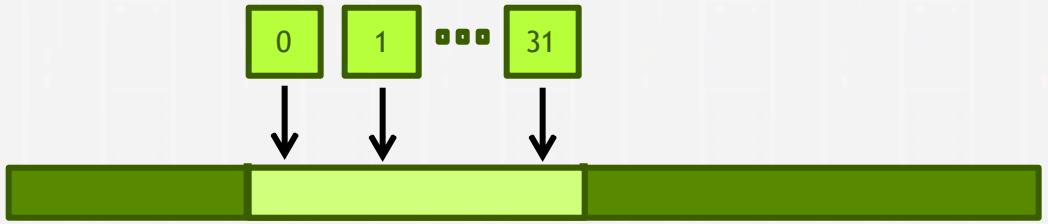
GPU architecture **hides latency** with computation from other thread warps



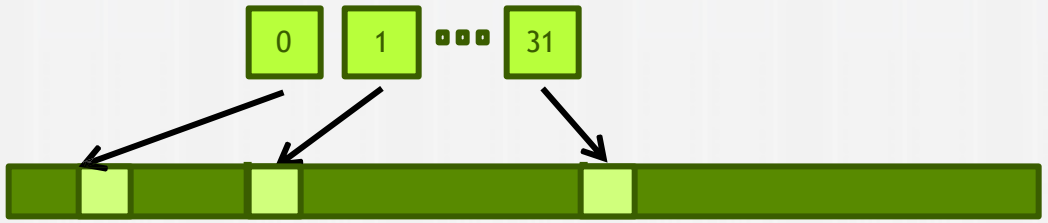
Memory Coalescing

- Global memory access happens in transactions of 32 or 128 bytes
- The hardware will try to reduce to as few transactions as possible

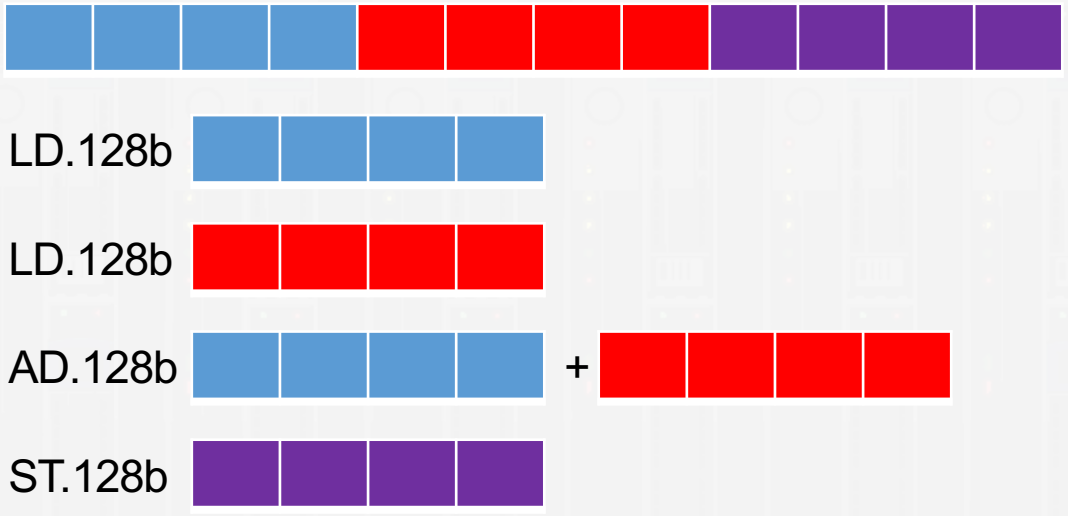
- *Coalesced* access:
  - A group of 32 contiguous threads
  - (“warp”) accessing adjacent words
  - Few transactions and high utilization



- *Uncoalesced* access:
  - A warp of 32 threads accessing scattered words
  - Many transactions and low utilization

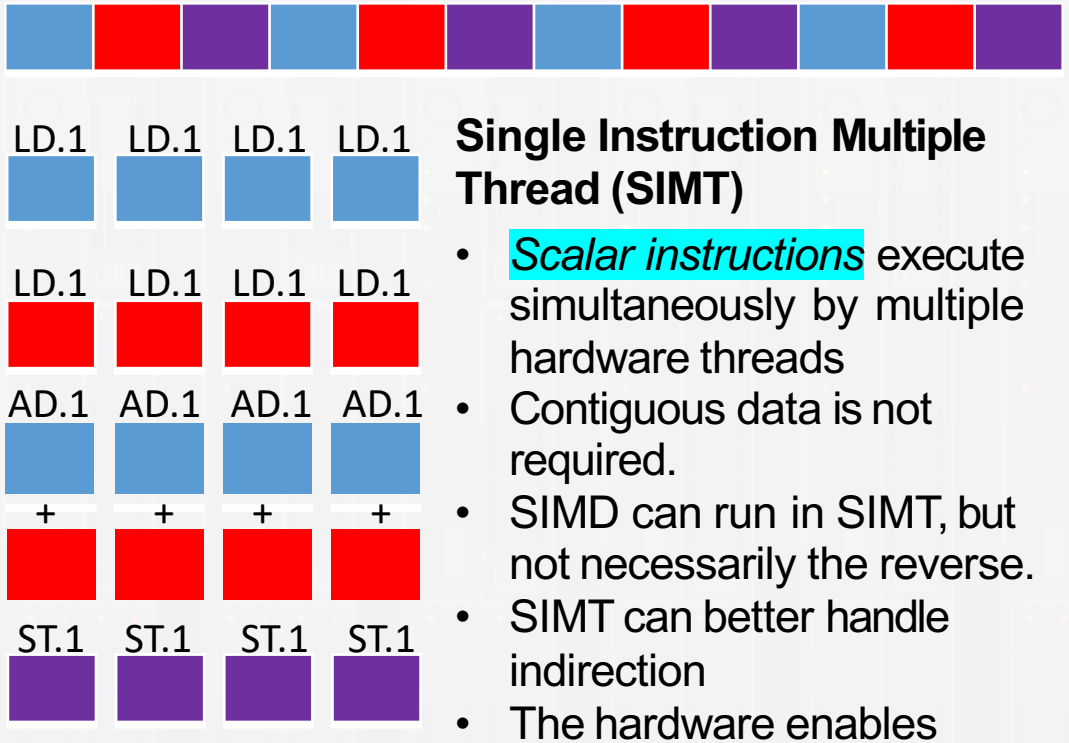


SIMD and SIMT



Single Instruction Multiple Data (SIMD)

- *Vector instructions* perform the same operation on multiple data elements.
- Data must be loaded and stored in contiguous buffers
- Either the programmer or the compiler must generate vector instructions

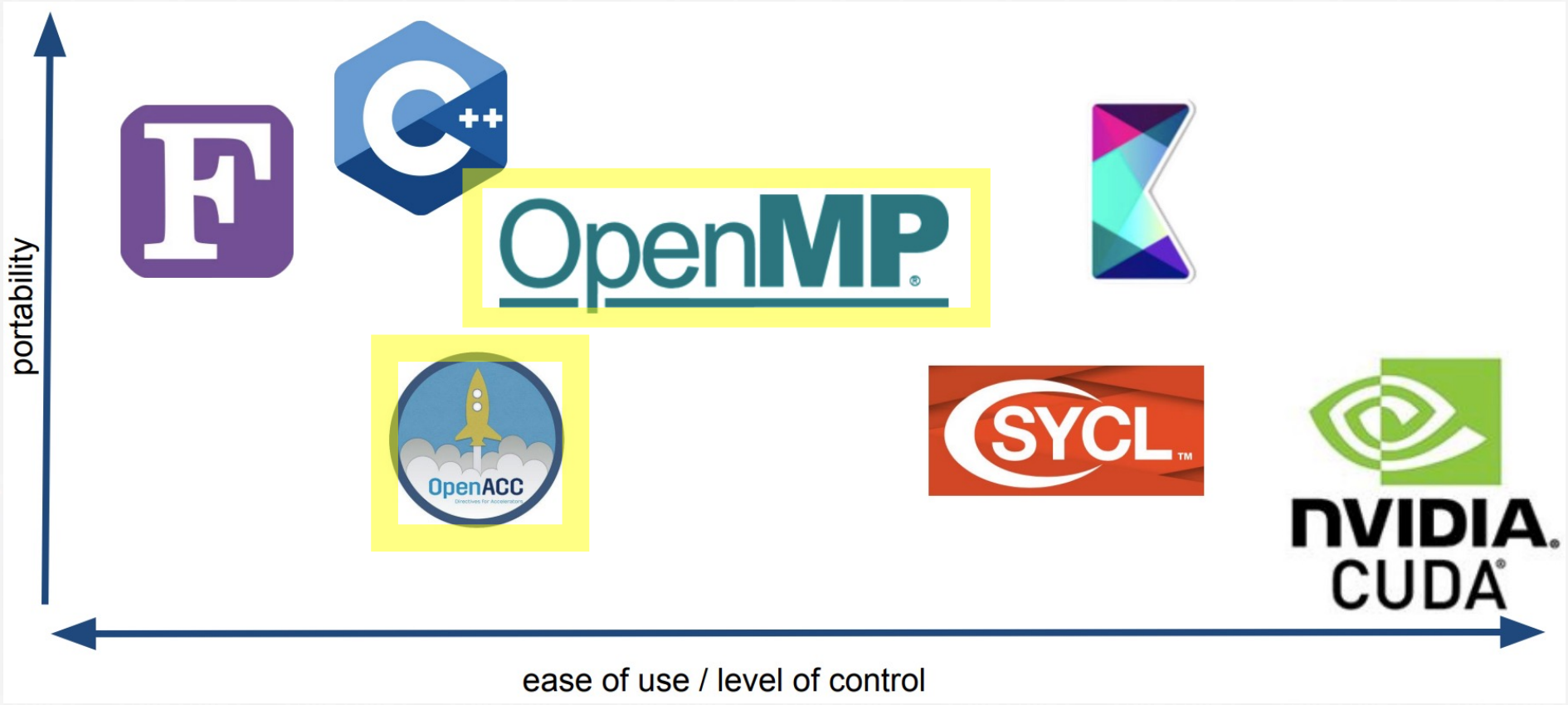


Single Instruction Multiple Thread (SIMT)

- *Scalar instructions* execute simultaneously by multiple hardware threads
- Contiguous data is not required.
- SIMD can run in SIMT, but not necessarily the reverse.
- SIMT can better handle indirection
- The hardware enables The parallel execution of scalar instructions

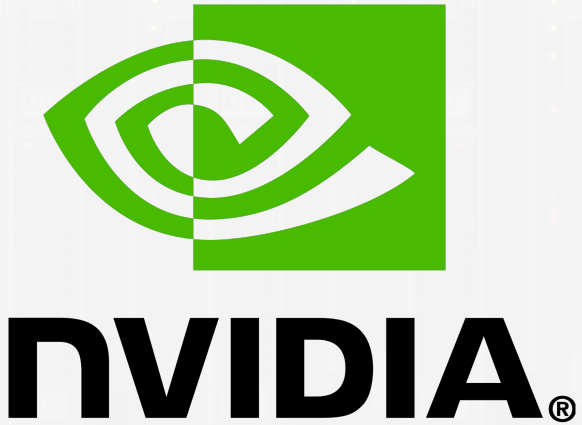
Taken from Jeff Larkin NVIDIA Presentation

Programming Models for GPGPU



CUDA

- Native model for NVIDIA GPUs.
- Reference model for other models.
- Full control
- Maximum performance possible
- Not portable (NVIDIA Only)
- Verbose



kokkos

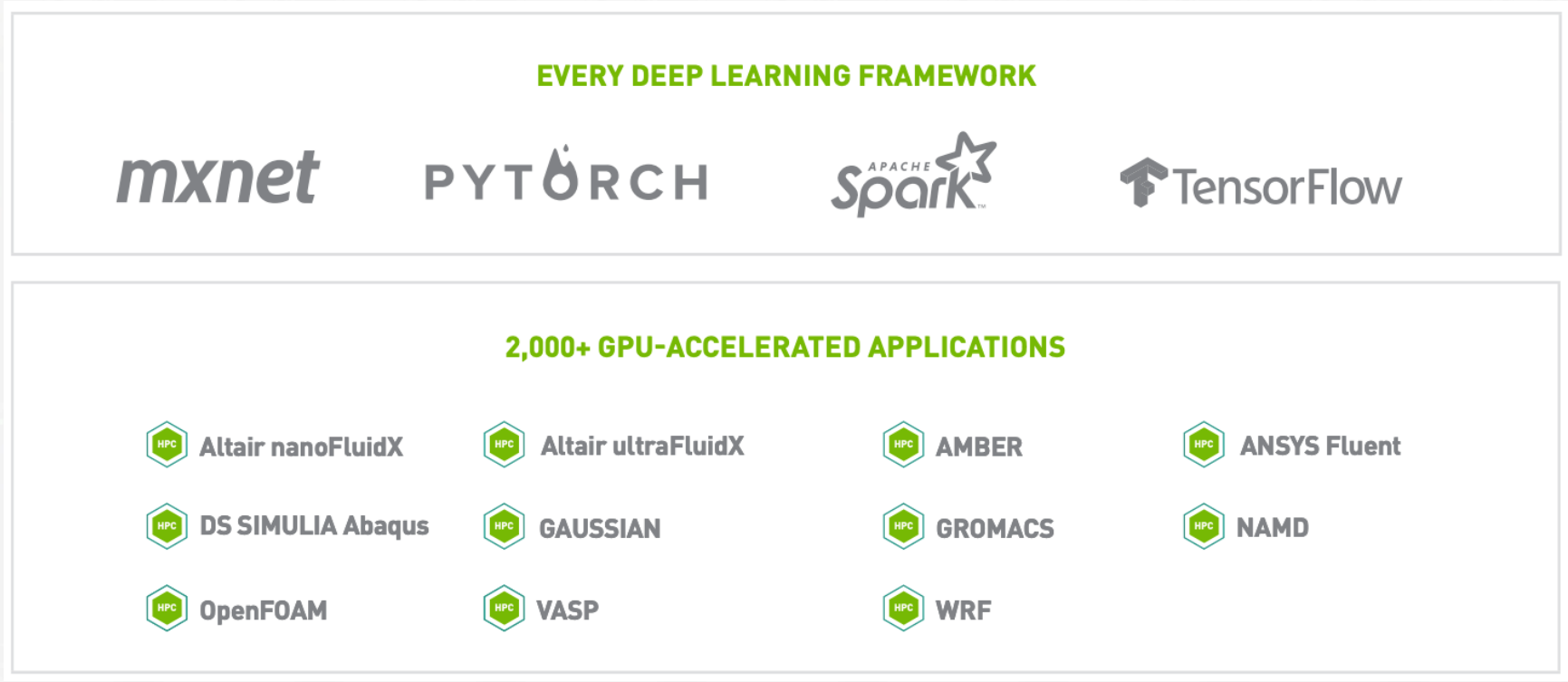
SYCL™

- An “Ecosystem” with programming model, memory abstractions, math, kernels, tools, etc.
- Funded by ECP, DoE and

- A cross-platform abstraction layer for heterogeneous processors (including NVIDIA) using backends.
- Relies on OpenCL, CUDA, ROCm, SPIR-V, etc.

Applications of GPGPU

- Scientific computing and deep learning



Challenges and Considerations

- Data transfer Host – GPU (PCIe Transfer)
- Memory hierarchy optimization
- Load balancing and synchronization
- Portability across GPU architectures