Parallel Computing with GPUs: GPU Architectures

Dr Mozhgan Kabiri Chimeh

http://mkchimeh.staff.shef.ac.uk/teaching/COM4521

The University Of Sheffield.

NVIDIA GPU RESEARCH CENTER

Last week

- Parallelism can add performance to our code
- We must identify parallel regions
- OpenMP can be both data and task parallel
- OpenMP data parallelism is parallel over data elements
  - but threads operate independently
- Critical sections cause serialisation which can slow performance
- Scheduling is required to achieve best performance

This Lecture

- What is a GPU?
- General Purpose Computation on GPUs (and GPU History)
- GPU CUDA Hardware Model
- Accelerated Systems

GPU Refresher

Latency vs. Throughput

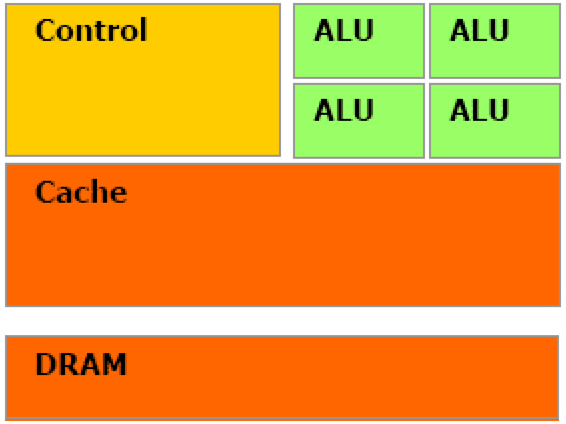
- Latency: The time required to perform some action
  - Measure in units of time
- Throughput: The number of actions executed per unit of time
  - Measured in units of what is produced

- E.g. *An assembly line manufactures GPUs. It takes* ***6 hours*** *to manufacture a GPU but the assembly line can manufacture* ***100 GPUs per day****.*

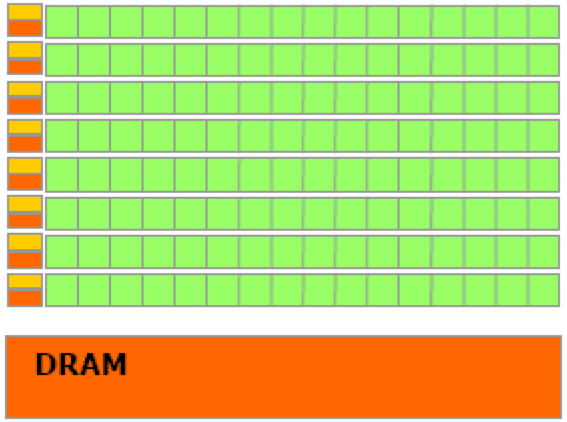
CPU vs GPU

- CPU
  - Latency oriented
  - Optimised for serial code performance
  - Good for single complex tasks
- GPU
  - Throughput oriented
  - Massively parallel architecture
  - Optimised for performing many similar tasks simultaneously (data parallel)

CPU vs GPU



CPU



GPU

- ❑Large Cache
  - ❑Hide long latency memory access
- ❑Powerful Arithmetic Logical Unit (ALU)
  - ❑Low Operation Latency
- ❑Complex Control mechanisms
  - ❑Branch prediction etc.

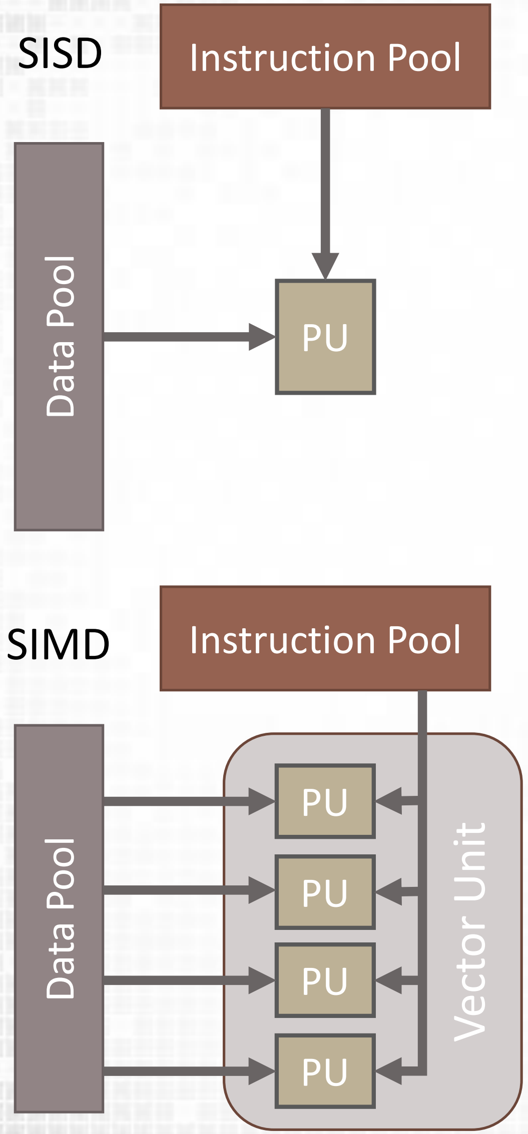
- ❑Small cache
  - ❑But faster memory throughput
- ❑Energy efficient ALUs
  - ❑Long latency but high throughput
- ❑Simple control
  - ❑No branch prediction

Data Parallelism

- Program has many similar threads of execution
  - Each thread performs the same behaviour on different data
  - Good for high throughput
- We can classify an architecture based on instructions and data (Flynn's Taxonomy)
  - Instructions:
    - Single instruction (SI)
    - Multiple Instruction (MI)
    - *Single Program (SP)*
    - *Multiple Program (MP)*

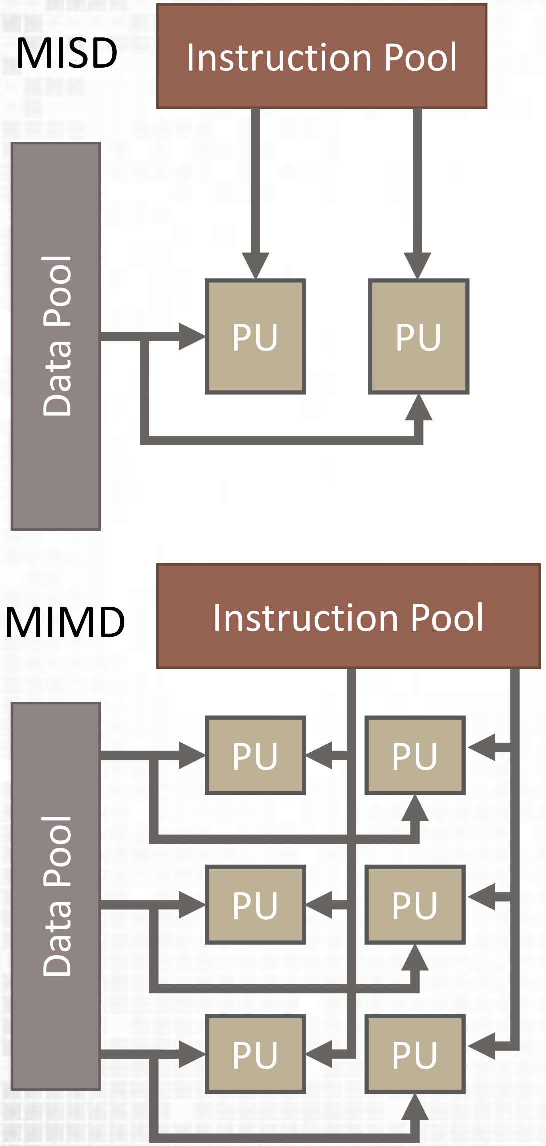
    Not part of the original taxonomy
  - Data:
    - Single Data (SD) – w.r.t. *work item not necessarily single word*
    - Multiple Data (MD)
  - e.g. SIMD = Single Instruction and Multiple Data

SISD and SIMD



- ❑SISD
  - ❑Classic von Neumann architecture
  - ❑PU = Processing Unit
- ❑SIMD
  - ❑Multiple processing elements performing the same operation simultaneously
  - ❑E.g. Early vector super computers
  - ❑Modern CPUs have SIMD instructions
    - ❑But are not SIMD in general

MISD and MIMD



- MISD
  - structure is not commercially implemented
  - E.g. Pipelining architectures

- MIMD
  - Processors as functionally asynchronous and independent
  - Different processors may execute different instructions on different data
  - E.g. Most parallel computers
  - E.g. OpenMP programming model

SPMD and MPMD

- SPMD
  - Multiple autonomous processors simultaneously executing a program on different data
  - Program execution can have an independent path for each data point
  - E.g. Message passing on distributed memory machines.
- MPMD
  - Multiple autonomous processors simultaneously executing at least two independent programs.
  - Typically client & host programming models fit this description.
  - E.g. Sony PlayStation 3 SPU/PPU combination, Some system on chip configurations with CPU and GPUs

The University Of Sheffield. GPU RESEARCH CENTER

Taxonomy of a GPU

- What taxonomy best describes data parallelism with a GPU?
  - SISD?
  - SIMD?
  - MISD?
  - MIMD?
  - SPMD?
  - MPMD?

Taxonomy of a GPU

- What taxonomy best describes data parallelism with a GPU?

- **Obvious Answer**: SIMD

- **Less Obvious answer**: SPMD

- **Slightly confusing answer**: SIMT (Single Instruction Multiple Thread)
  - This is a combination of both it differs from SIMD in that;
    1) Each thread has its own registers
    2) Each thread has multiple addresses
    3) Each thread has multiple flow paths
  - We will explore this in more detail when we look at the hardware!
  - http://yosefk.com/blog/simd-simt-smt-parallelism-in-nvidia-gpus.html

This Lecture

- ❑What is a GPU?
- ❑General Purpose Computation on GPUs (and GPU History)
- ❑GPU CUDA Hardware Model
- ❑Accelerated Systems

GPU Early History

- Hardware has evolved from the demand for increased quality of 3D computer graphics
- Initially specialised processors for each part of the graphics pipeline
  - Vertices (points of triangles) and Fragments (potential pixels) can be manipulated in parallel
- The stages of the graphics pipeline became programmable in early 2000's
  - NVIDIA GeForce 3 and ATI Radeon 9700
  - DirectX 9.0 required programmable pixel and vertex shaders

The University Of Sheffield.

NVIDIA GPU RESEARCH CENTER

GPGPU

- General Purpose computation on Graphics Hardware
  - First termed by Mark Harris (NVIDIA) in 2002
  - Recognised the use of GPUs for non graphics applications
- Requires mapping a problem into graphics concepts
  - Data into textures (images)
  - Computation into shaders
- Later unified processors were used rather than fixed stages
  - 2006: GeForce 8 series

Unified Processors and CUDA

- ❑Compute Unified Device Architecture (CUDA)
  - ❑First released in 2006/7
- ❑Targeted new bread of unified “streaming multiprocessors”
- ❑C like programming for GPUs
  - ❑No computer graphics: General purpose programming model
  - ❑Revolutionised GPU programming for general purpose use

Other GPU Programming Techniques

- GPU Accelerated Libraries and Applications (MATLAB, Ansys, etc)
  - GPU mostly abstracted from end user
  - *Pros: ?*
  - *Cons: ?*
- GPU Accelerated Directives (OpenACC)
  - Helps compiler auto generate code for the GPU
  - Very similar to OpenMP
  - *Pros: ?*
  - *Cons: ?*
- OpenCL
  - Inspired by CUDA but targeted at more general data parallel architectures
  - *Pros: ?*
  - *Cons: ?*

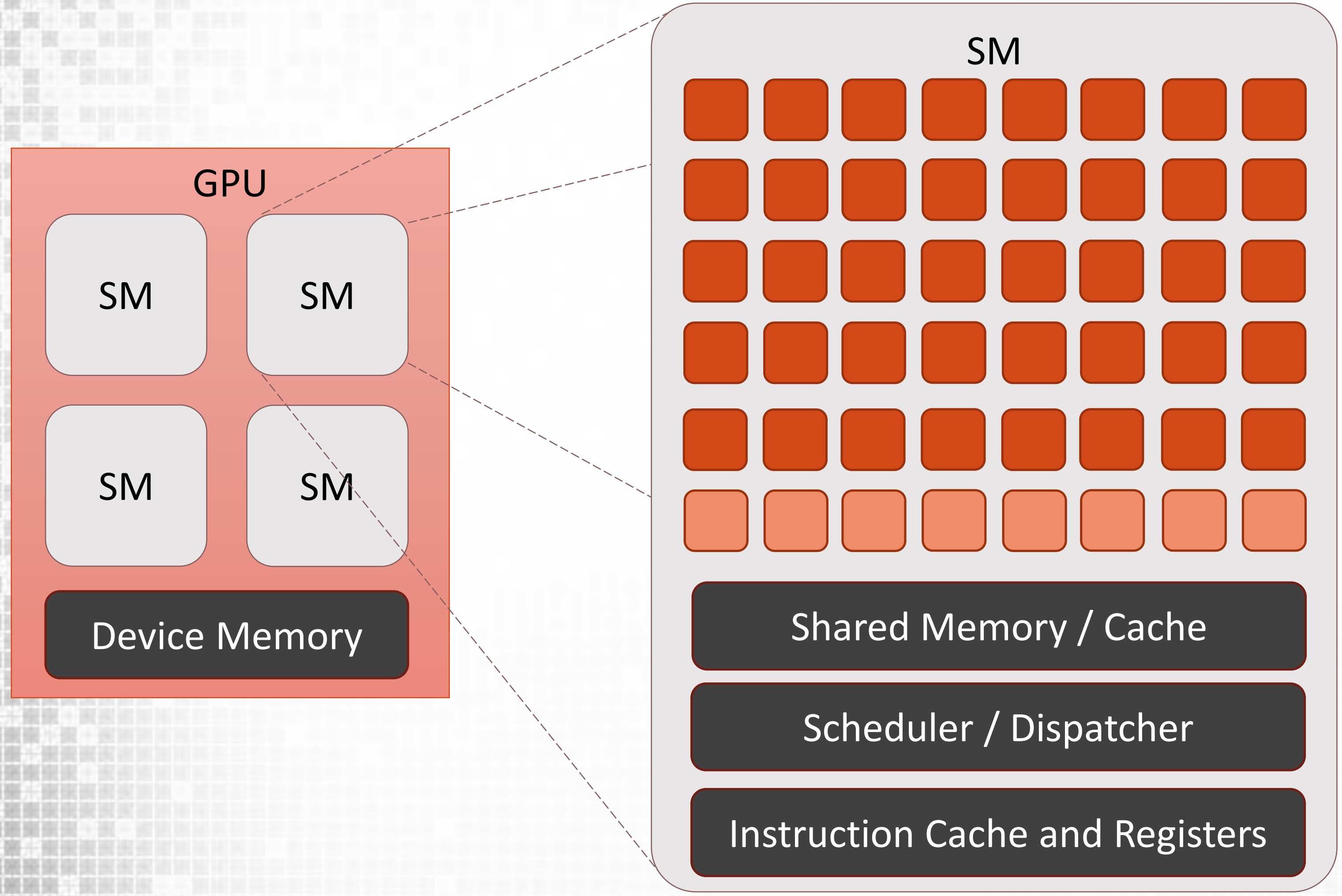
Other GPU Programming Techniques

- GPU Accelerated Libraries and Applications (MATLAB, Ansys, etc)
  - GPU mostly abstracted from end user
  - *Pros: Easy to learn and use*
  - *Cons: … difficult to master (*High level of abstraction reduces ability to perform bespoke optimisation)
- GPU Accelerated Directives (OpenACC)
  - Helps compiler auto generate code for the GPU
  - Very similar to OpenMP
  - *Pros: Performance portability, limited understanding of hardware required*
  - *Cons: Limited fine grained control of optimisation*
- OpenCL
  - Inspired by CUDA but targeted at more general data parallel architectures
  - *Pros: Cross platform*
  - *Cons: Limited access to cutting edge NVIDIA specific functionality, limited support*

This Lecture

- ❑What is a GPU?
- ❑General Purpose Computation on GPUs (and GPU History)
- ❑GPU CUDA Hardware Model
- ❑Accelerated Systems

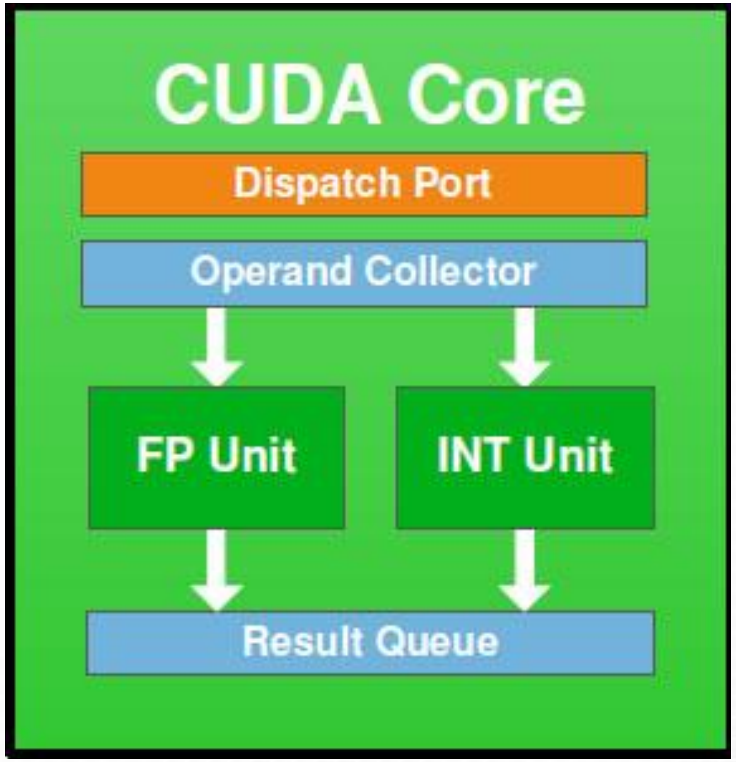
Hardware Model



- NVIDIA GPUs have a 2-level hierarchy
  - Each Streaming Multiprocessor (SMP) has multiple vector “CUDA” cores
  - The number of SMs varies across different hardware implementations
  - The design of SMPs varies between GPU families
  - The number of cores per SMP varies between GPU families

NVIDIA CUDA Core

- CUDA Core
  - Vector processing unit
  - Stream processor
  - Works on a single operation



NVIDIA GPU Range

- GeForce
  - Consumer range
  - Gaming oriented for mass market
- Quadro Range
  - Workstation and professional graphics
- Tesla
  - Number crunching boxes
  - Much better support for double precision
  - Faster memory bandwidth
  - Better Interconnects

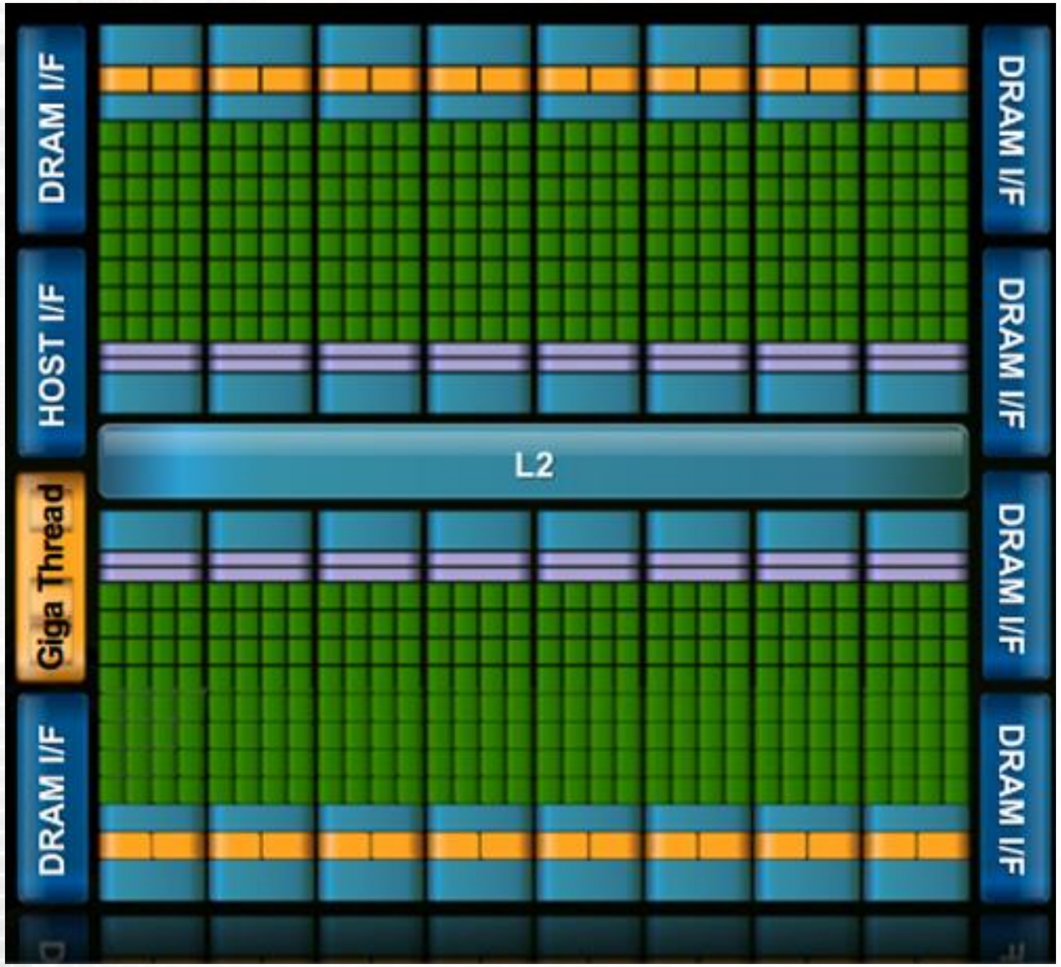
Tesla Range Specifications

	"Kepler" K20	"Kepler" K40	"Maxwell" M40	Pascal P100	Volta V100
CUDA cores	2496	2880	3072	3584	5120
Chip Variant	GK110	GK110B	GM200	GP100	GV100
Cores per SM	192	192	128	64	64
Single Precision Performance	3.52 Tflops	4.29 Tflops	7.0 Tflops	9.5TFlops	**15TFFlops**
Double Precision Performance	1.17 TFlops	1.43 Tflops	*0.21 Tflops*	4.7 Tflops	**7.5Tflops**
Memory Bandwidth	208 GB/s	288 GB/s	288GB/s	720GB/s	**900GB/s**
Memory	5 GB	12 GB	12GB	12/16GB	16GB

The University Of Sheffield.

NVIDIA GPU RESEARCH CENTER

Fermi Family of Tesla GPUs



- ❑ Chip partitioned into Streaming Multiprocessors (SMPs)
- ❑ 32 vector cores per SMP
- ❑ Not cache coherent. No communication possible across SMPs.

Kepler Family of Tesla GPUs

- ❑Streaming Multiprocessor Extreme (SMX)
- ❑Huge increase in the number of cores per SMX
  - ❑Smaller 28nm processes
- ❑Increased L2 Cache
- ❑Cache coherency at L2 not at L1

Maxwell Family Tesla GPUs

- ❑Streaming Multiprocessor Module (SMM)
- ❑SMM Divided into 4 quadrants (GPC)
  - ❑Each has own instruction buffer, registers and scheduler for each of the 32 vector cores
- ❑SMM has 90% performance of SMX at 2x energy efficiency
  - ❑128 cores vs. 192 in Kepler
  - ❑BUT small die space = more SMMs
- ❑8x the L2 cache of Kepler (2MB)

Pascal P100 GPU

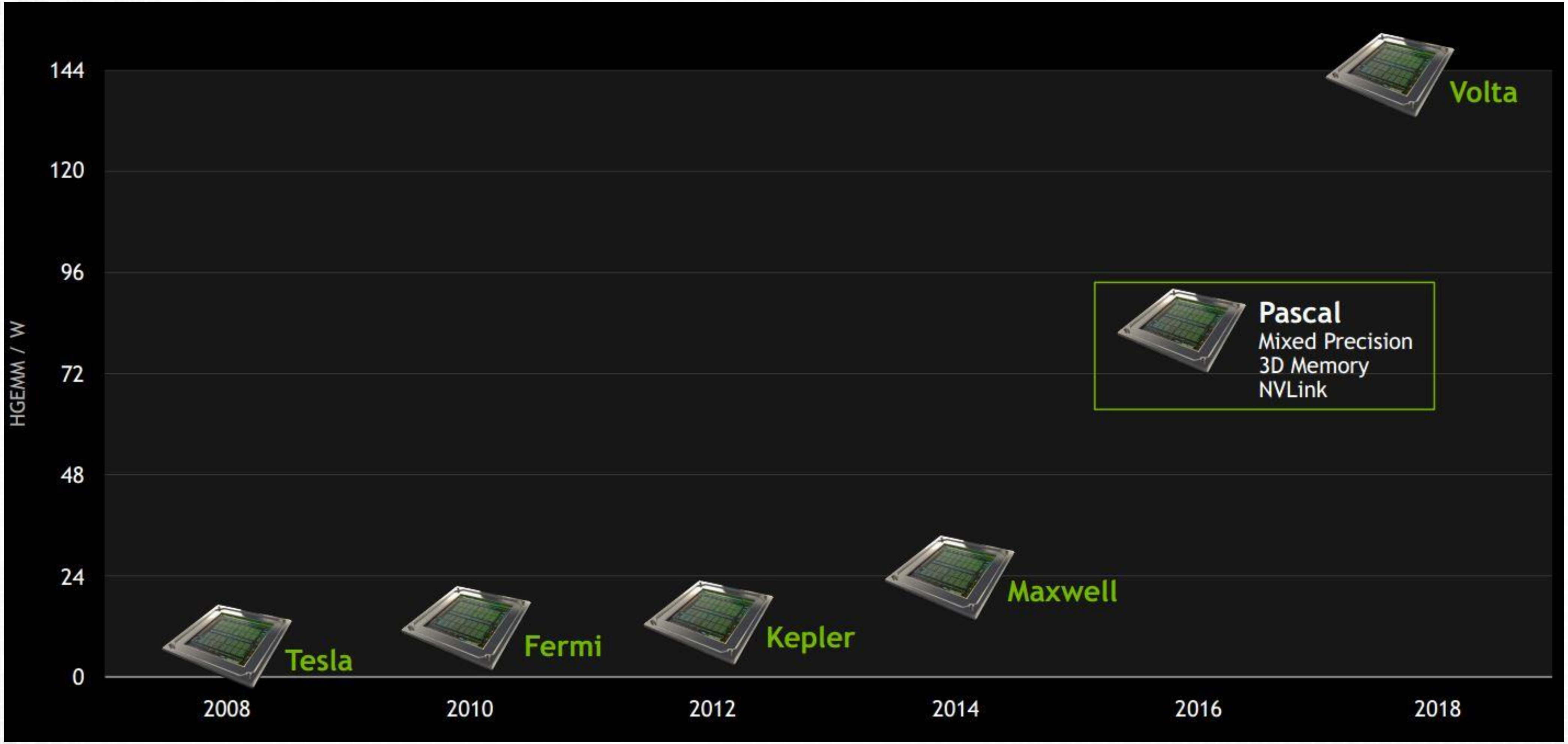
- ❑Many more SMPs
- ❑More GPCs
- ❑Each CUDA core is more efficient
  - ❑More registers available
- ❑Same die size as Maxwell
- ❑Memory bandwidth improved drastically
  - ❑NVLink

Warp Scheduling

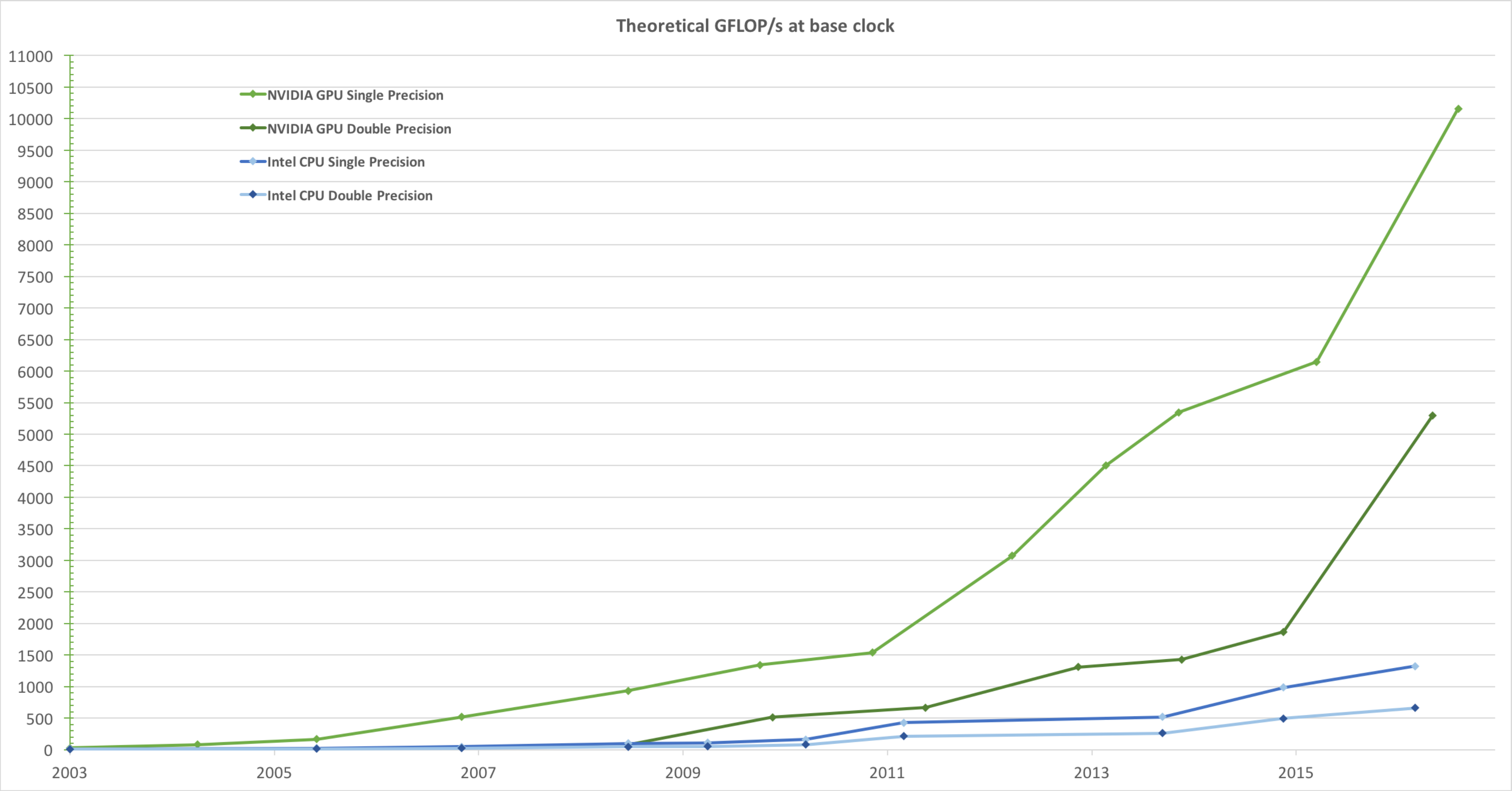
- GPU Threads are always executed in groups called warps (32 threads)
  - **Warps are transparent to users**

- SMPs have zero overhead warp scheduling
  - Warps with instructions ready to execute are eligible for scheduling
  - Eligible warps are selected for execution on priority (context switching)
  - All threads execute the same instruction (SIMD) when executed on the vector processors (CUDA cores)
- The specific way in which warps are scheduled varies across families
  - Fermi, Kepler and Maxwell have different numbers of warp schedulers and dispatchers

NVIDIA Roadmap

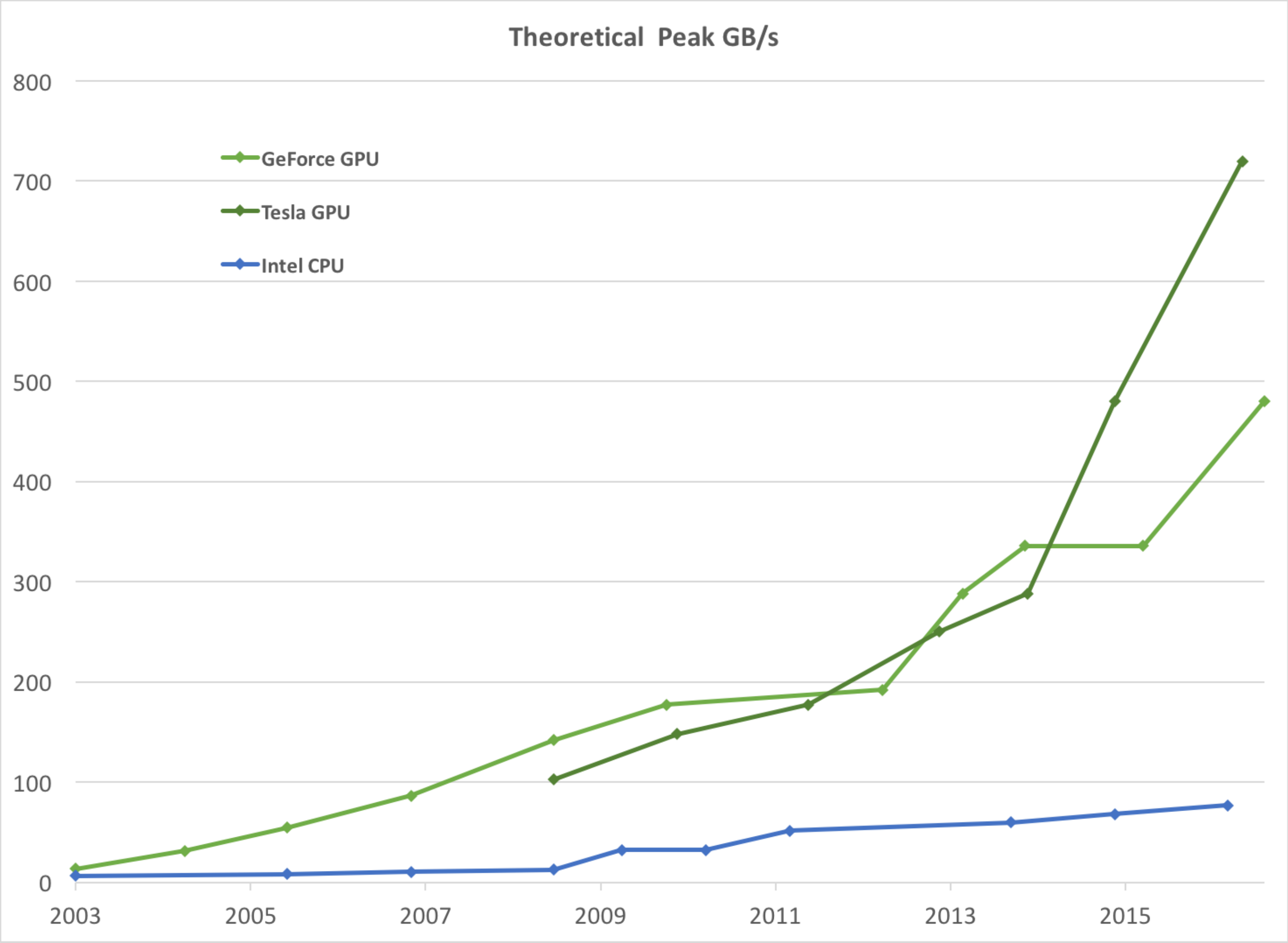


Performance Characteristics



Source: NVIDIA Programming Guide (http://docs.nvidia.com/cuda/cuda-c-programming-guide)

Performance Characteristics



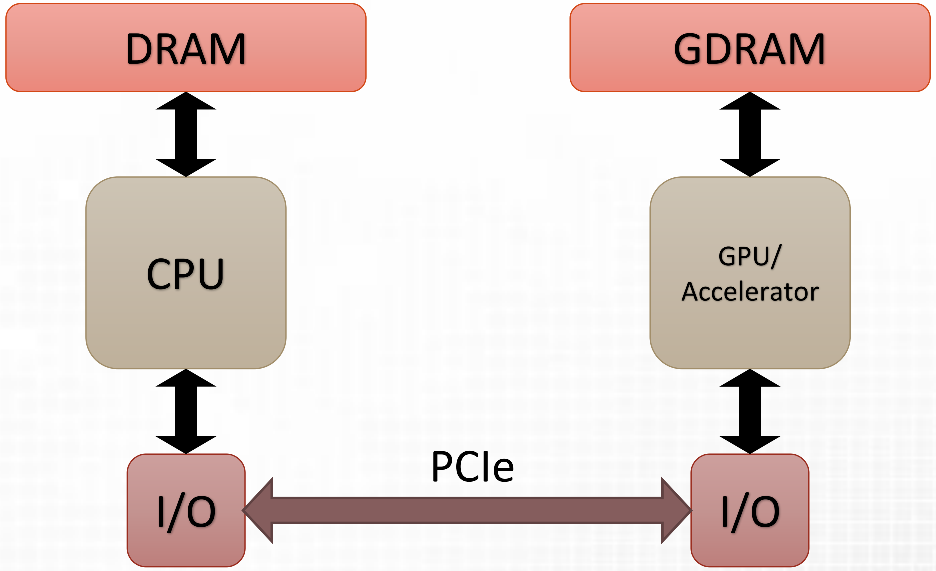
Source: NVIDIA Programming Guide (http://docs.nvidia.com/cuda/cuda-c-programming-guide)

This Lecture

- ❑What is a GPU?
- ❑General Purpose Computation on GPUs (and GPU History)
- ❑GPU CUDA Hardware Model
- ❑Accelerated Systems

Accelerated Systems

- CPUs and Accelerators are used together
  - GPUs cannot be used instead of CPUs
  - GPUs perform compute heavy parts
- Communication is via PCIe bus
  - PCIe 3.0: up to 8 GB per second throughput
  - NVLINK: 5-12x faster than PCIe 3.0

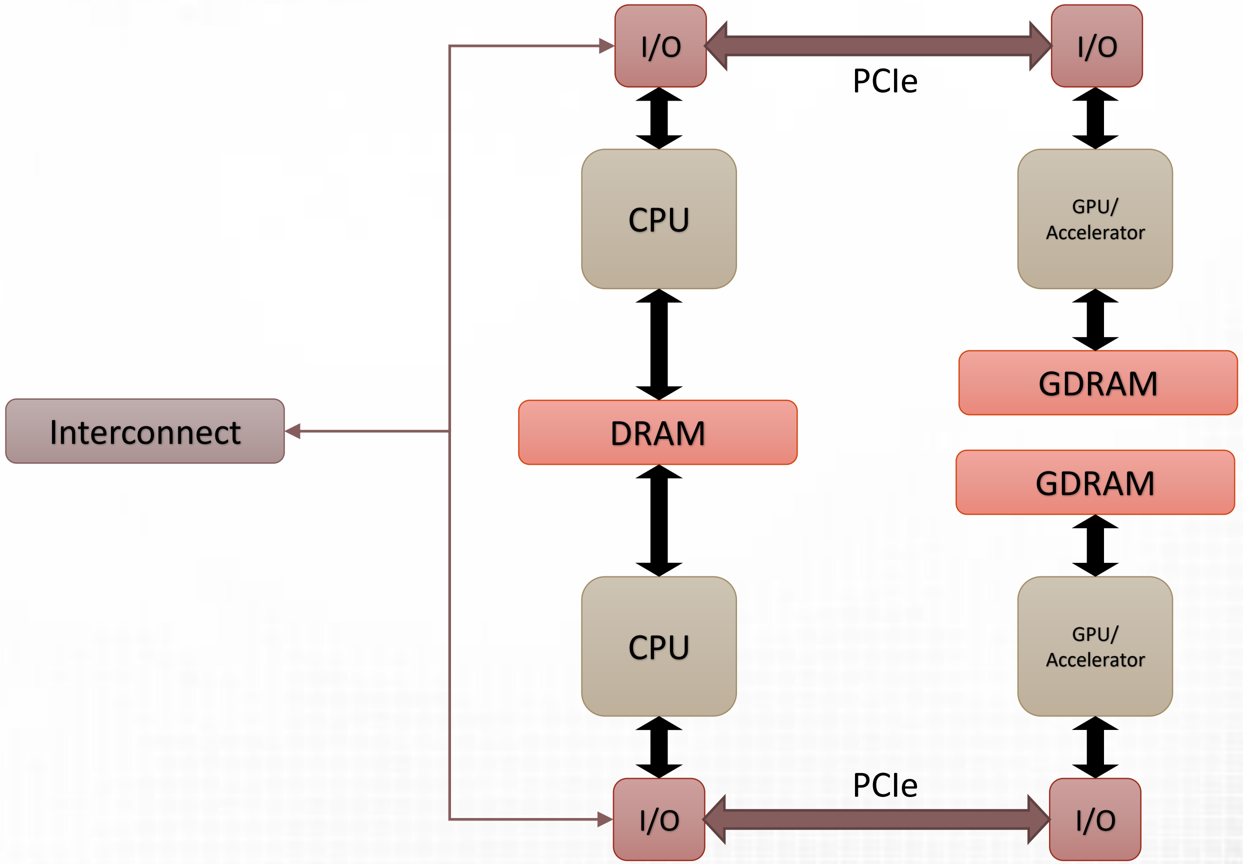


Simple Accelerated Workstation

- Insert your accelerator into PCI-e
- Make sure that
  - There is enough space
  - Your power supply unit (PSU)is up to the job
  - You install the latest GPU drivers

Larger Accelerated Systems

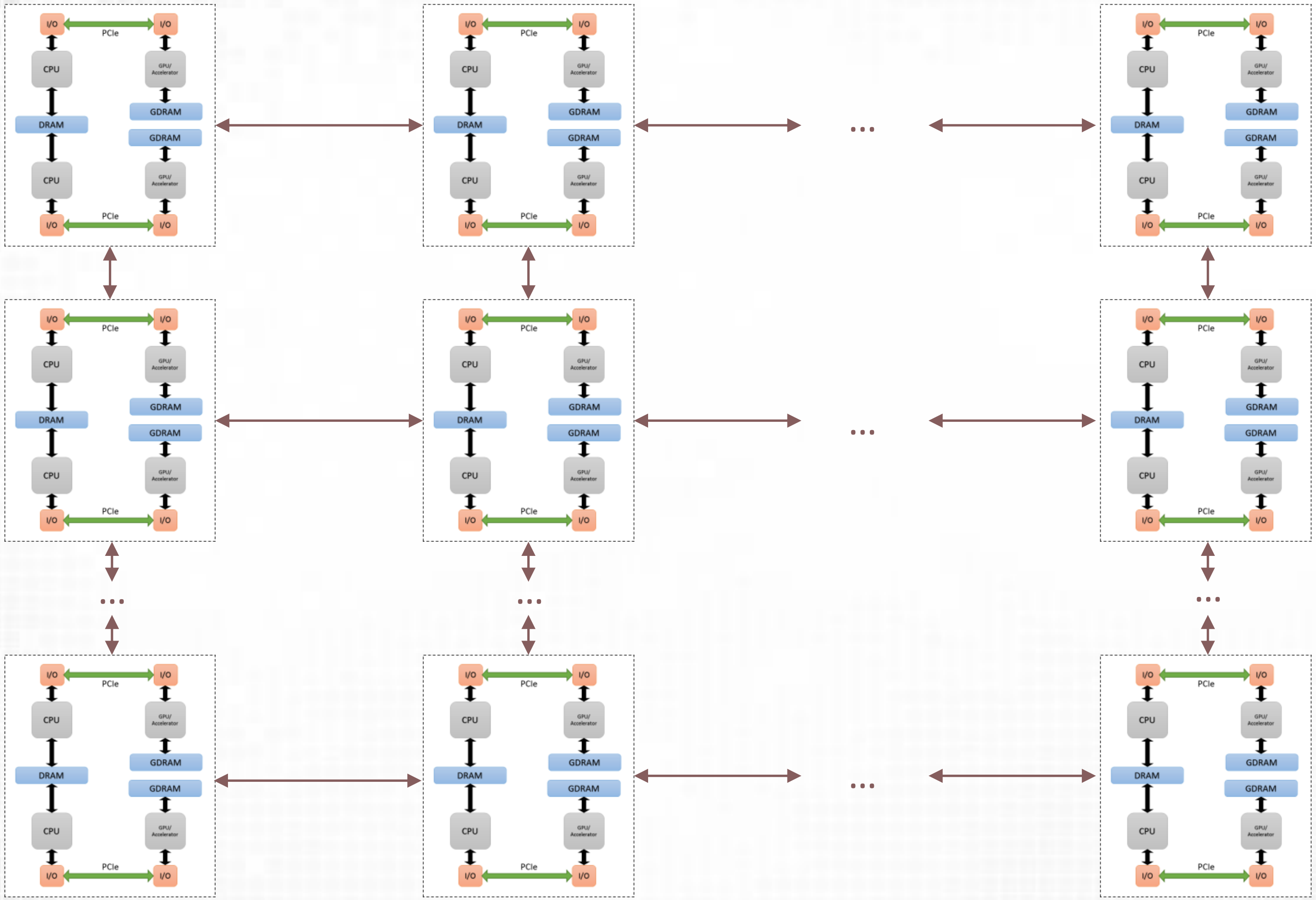
- ❑Can have multiple CPUs and Accelerators within each “Shared Memory Node”
  - ❑CPUs share memory but accelerators do not!



GPU Workstation Server

- ❑Multiple Servers can be connected via interconnect
- ❑Several vendors offer GPU servers
- ❑For example 2 multi core CPUs + 4 GPUS
- ❑Make sure your case and power supply are upto the job!

Accelerated Supercomputers

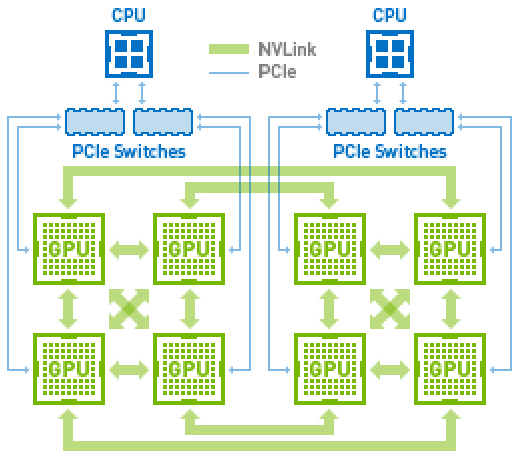


DGX-1 (Volta V100)

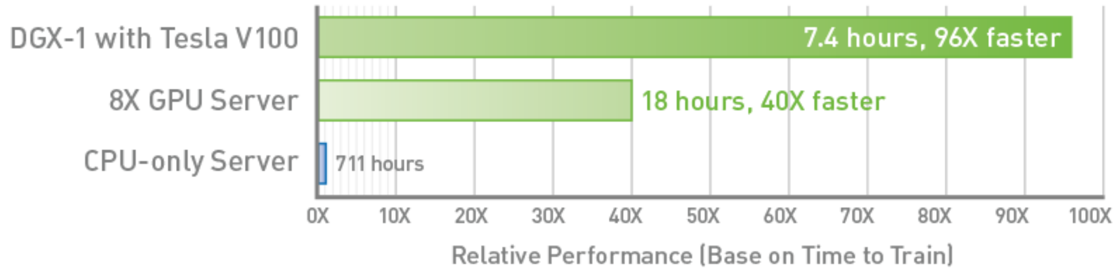
SYSTEM SPECIFICATIONS

GPUs	8X Tesla V100	8X Tesla P100
TFLOPS (GPU FP16)	960	170
GPU Memory	128 GB total system	
CPU	Dual 20-Core Intel Xeon E5-2698 v4 2.2 GHz	
NVIDIA CUDA® Cores	40,960	28,672
NVIDIA Tensor Cores (on V100 based systems)	5,120	N/A
Maximum Power Requirements	3,200 W	
System Memory	512 GB 2,133 MHz DDR4 LRDIMM	
Storage	4X 1.92 TB SSD RAID 0	
Network	Dual 10 GbE, 4 IB EDR	
Software	Ubuntu Linux Host OS See Software Stack for Details	
System Weight	134 lbs	
System Dimensions	866 D x 444 W x 131 H (mm)	
Packing Dimensions	1,180 D x 730 W x 284 H (mm)	
Operating Temperature Range	10–35 °C	

NVIDIA® NVLink™ Hybrid Cube Mesh



NVIDIA DGX-1 Delivers 96X Faster Training



Workload: ResNet50, 90 epochs to solution | CPU Server: Dual Xeon E5-2699 v4, 2.6GHz

Capabilities of Machines Available to you

- Diamond High Spec Lab (for lab sessions)
  - Quadro K5200 GPUs
    - Kepler Architecture
    - 2.9 Tflops Single Precision
- VAR Lab
  - Same machines as High Spec Lab (no managed desktop)
  - Must be booked to access (link)
- ShARC Facility
  - Kepler Tesla K80 GPUs (general pool)
  - Pascal Tesla P100 GPUs in DGX-1 (DCS only)
  - Lab in week 8

CUDA 10, NSIGHT Visual Profiler available in all locations

Summary

- GPUs are better suited to parallel tasks than CPUs
- Accelerators are typically not used alone, but work in tandem with CPUs
- GPU hardware is constantly evolving
- GPU accelerated systems scale from simple workstations to large-scale supercomputers
- CUDA is a language for general purpose GPU (NVIDIA only) programming