Modern HPC Architecture: Fundamental Overview

NRIS

Norwegian research infrastructure services

Hicham Agueny, PhD
Scientific Computing Group, UiB/NRIS

TOP4 Powerful Supercomputers as of Nov 2025

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,340,000	1,809.00	2,821.10	29,685
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	1,000.00	1,226.28	15,794

US$600 million

Advance nuclear weapon science and scientific discovery.

€500 million

Motivation: Simulating Real-World Phenomena

e.g. Weather prediction| Epidemic spread | Molecular Dynamics | Autonomous systems

Example in quantum physics

TDSE $$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$$

A system with one electron **Psi(x,y,z)** N1e=nx*ny*nz nx=ny=nz= 1000 points **N1e=10^9x8byte = 2x8 GB**

A system with two electrons **Psi(x1,y1,z1,x2,y2,z2)** N2e = N1e^2=10^18x8byte = **2x8x10^9 GB**

The powerful supercomputer in the word - **El Capitan**; has **11136 nodes ~ 5.7 x10^7 GB**

A system with 100 electrons N = N1e^100 = 10^900 x 8byte = **8 x 10^891 GB**

Motivation: Simulating Real-World Phenomena

e.g. Weather prediction| Epidemic spread | Protein structure | Autonomous systems

Core Challenges

- **Uncertainty in Predictions**
  - Simplifying mathematical models to make predictions → simplifications introduce uncertainty
  - **Critical data scarcity** in many science & engineering scenarios
  - Approximations increase prediction errors
- **Trade-off dilemma**: More realistic models → higher costs → forced approximations → increased error
- **Computational limits**: Even supercomputers struggle with real simulation
  motivating **hybrid approaches** (e.g., AI surrogates + high-fidelity models)
- **HPC opportunities & challenges**:
  - Enables higher-resolution models and lower error
  - Requires optimizing for **heterogeneous architectures** (GPUs, etc.)

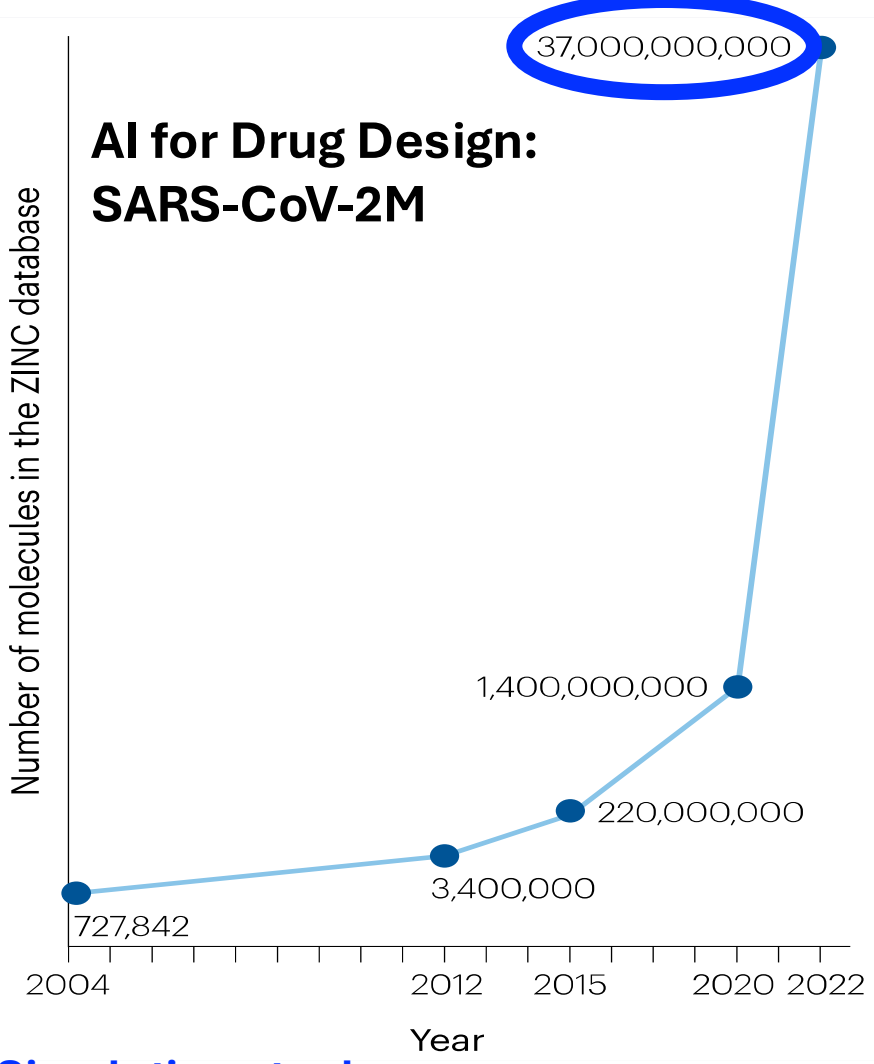
HPC Constraints

Storage & Memory

- GPT-3 (175B) checkpoint ≈ **350 GB**
  - ≈ **3.5 TB** for 10 concurrent tasks
- GPU memory for training can exceed **1.2 TB**
- What about the newest Mistral **675B ?**

Computational Time

- Training **Llama-3.3-70B**: **7.0M GPU-hours**
  - ≈ **1 week** on El Capitan (44,544 GPUs)
  - ≈ **960 days** (~2.6 years) on Olivia cluster (304 GPUs)



Simulations took:

90 days on **250 GPUs & 640 cpu-core**

1 day on **27000 GPU** Summit supercomputer

Nature Reviews Drug Discovery **23**, 141–155 (2024)

Agenda

- Short introduction to HPC
- Compute **Architecture**
  - Cluster **Architecture**
  - CPU & GPU **Architectures**
  - High-speed Network **Architecture**
- File System/Storage **Architecture**
- Optimization Strategies: GPU-aware data movement

Why Care About HPC Architecture ?

- **Optimizing your code:** for specific hardware you are targeting.
  - Knowing the architecture helps optimizing code for better performance.
  - Use features that provide higher bandwidth and lower latency.
  - Ex. GPUDirect P2P, RDMA, Storage
    (memory hierarchy, cpu-affinity, NUMA domains)

- **Computing is expensive:** GPU time costs: **15 - 35 kr GPU-hour** → **ex. 20 GPUs for 10 days/month**
  → **72000 - 168000 kr x 12month = 864,000 - 2,016,000 kr**

- **Troubleshooting & Diagnostics:** When things go wrong, understanding the architecture of software Stack helps users **diagnose issues**
  Ex. Code works on login node; but crashes on compute nodes.
  Ex. OpenMPI in Cray systems, LibFabric, UCX

- **Not abusing the system:** Direct installation of ML frameworks creates **tens of thousands** of small files, overwhelming the entire system.

◆ **Just Curious about HPC architecture**
It is a JOY to understand how things work!

Terminology: Traditional cluster vs HPC cluster

- **How is a traditional computer cluster different from an HPC cluster?**

Terminology

- **Server or compute node**: a single unconnected node.

- **Cluster:** Collection of computers (nodes) that are just connected.

Cluster

- **HPC system**: Collection of compute nodes connected via a **high-speed network** called interconnect & forming a **single system**.
  - **High bandwidth**: Fast data transfer capacity.
  - **Low latency**: Minimal data transfer delay.
  - **High scalability**: Ability to handle increased load efficiently.

HPC Cluster

- **What is a supercomputer ?**

Oxford English Dictionary, a **supercomputer** refers to
"*The design and use of computers with* ***exceptionally high processing power*** *or* ***speed***".

Ex. **Cray-2** from the 1980s – with just **four processors**, it represented the *state-of-the-art* in **supercomputing**
Today: El Capitan – Over 11 millions cores ....

To understand how powerful an HPC system is from a theoretical point of view, what quantity should we look at?

Performance of a computer

- **The performance of a processor is measured by the quantity:**

FLOPS (**F**loiting-**P**oint of **O**pertaions **P**er **S**econd).

It is a measure of the speed of a computer to perform arithmetic operations.

For a single processor:

FLOPS = (Clock speed)x(cores)x(FLOPs/cycle)

=Peak performance

FLOP is a way of encoding real numbers (i.e. FP64 or FP32, FP16…)

- 1 TeraFLOPS = 10^12 calculations per second.
- 1 PetaFLOPS = 10^15 calculations per second.
- **1 ExaFLOPS** = 10^18 calculations per second

(quintillion floating point operations per second)

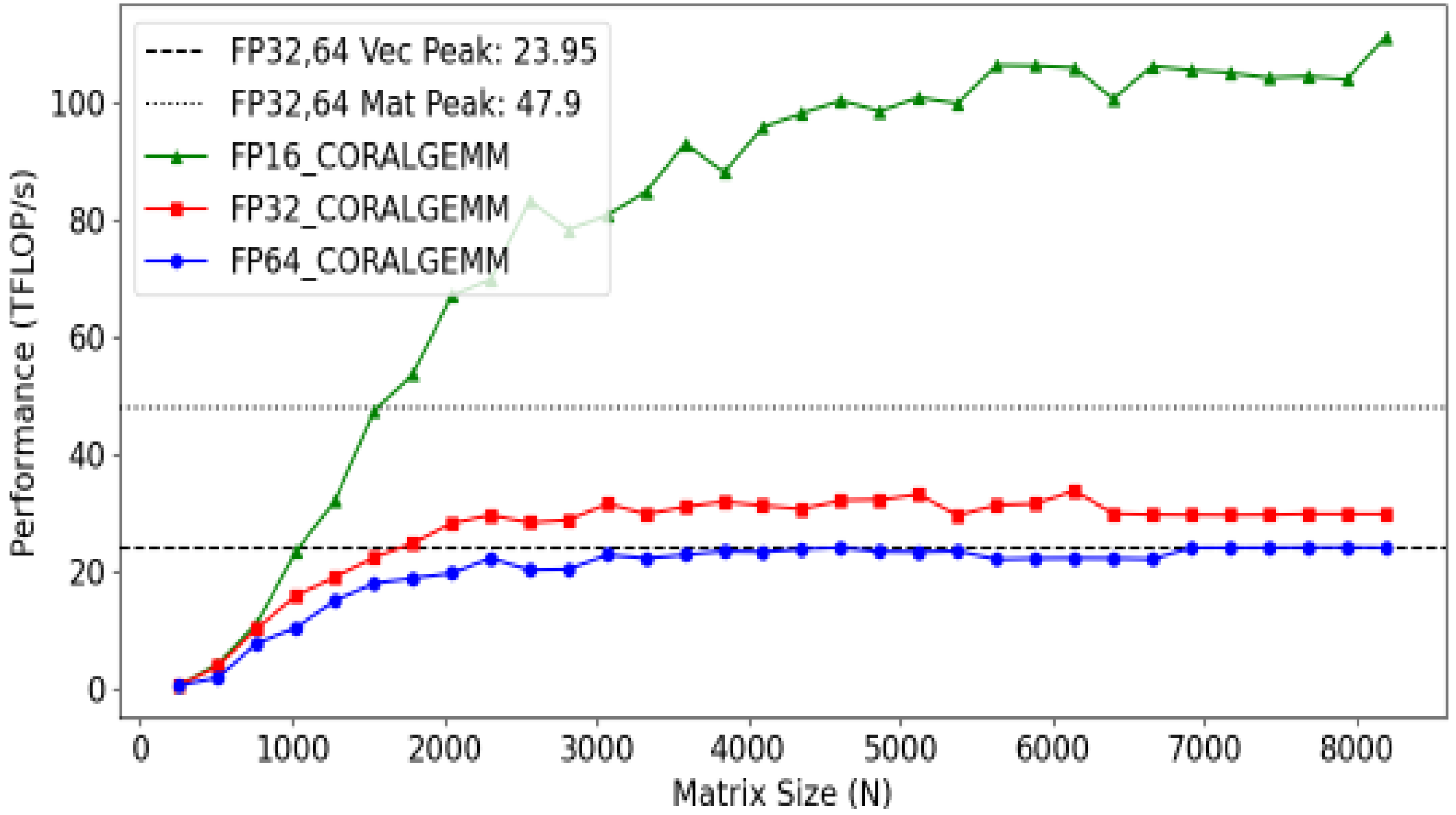
1.000.000.000.000.000.000

- **1 GigaFLOPS**: processor can handel **billion floating-point (64 bit) operations every second**.
- **For matching: 1GigaFLOPS ~performing one calculation every second for 31.69 years**.

TOP4 Powerful Supercomputers as of Nov 2025

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,340,000	1,809.00	2,821.10	29,685
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	1,000.00	1,226.28	15,794

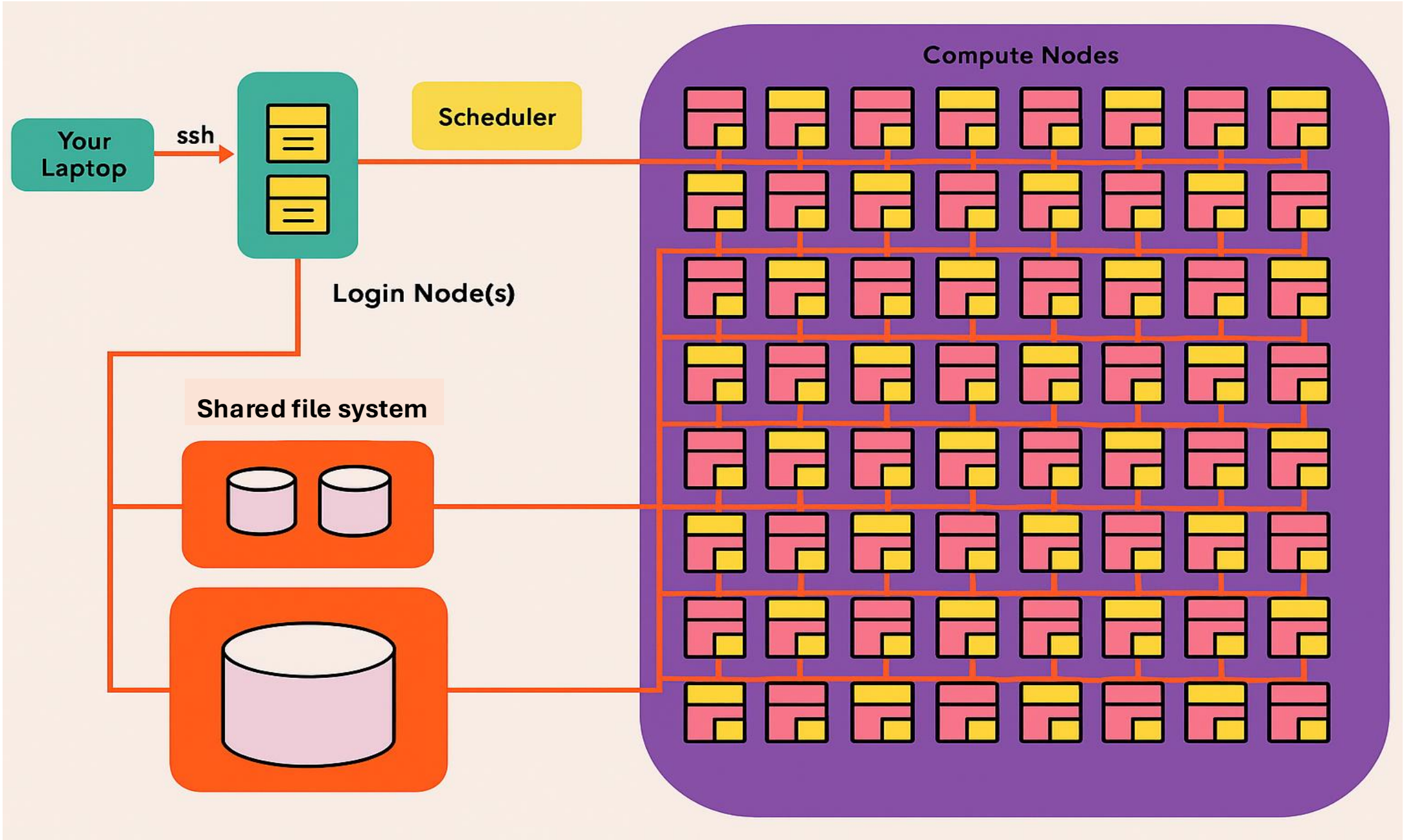
Comparison of peak performance: FP64 vs FP32 vs FP16



https://www.osti.gov/servlets/purl/2224161

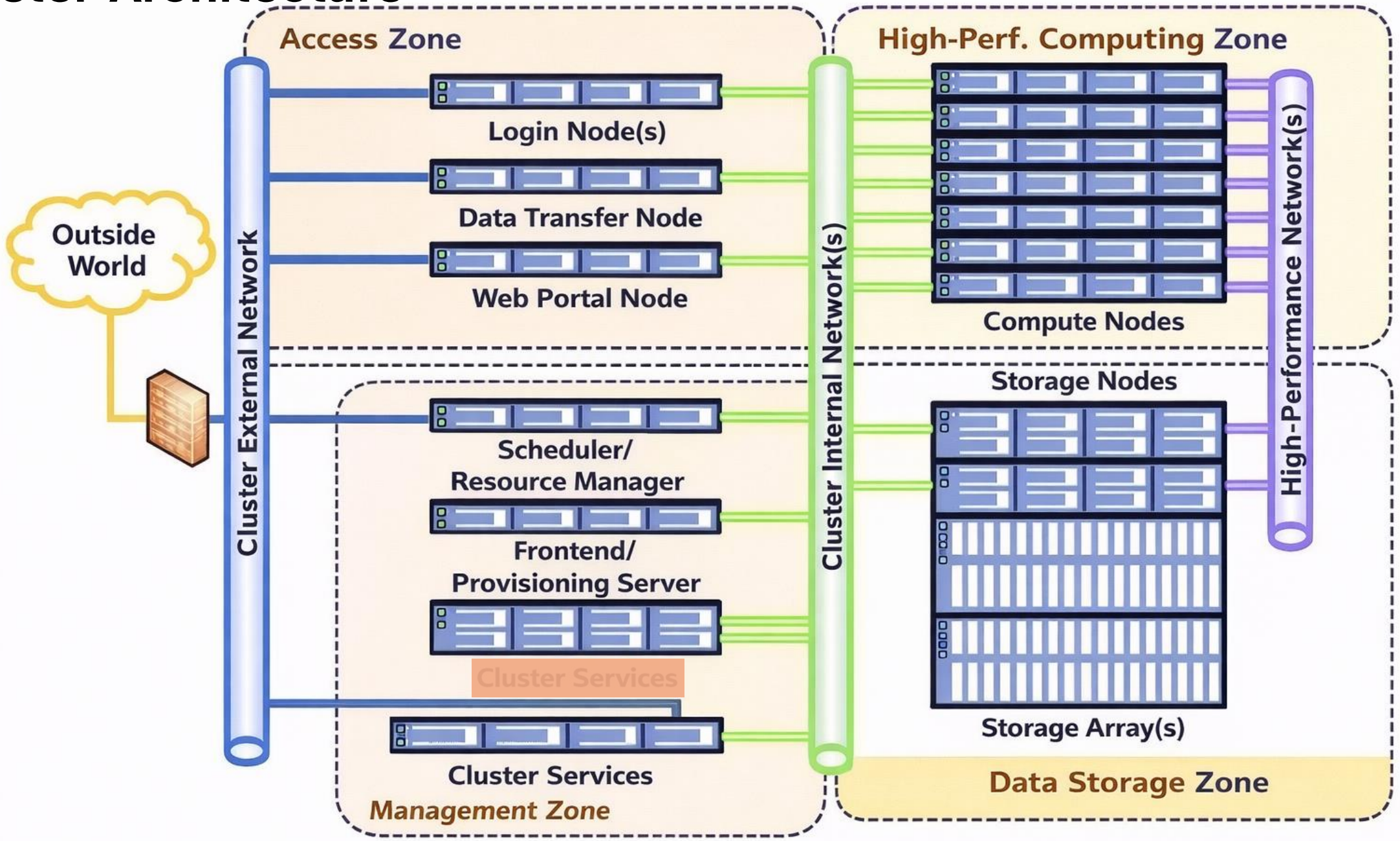
Basic Cluster-Architecture

Basic Cluster-Architecture



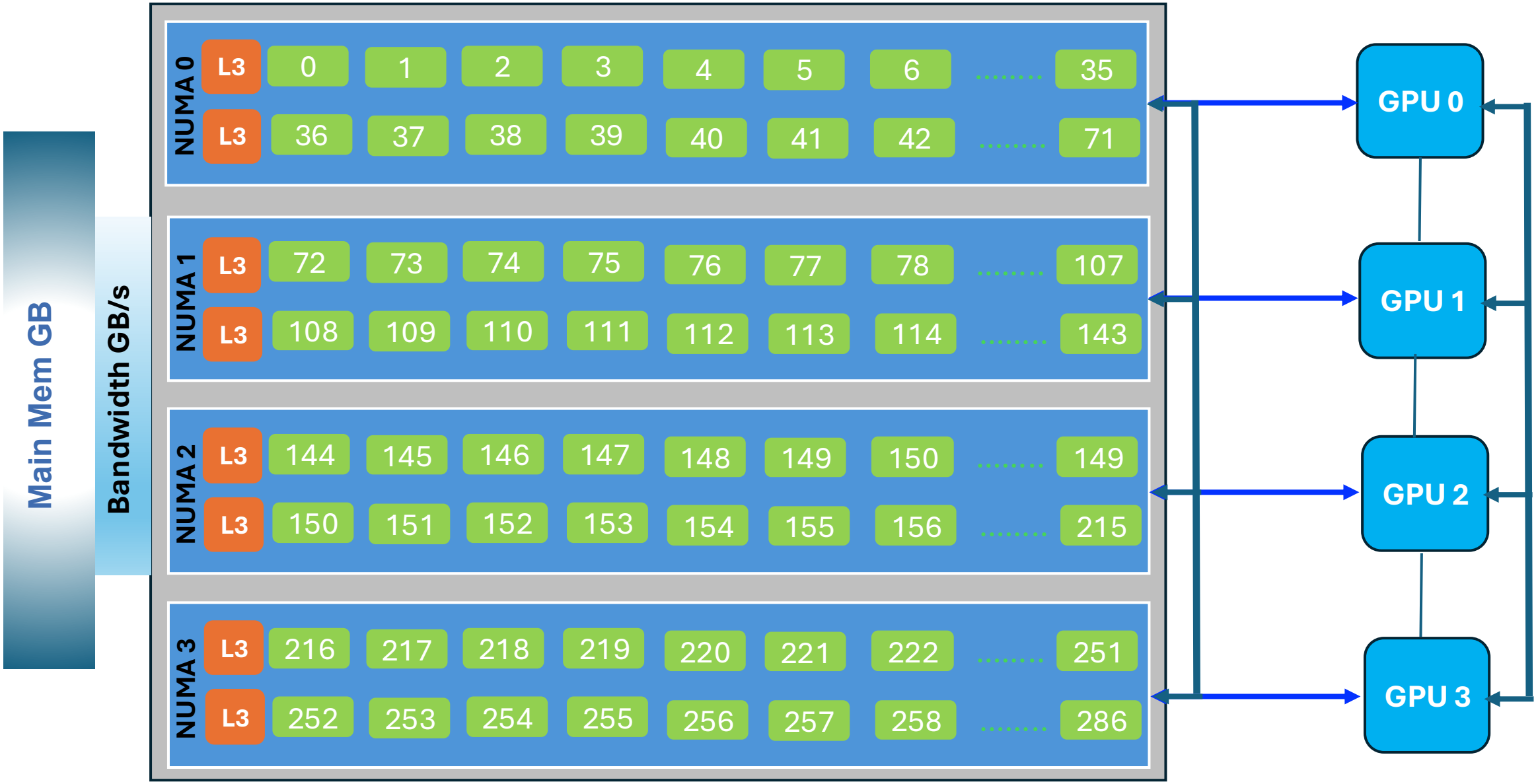
Adapted from https://indico.cern.ch/event/814979/contributions/3401193/attachments/1831477/3105158/comp_arch_codas_2019.pdf

Basic Cluster-Architecture



Adapted from: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-223.pdf

Compute node diagram (Simplified)



Hands-on Example:

Slurm Quick-Start Commands

Goal: How compute nodes are represented in Slurm and how to inspect their architectural details.

- List available compute nodes & their states using **sinfo** command.
- Inspect node and job details (CPU, memory, GPU) using **scontrol**.

SSH setup to Olivia

$ssh username@olivia.sigma2.no

Out:(hicham@login.olivia.sigma2.no) One-time password (OATH) for `hicham':

Out:(hicham@login.olivia.sigma2.no) Password:

$mkdir /cluster/work/projects/nn14000k/$USER

$cd /cluster/work/projects/nn14000k/$USER

$ git clone https://github.com/HichamAgueny/hpc-course.git

$cd hpc-course

Hands-on Example

hpc-course / examples / 01_slurm_basics / README.md

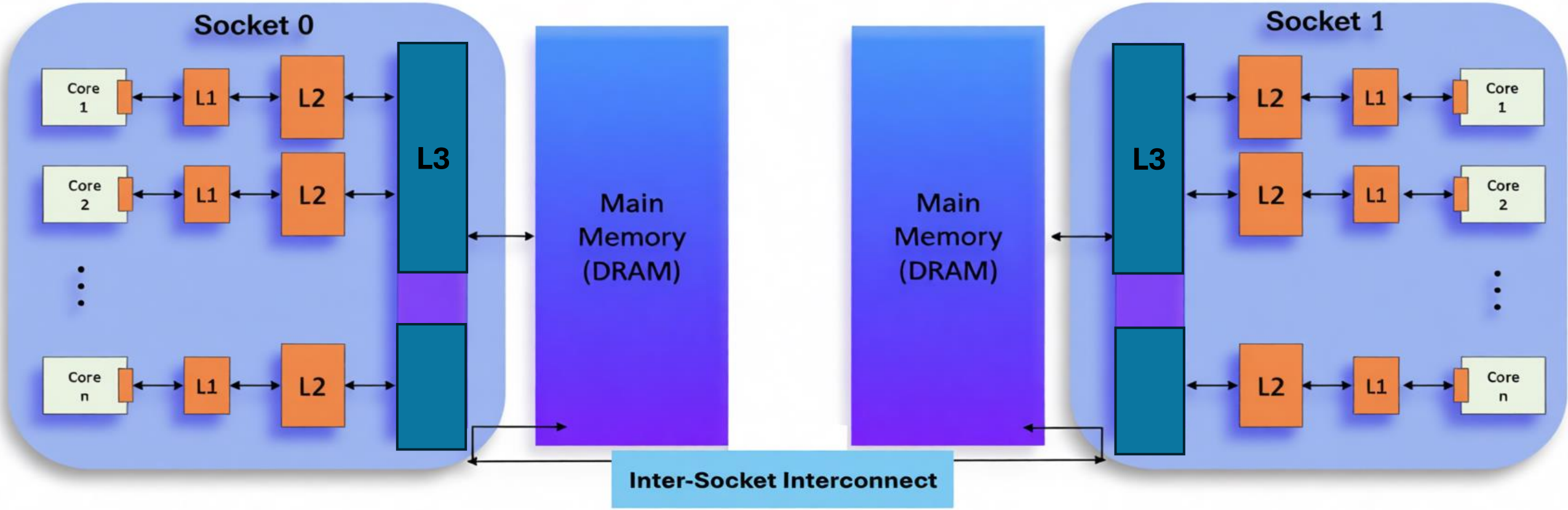
Slurm Quick-Start Commands

1.0 A tiny workflow example: Details below

```
# 1️⃣ See the whole cluster status
sinfo

# 2️⃣ Look at the "accel" partition details
sinfo -p accel

# 3️⃣ Peek at a specific node that you know belongs to that partition
scontrol show node <node-name>

# 4️⃣ Submit a job, wait a few seconds, then check its status
squeue -p accel   # List all jobs that are currently running in the "accel" partition
scontrol show jobid <JobID>        # show allocation, state, node list, etc.
```

CPU & GPU architectures

Traditional CPU architecture

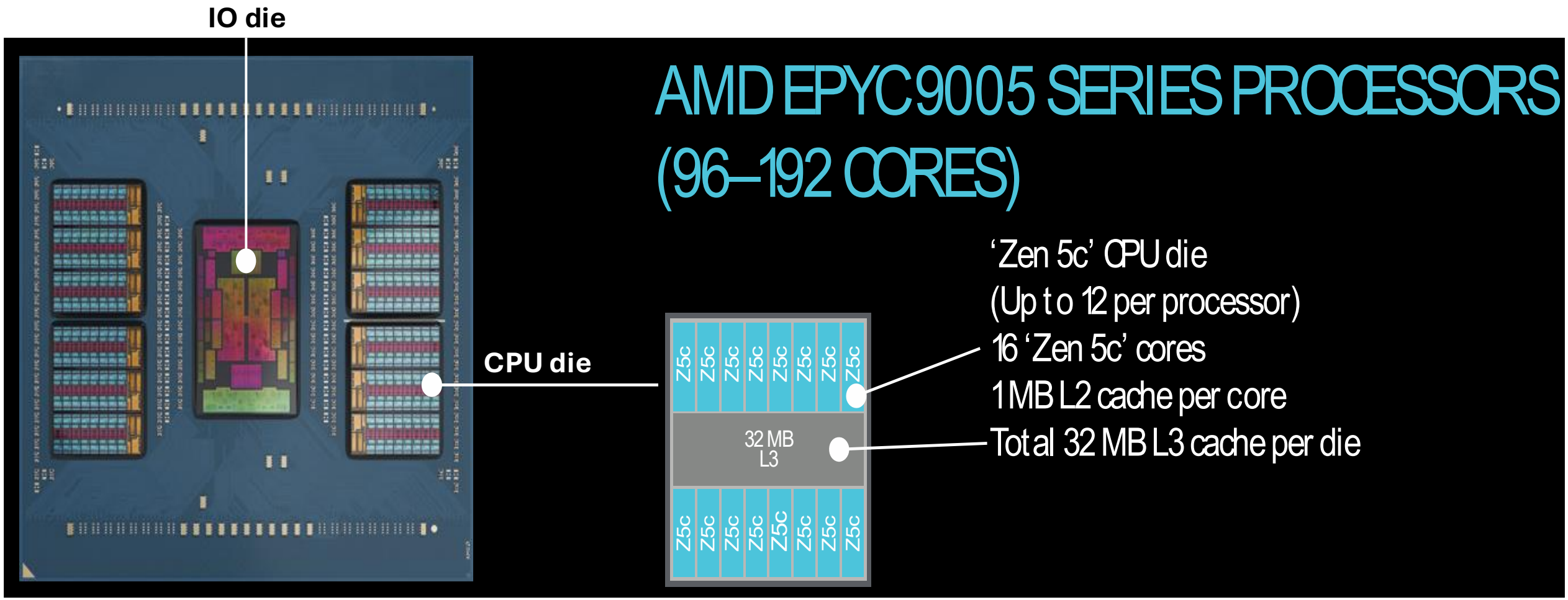


Adapted from *https://indico.cern.ch/event/814979/contributions/3401193/attachments/1831477/3105158/comp_arch_codas_2019.pdf*

AMD EPYC 7742, Zen 2 architecture, 2 sockets, 128 cores total.

	L1 cache	L2 cache	L3 cache
Size	32 KiB per core	256 KiB per core	256 MiB per socket
Latency	~4 cycles	~12–15 cycles	~35–45 cycles
Scope	Private per core	Private per core	Shared by 4 cores (per CCX)

CPU Processor Architecture: 5th Gen AMD EPYC



12 CPU die
Each die has 16 cores
16 cores share 32 M L3 cache
Max Memory Capacity **6 TB**

'Zen 5c' core is produced at **3nm**
I/O die remains at **6nm**
Memory channels / max per socket
theoretical bandwidth **12 / 614 GB/s**

Core density up **192 cores**
The highest of any x86- architecture
CPU available today

https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/5th-gen-amd-epyc-processor-architecture-white-paper.pdf

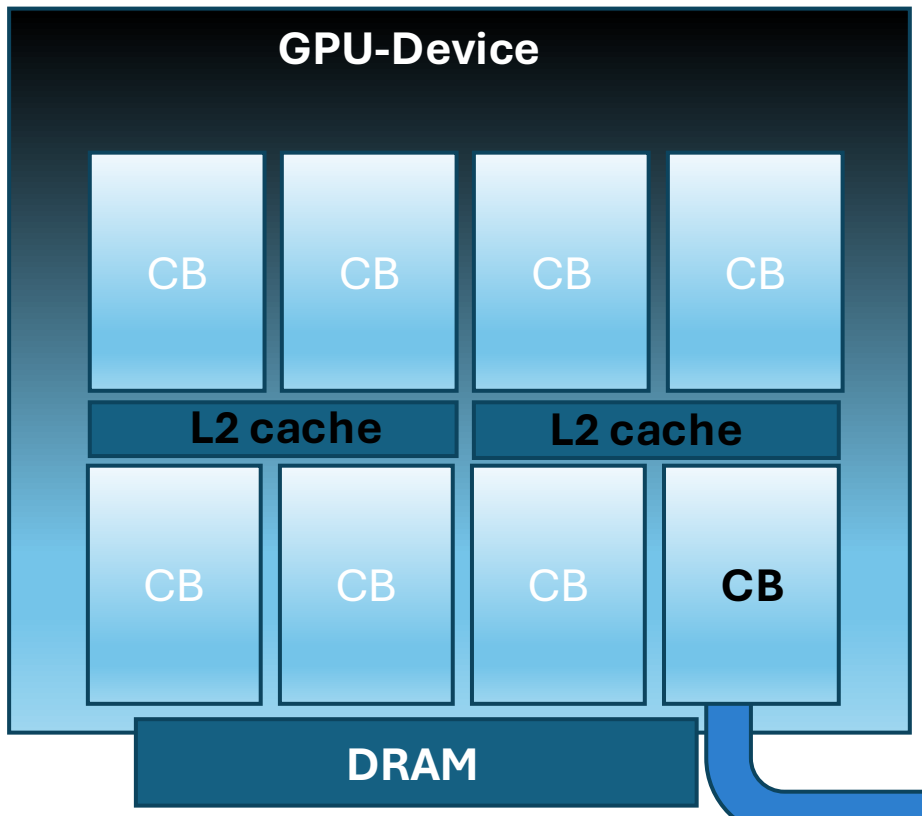
Traditional CPU architecture

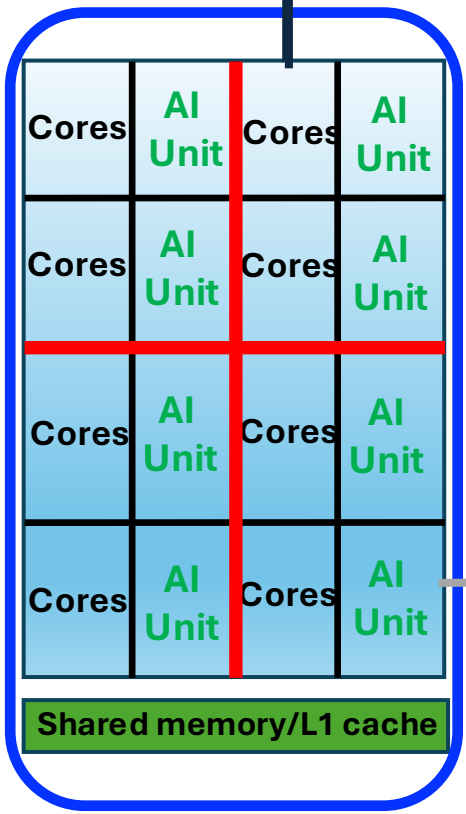
CPUs are designed for

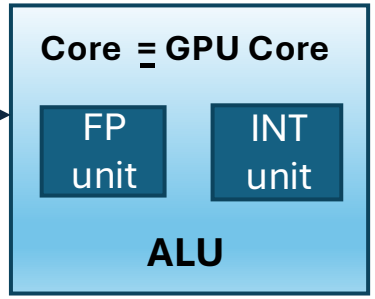
- Low latency (use caches L1/L2/L3)
- Responsive (fast change of instruction e.g. “if” block, loops with variable lengths)
- single-threaded performance (running one main task faster, even with **irregular steps**)

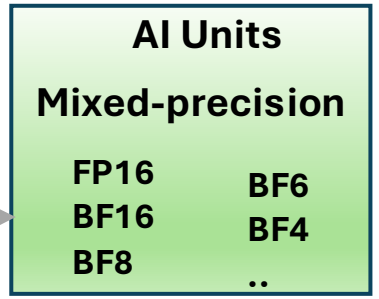
The CPU design becomes less efficient when processing millions of simple operations in parallel.

Simplified GPU architecture







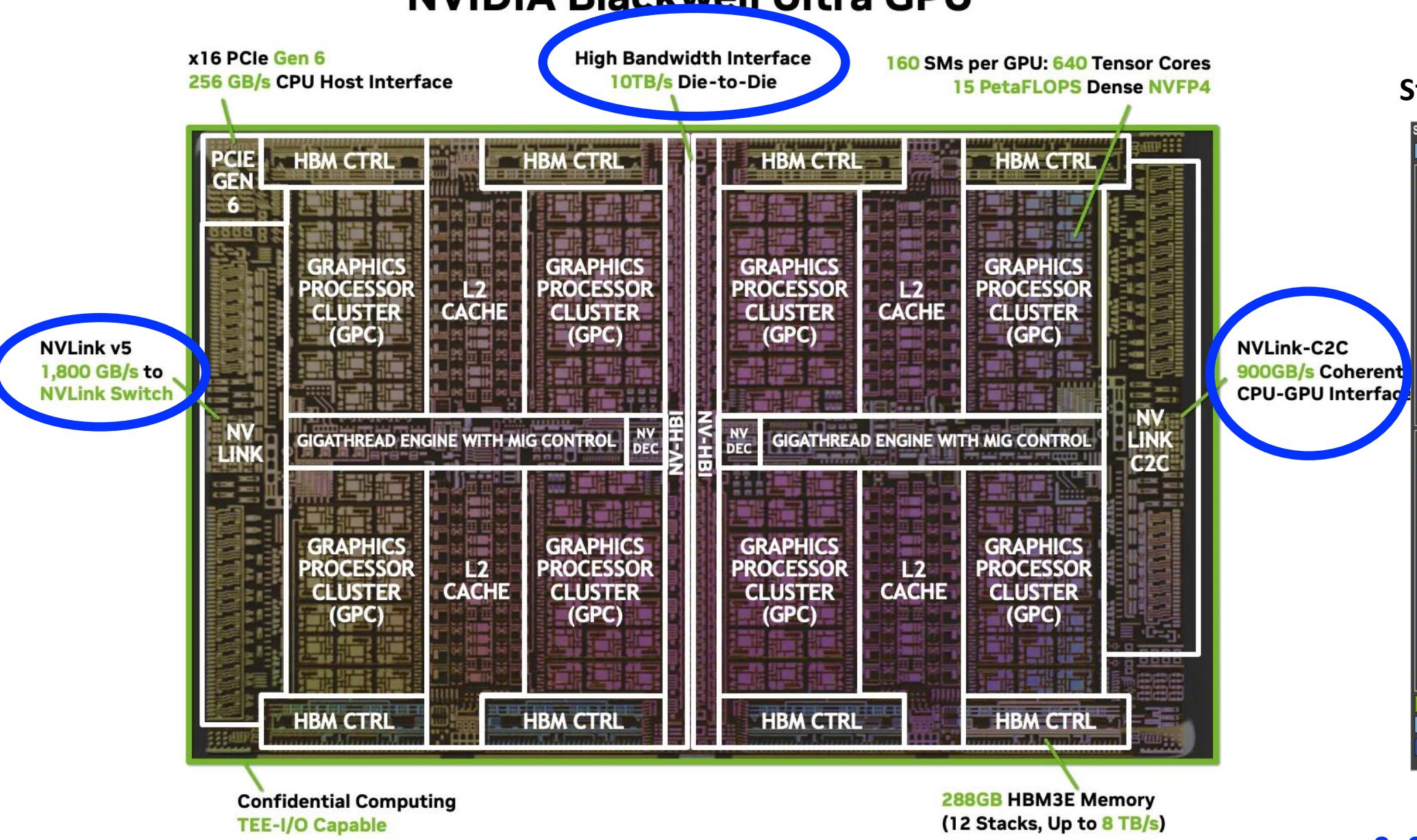


Compute Block (CB)

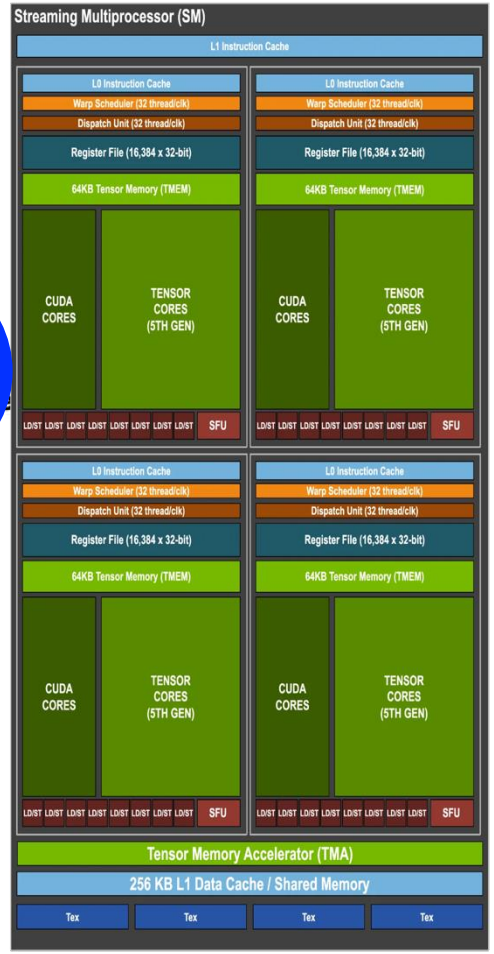
All threads share L1 cache.

- Core includes ALU. Various logical operations for computations (FP32, FP64) e.g. addition, multiplication and ...
- ALU executes SIMT/SIMD instructions. (processing multiple data with the same operation concurently (in parallel))

NVIDIA Blackwell Ultra GPU



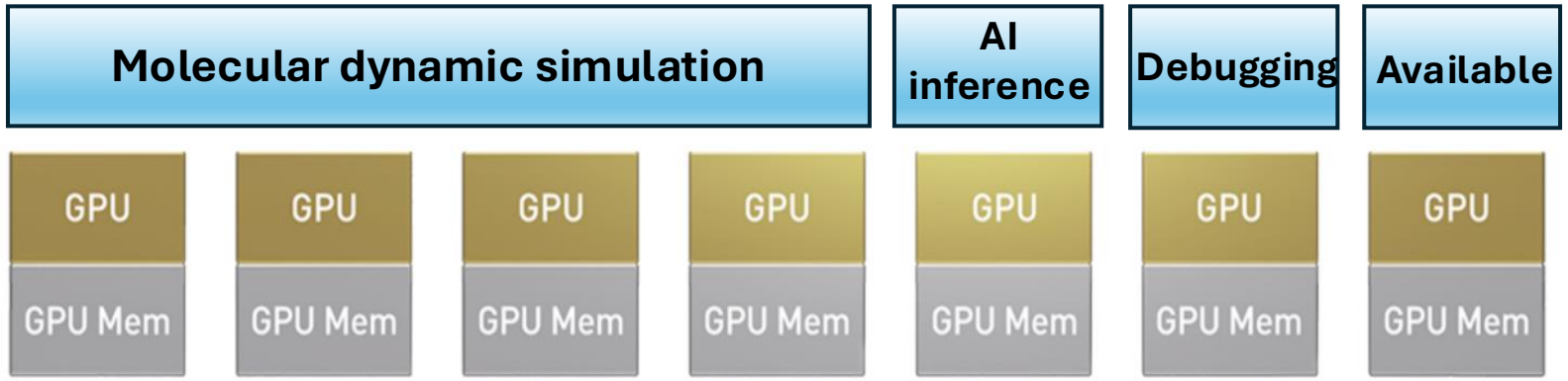
Streaming multiprocessor

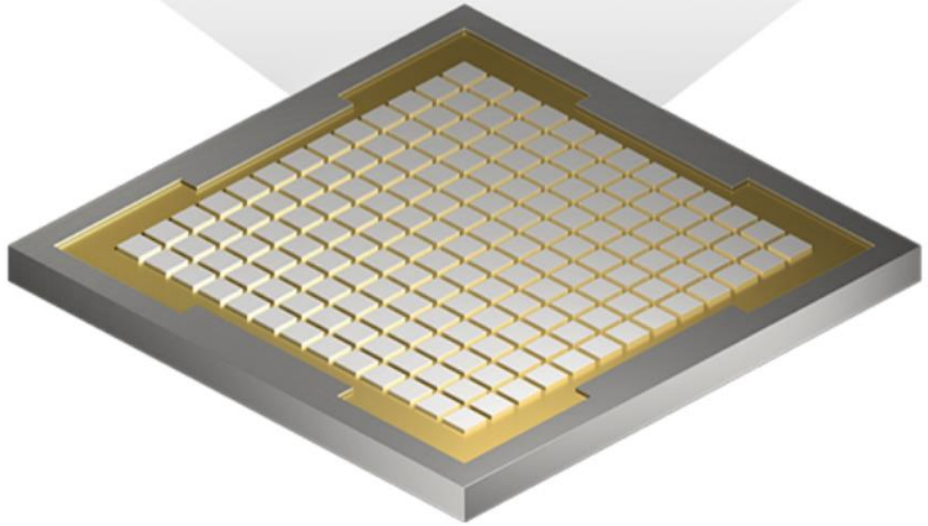


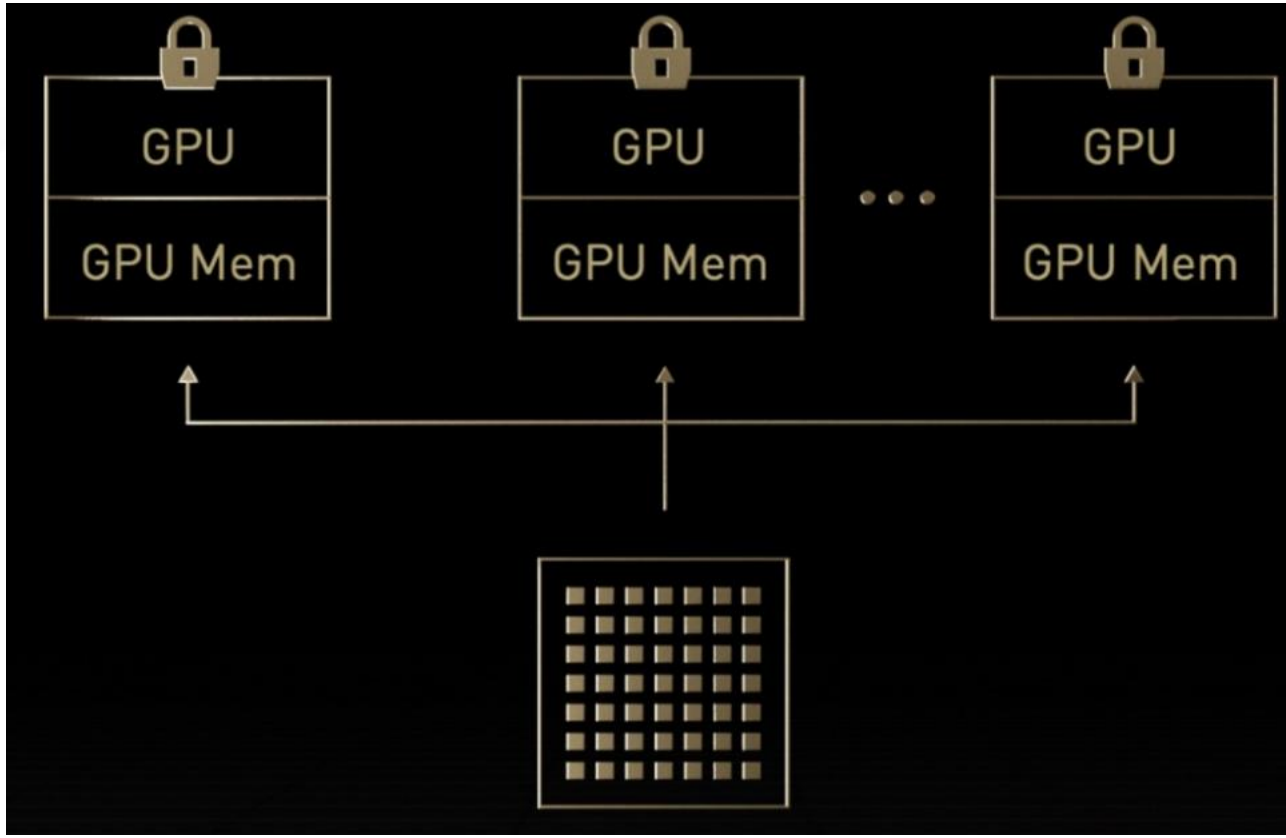
8x20x160-→25.000 cores

https://developer.nvidia.com/blog/inside-nvidia-blackwell-ultra-the-chip-powering-the-ai-factory-era/

Multi-Instance GPU (MIG): NVIDIA GPU

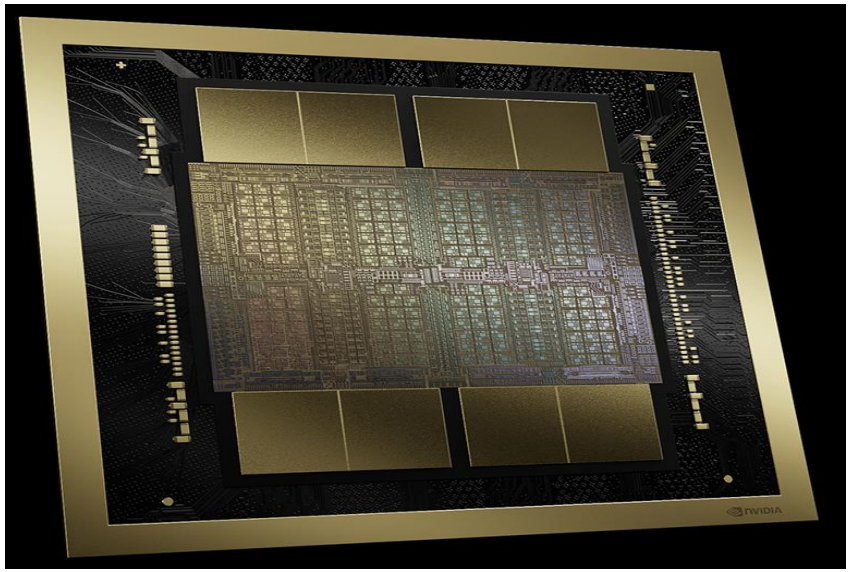






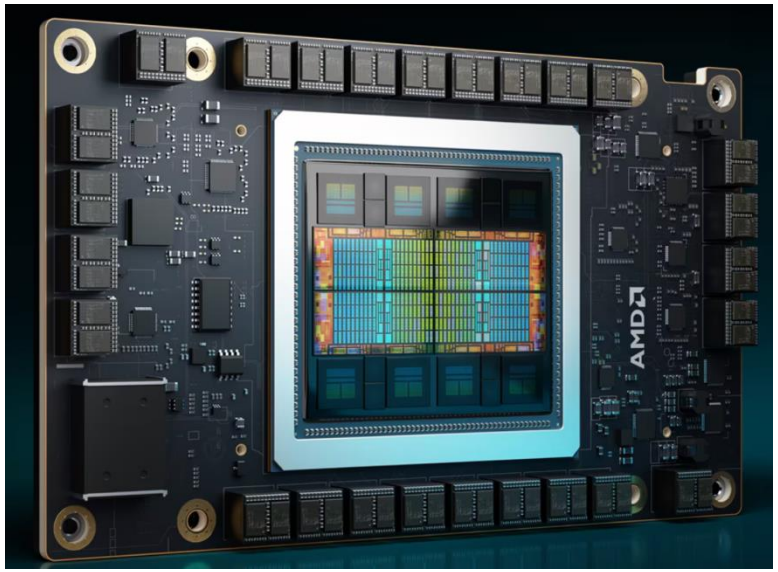
https://www.nvidia.com/en-us/technologies/multi-instance-gpu/

GPUs – Graphic Processing Units



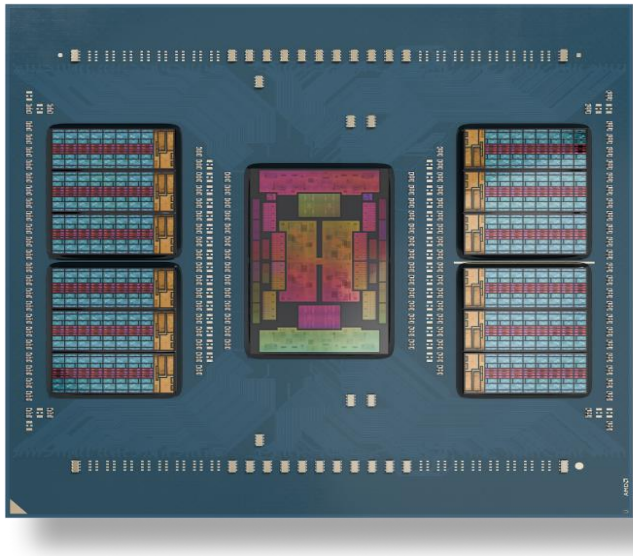
NVIDIA BlackWell GB202 GPU

192 Streaming Multiprocessors (SMs)
24,576 CUDA-cores
92.2 Billion transistors
186GB
8 TB/s
80 TFLOPS (FP32)



AMD MI325 X GPU

304 Compute Units (CUs)
19,456 Stream processors
153 Billion transistors
256 GB
6 TB/s
163.4 TFLOPs (FP32)



AMD EPYC 9005 CPU

12 CPU Dies
192 cores
6TB
614 GB/s

https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf

https://www.amd.com/en/products/accelerators/instinct/mi300/mi325x.html

CPU vs GPU

Traditional CPUs built for:

- Low latency (L1/L2 private cache and shared L3 cache)
- Complex control flow (if, loops, function calls)
- Single-thread speed
- Not ideal for massive parallelism

Modern GPUs built for:

- High throughput
- Thousands of simple, synchronized threads
- Regular, data-parallel workloads (SIMT/SIMD)
- Not ideal for branching or irregular code

Hands-on Example:

Monitoring GPUs with nvidia-smi

Goal: To monitor and interpret real-time GPU performance metrics using **nvidia-smi.**

- Display basic GPU information and status using nvidia-smi.
- Extract specific metrics (e.g., memory, temperature, power) in structured formats for analysis.
- Understand GPU interconnect topology to assess communication efficiency in multi-GPU systems.

Hands-on Example

hpc-course / examples / 02_nvidia_smi / README.md

1.0 Launch an interactive session

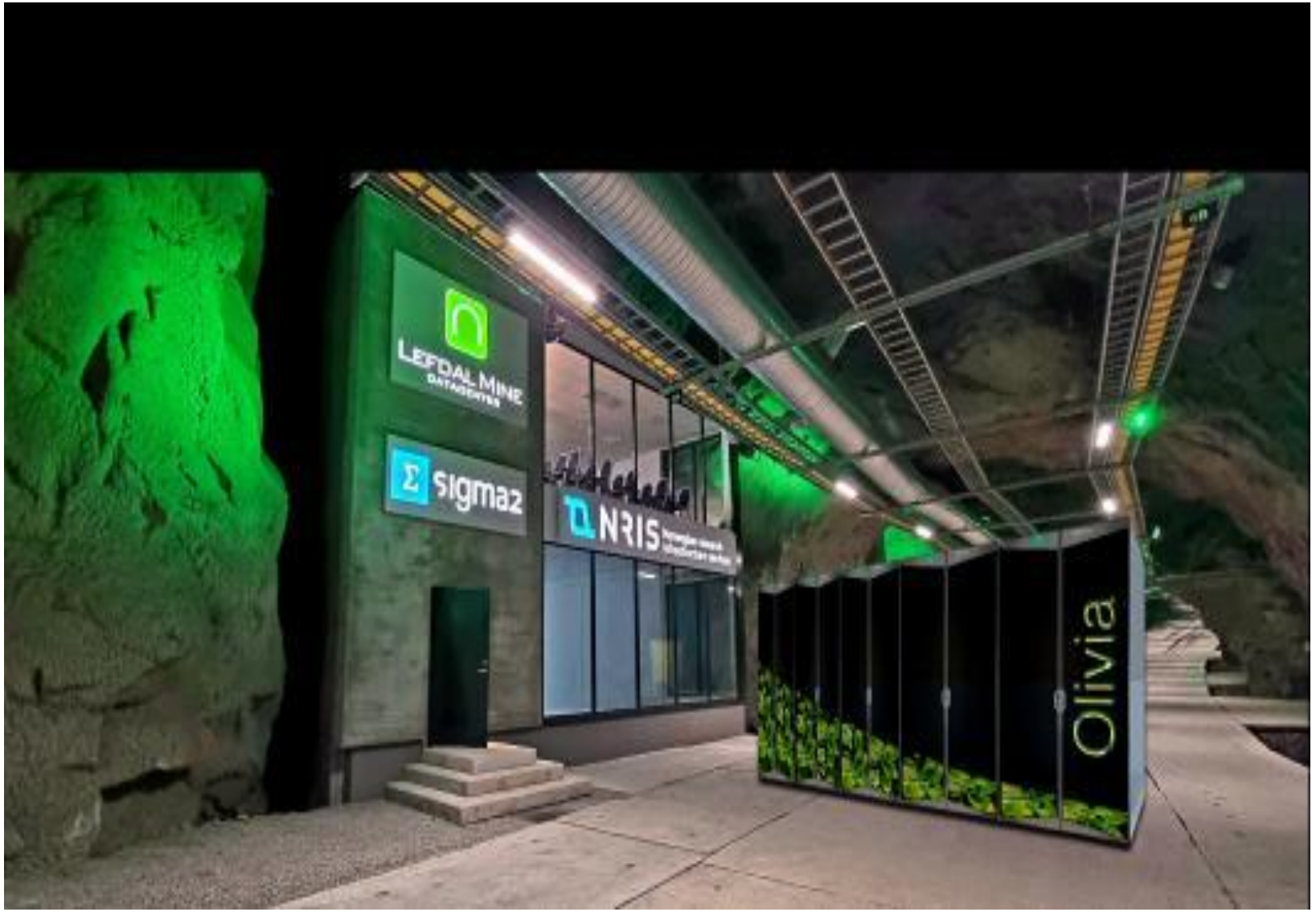
```
srun -A nn14000k -p accel --nodes=1 --gpus=1 --ntasks-per-node=1  --cpus-per-task 1 --mem
```

1.1 GPU-monitoring with nvidia-smi (details below)

```
# 1. All GPUs
nvidia-smi

# 2. Names of GPUs 0,1,2 (CSV, no header)
nvidia-smi -i 0,1,2 --query-gpu=name --format=csv,noheader

# 3. Memory / Temperature / Power (CSV)
nvidia-smi --query-gpu=memory.used,memory.total,temperature.gpu,power.draw \
           --format=csv,noheader

# 4. Export to CSV for later plotting
nvidia-smi --query-gpu=memory.used,memory.total,temperature.gpu,power.draw \
           --format=csv,noheader > gpu_stats.csv

# 5. GPU interconnect topology
nvidia-smi topo -m
```

Norway's New Supercomputer: Olivia

- System: **HPE Cray** Supercomputing EX
- 76 GPU-nodes (**304** GPUs)
- NVIDIA **Grace Hopper Superchip** (**GH200** 96 GB)
- **HPE Slingshot Interconnect**



https://www.sigma2.no/news/2024/olivia-becomes-name-norways-next-supercomputer

Overview of the Grace Hopper System

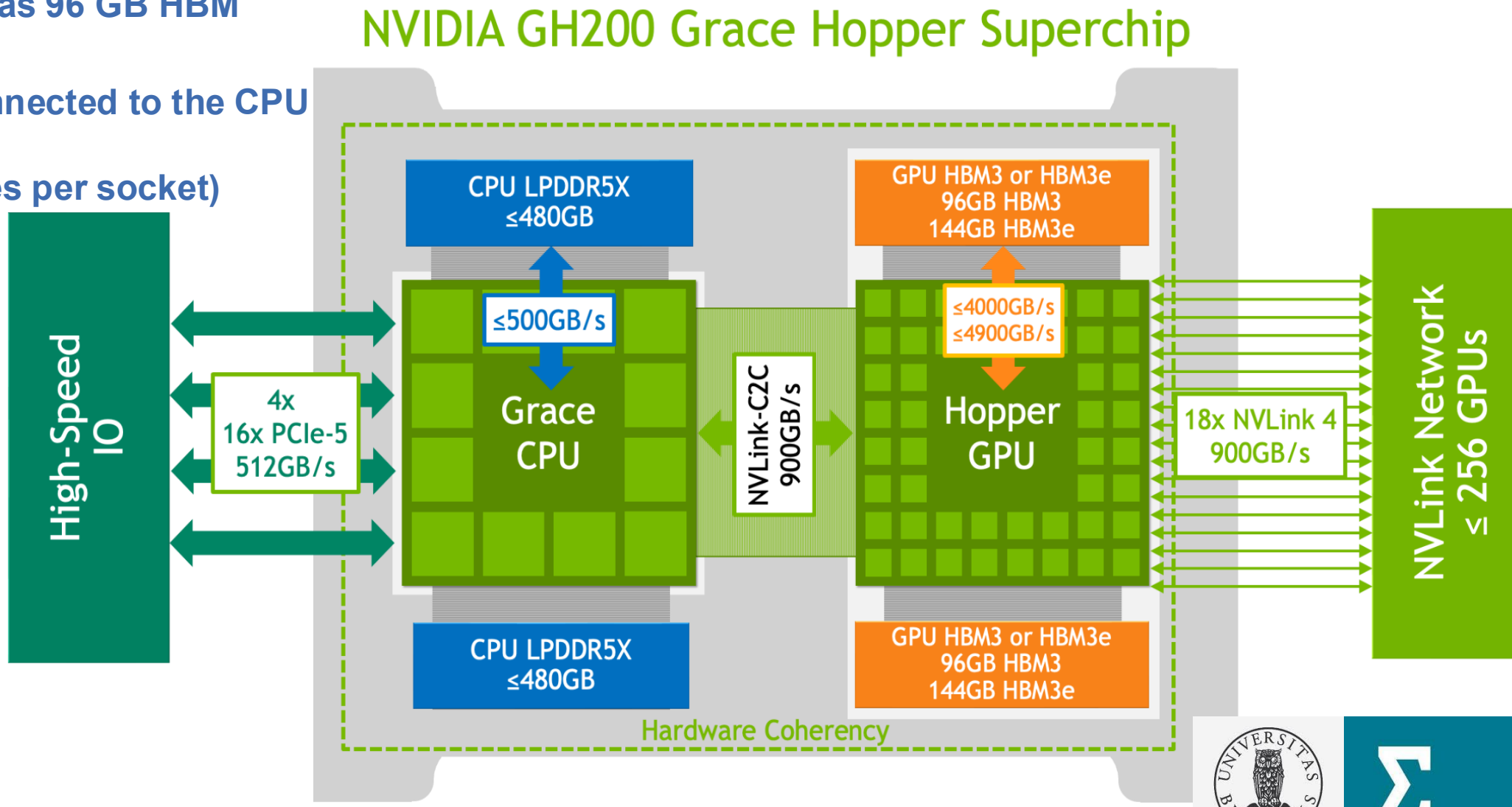
Olivia:

Each GH200 device has 96 GB HBM

120 GB LPDDR5x connected to the CPU

288 cpu-core (72 cores per socket)

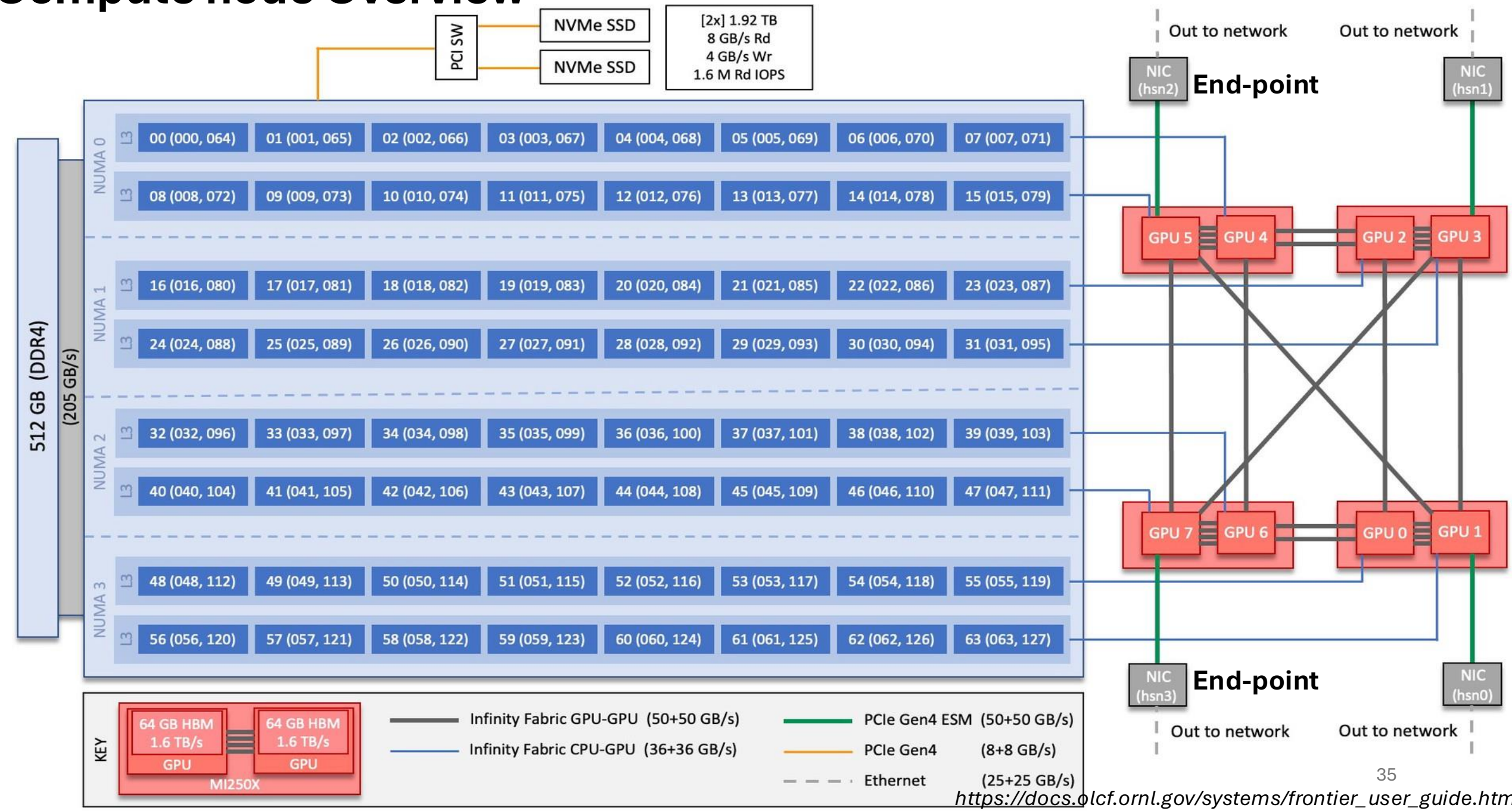
864 GB total memory



https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper

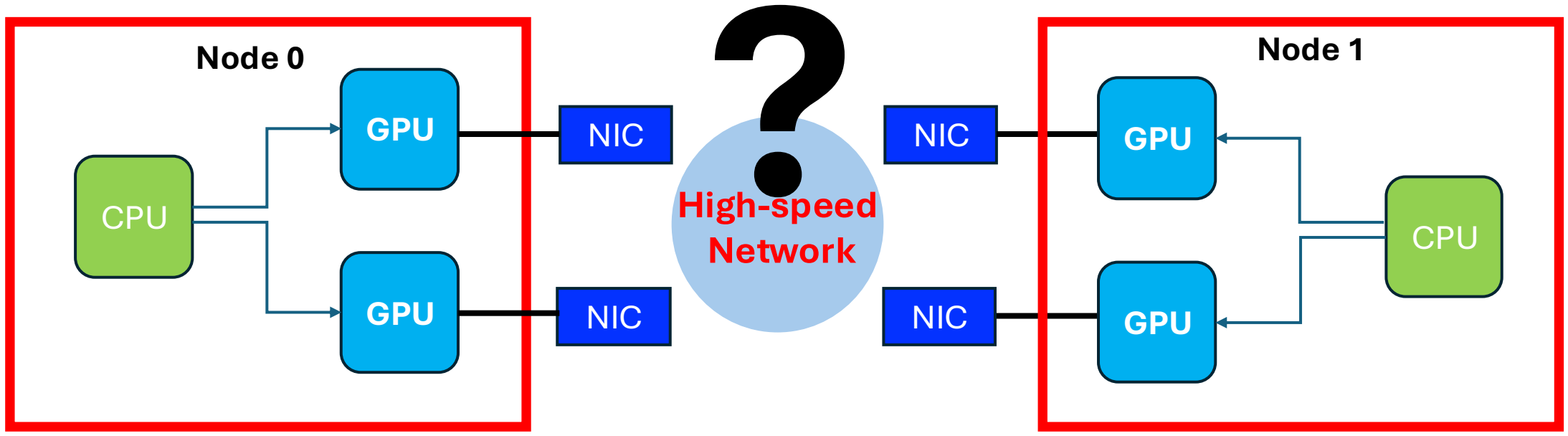
Network Fabric Architecture

Compute node Overview



https://docs.olcf.ornl.gov/systems/frontier_user_guide.htm

Network Interface Card - NIC



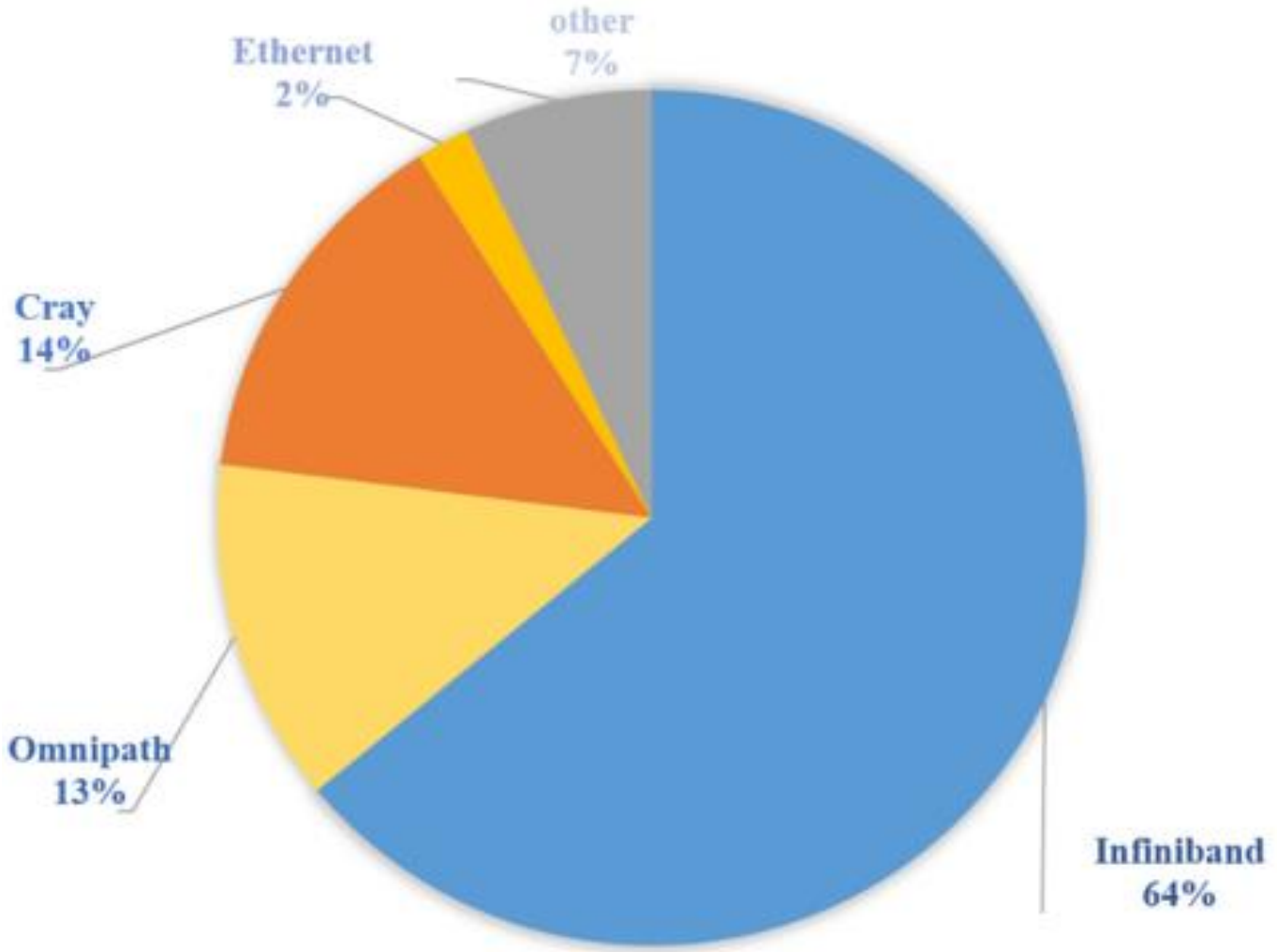
How does a compute node connect to thousands of other compute nodes in an HPC system ??

Challenge: Design a high-speed network architecture that:

- **Maximizes bandwidth**: Enable fast data transfer.
- **Minimizes latency**: Achieve near-instantaneous communication.
- **Ensures scalability:** Supporting hundreds of thousands of endpoints without significant performance loss.

Requires High-Performance Interconnects

high-performance interconnects - HPIs

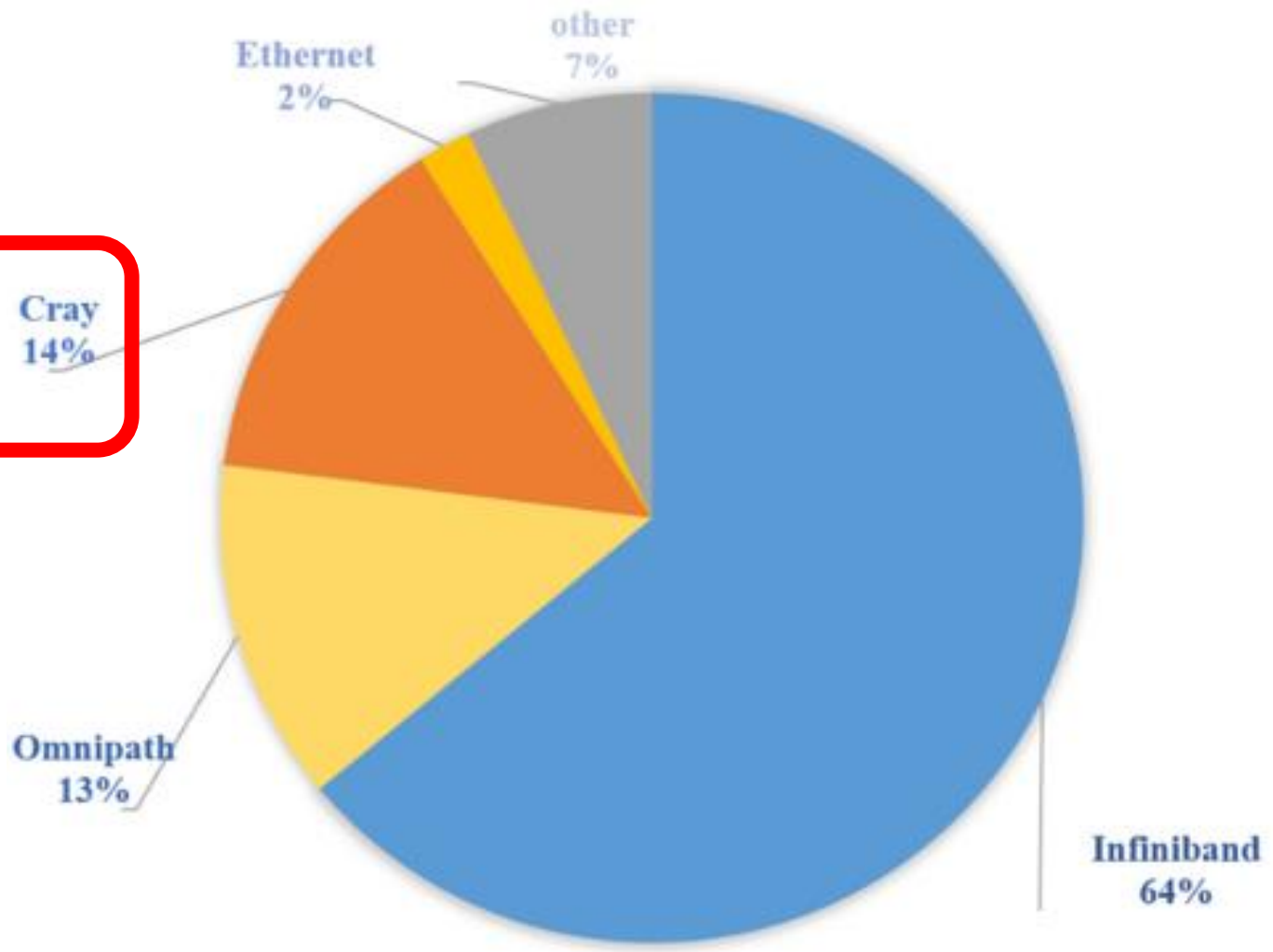


Distribution map of HPIs in the Top100 as of Nov 2021

Electronics 2022, **11**, 1369 10.3390/electronics11091369

high-performance interconnects - HPIs

HPE Slingshot Network



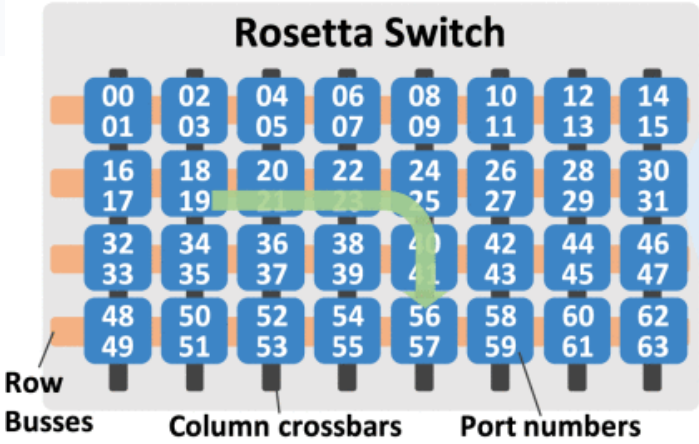
Distribution map of HPIs in the Top100 as of Nov 2021

Electronics 2022, **11**, 1369 10.3390/electronics11091369

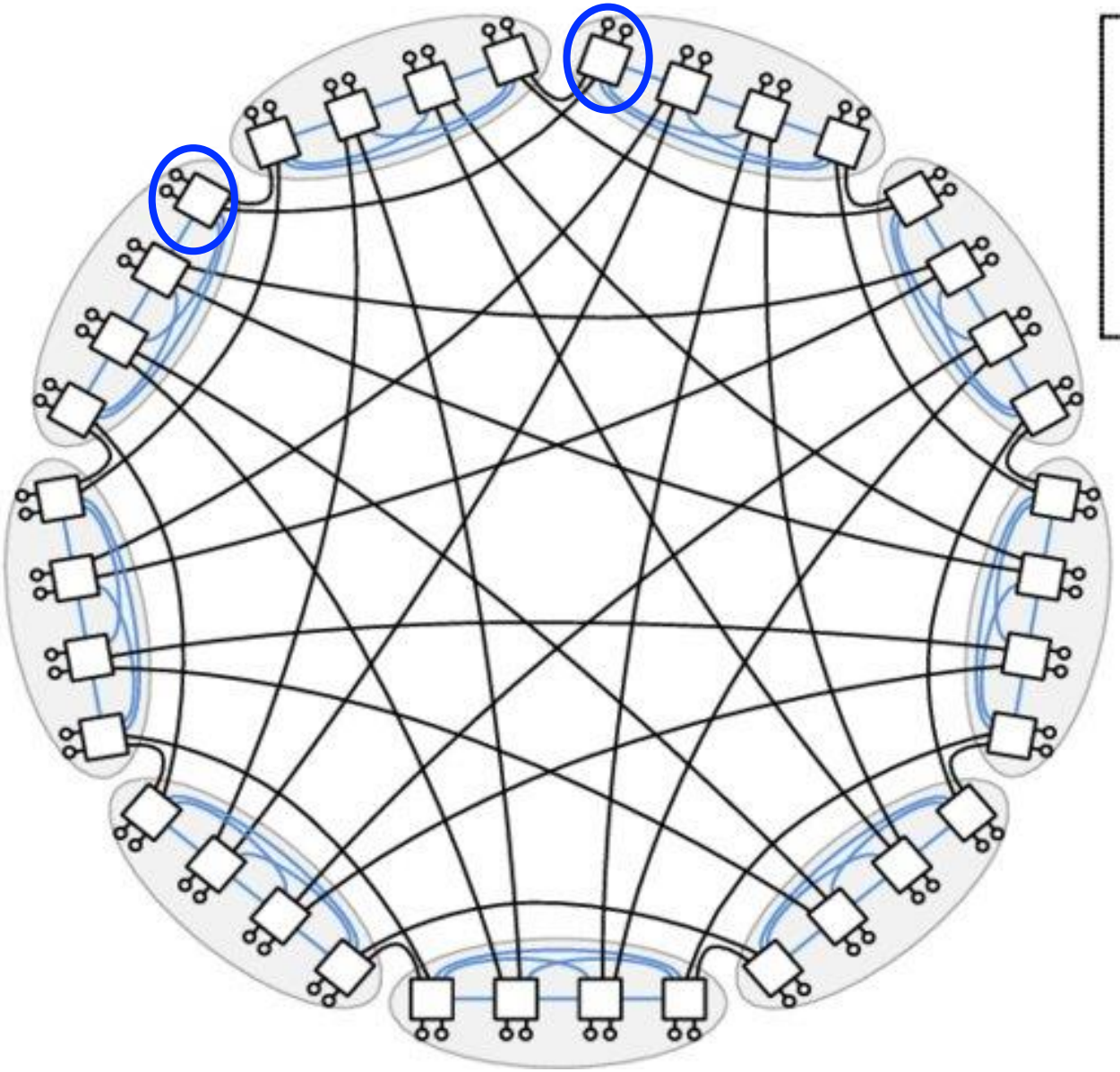
HPE Slingshot interconnects

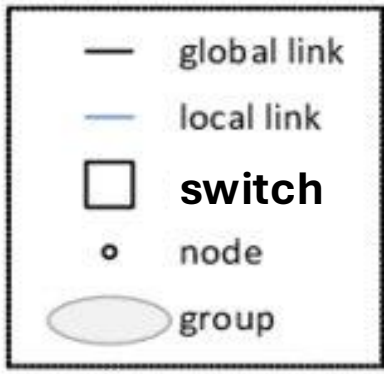
- **HPE Slingshot networks** are constructed from two components:
  - **Cassini – a** PCIe Gen4 Network Interface cards (NIC)
  - **Rosetta – a** 64-port **200 Gbps Ethernet switch** (Slingshot Switch)

- HPE Slingshot uses **Dragonfly topology to ensure**
  - High-bandwidth
  - Low latency
  - Scalability
  - Cost effectiveness



Dragonfly topology: Example of 72 end-points (compute nodes)





Dragonflies are organized as groups of switches.

Switches form a network that allows any node to reach any other node.

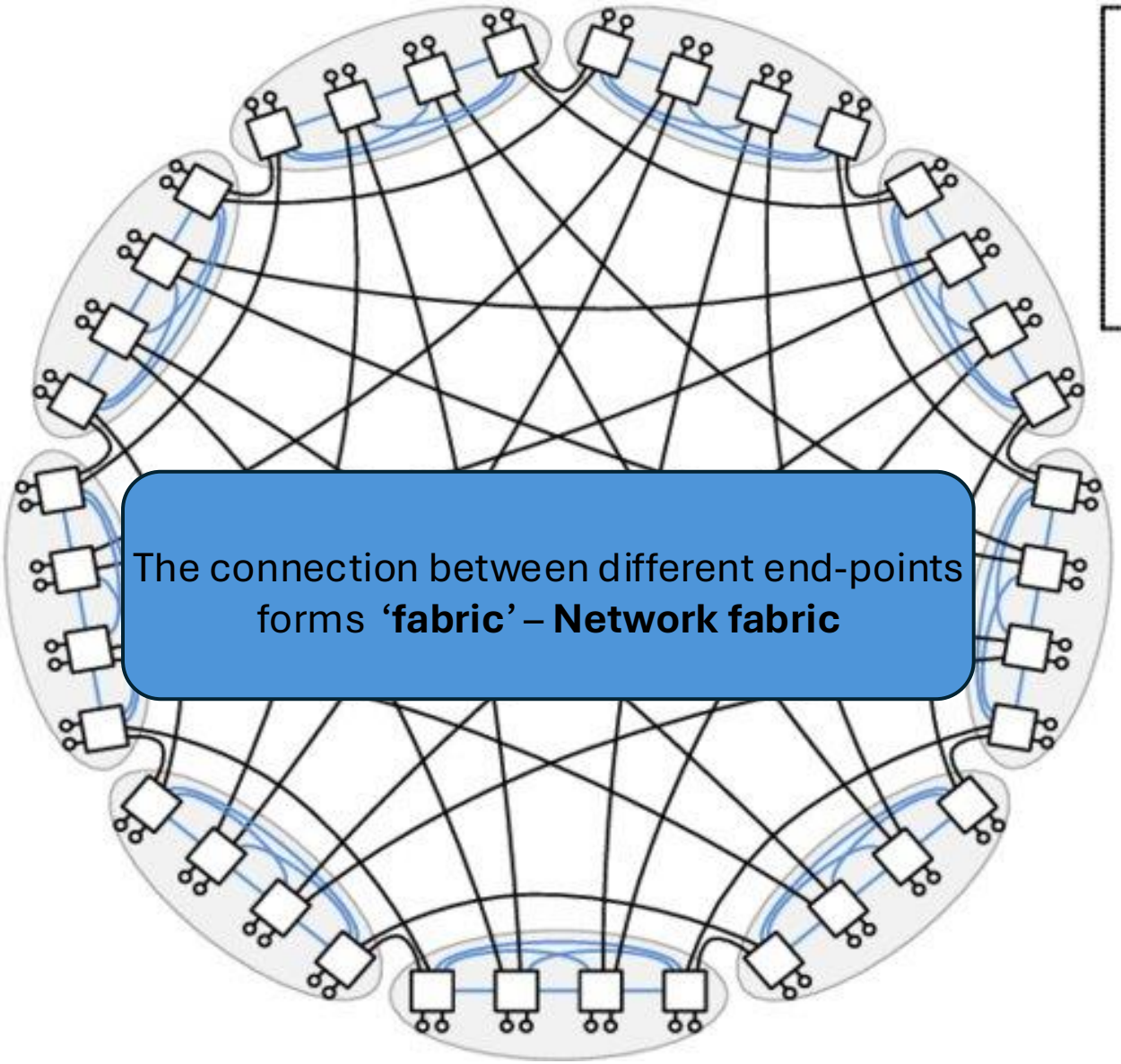
Ex. 9 groups
Each has 4 switches
Each switch connects 2 compute nodes
Total of 9x4x2 = **72 end-points (compute nodes)**

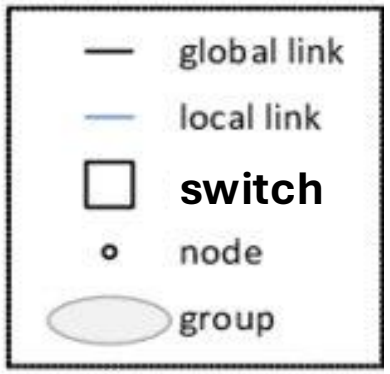
A compute node can communicate with any other node in at most 3 hops (steps)

1. **Node A** (Source) → **Switch 1** (1st hop)
2. **Switch 1** → **Switch 2** (2nd hop)
3. **Switch 2** → **Node B** (Destination)

Figure adapted from: https://ieeexplore.ieee.org/document/6337589

Dragonfly topology: Example of 72 end-points (compute nodes)





Dragonflies are organized as groups of switches.

Switches form a network that allows any node to reach any other node.

Ex. 9 groups
Each has 4 switches
Each switch connects 2 compute nodes
Total of 9x4x2 = **72 end-points (compute nodes)**

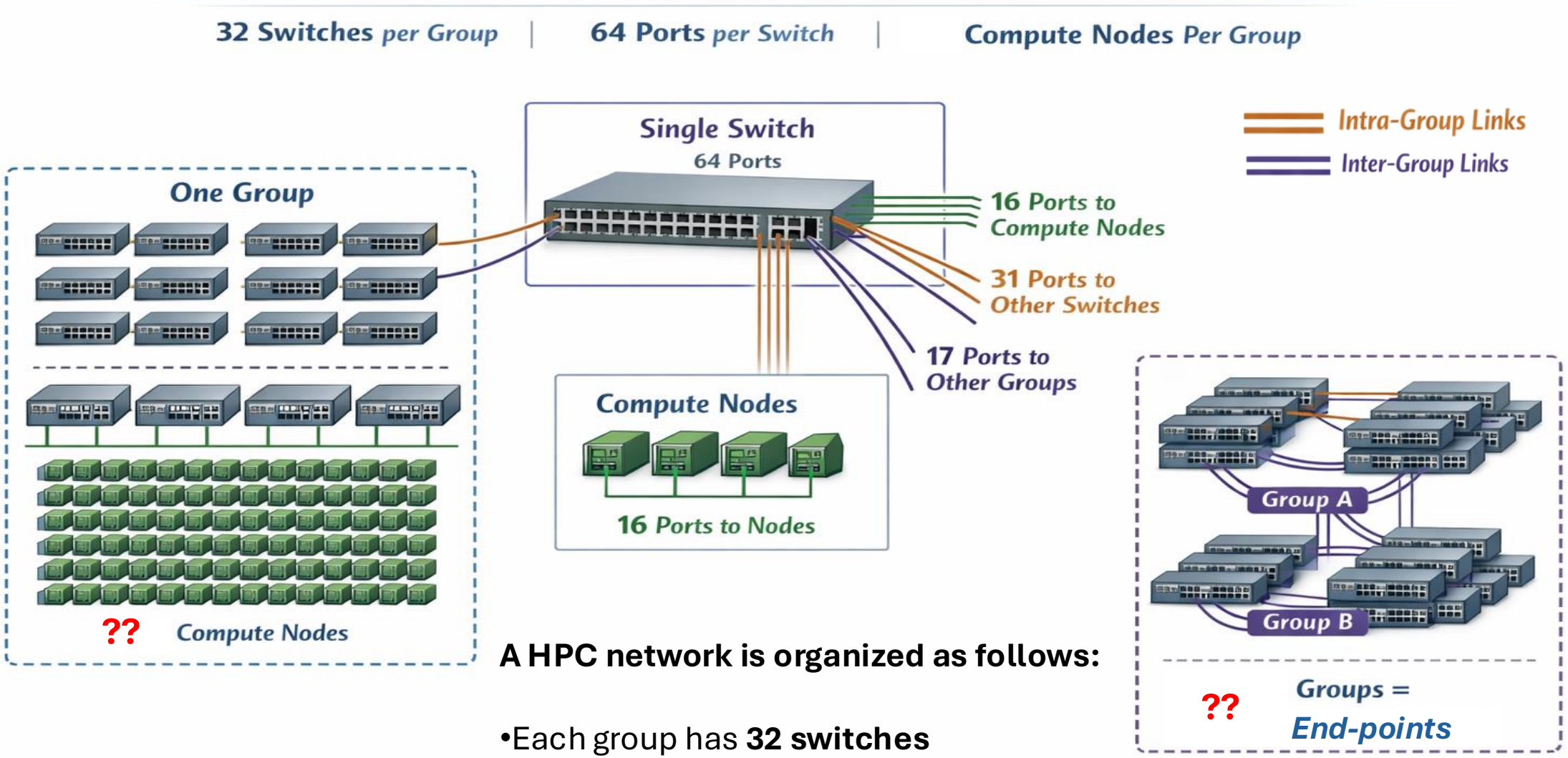
A compute node can communicate with any other node in at most 3 hops (steps)

1. **Node A** (Source) → **Switch 1** (1st hop)
2. **Switch 1** → **Switch 2** (2nd hop)
3. **Switch 2** → **Switch 3** (3rd hop)
4. **Switch 3** → **Node B** (Destination)

Figure adapted from: https://ieeexplore.ieee.org/document/6337589

Quiz:

HPC Network Structure



A HPC network is organized as follows:

- Each group has **32 switches**
- Each switch has **64 physical ports**.
- **16 ports** of each switch are used to connect to compute nodes.
- **31 ports** are used to connect switches to each other.
- **17 ports** are used to connect to other groups.

Quiz: Understanding the HPC Network Structure

Question 1
Each switch has **16 ports** connected to compute nodes.
If one group contains **32 switches**, how many compute nodes are in one group?

A) 256
B) 512
C) 544
D) 1024

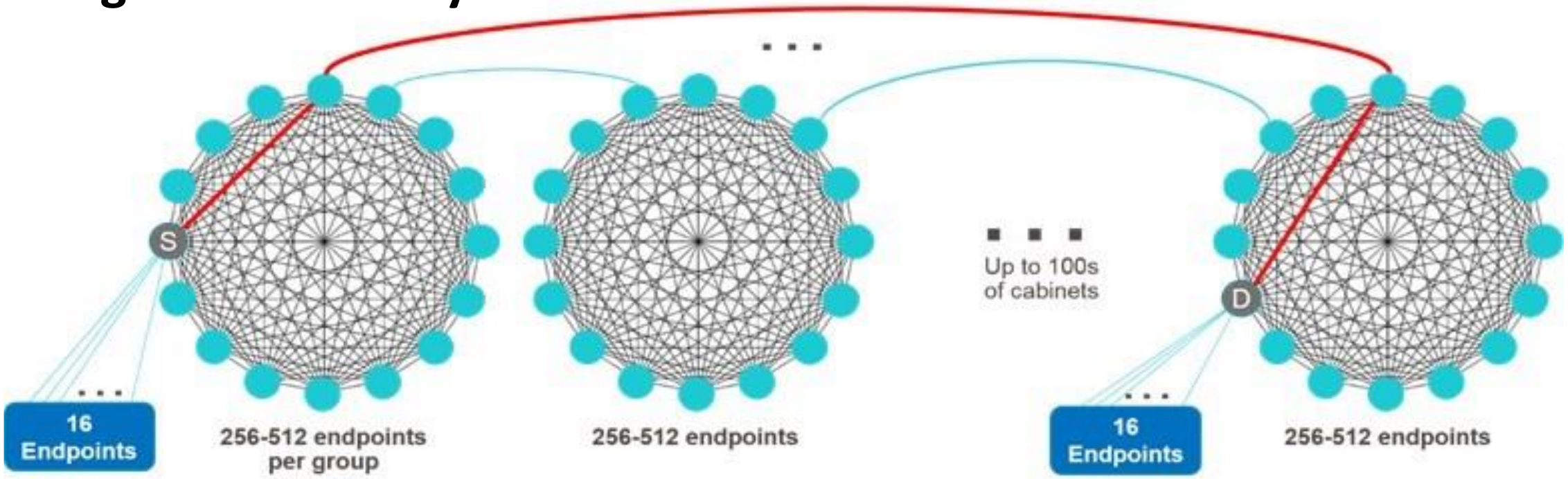
Question 2
Each group contains **512 compute nodes**.
If the system has **545 groups**, how many compute nodes are there in total?

A) 256,000
B) 279,040
C) 544,512
D) 1,024,000

Question 3
If each compute node requires **4 NICs**,
how many compute nodes can the system support in total?

A) 69,760
B) 139,520
C) 279,040
D) 545,000

HPE Slingshot scalability



Each group: **32 switches** each switch has **64 physical ports**

16 ports of each switch are used to connect to compute nodes

32x16 = 512 nodes on each group

31 ports are used to connect switches to each others

17 ports to connect to other groups

How many groups: **32 switches x 17 = 544 groups**

A system having **512 nodes x 545 groups** = **279 040** endpoints

HPE Slingshot Launched into Network Space: https://cug.org/proceedings/cug2022_proceedings/includes/files/pap121s2-file1.pdf
An In-Depth Analysis of the Slingshot Interconnect: http://ww.w.unixer.de/publications/img/sensi-slingshot.pdf

Parallel File System - LUSTRE

What's LUSTRE ?

- **Lustre** is parallel **filesystem** software platform designed for **HPC** systems.
- **Lustre** primarily focuses on the **management** of storage devices and file systems.
- Allows **multiple** clients (login nodes, compute nodes, …) to **read/write** data **simultaneously**.
- All clients see a **unified filesystem** (concurrent **coherent** access to files).
- **Lustre** differentiates between **servers that store data** and **servers that store metadata**

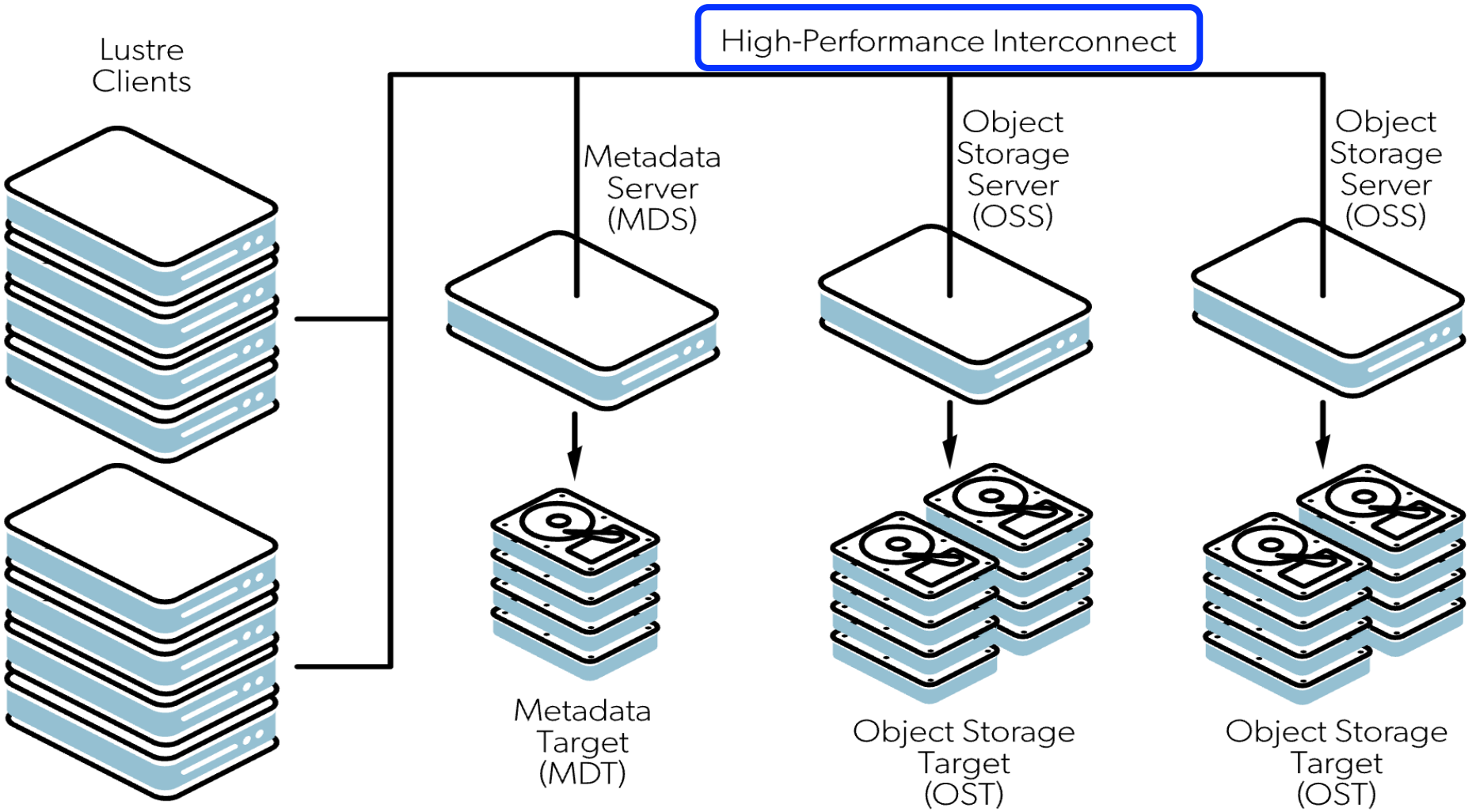
  (e.g. filename, directories, etc)
- **Key feature**: Ability to **scale** both **storage** capacity and **bandwidth** independently.

  Support hundred's of **PB of data storage** and hundreds of **GB/s** simultaneously.

https://wiki.lustre.org/Main_Page

Intel: https://www.lustre.org/wp-content/uploads/architecting-lustre-storage-white-paper.pdf

Architecture of Lustre



Metadata Server (MDS):
manages access to MDT
Handles metadata operations
e.g. file creation, deletion, etc
Each MDS is responsible of one or more MDTs

Metadata Targets (MDT):
stores metadata information
filesystem namespace information (directories, filenames, permissions etc.)

Object Storage Servers (OSS)

Object Storage Targets (OST):
hosts a local file system
stores the actual data
(file content).

Clients: Login nodes, compute nodes
(access global file system)

https://docs.lumi-supercomputer.eu/storage/parallel-filesystems/lustre/

Lustre Capabilities

Lustre is designed for large amounts of data

Storage System Requirements	Lustre File System Capabilities
Large file system	Up to 512 PB for one file system.
Large files	Up to 32 PB for one file.
Global name space	A consistent abstraction of all files allows users to access file system information heterogeneously.
High throughput	2 TB/s in a production system. Higher throughput being tested.
Many files	Up to 10 million files in one directory and 2 billion files in the file system. Virtually unlimited with Distributed Name Space.
Large number of clients accessing the file system in parallel	Up to 25,000+ clients in a production system.
High metadata operation rate	Support for 80,000/s create operations and 200,000/s metadata stat operations.

Intel: https://www.lustre.org/wp-content/uploads/architecting-lustre-storage-white-paper.pdf

Storage System Overview: Example supercomputer LUMI

Divide storage into several filesystems: **/scratch; /flash; /home; /project**; etc
Spaces can be physically or logically separated

Storage Type	Filesystem	Capacity	Bandwidth	Metadata Servers (MDSs)	Object Storage Targets (OSTs)	Storage Type
LUMI-P (Main Storage)	Lustre	20 PB	240 GB/s	1 MDS	32 OSTs	HDD (Spinning Disks)
LUMI-F (Flash Storage)	Lustre	8 PB	1,740 GB/s	2 MDSs	58 OSTs	SSD (Solid-State Drives)

filesystem summary

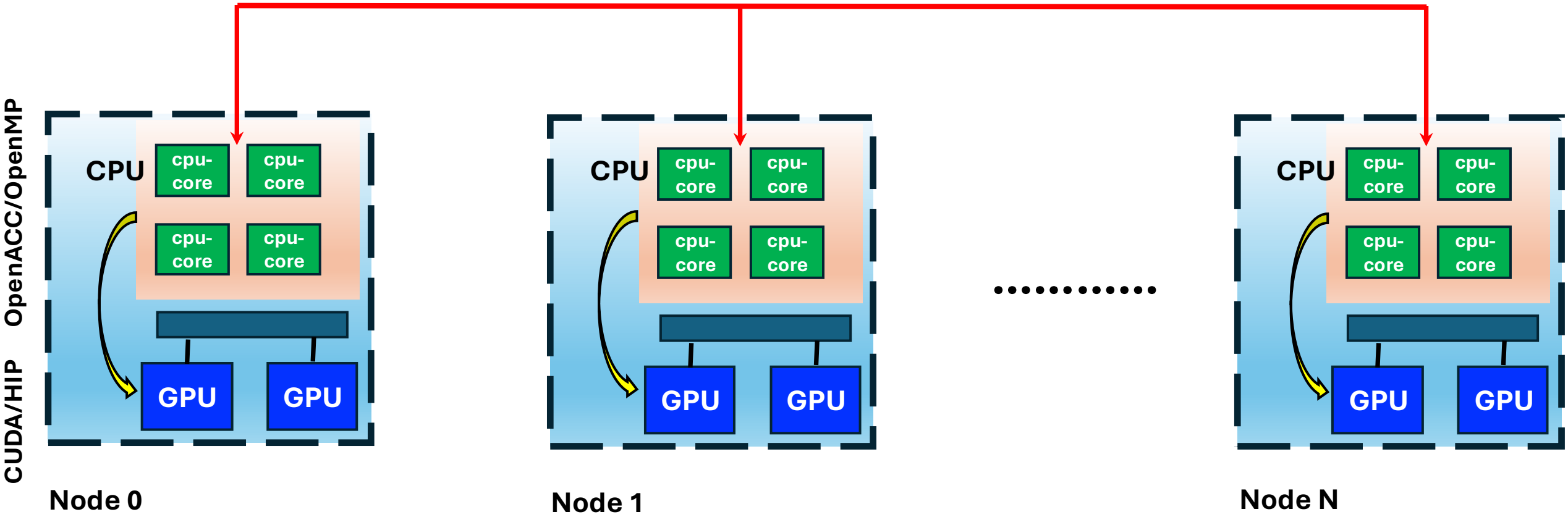
$ lfs osts

```
hiagueny@uan04:~> lfs df -h | grep 'filesystem_summary'
filesystem_summary:          8.5P        1.5P         6.9P  19% /pfs/lustref1
filesystem_summary:         18.2P        4.4P        13.6P  25% /pfs/lustrep1
filesystem_summary:         18.2P        4.6P        13.4P  26% /pfs/lustrep2
filesystem_summary:         18.2P        7.7P        10.3P  43% /pfs/lustrep3
filesystem_summary:         18.2P        4.7P        13.3P  26% /pfs/lustrep4
```

capacity used available use%

GPUDirect Technology

Hybrid MPI + GPU-model



GPUDirect Technology

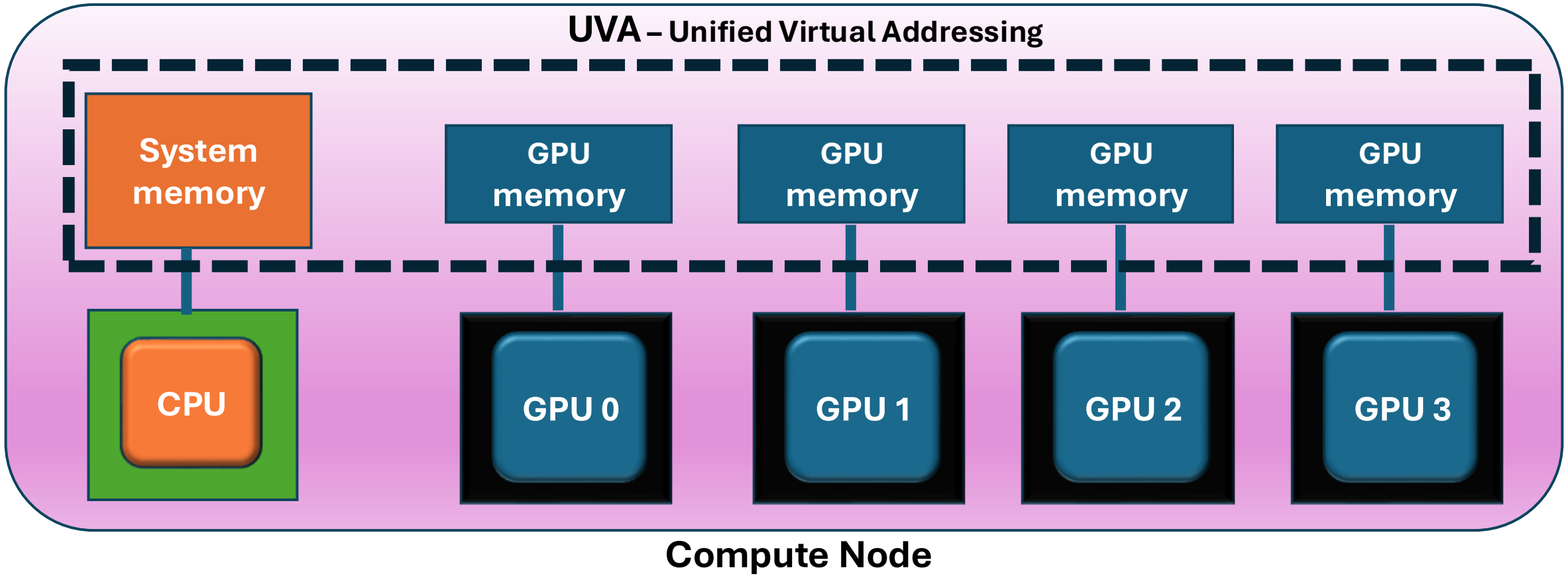
In GPU-aware MPI concept:

MPI library can directly access the GPU memory without necessarily passing by the CPU memory.

Takes advantage of GPUDirect Technology (RDMA and Peer-To-Peer)

Device pointers are passed as arguments to an MPI routine

Single node: MPI with GPU awareness

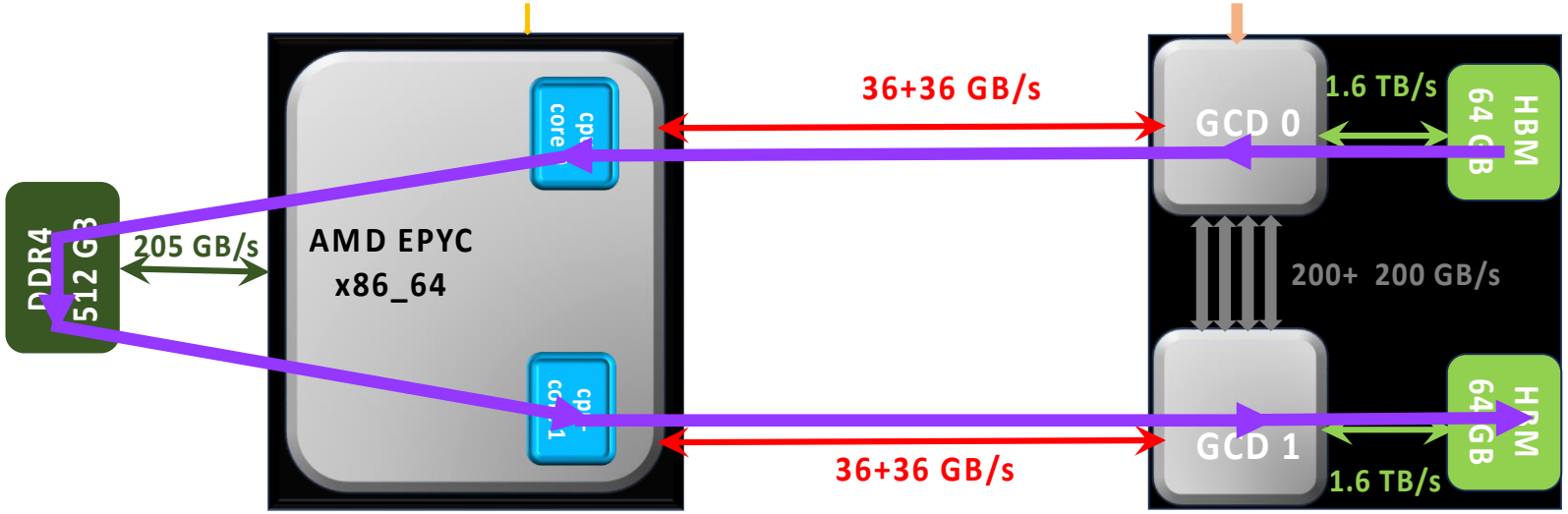


- Device pointers are passed as arguments to an MPI routine
- MPI library will detect that the pointer is a device pointer
- Unified Virtual Addressing (UVA): single virtual memory system for all memory (GPUs, CPU)
- Direct GPU-to-GPU communication

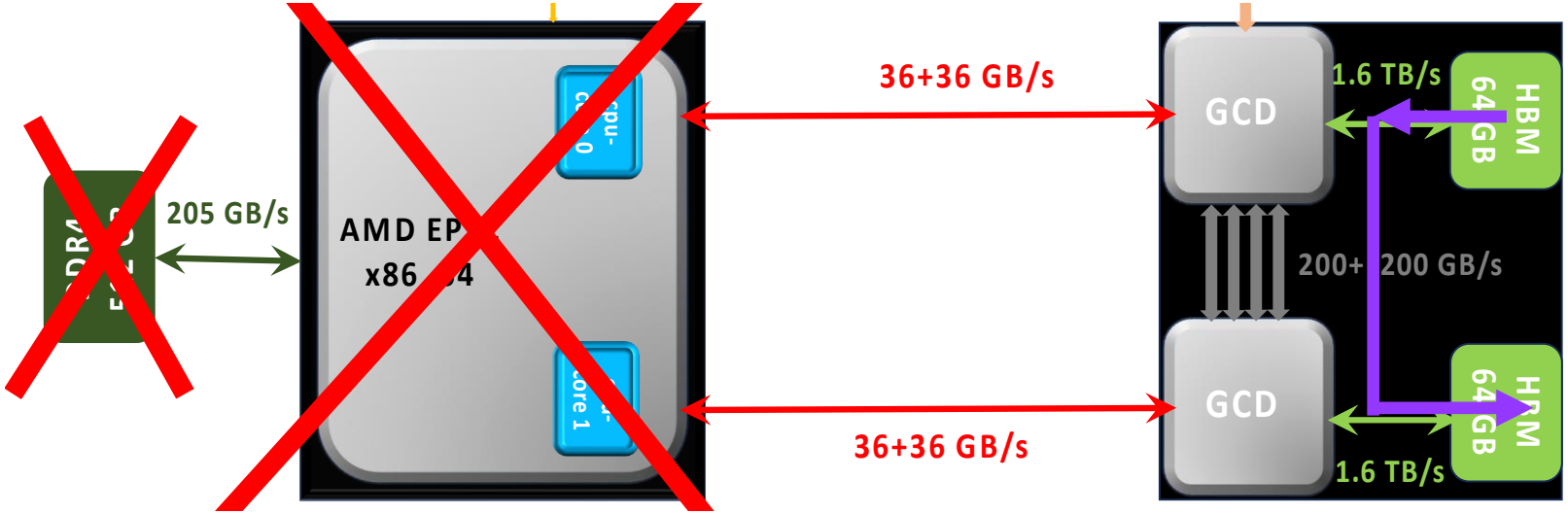
Adapted from: https://developer.nvidia.com/blog/introduction-cuda-aware-mpi/

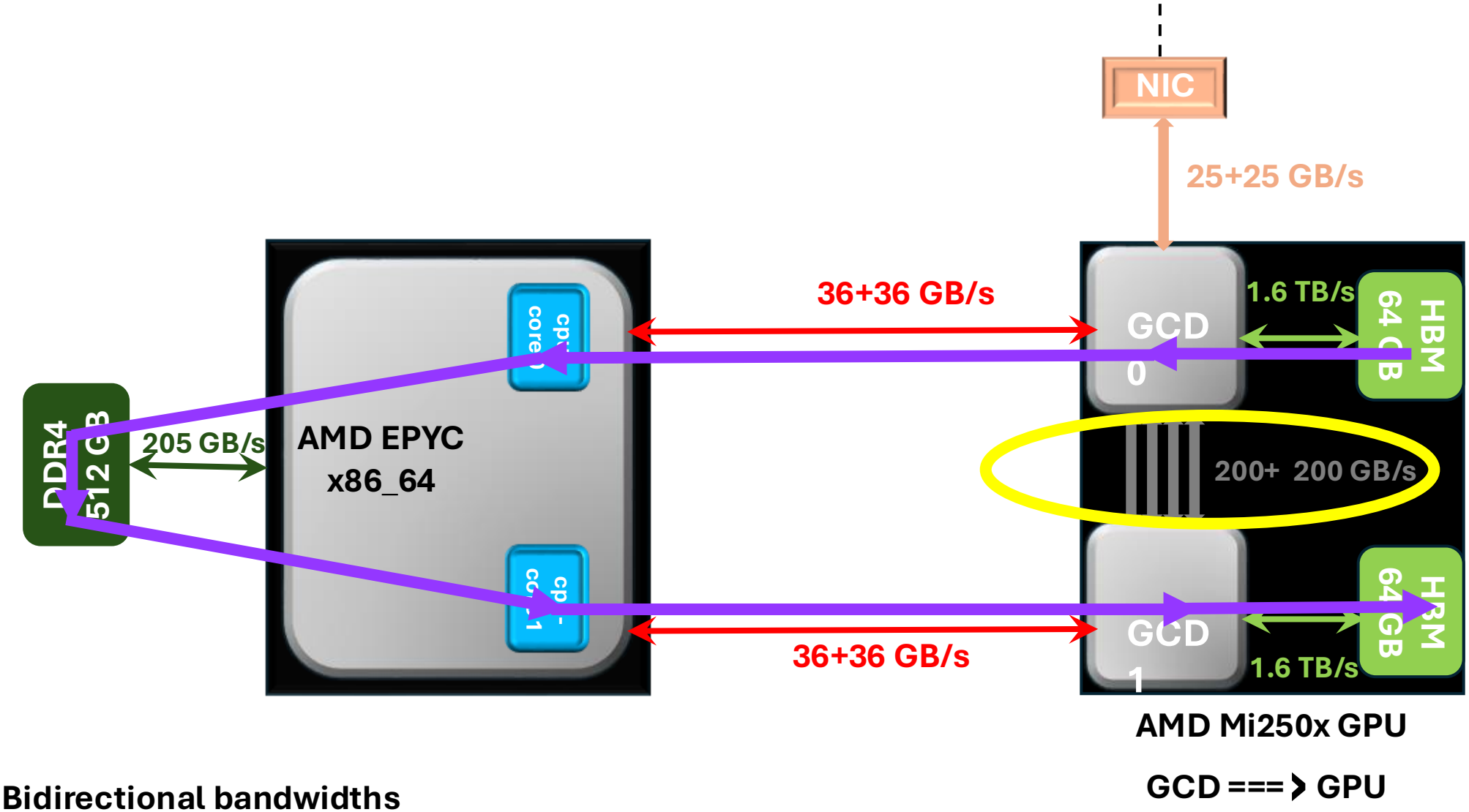
GPU-aware feature

GPU-aware disabled



GPU-aware enabled

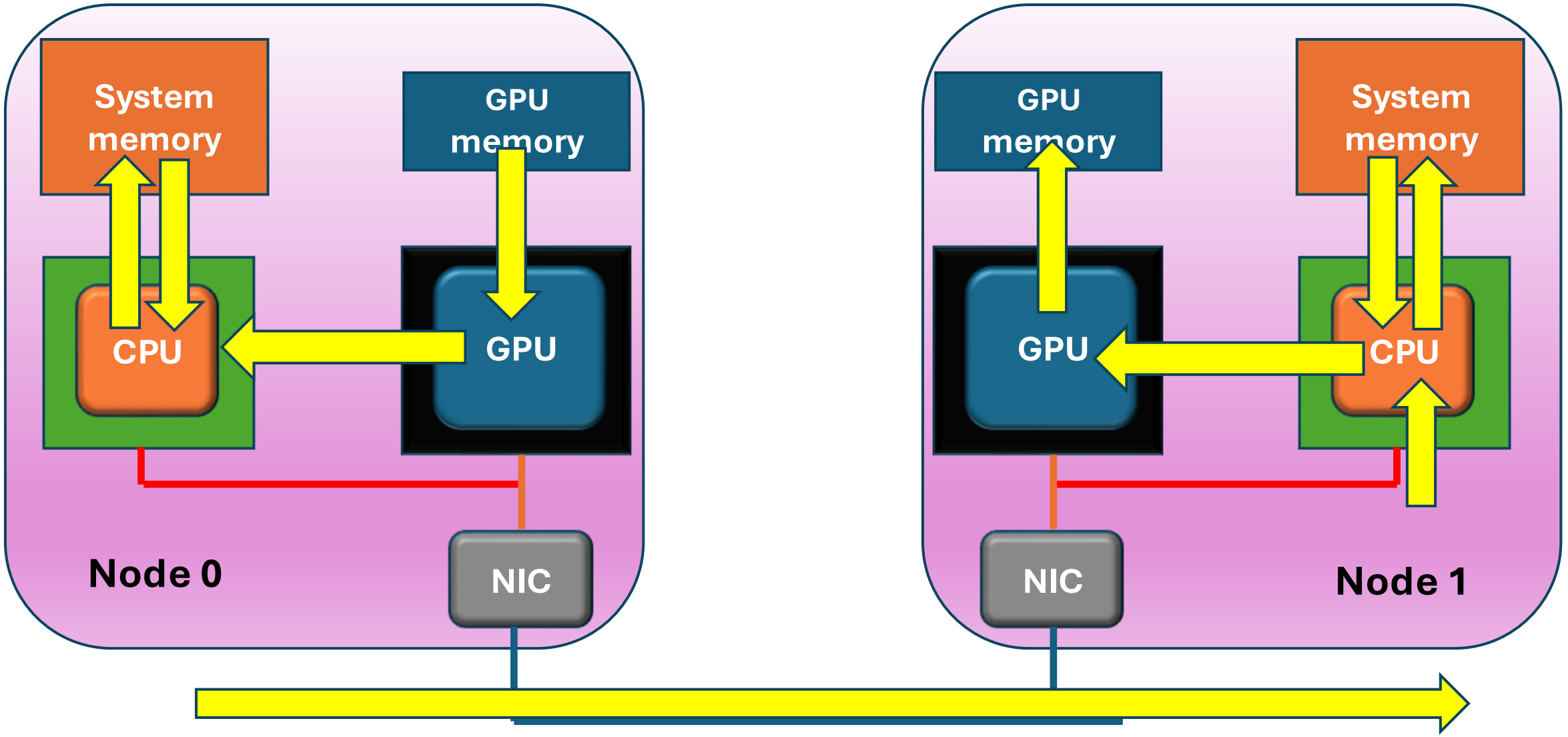




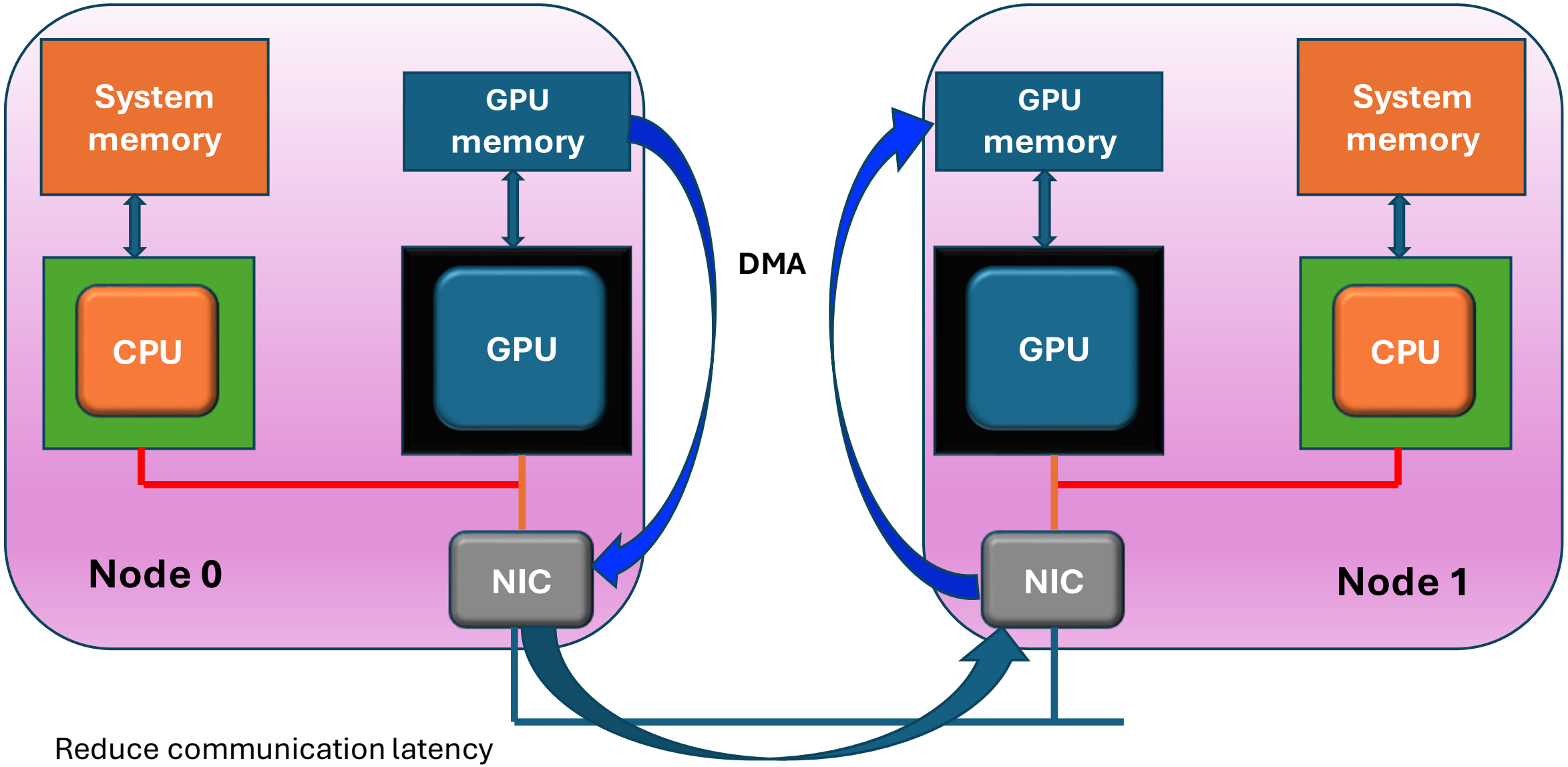
Bidirectional bandwidths

Infinity Fabric	**36 + 36**	**GB/s**
Infinity Fabric	**200 + 200**	**GB/s**
PCIe Gen4 ESM	**50 + 50**	**GB/s**
Ethernet	**25 + 25**	**GB/s**

Multiple nodes: Traditional way of transferring data



Multiple nodes: GPUDirect RDMA (GDR)



Reduce communication latency
Improve performance

Hands-on Example:

Measuring Bandwidth

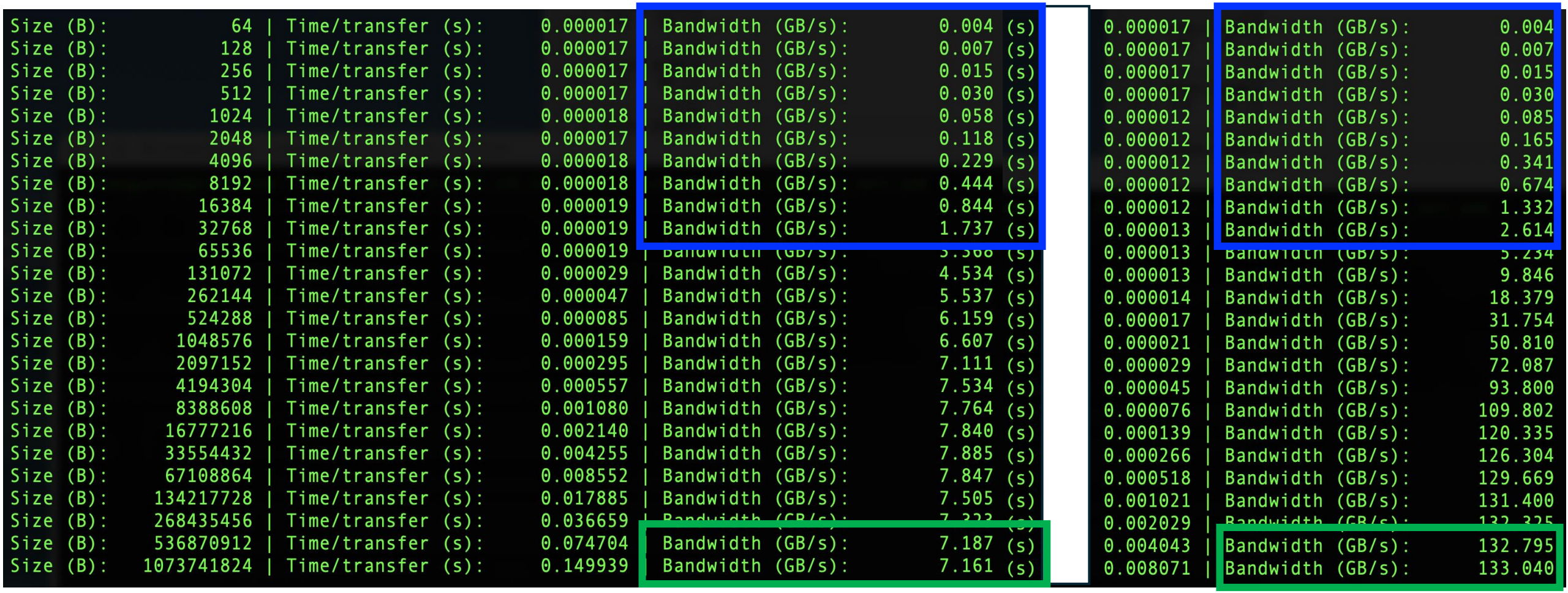
Goal: Evaluate the performance difference between **GPU-aware MPI** and **host-mediated transfers** for **GPU-to-GPU communication**.

- Measure and compare data transfer times between GPUs using GPU-aware MPI versus host-staged transfers.
- Understand the performance impact of direct GPU-to-GPU communication versus CPU-mediated transfers in MPI.

Measruing the bandwidth: 2 MPI processes and 2 GPUs, a single Node

Host-mediated transfer

GPU-aware feature enabled



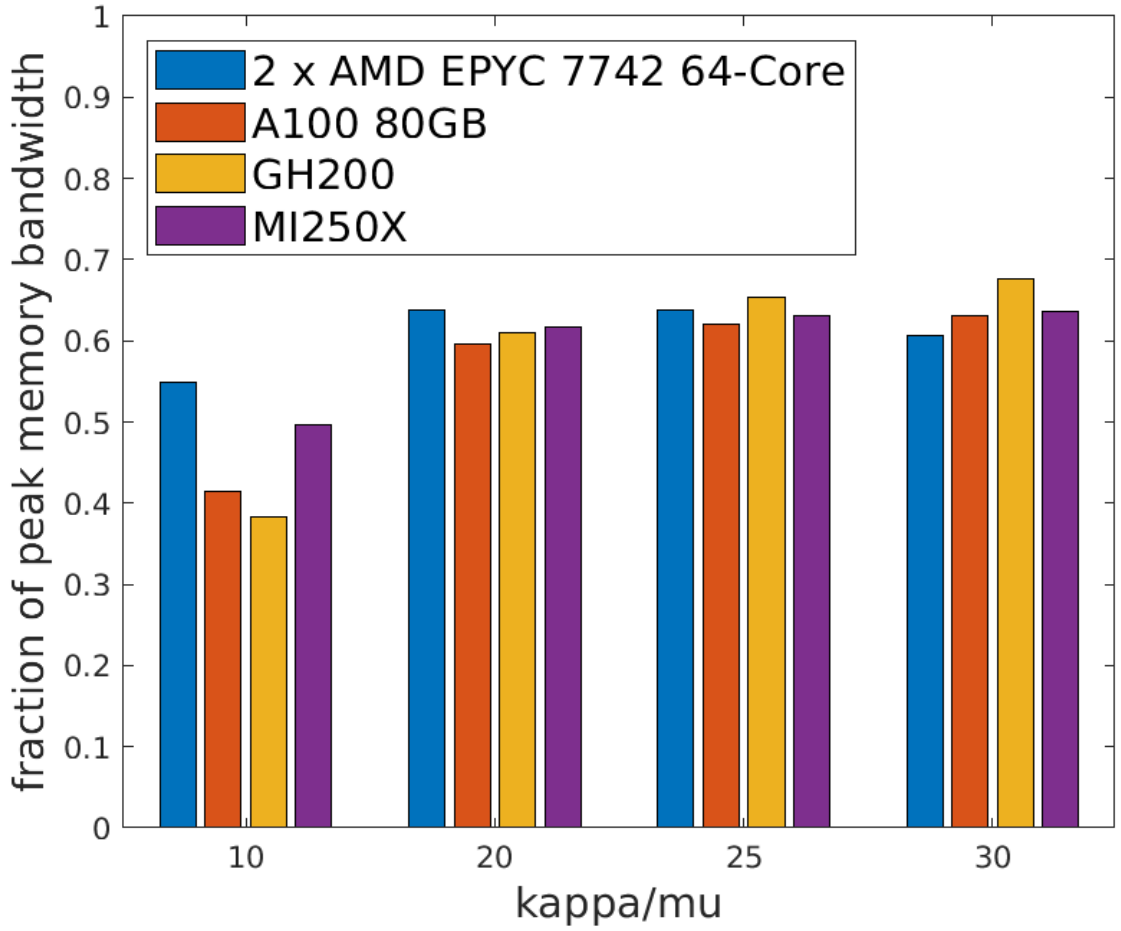
Size (B):	Time/transfer (s):	Bandwidth (GB/s):	Time/transfer (s):	Bandwidth (GB/s):
64	0.000017	0.004 (s)	0.000017	0.004
128	0.000017	0.007 (s)	0.000017	0.007
256	0.000017	0.015 (s)	0.000017	0.015
512	0.000017	0.030 (s)	0.000017	0.030
1024	0.000018	0.058 (s)	0.000012	0.085
2048	0.000017	0.118 (s)	0.000012	0.165
4096	0.000018	0.229 (s)	0.000012	0.341
8192	0.000018	0.444 (s)	0.000012	0.674
16384	0.000019	0.844 (s)	0.000012	1.332
32768	0.000019	1.737 (s)	0.000013	2.614
65536	0.000019	[illegible] (s)	0.000013	[illegible]
131072	0.000029	4.534 (s)	0.000013	9.846
262144	0.000047	5.537 (s)	0.000014	18.379
524288	0.000085	6.159 (s)	0.000017	31.754
1048576	0.000159	6.607 (s)	0.000021	50.810
2097152	0.000295	7.111 (s)	0.000029	72.087
4194304	0.000557	7.534 (s)	0.000045	93.800
8388608	0.001080	7.764 (s)	0.000076	109.802
16777216	0.002140	7.840 (s)	0.000139	120.335
33554432	0.004255	7.885 (s)	0.000266	126.304
67108864	0.008552	7.847 (s)	0.000518	129.669
134217728	0.017885	7.505 (s)	0.001021	131.400
268435456	0.036659	[illegible] (s)	0.002029	[illegible]
536870912	0.074704	7.187 (s)	0.004043	132.795
1073741824	0.149939	7.161 (s)	0.008071	133.040

Speed of transferring 1 GB of data between 2 MPI-processes is about 7 GB/s .....Too slow!!

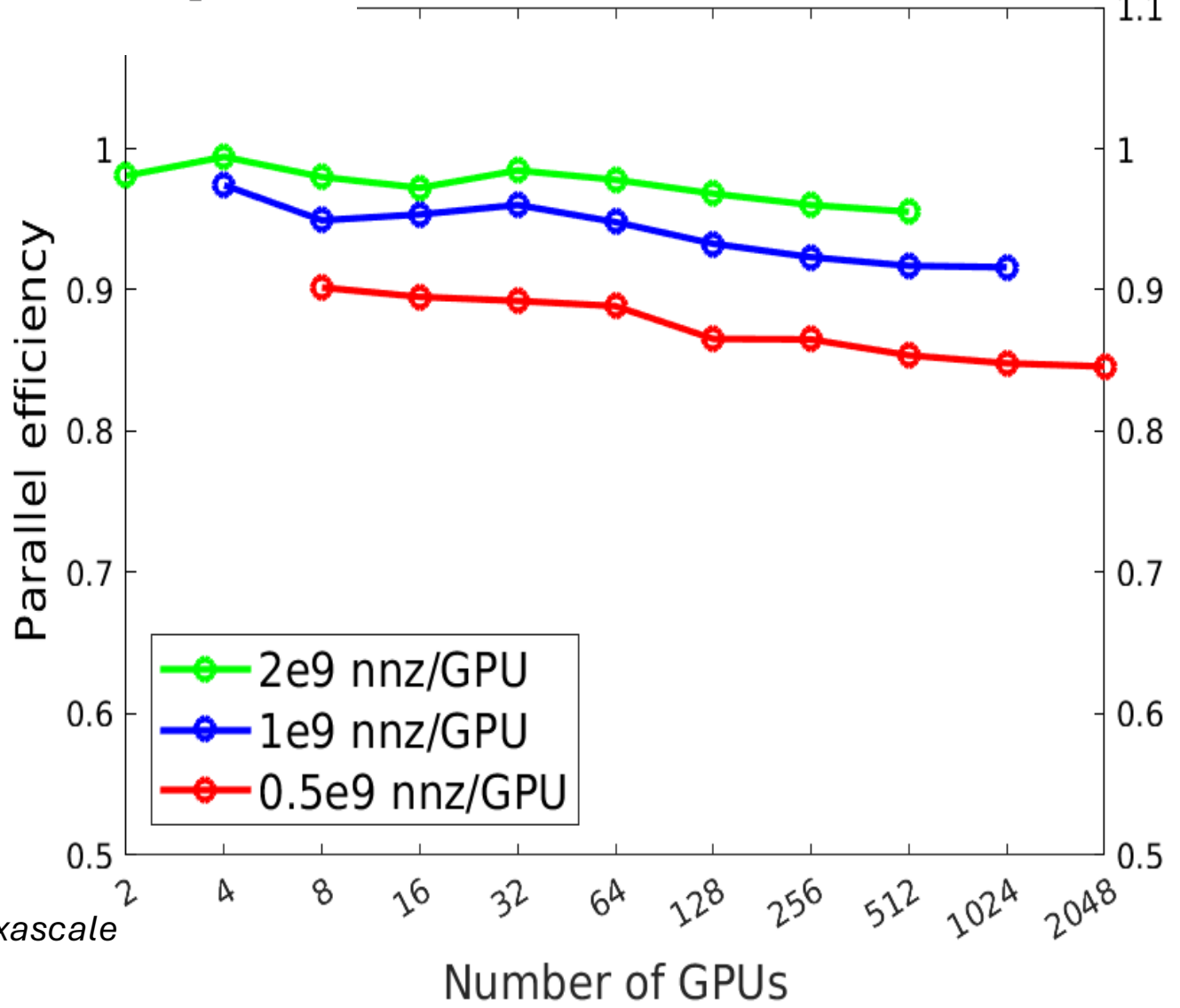
Speed of transferring 1 GB of data between 2 MPI-processes is about 133 GB/s .....!!

Benchmark: GPU-accelerated solver Time-dependent Dirac Eq.

$$i\frac{\partial}{\partial t}\psi(t) = \left[\beta c^2 - I_4 V(r) + c\boldsymbol{\alpha}\cdot\mathbf{p} - c\boldsymbol{\alpha}\cdot\mathbf{A}(t,\mathbf{r})\right]\psi(t).$$



LUMI-G



GaDE - GPU-Acceleration of Time-Dependent Dirac Equation for Exascale

Comput. Phys. Commun. 321, 110015 (2026).

Conclusion

- **Compute node architecture**
  NUMA node: share L3 cache with 8 threads
  CPU-affinity: selecting the closest GPU to a NUMA node for faster communication

- **CPU & GPU architectures**
  Memory hierarchy for optimizing communication latency
  GPUs offer high-throughput with 10000-20000 cores for heavy computing

- **Network Fabric architecture**
  Slingshot Network: Cassini NIC & Rosetta Switch
  NIC handles the processing of network protocol.
  Rosetta Switch uses DragonFly topology to communication between a large number of endpoints.
  Congestion: HPE Slingshot provides advanced congestion control to quickly **detect congestion**.
  RDMA capability to reduce communication latency
  Open-source APIs for **network communications: Libfabric & UCX**

- **Lustre architecture**
  Lustre focuses on the **management** of storage devices and file systems.
  Up to **512 PB** in one filesystem & up to **32 PB** for one file.
  Support of up to **25000+ clients** (access the file system in parallel).

General comments:

- In general: “**idle**” is associated with computation on either CPU or GPU
- But also to communication: **Network idle**
- Overlapping communication and computation
- CPU: Use **nonblocking** Sends and Recvs
- GPU: Use **asynchronous** programming via streams
- Further development of **new hardware features** – but require their use by programmers
- Design your code to take advantage of high-performance interconnects (**GPU-awareness feature**, RDMA)
- **Profiling** to identify bottelnecks
- If a system offers both protocols: **OFI & UCX** – benchmarking helps identify which one offers better performance for your application.
- High-speed Network is designed to handle **large amount of data**.
  It will struggle with small data transfers (the system is not optimized to handle lot of small files)