GPU Architectures

Reusing this material

|epcc|

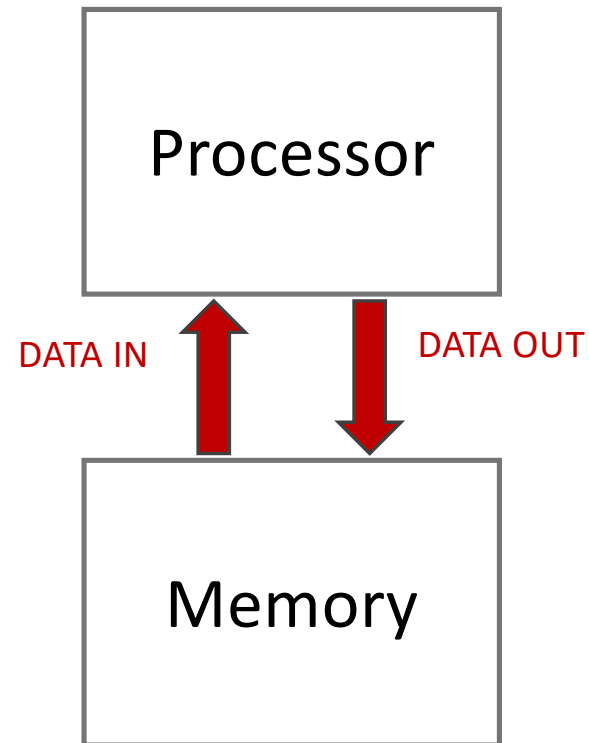# Partners

A Simple Model of Performance

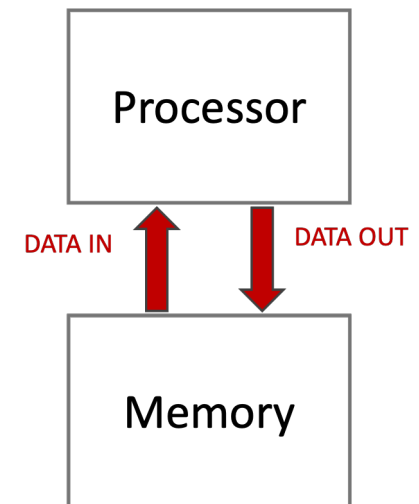# A Simple Model of Performance

A number of factors affect the performance of the model:

- **Clock speed:** the rate of issue of instructions by the processor

- **Memory latency:** time taken to retrieve a data item from memory

- **Memory bandwidth:** amount of data transferred in unit time

- **Parallelism:** can I replicate the basic processing unit?

Processor

DATA IN   DATA OUT

Memory

CPU Clock Speed

- Clock speed has plateaued since the early 2000s.
- This is primarily due to power limits being reached.
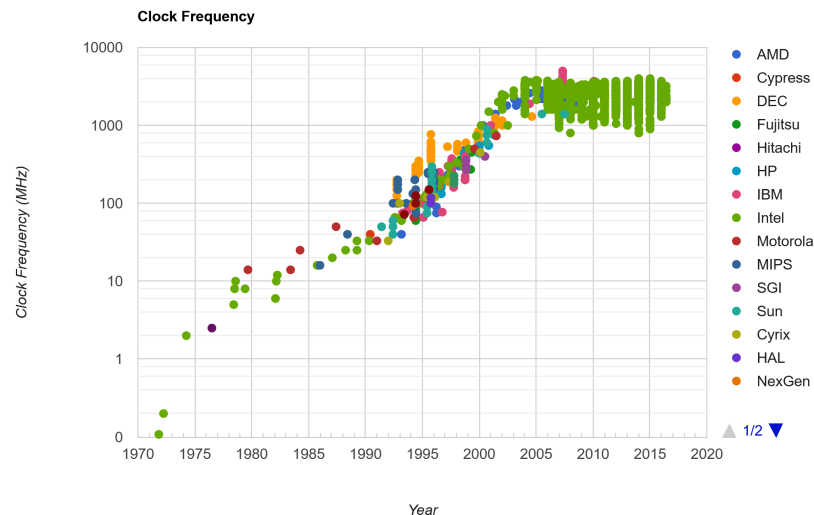
Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, and Mark Horowitz. 2012. CPU DB: recording microprocessor history. Commun. ACM 55, 4 (April 2012), 55–63. https://doi.org/10.1145/2133806.2133822

CPU Memory

Latency

- Retrieving data from RAM can take O(100-1000) clock cycles
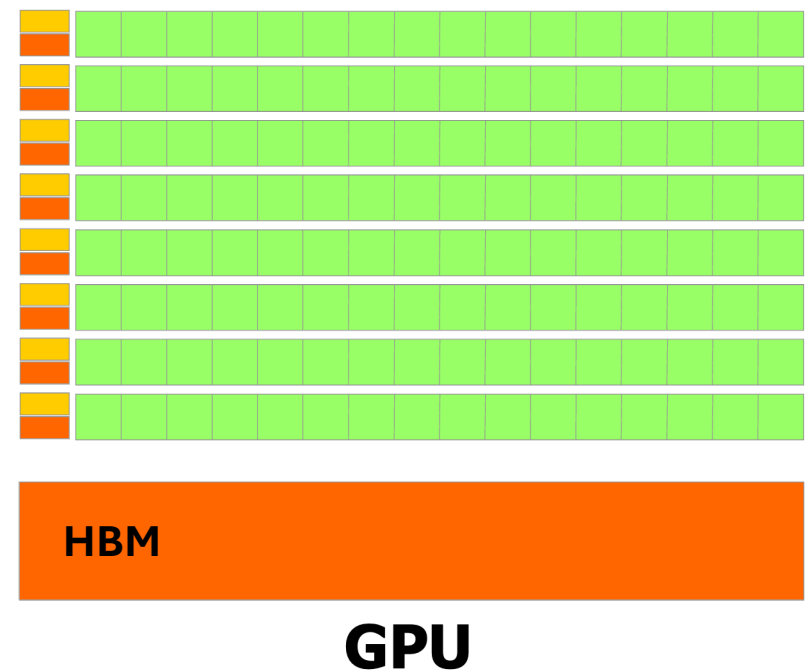- CPUs reduce the impact of this using cache hierarchy

Bandwidth

- Size O(100) GB, bandwidth O(100) GB/second
- Often bottleneck

CPU Parallelism

- Multi-core systems
- Applications must be parallel (MPI, OpenMP, threading etc)

CPU vs GPU Hardware

- Much more area dedicated to compute

Control	ALU	ALU
	ALU	ALU

Cache

DRAM

CPU

HBM

GPU
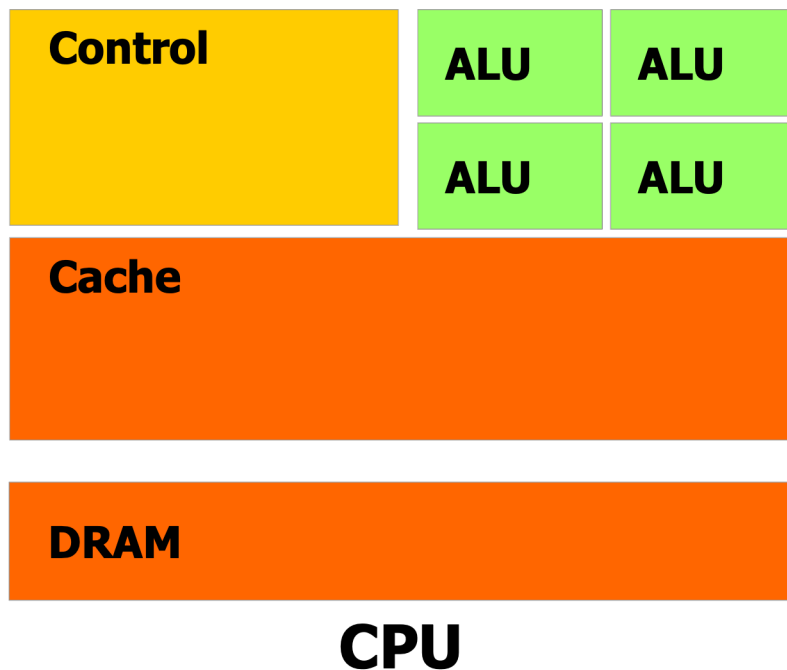
Modified from NVIDIA, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons

CPU vs GPU Hardware

- CPU computational units are "bigger" but smaller in number
- CPU cores are faster and "smarter" than a GPU core
  - E.g. out-of-order executions, branch prediction, speculative execution
- GPUs specialise in floating point operations:
  - E.g. multiply-add, fused multiply-add, tensor core, ray tracing core
- Performance of a GPU does not come from the processing capabilities of individual cores, rather the massive parallelism
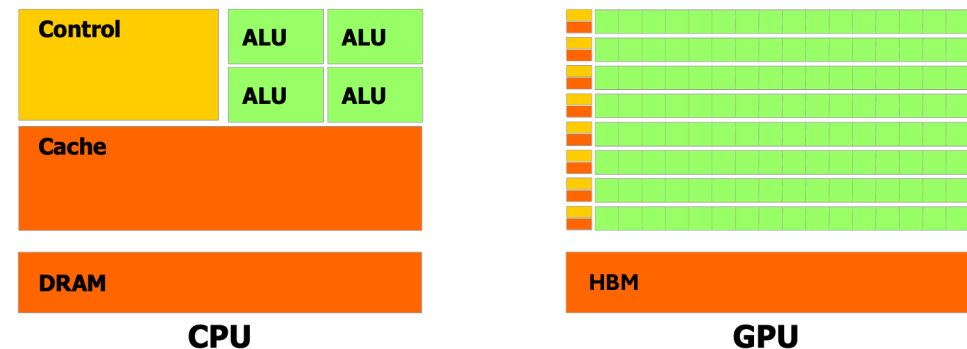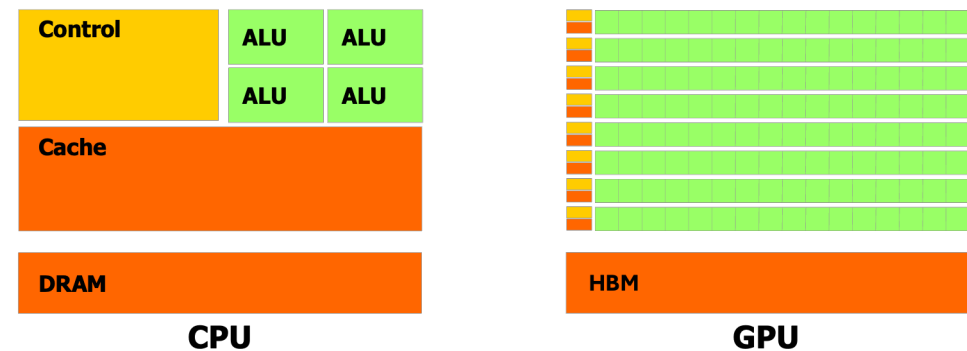

Modified from NVIDIA, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons

CPU vs GPU Hardware

- GPUs have high bandwidth, but high latency .
- Hide latency with multiple threads and fast switching contexts
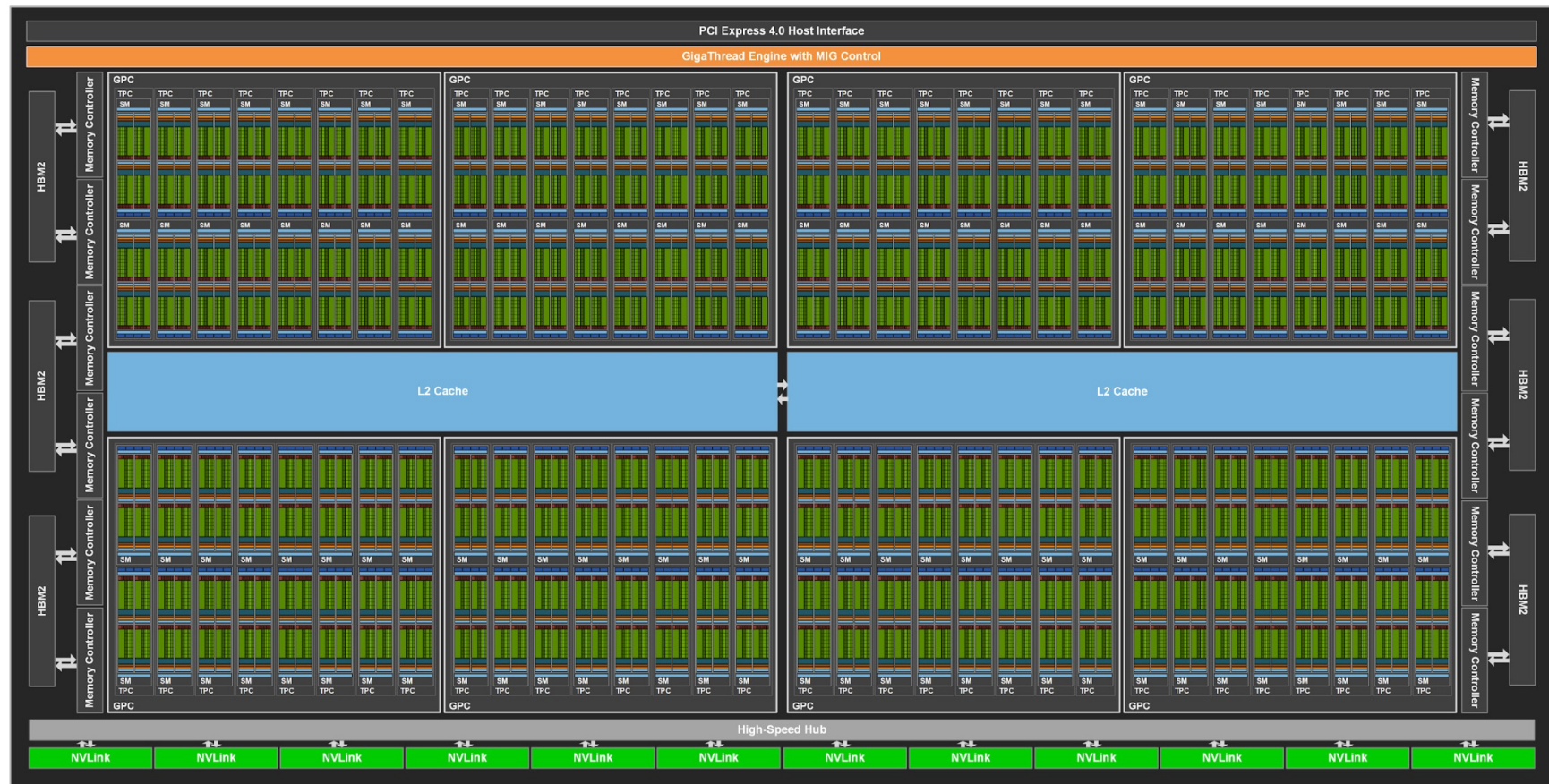

Modified from NVIDIA, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons
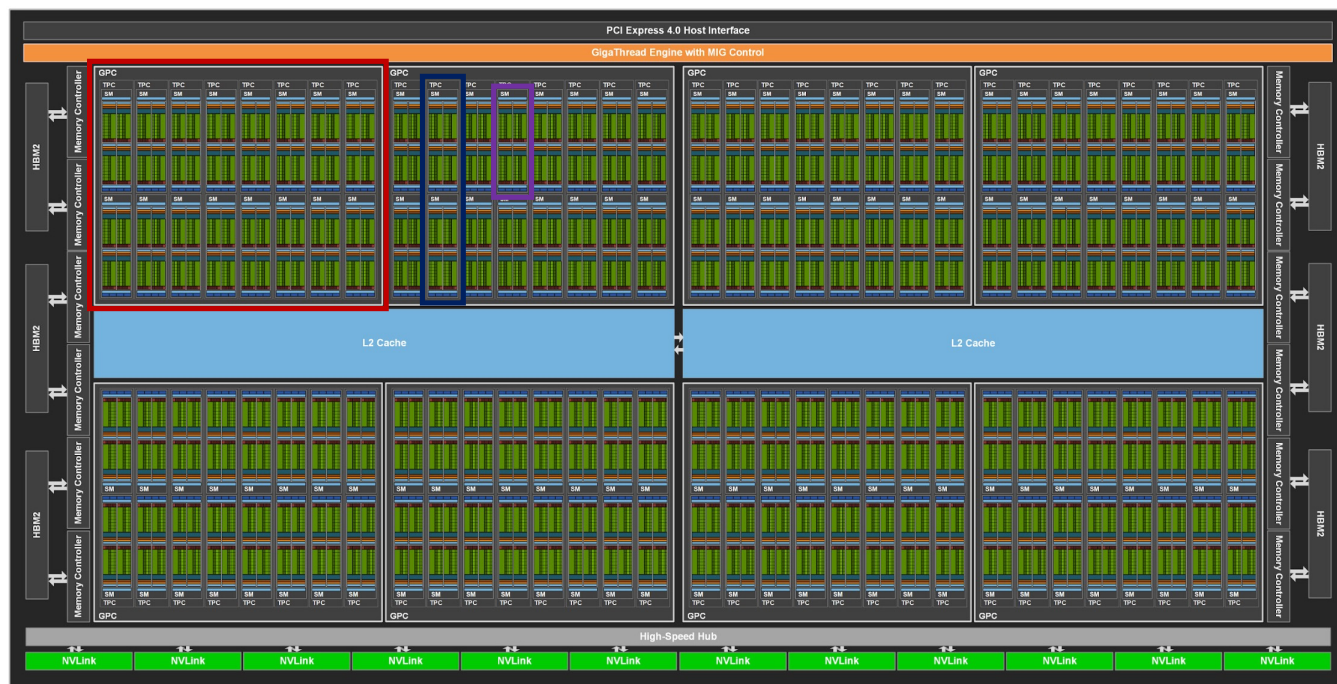
Single Instruction Multiple Data (SIMD)

- All the GPU cores execute exactly the same operation, but over different data.

- Rather than N clock steps to process N data points, N compute units process N data points in a single clock step (if sufficient cores).

- SIMD is effective for many applications:
    - Computer graphics – a single operation/series of operations needs applying to all pixels in an image.
    - Training AI models – input data in the same batch can be processed in parallel
    - Scientific simulation – many algorithms can be parallelised.

GPU Structure – NVIDIA A100

epcc

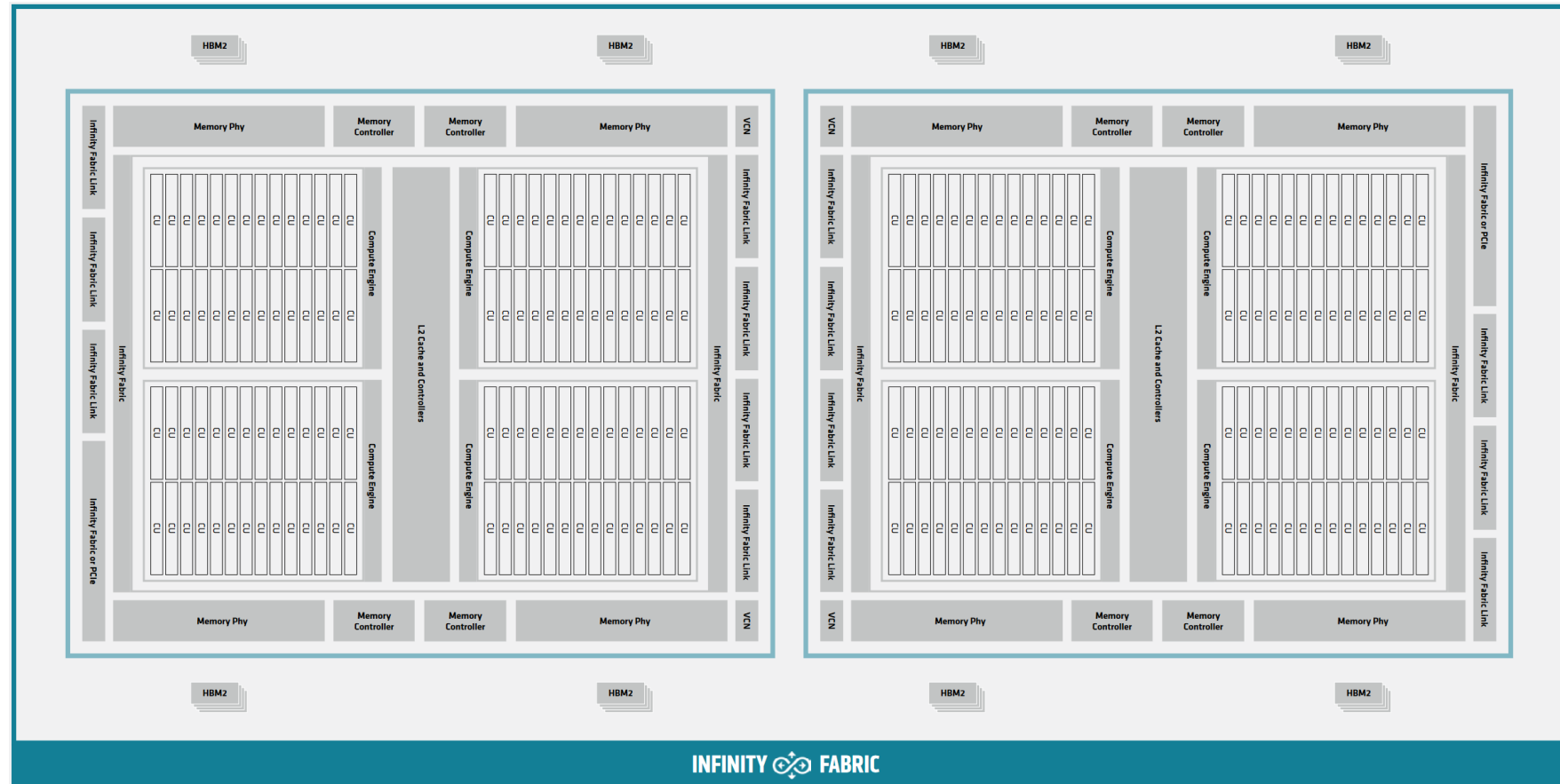https://resources.nvidia.com/en-us-it/ampere-architecture-white-paper (Figure 6) - GA100 Full GPU with 128 SMs (A100 Tensor Core GPU has 108 SMs)

GPU Structure – NVIDIA A100

GPU Processing Cluster (GPC)

Texture Processing Cluster (TPC)
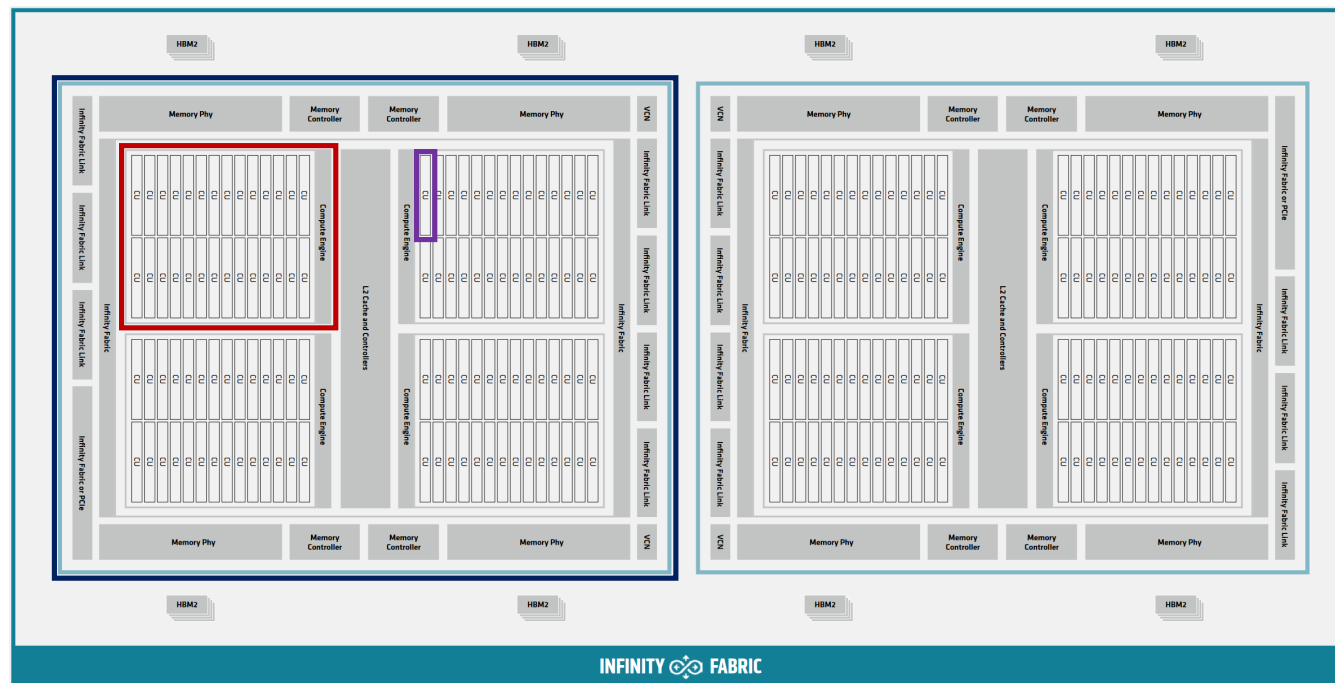
Streaming Multiprocessor (SM)

https://resources.nvidia.com/en-us-it/ampere-architecture-white-paper (Figure 6) - GA100 Full GPU with 128 SMs (A100 Tensor Core GPU has 108 SMs)

GPU Structure – NVIDIA A100

- Each SM is subdivided into 4 processing blocks.
- This allows for greater flexibility in scheduling.

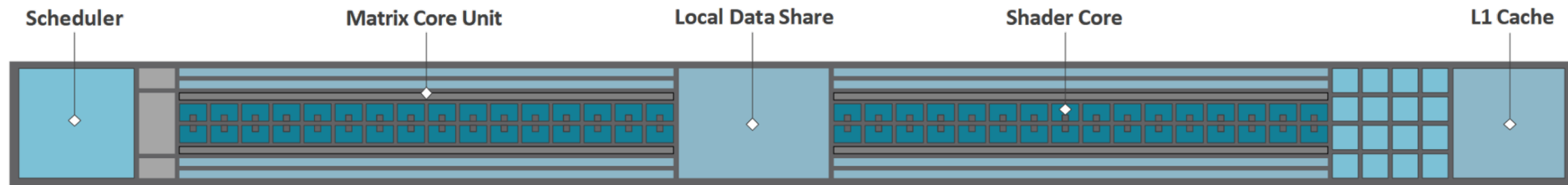https://resources.nvidia.com/en-us-it/ampere-architecture-white-paper (Figure 7) - GA100 Streaming Multiprocessor (SM)

GPU Structure – AMD MI200

GPU Structure – AMD MI200

Graphics Compute Die (GCD)

Compute Engine (CE)

Compute Unit (CU)

GPU Structure – AMD MI200

Scheduler · **Matrix Core Unit** · **Local Data Share** · **Shader Core** · **L1 Cache**

Notes:

4x Matrix Core Units (per CU)

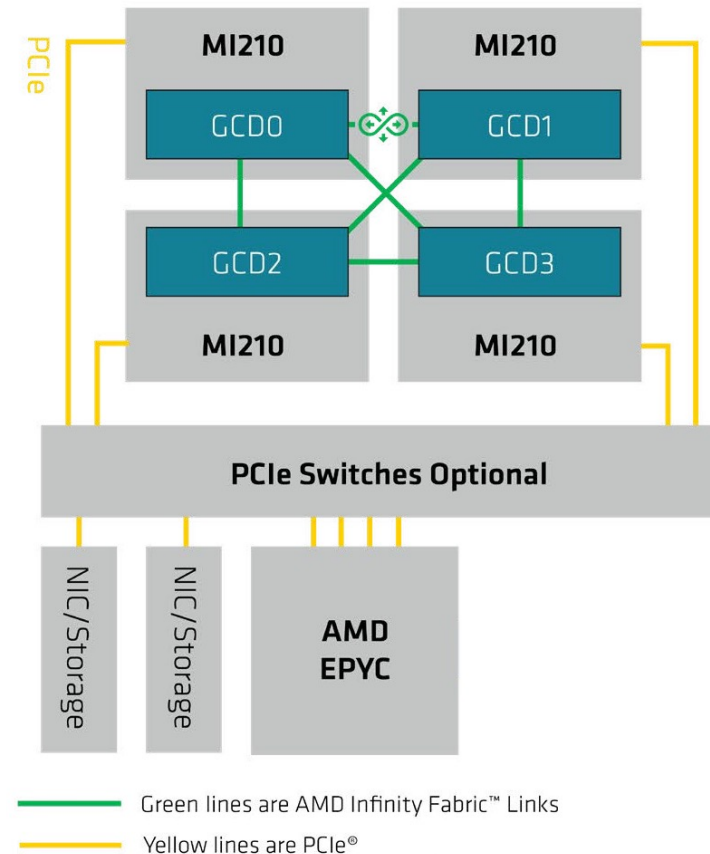4x 16-wide SIMD (per CU) for total of 64 Shader Cores per CU

GPU Structure – ARCHER2 (AMD MI210)

epcc

4 compute nodes - each with:

- 4x MI210 GPUs
  - Each with one Graphics Compute Die
  - 104 compute units
  - 64 GB HBM2e Memory
- 1x AMD EPYC 7543P (Milan) CPU
  - 32 core @ 2.8 GHz
  - 512 GiB host memory

Documentation:
https://docs.archer2.ac.uk/user-guide/gpu/

Adapted from https://www.amd.com/content/dam/amd/en/documents/instinct-business-docs/white-papers/amd-cdna2-white-paper.pdf Figure 2e – Block diagram of a flagship HPC/AI server node built using the AMD Instinct™ MI210 accelerators in a quad GPU hive with direct connect xGMI bridge boards and AMD EPYC™ processor

Threads, blocks, and warps

- **Thread** – item of work (i.e. a data point)
- **Block (Nvidia)/Workgroup (AMD)** - collection of threads

- Blocks/workgroups are assigned to SMs (Nvidia) or CUs (AMD)
- Threads execute in warps (Nvidia – 32 threads) or wavefronts (AMD – 64 threads)

Memory Hierarchy

|epcc|

Registers:

- Private to each thread – used to store local variables and intermediate results.

- Each thread has access to a set number of registers.

- If a kernel requires too many registers, it will utilise a portion of (slower) global memory as a local memory cache/scratchpad.
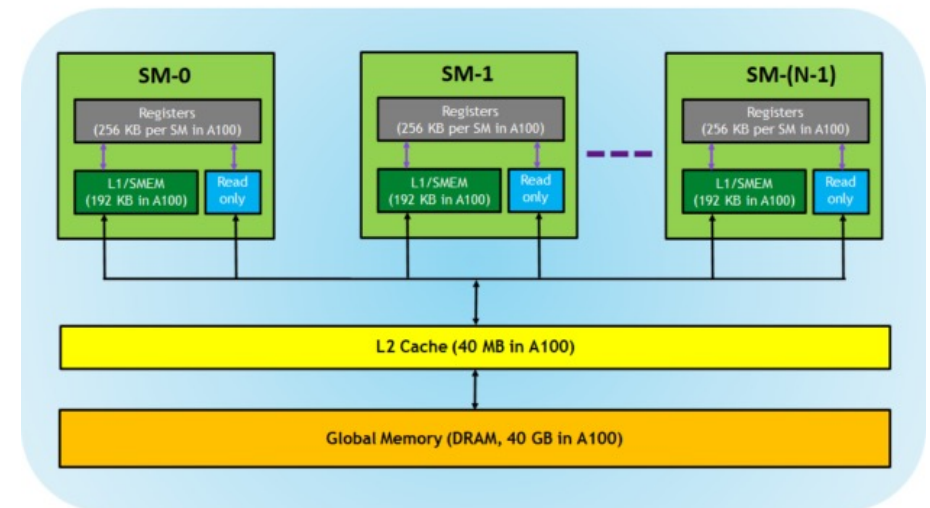

https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ - Figure 4
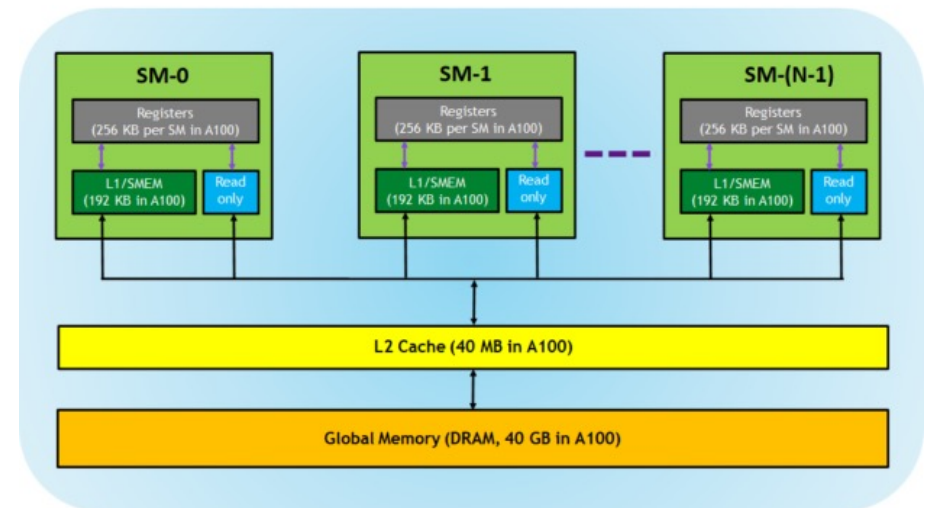
Memory Hierarchy

L1 Cache/Shared Memory:

- Physical L1 cache resource is shared between blocks/workgroups on a given SM/CU.

- Shared memory is shared between threads in the same block/workgroup.
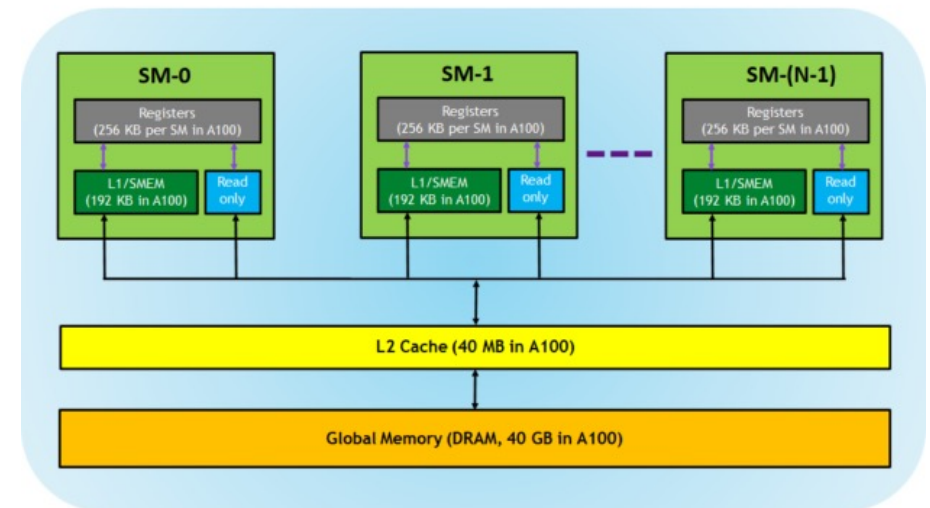

https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ - Figure 4

Memory Hierarchy

Read-only memory:

- Each SM/CU has memory which is read-only to kernel code.

- It may be used as an instruction cache, constant memory, texture memory, RO cache, etc.

https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ - Figure 4

Memory Hierarchy

|epcc|

L2 Cache:

- Shared across all SMs/CUs

- Hardware-managed cache that stores frequently accessed data to reduce memory latency

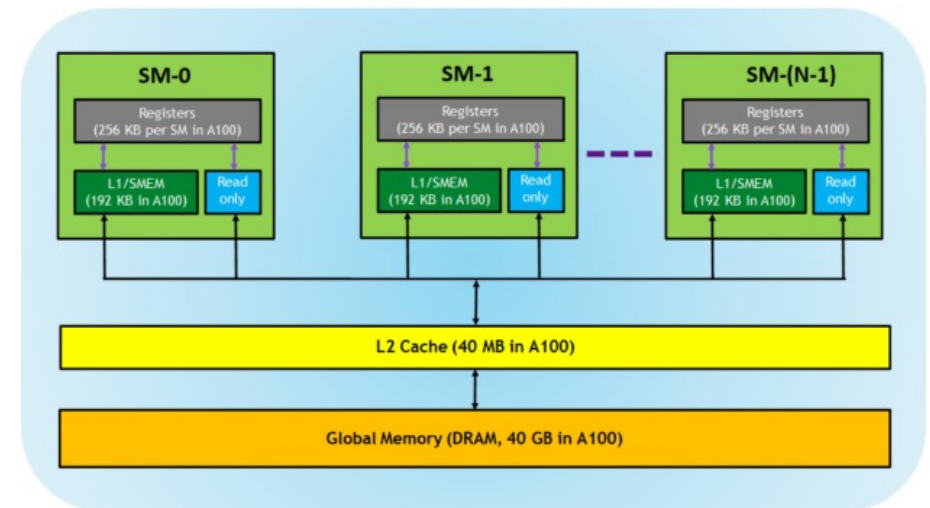https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ - Figure 4
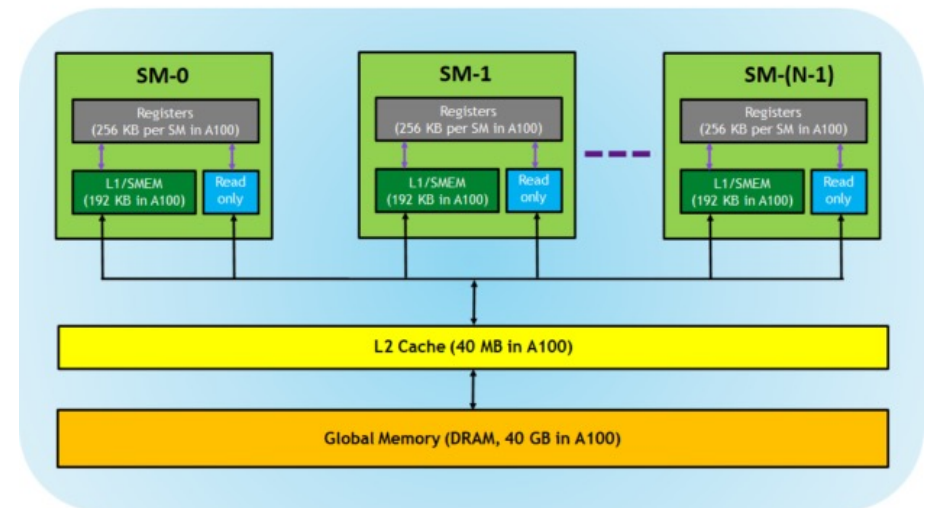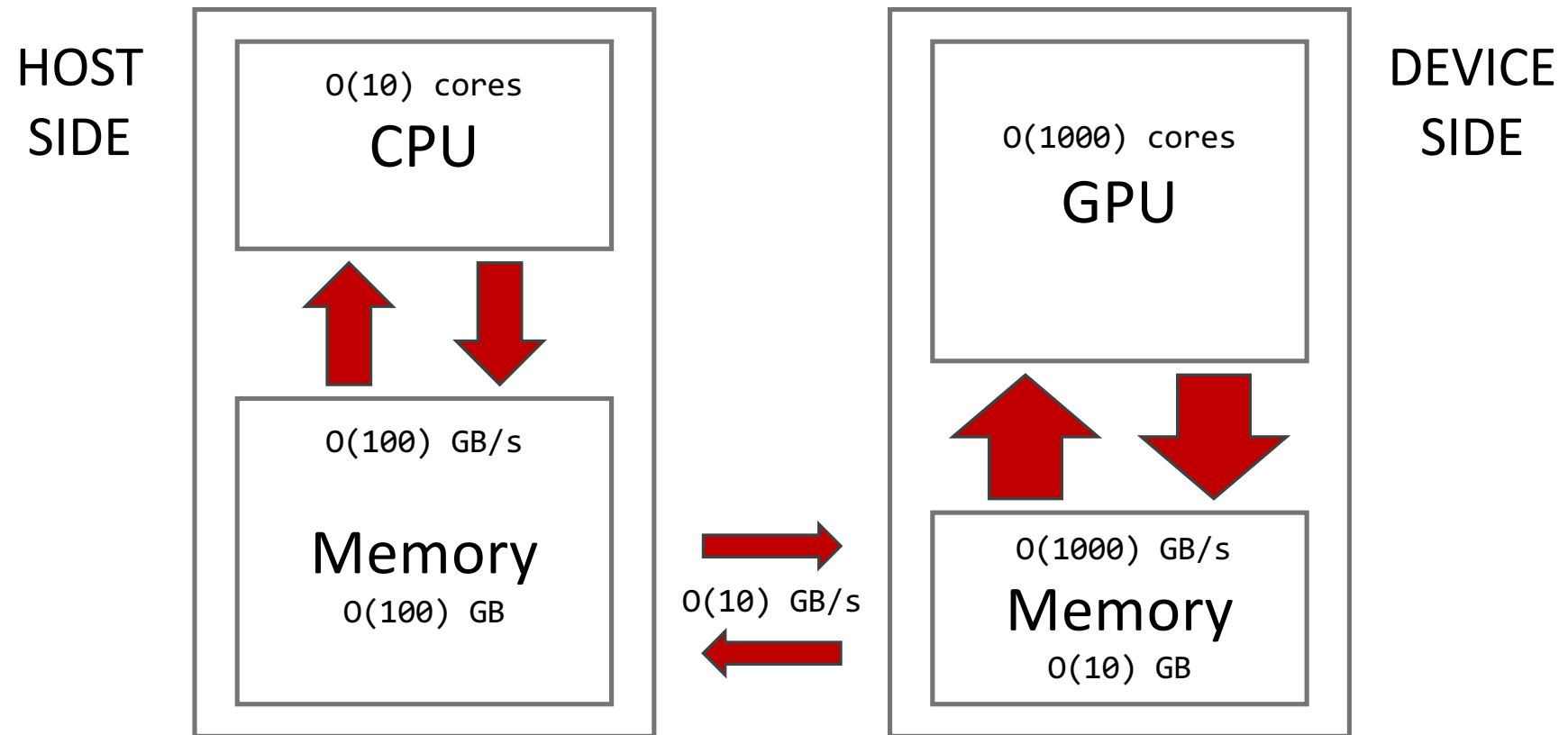
Memory Hierarchy

Global Memory

- Accessible by all threads, but access is relatively slow

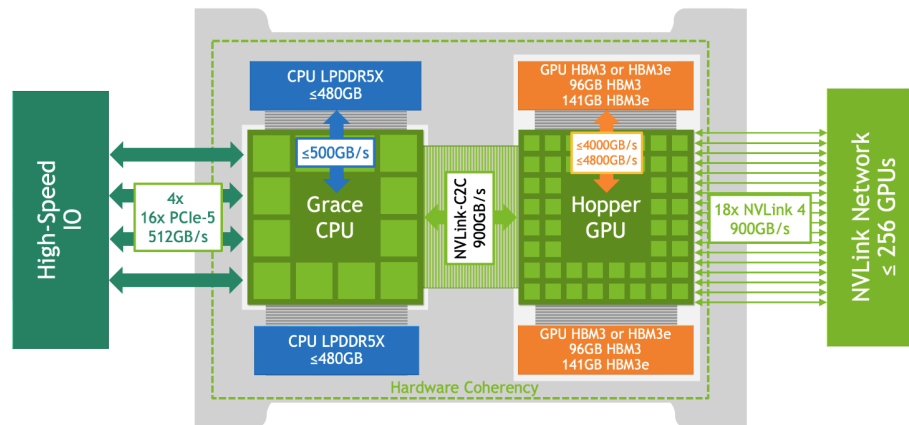- Primary storage for input data, output data, and global constants

https://developer.nvidia.com/blog/cuda-refresher-cuda-programming-model/ - Figure 4

Host Device Picture

epcc

HOST SIDE

DEVICE SIDE

O(10) cores
CPU

O(1000) cores
GPU

O(100) GB/s

Memory
O(100) GB

O(10) GB/s

O(1000) GB/s

Memory
O(10) GB

Unified Memory – Gracehopper (NVIDIA)

Seems to be they have separate CPU and GPU memory, but share an address space?

Whether data is stored on the CPU or GPU will therefore influence access time
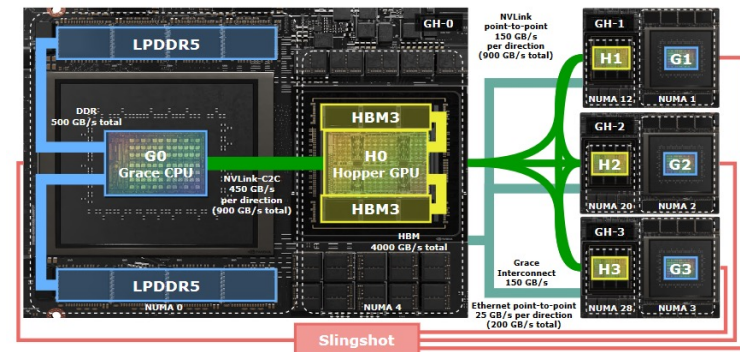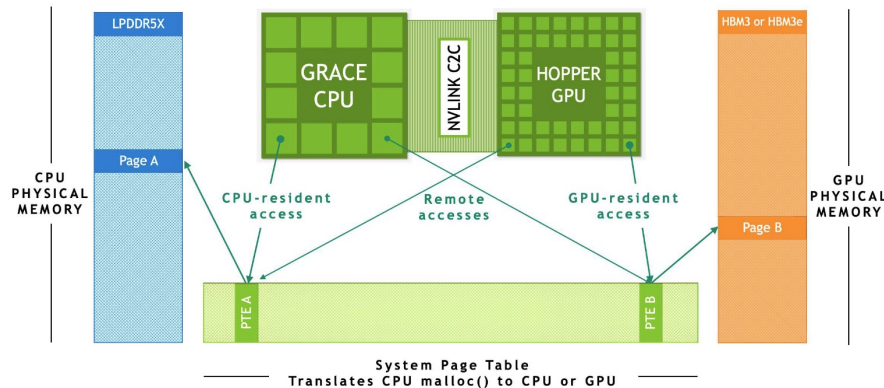

Fig. 1: Architecture of the Quad GH200 node of the Alps supercomputer. Every node is composed of four GH200 fully connected using NVLink and a cache coherent interconnect. Every GH200 is connected to a Slingshot network through a separate NIC.
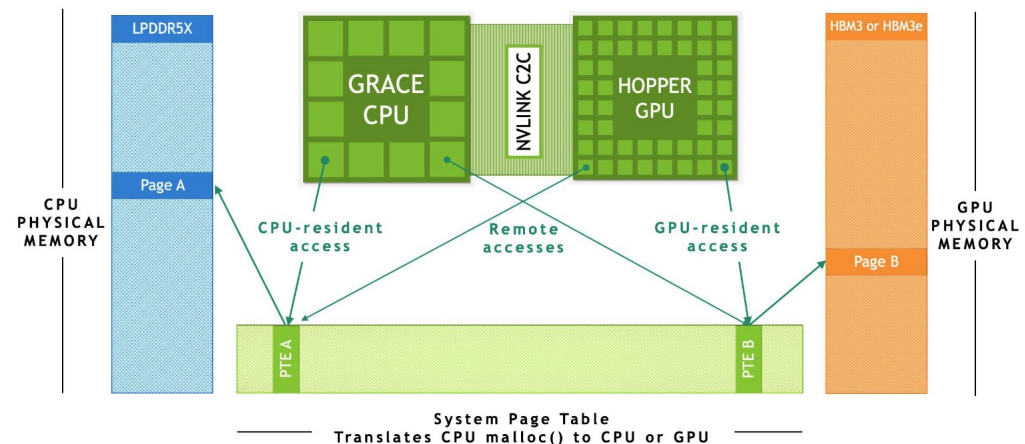
Unified Memory

- New 'Superchips' or APUs (Advanced Processing Units) have unified memory between the GPU and CPU.

- No longer need to move data between host and device.

- However!
  - Does not guarantee equal latency or bandwidth across the whole memory space.
  - Location of your data (i.e. host or device) is still important.

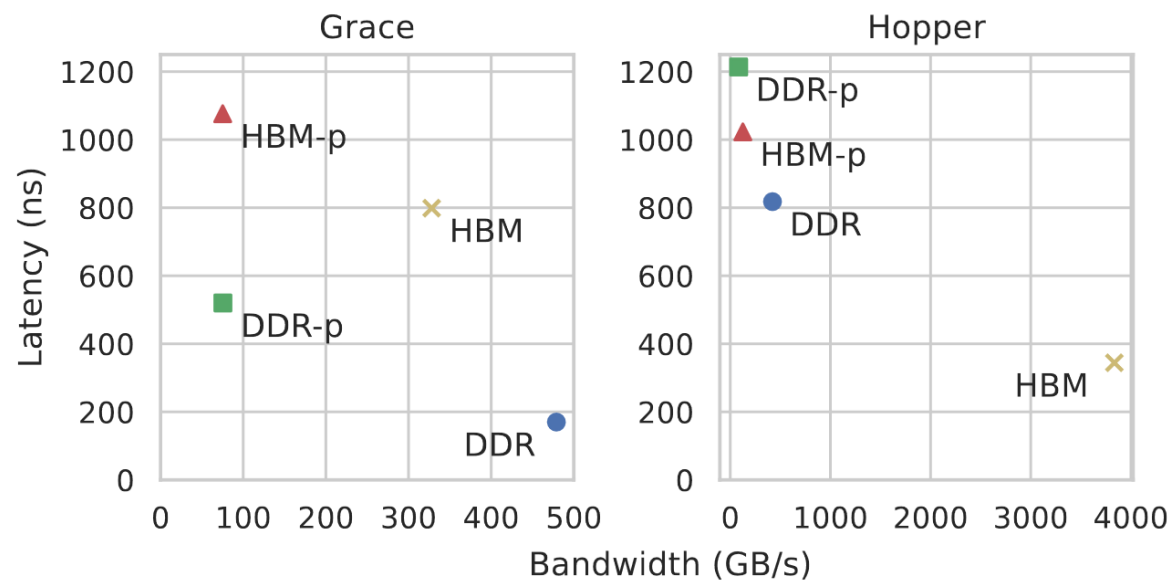# Unified Memory – Gracehopper (NVIDIA)
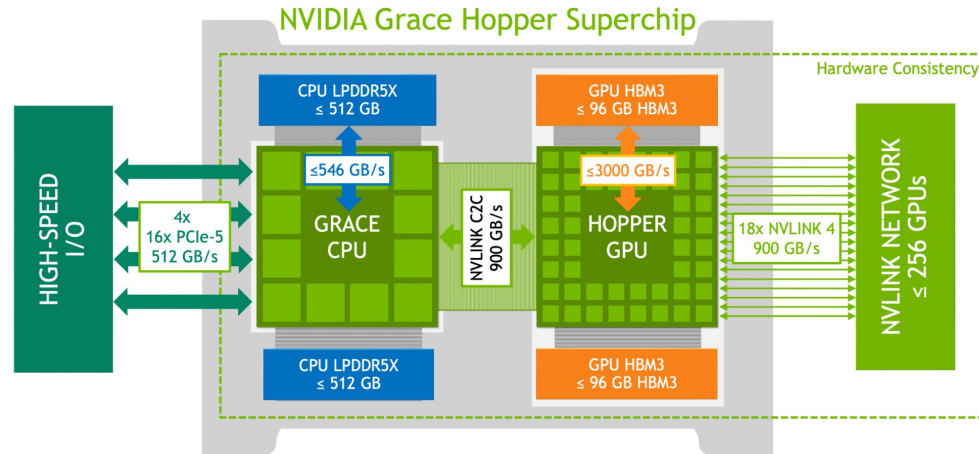
NVIDIA GH200 Grace Hopper Superchip combines a Grace CPU and Hopper GPU

System page table translates from CPU -> GPU and GPU -> CPU address spaces

Unified Memory – Gracehopper (NVIDIA)

Maximum bandwidth plotted against access latency achieved by Grace (left) and Hopper (right) to different memories of the system. The suffix "-p" indicates memory on a peer GH200.

Fusco, L., Khalilov, M., Chrapek, M., Chukkapalli, G., Schulthess, T., & Hoefler, T. (2024, August 26). Understanding Data Movement in Tightly Coupled Heterogeneous Systems: A Case Study with the Grace Hopper Superchip. arXiv:2408.11556v2 [cs.DC]. doi: https://doi.org/10.48550/arXiv.2408.11556

Unified Memory – MI300A (AMD)
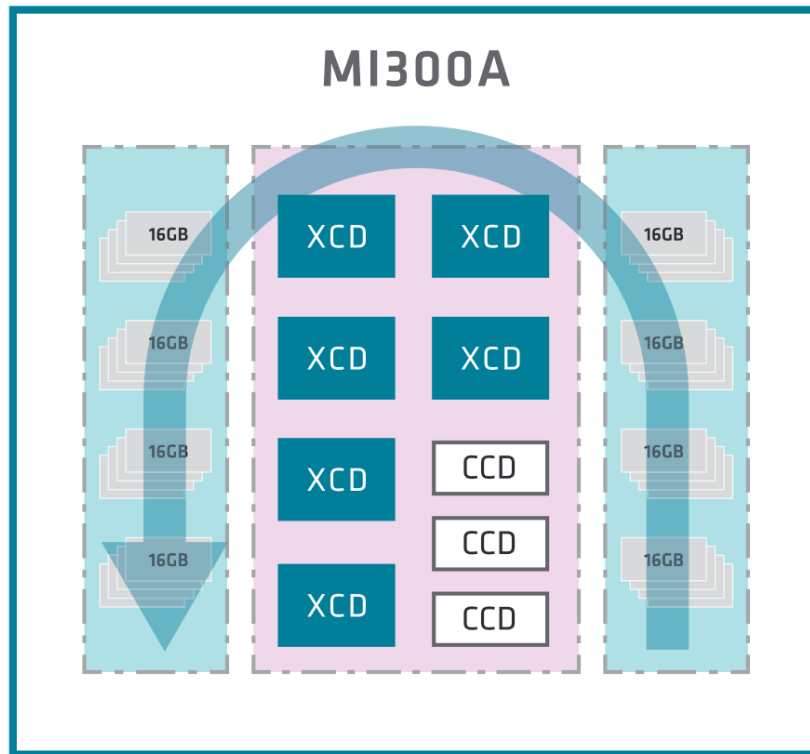
XCD - Accelerator Complex Dies

CCD - CPU Complex Dies

Unified Memory – MI300A (AMD)

- 128GB unified HBM3 memory

- Unlike GH200, GPU and CPU memory is in the same physical location.

- Exact memory layout is unclear:
  - Most places claim 128GB of HBM3 memory.
  - But white paper mentions 128 GiB of DRAM interleaved between HBM stacks: switch stack every 4KiB through physical memory space.

- Theoretical bandwidth:
  - 384 GB/s peer-to-peer
  - 5.3 TB/s on-package peak throughput

- Likely still latency and bandwidth differences between CPU and GPU chiplets.