High-Performance Computing

Daniele De Sensi

Organization Info

Schedule

- 27/11/2025 17:00-19:00: Introduction and Outlook
- 01/12/2025 16:00-19:00: MPI & CUDA Profiling (Laura Bellantani, CINECA)
- 04/12/2025 17:00-19:00: Multi-GPU: NCCL
- Extra Lec. 1 (Friday XX 15-18 (?)): Multi-GPU: NCCL Hands-On
- 11/12/2025 17:00-19:00: Multi-GPU: NVSHMEM
- Extra Lec. 2 (Friday YY 15-18 (?)): Multi-GPU: NVSHMEM Hands-on
- 18/12/2025 17:00-19:00: Advanced GPU Programming
- Extra/Non-Extra Lecture (TBD): Marco Faltelli - ENEA HPC Lab

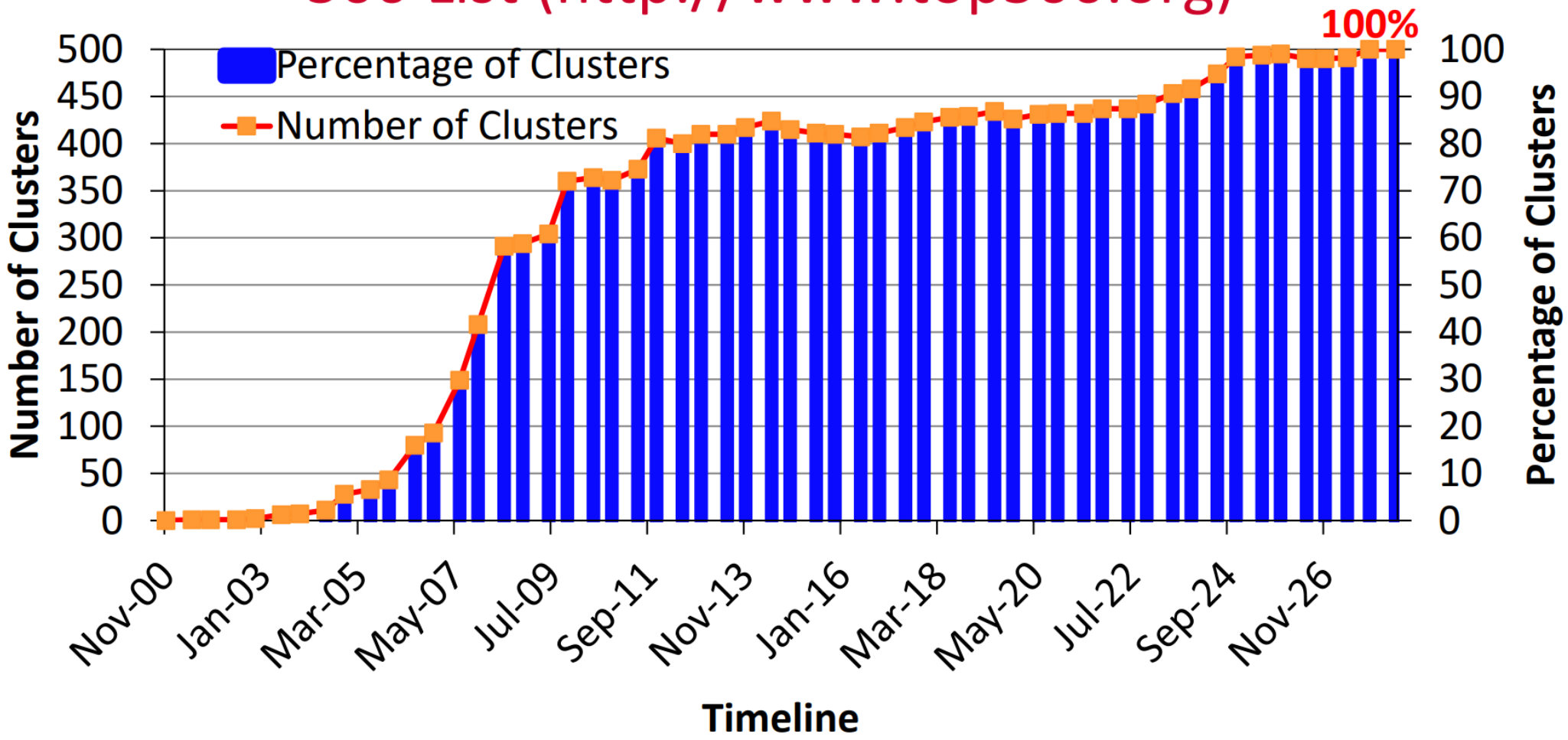
Computational Resources Access

- Computer Science Department Cluster
- If you do not have access already, fill the form https://docs.google.com/forms/d/e/1FAIpQLScJbGoKOKEajatPkBRB6bcKDe3_IQTYSZHVzA7vFnjpYmBxsA/viewform?usp=sf_link

HPC Systems Anatomy

HPC Systems Today – Top500

Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)



https://hoti.org/assets/slides/2025_08_22_day3_Tutorial_Track1_HPNetworks.pdf

HPC Systems Today – Top500

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
5	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
6	**HPC6** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461
7	**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
8	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	2,121,600	434.90	574.84	7,124
9	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
10	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	306.31	7,494

1.8×10^{18} Flop/s

HPC Systems Today

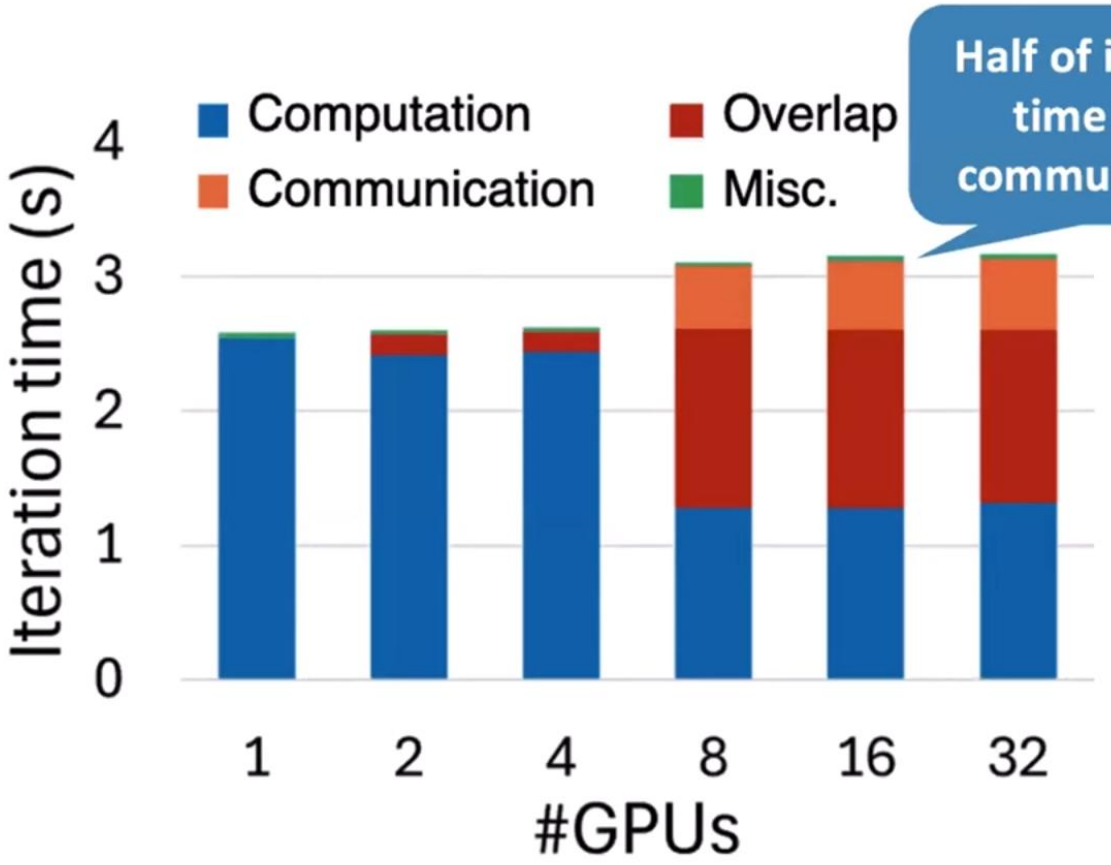
HPL

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)
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

HPCG

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	HPCG (TFlop/s)
1	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	17406.00

100x lower Flop/s on HPCG

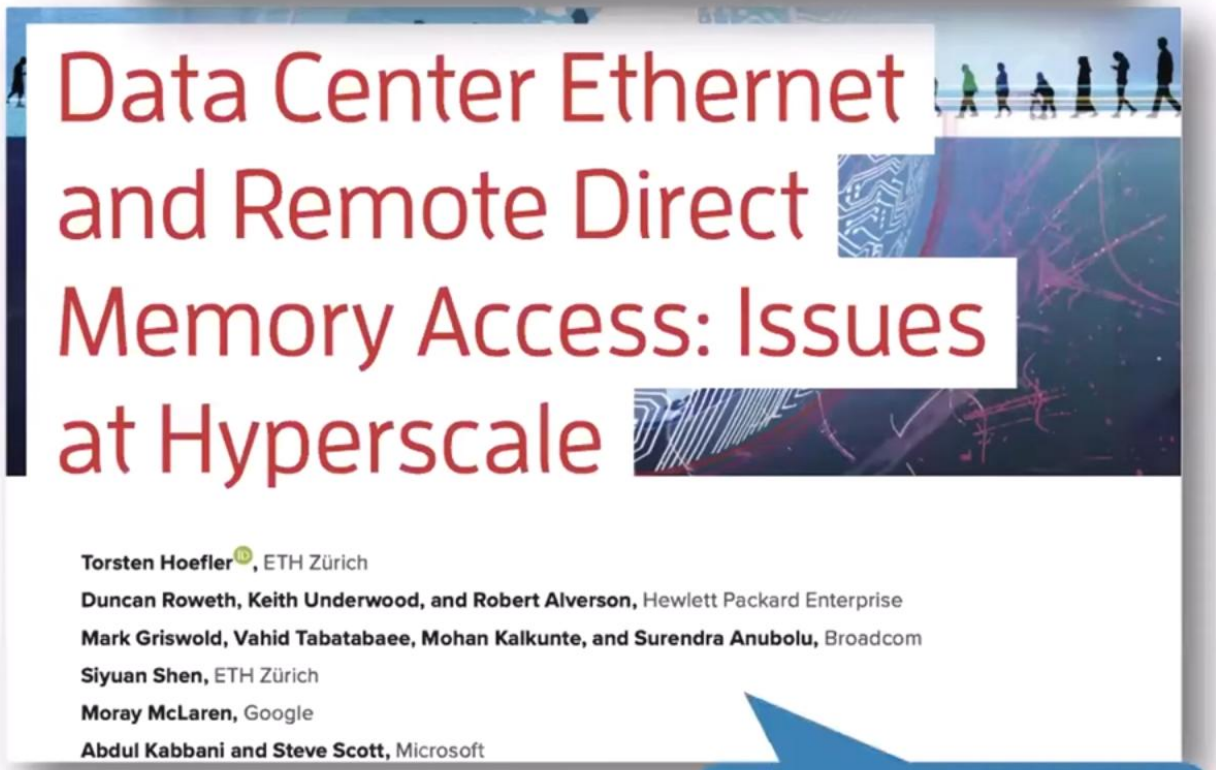


Half of iteration time is for communication

Iteration time breakdown of the full fine-tuning method on a 1B parameters Transformer decoder model [1]

Is Network the Bottleneck of Distributed Training?

Zhen Zhang[1], Chaokun Chang[2], Haibin Lin[2], Yida Wang[2], Raman Arora[1], Xin Jin[1]
[1]Johns Hopkins University, [2]Amazon Web Services



Network bandwidth is not efficiently utilized

[1] N. Alnaasan et al., "Characterizing Communication in Distributed Parameter-Efficient Fine-Tuning for Large Language Models," HOTI 2024, pp. 11–19.

https://www.youtube.com/watch?v=UDCLvBqEPcA

HPC Systems Today

Data movement is one of the main bottlenecks

Different ways to improve systems efficiency:

- Increase data movement efficiency within the node
  - Traditional von-Neumann architectures
  - Non-traditional architectures
  - Reduced-precision Data Types
- Increase data movement efficiency between the nodes
  - Scale-out Network
  - Scale-up Network
  - Programming models/APIs

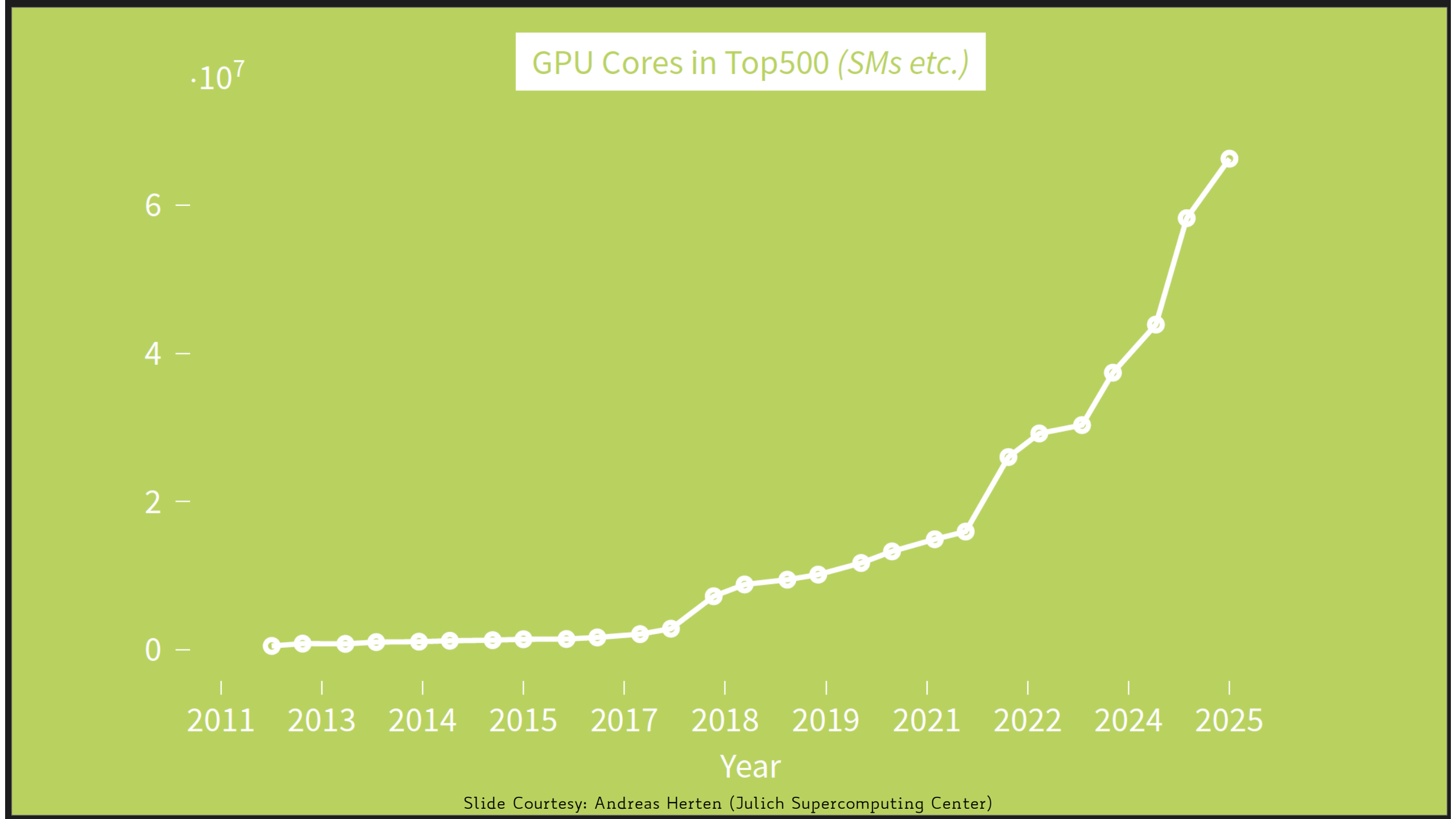
Questions?

HPC Systems Today

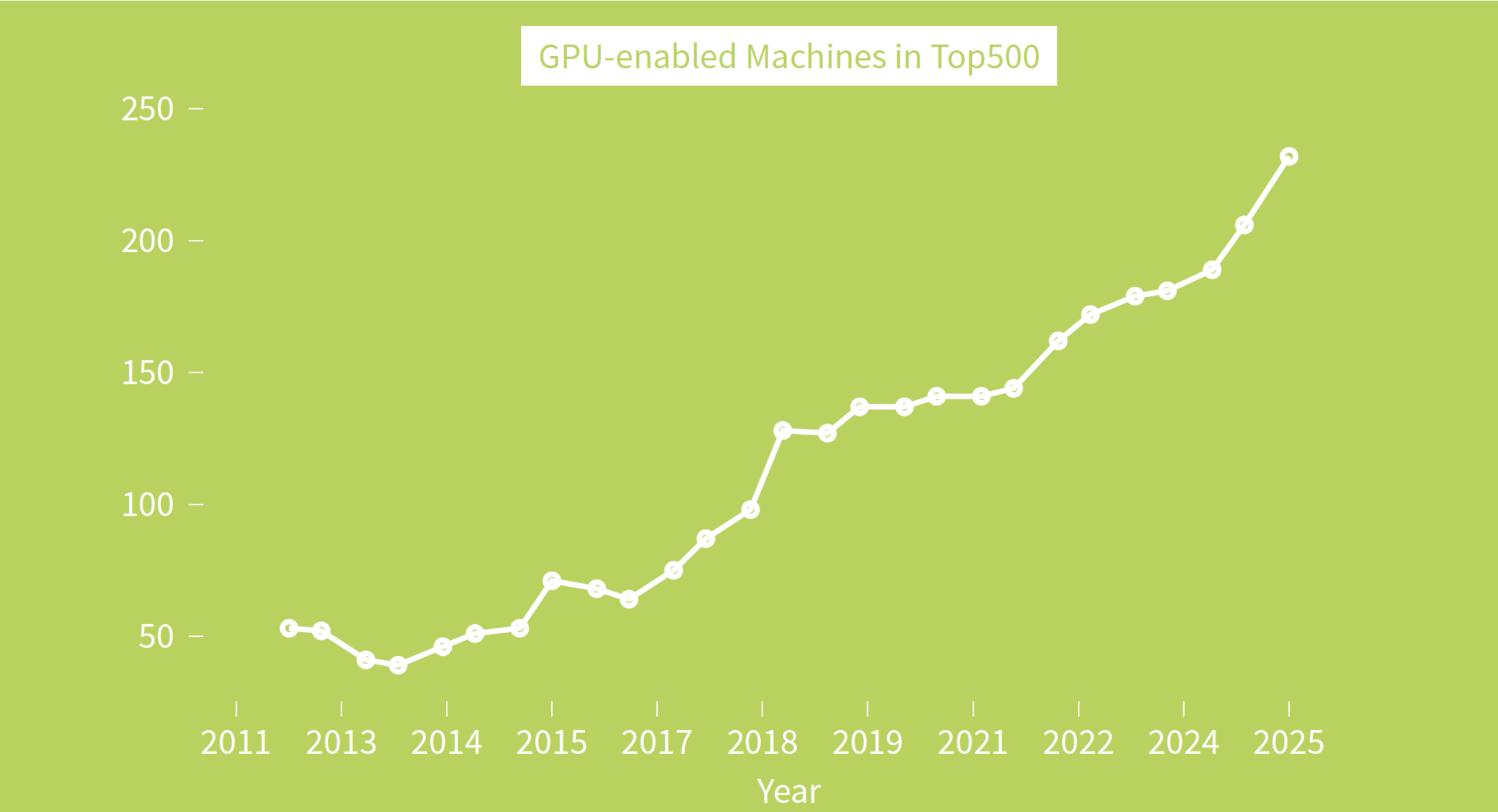
HPCG lower efficiency caused by data movements

Different ways to improve systems efficiency:

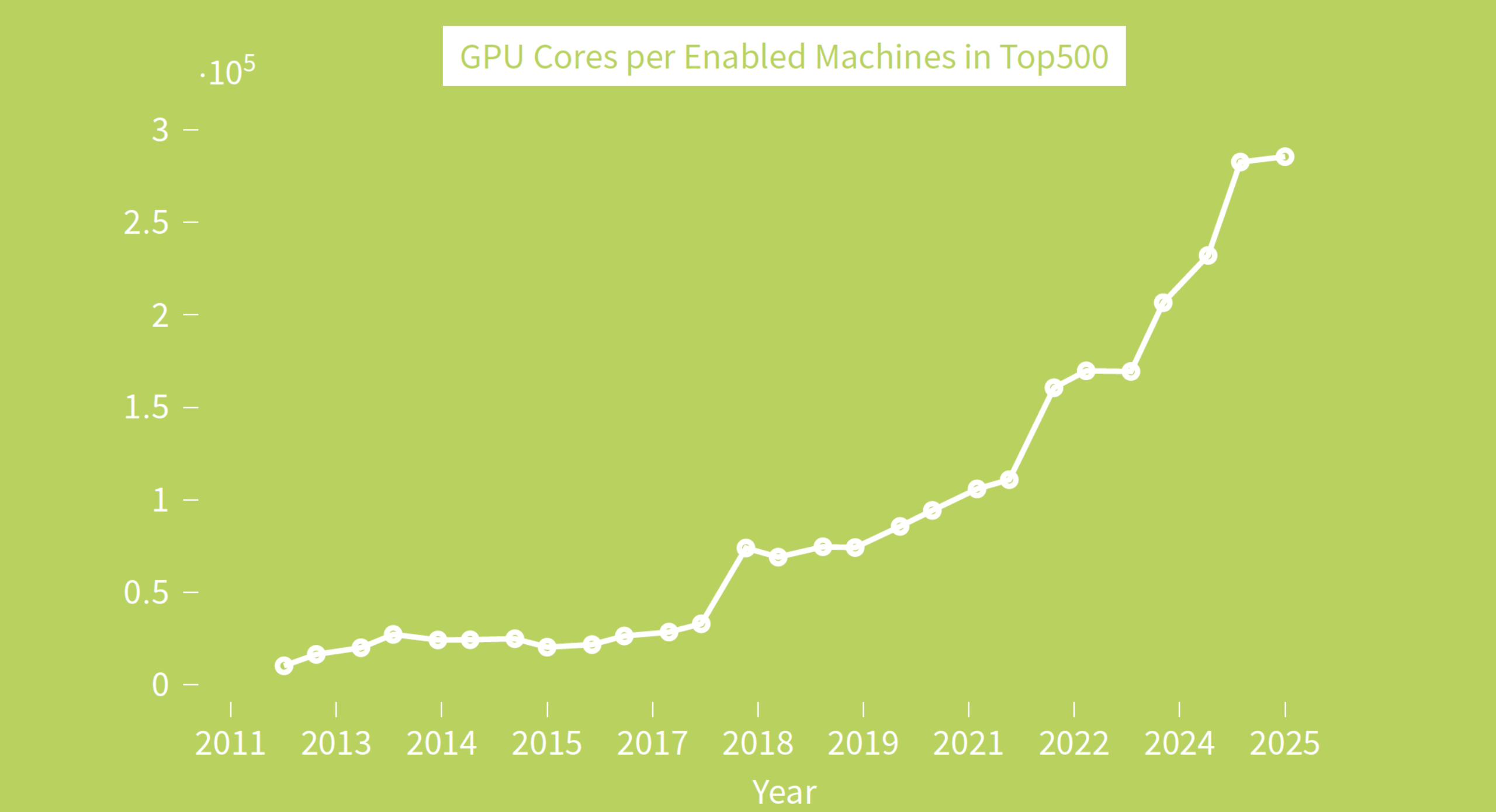
- Increase data movement efficiency within the node
  - Traditional von-Neumann architectures
  - Non-traditional architectures
  - Reduced-precision Data Types
- Increase data movement efficiency between the nodes
  - Scale-out Network
  - Scale-up Network
  - Programming models/APIs



Slide Courtesy: Andreas Herten (Julich Supercomputing Center)



Slide Courtesy: Andreas Herten (Julich Supercomputing Center)



Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

HPC Systems Today – Accelerators

"The history of AI is inseperable from hardware developments. A lot of progress we have been seeing in AI [...] have been enabled by GPUs. [...] Our imagination is limited by the hardware at our disposal."
Yann LeCun - 2018 Turing Award

Yann LeCun on the future of deep learning hardware (https://www.youtube.com/watch?v=9r2lqrtQXFQ)

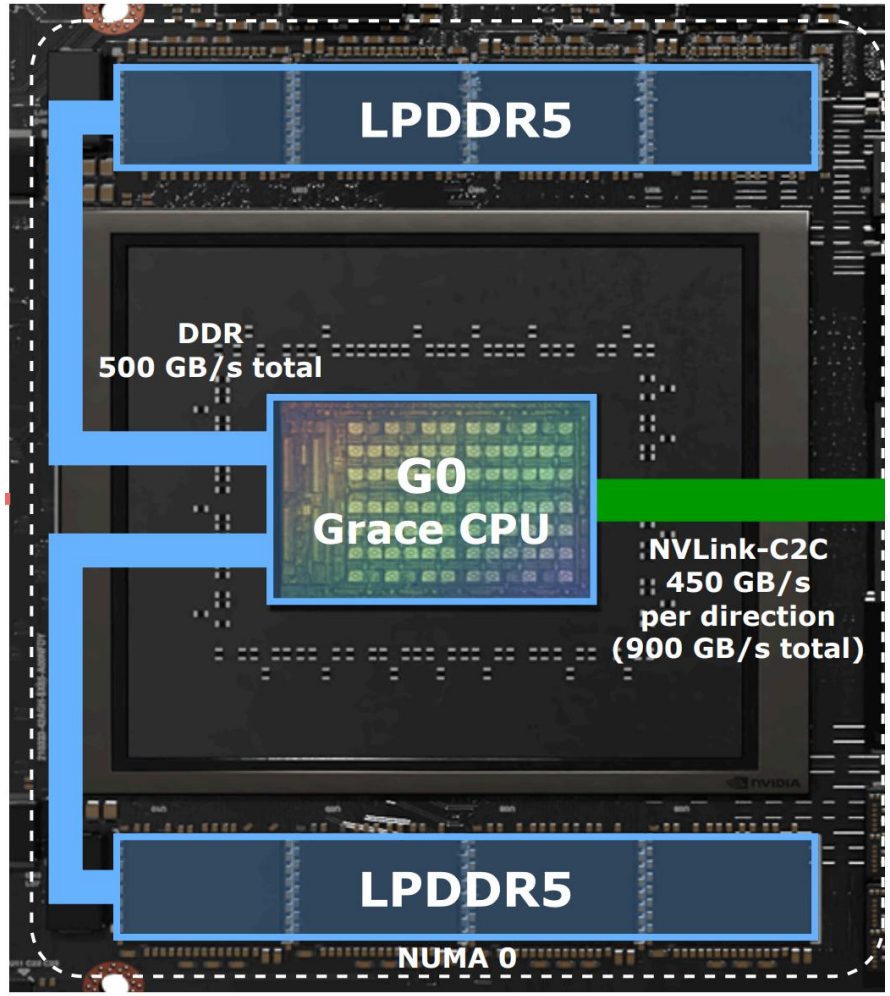
Why are GPUs particularly suited for ML?

- ML is data intensive, and GPUs have higher memory bandwidth than CPUs
- ML involves extensive matrix operations such as matrix multiplication and convolutions
- GPUs are particularly well-suited for such kind of operations (parallelization of matrix operations is an active area of research since the early days of computer science)

*"ACM named Jack J. Dongarra recipient of the 2021 ACM A.M. Turing Award for **pioneering contributions to numerical algorithms and libraries that enabled high performance computational software to keep pace with exponential hardware improvements for over four decades.** [...] These contributions laid a framework from which scientists and engineers made important discoveries and game-changing innovations in areas including **big data analytics,** [...] and supported revolutions in computer graphics and **deep learning.**"*

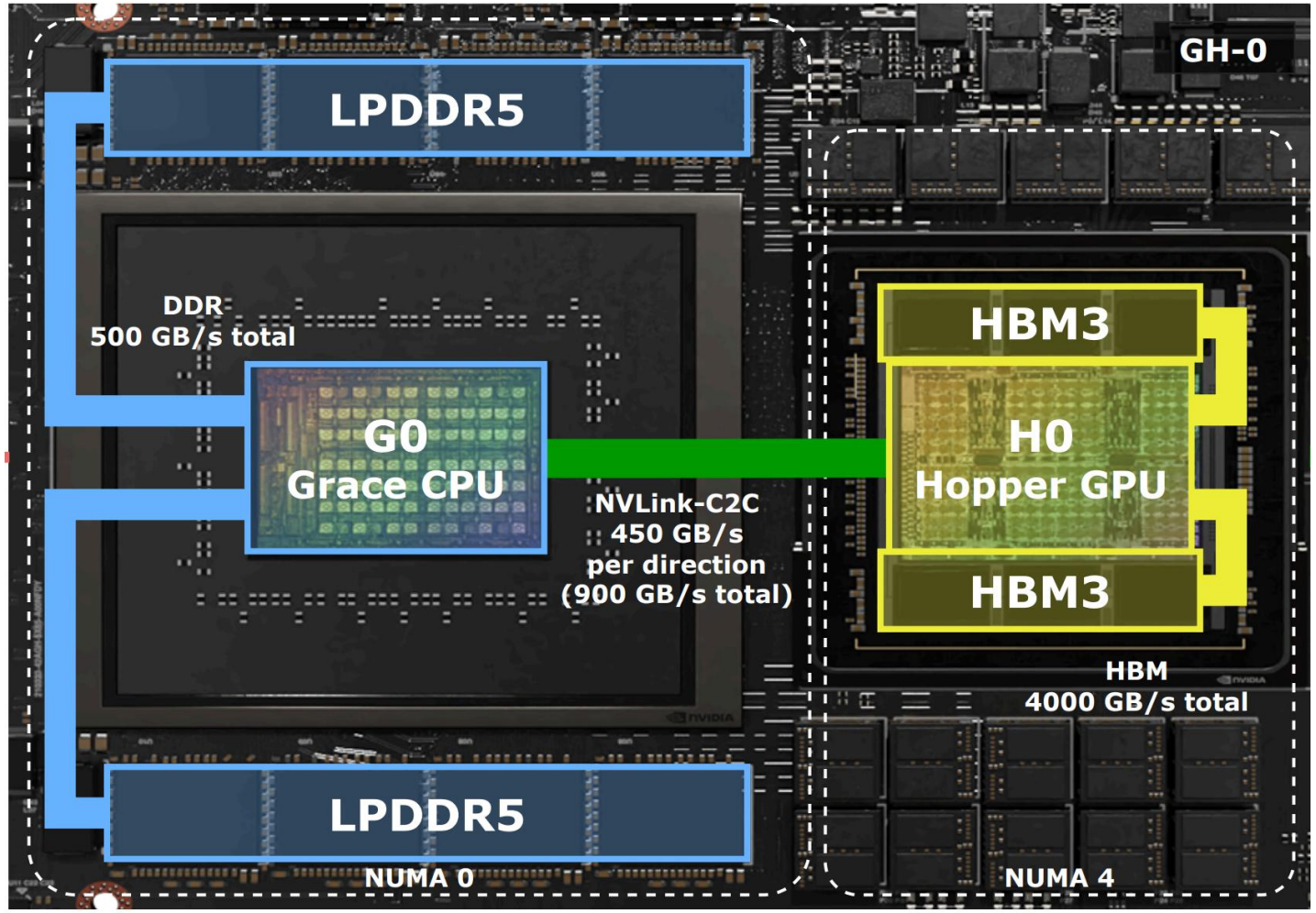
Jack Dongarra – 2021 Turing Award

Node Architecture Today – GH200



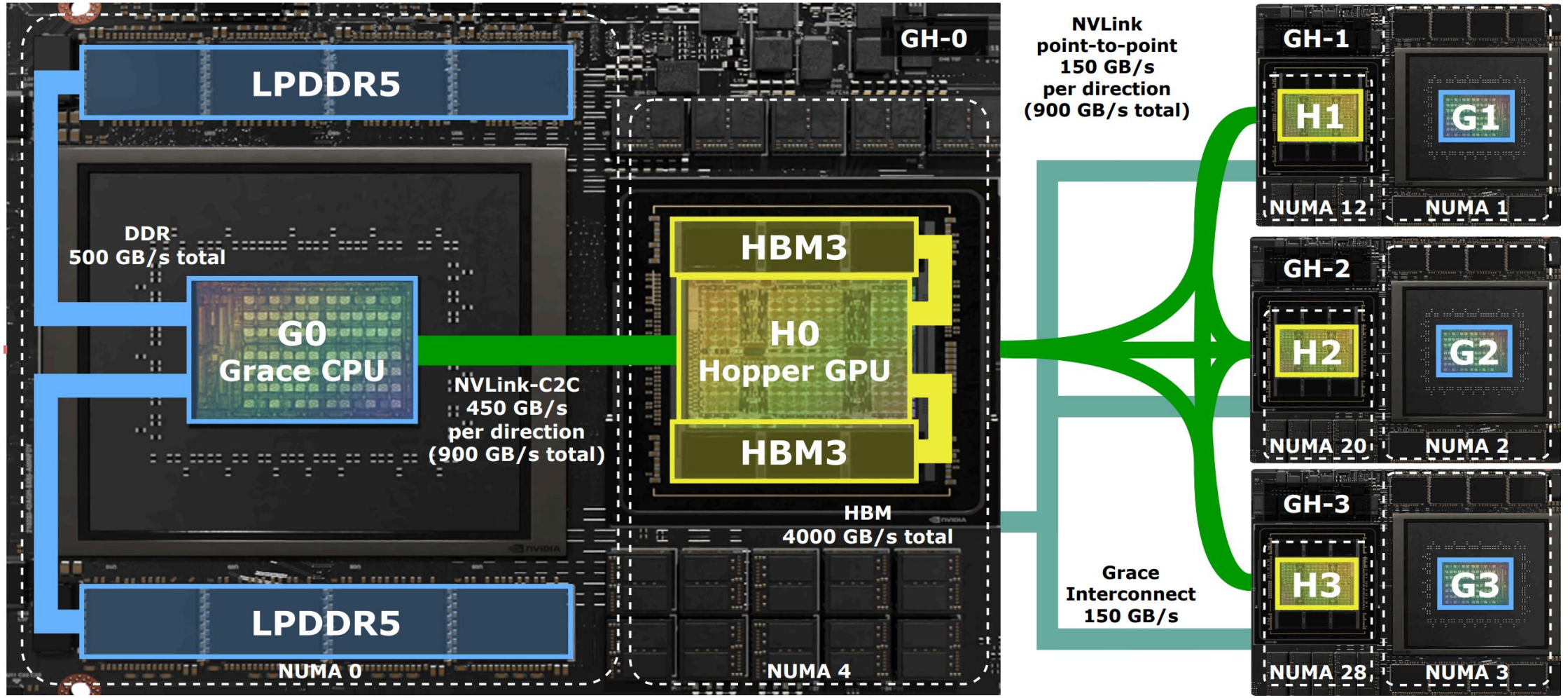
https://arxiv.org/pdf/2408.11556

Node Architecture Today – GH200



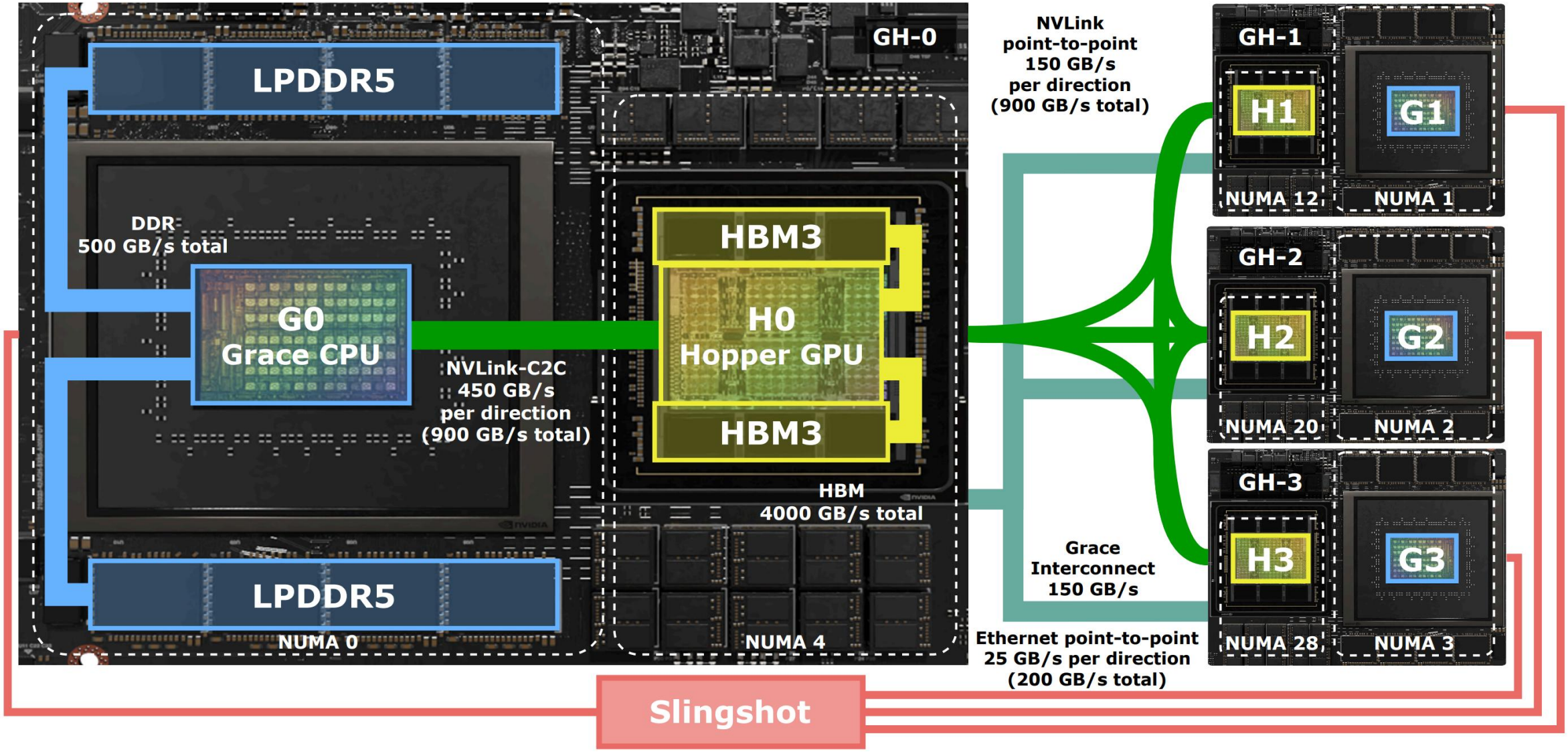
https://arxiv.org/pdf/2408.11556

Node ArchitectureToday – GH200/Alps



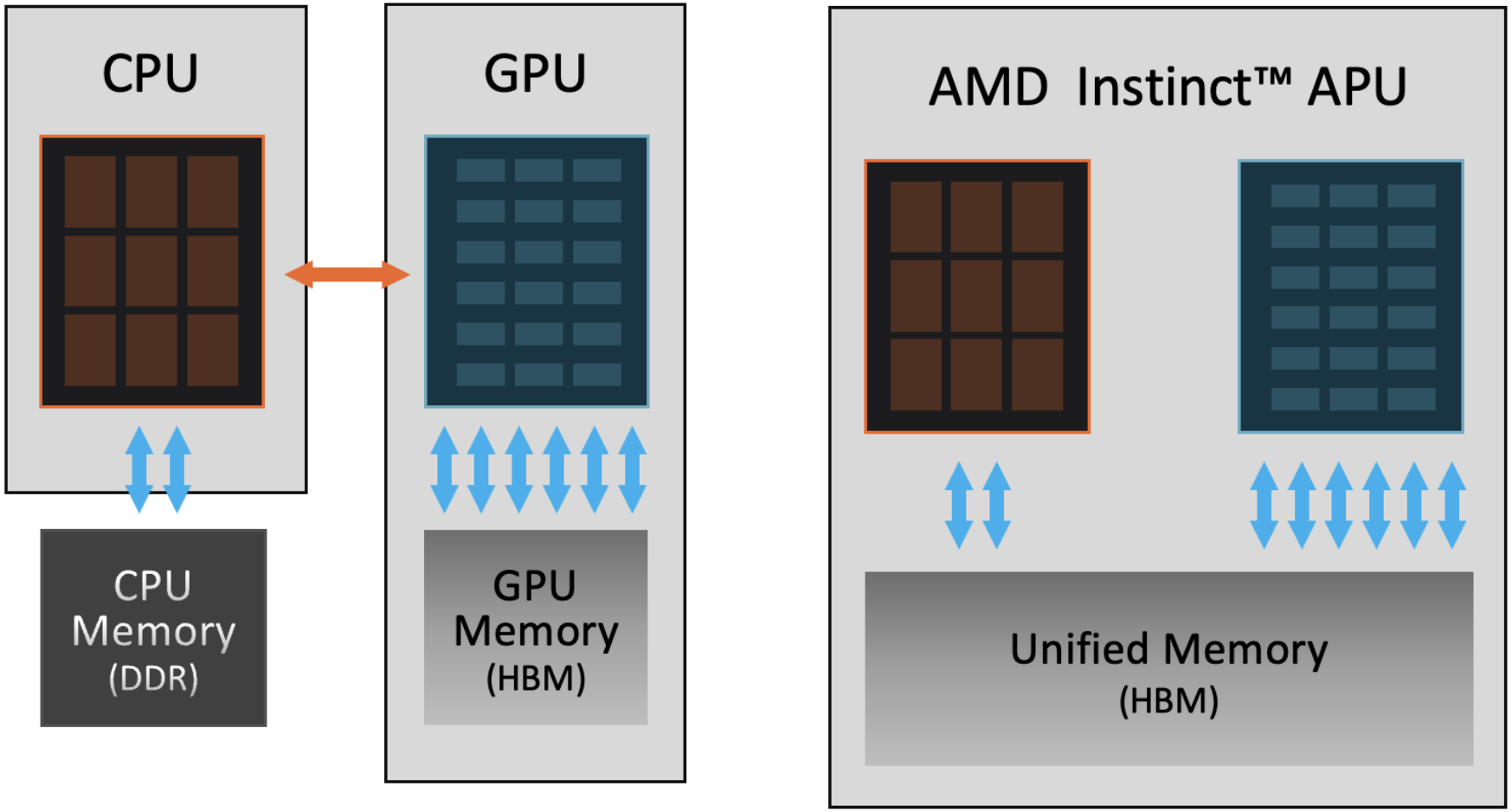
https://arxiv.org/pdf/2408.11556

HPC Systems Today – GH200/Alps



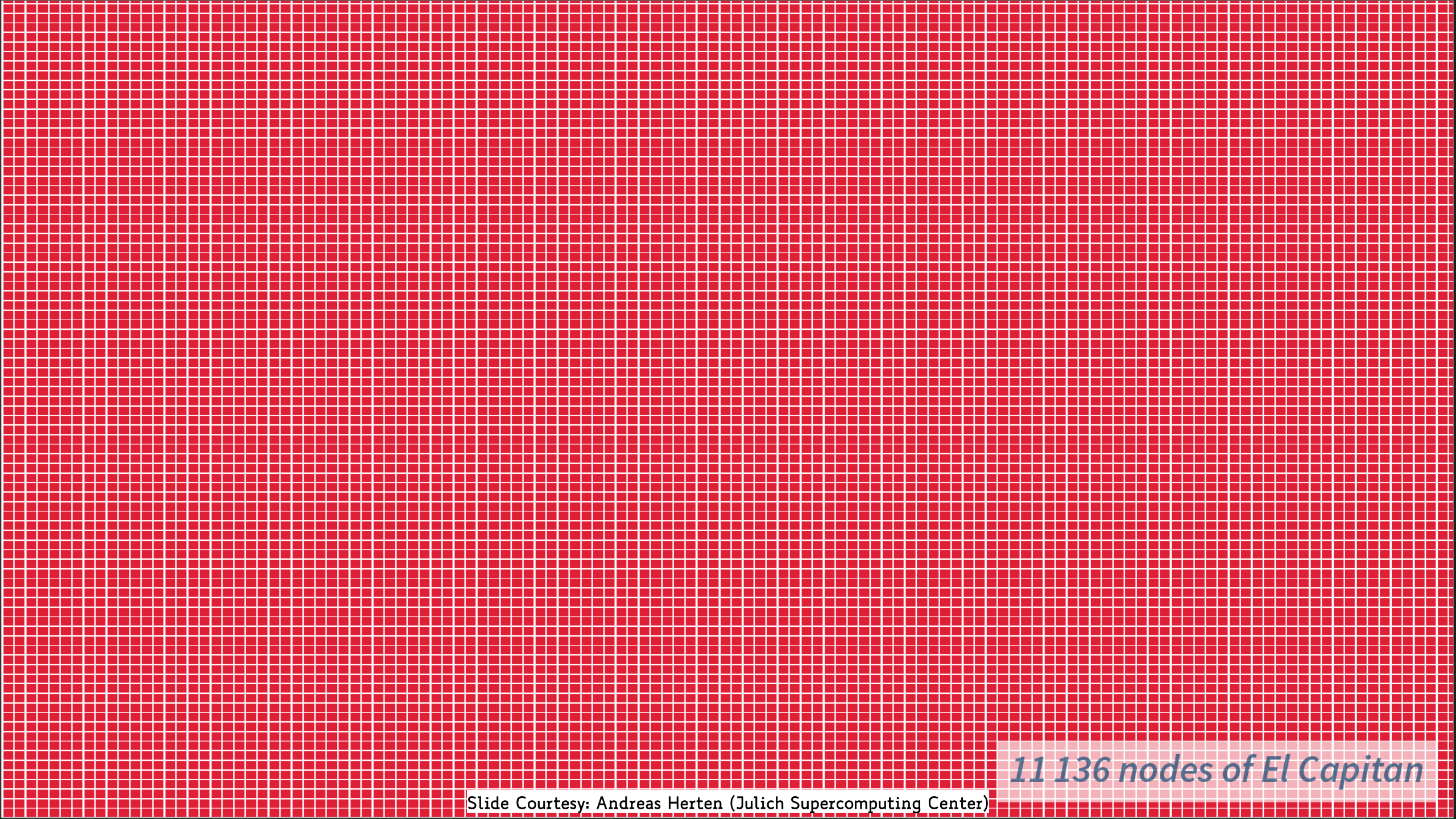
https://arxiv.org/pdf/2408.11556

Node Architecture Today – MI300A APU



(a) Discrete CPU and GPU.

(b) APU.

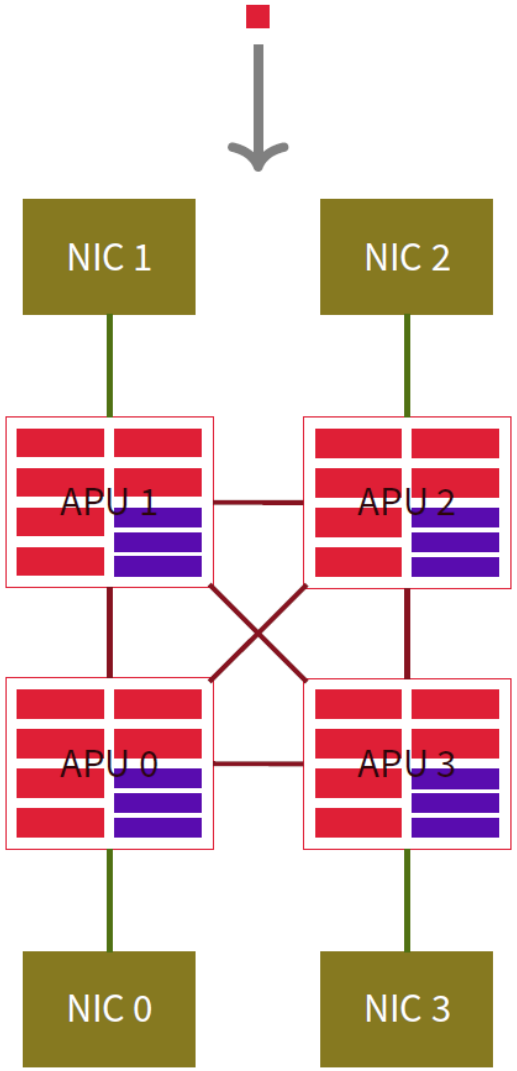


Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

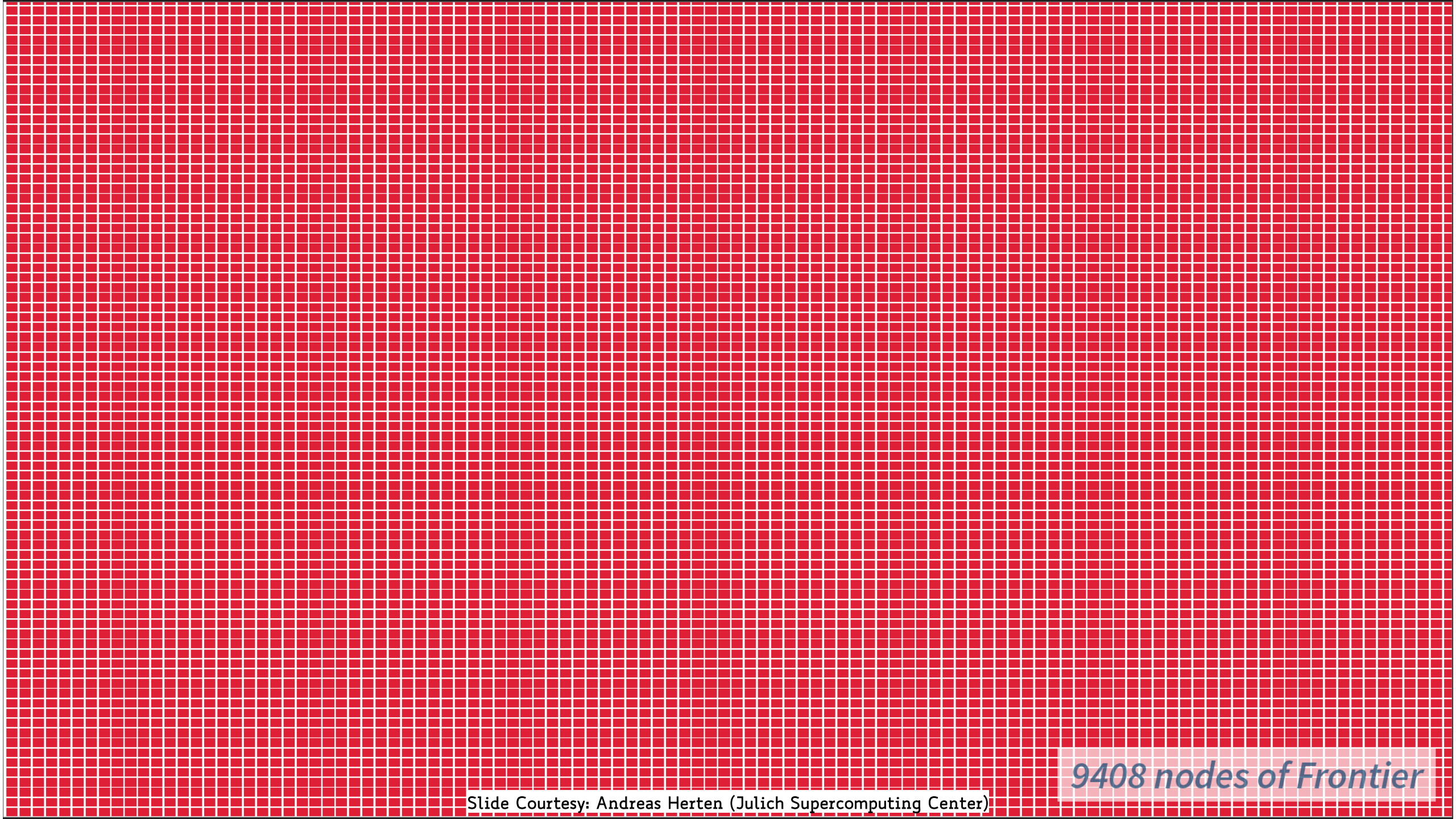
El Capitan

- At Lawrence Livermore National Lab, US
- Largest Exascale system
- HPL: 1.7 EFLOP/s
- APU: 4× AMD Instinct MI300A
  CPU+GPU chiplets combined, 128 GB per APU (HBM)
- Network: 4× HPE Slingshot, 4 × 50 GB/s

→ 11 136 nodes, 44 544 GPUs, 44 544 network devices



Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

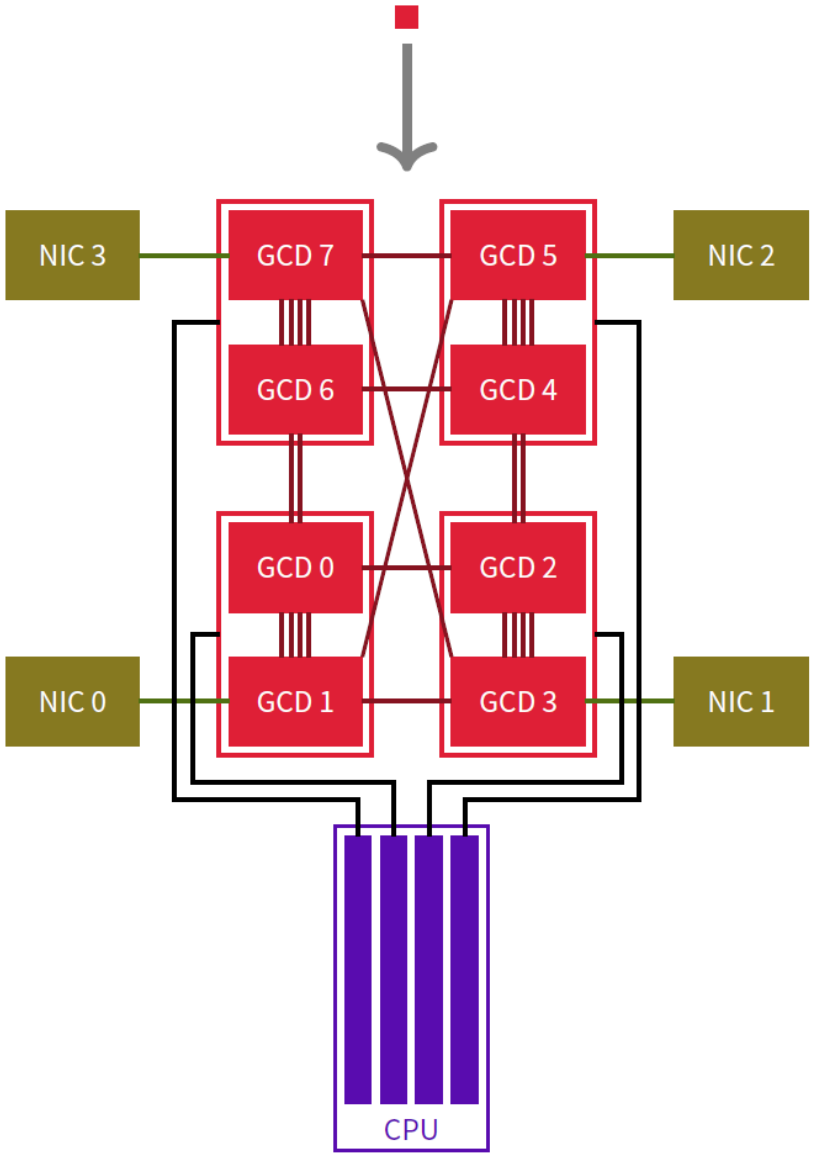


Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

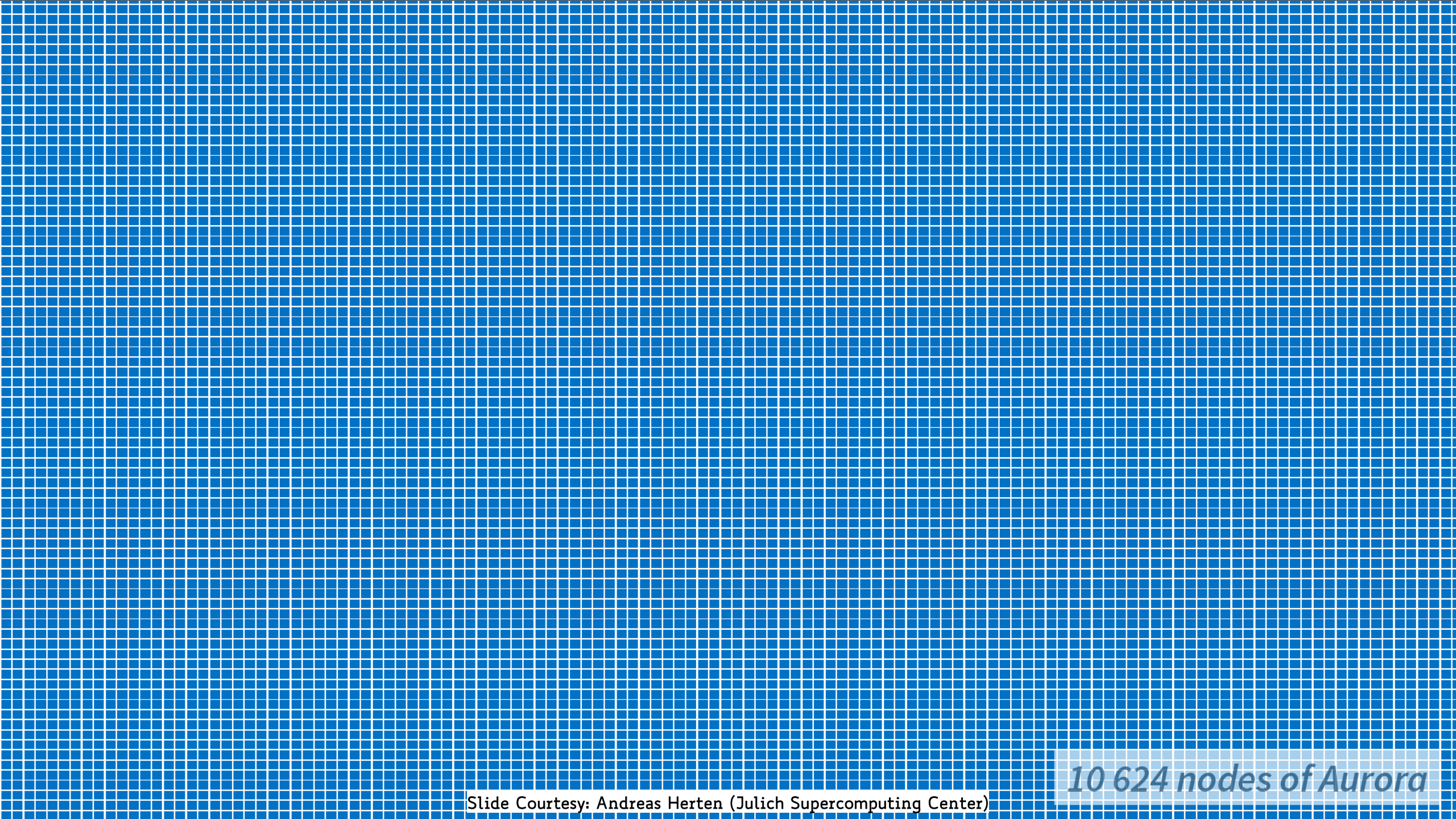
Frontier

- At Oak Ridge Nation Lab, US
- First Exascale system
- HPL: 1.4 EFLOP/s
- GPU: 4× AMD Radeon Instinct MI250X
  2 *GCDs* per GPU, 64 GB memory per GCD
- CPU: 1× AMD Epyc Trento, 64 cores; 4 NUMA domains, one per GCD; 512 GB DDR memory
- Network: 4× HPE Slingshot, 4 × 50 GB/s

→ 9408 nodes, 37 632 GPUs, 75 264 GCDs, 37 632 network devices



Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

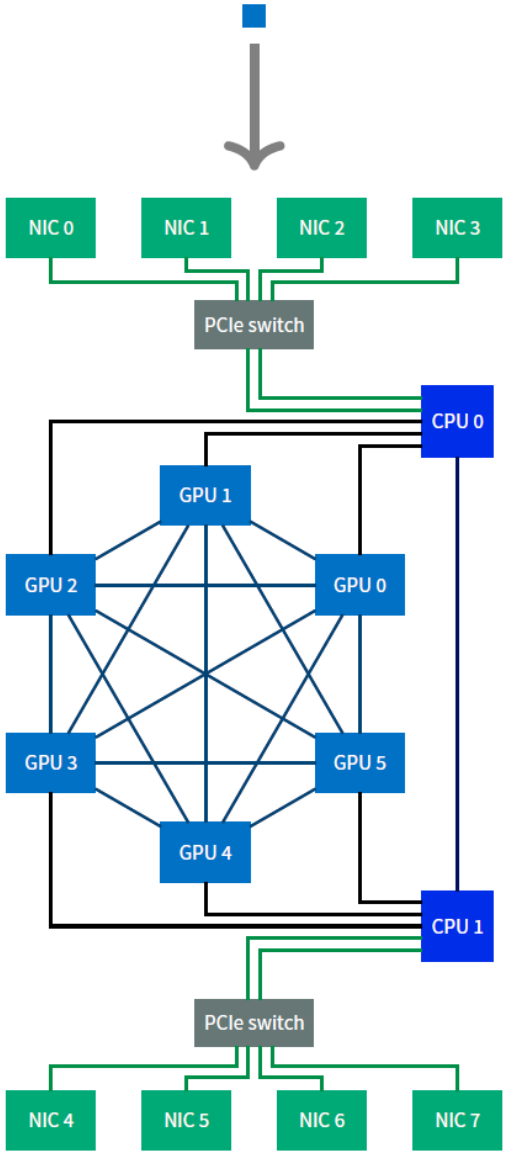


Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

Aurora

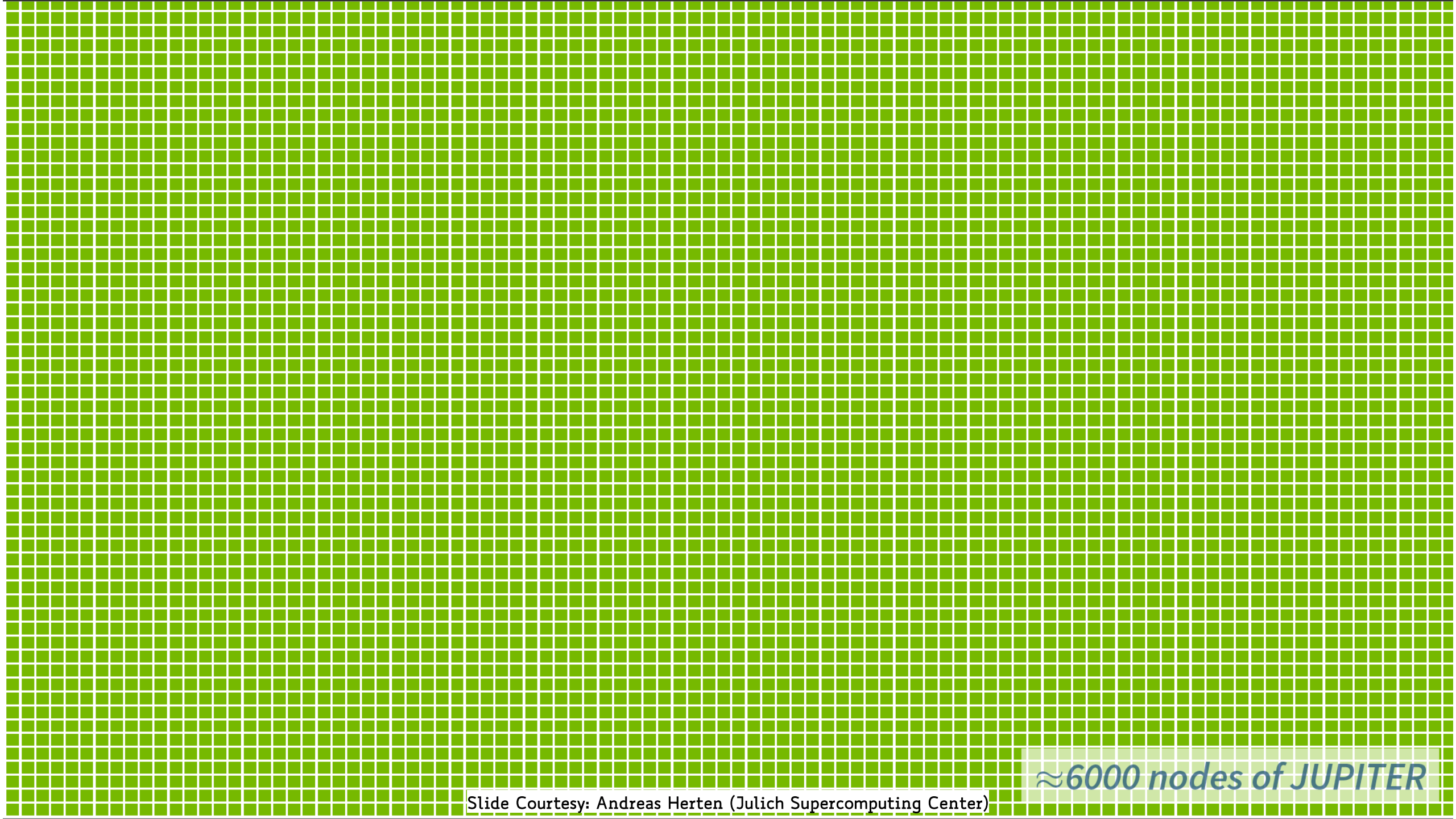
- At Argonne National Lab, US
- Highest theoretical peak
- HPL: 1 EFLOP/s
- **GPU**: 6× Intel Ponte Vecchio (*Data Center GPU Max*)
  128 GB memory per GPU
- **CPU**: 2× Intel Sapphire Rapids, 2× 56 cores; 2×
  56 GB HBM memory; 1 TB DDR memory
- **Network**: 4× HPE Slingshot, 8 × 50 GB/s

→ 10 624 nodes, 63 744 GPUs, 84 992 network devices



JÜLICH Forschungszentrum

Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

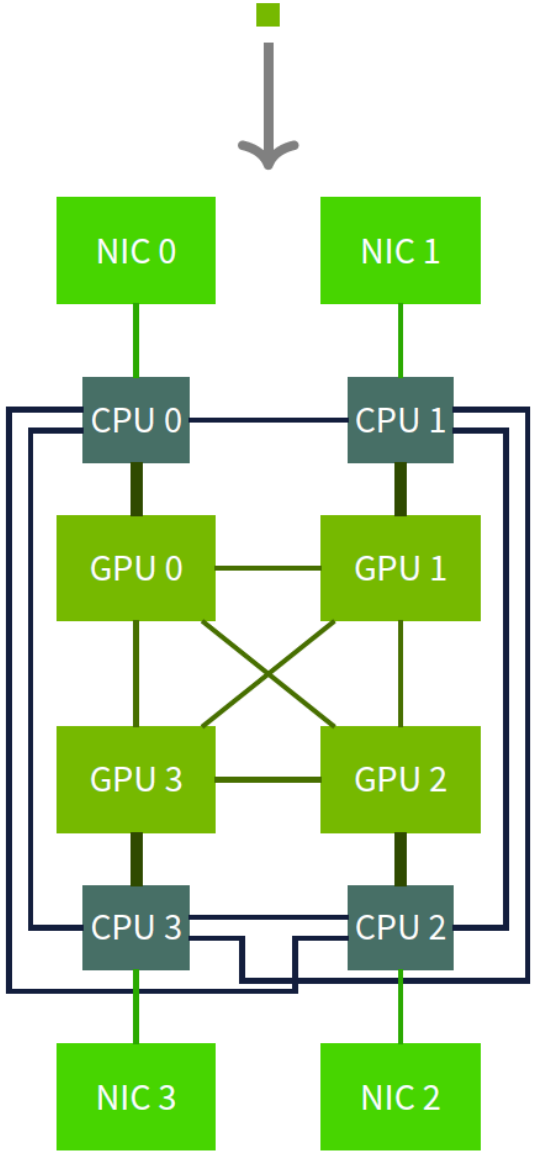


Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

JUPITER

- At Jülich Supercomputing Center, Germany; procured by EuroHPC JU
- First European Exascale system
- **Booster**, Cluster
- HPL June 2025: 0.8 EFLOP/s
- GPU: 4× NVIDIA H100 *Grace-Hopper flavor* 96 GB memory per GPU
- CPU: 4× NVIDIA Grace, 4× 72 cores; 4× 120 GB LPDDR5X memory
- Network: 4× NVIDIA InfiniBand NDR200, 4 × 25 GB/s

→ ≈6000 nodes, ≈24 000 GPUs, ≈24 000 network devices



Slide Courtesy: Andreas Herten (Julich Supercomputing Center)

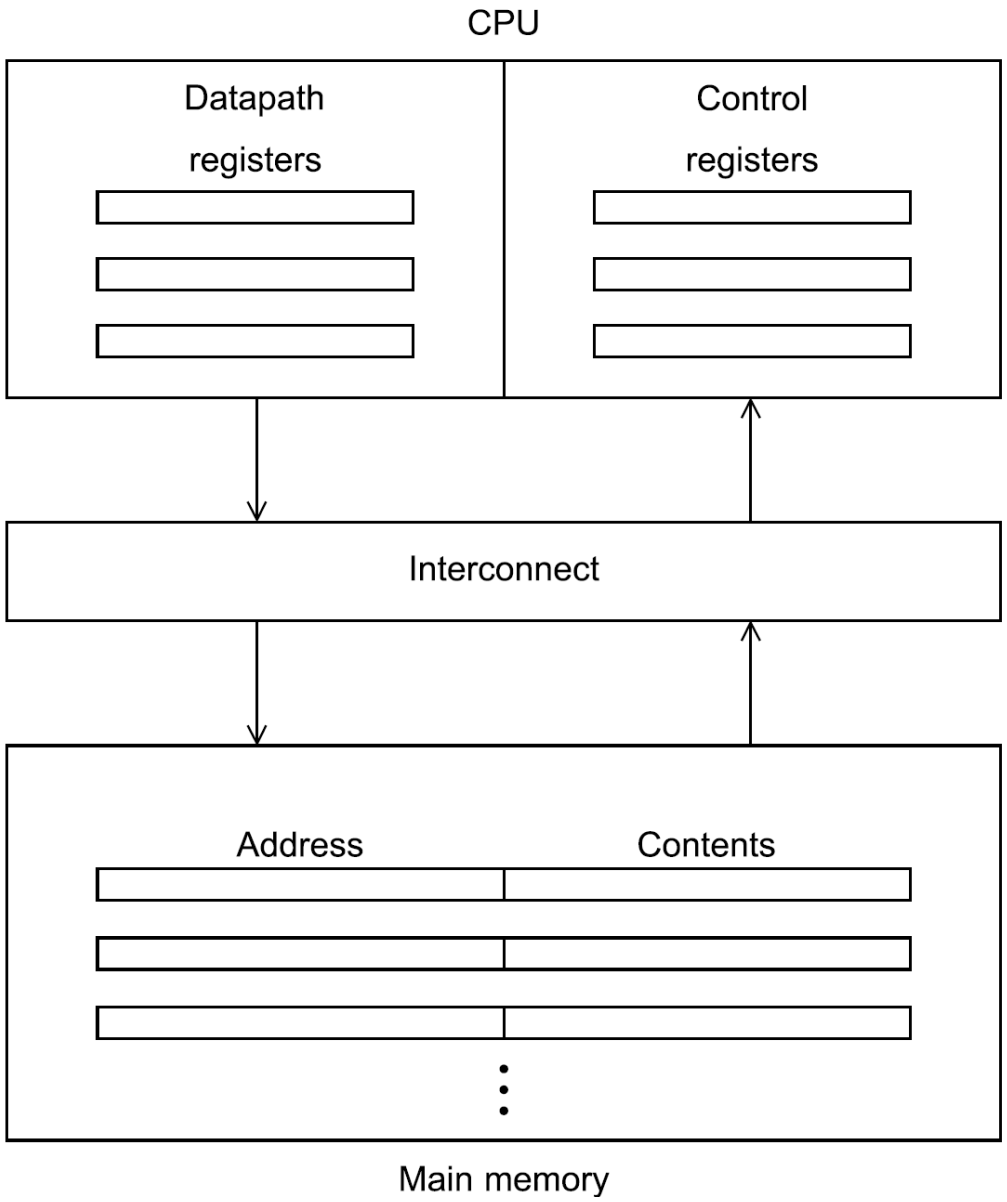
HPC Systems Today

HPCG lower efficiency caused by data movements

Different ways to improve systems efficiency:

- Increase data movement efficiency within the node
  - Traditional von-Neumann architectures
  - Non-traditional architectures
  - Reduced-precision Data Types
- Increase data movement efficiency between the nodes
  - Scale-out Network
  - Scale-up Network
  - Programming models/APIs

von Neumann architecture

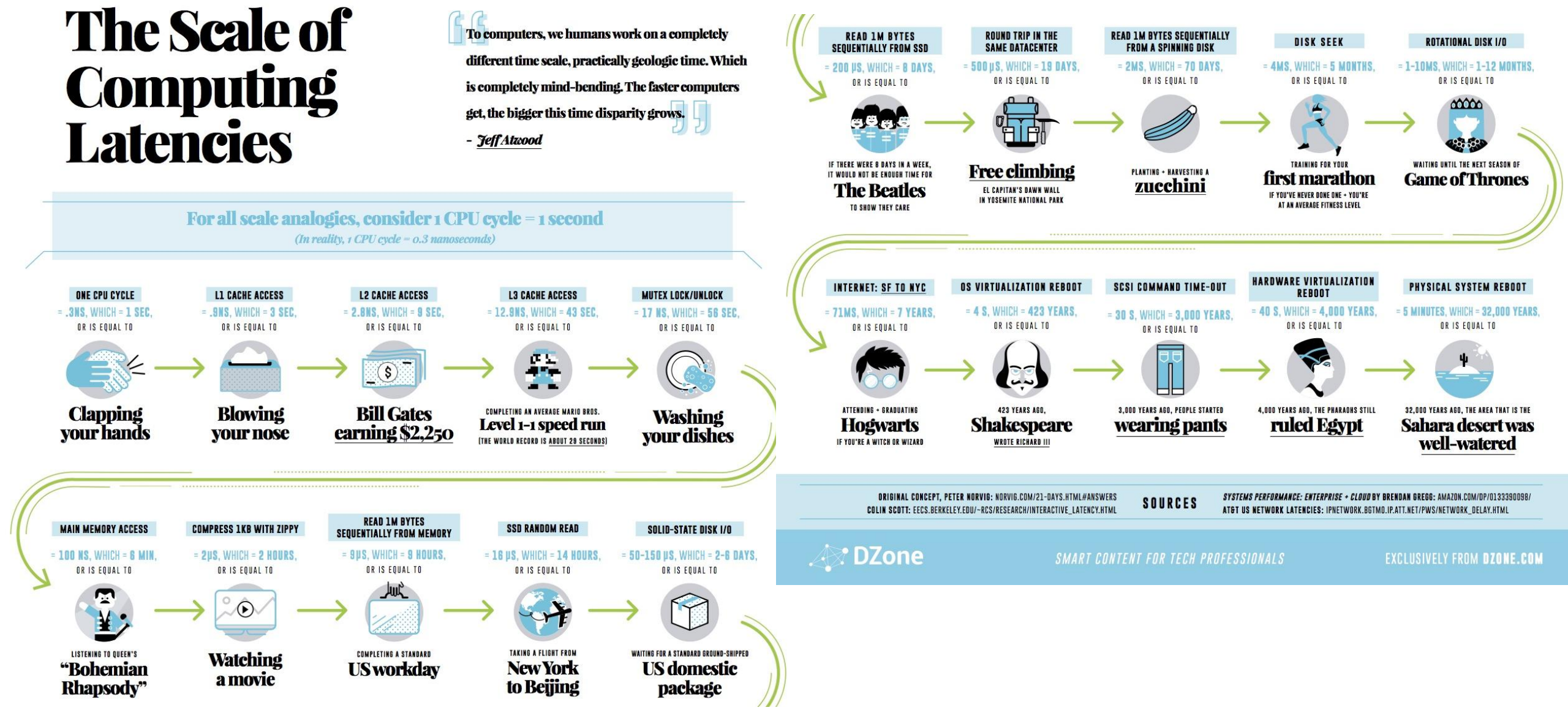


- **Main memory:** Collection of **locations**. Each has an address (used to access that location) and some content (**data** or **instruction**).
- **CPU/Processor/Core:** **Control unit** (decides which instructions execute) and **datapath** (executes the instructions)
  - The state of an executing program is stored in **registers** (very fast storage)
  - An important register in the control unit is the **program counter (PC)**, storing the address of the next instruction to execute
- **Interconnect:** used to transfer data between CPU and memory. Traditionally a **bus**, but can be much more complex (we will get back to it)

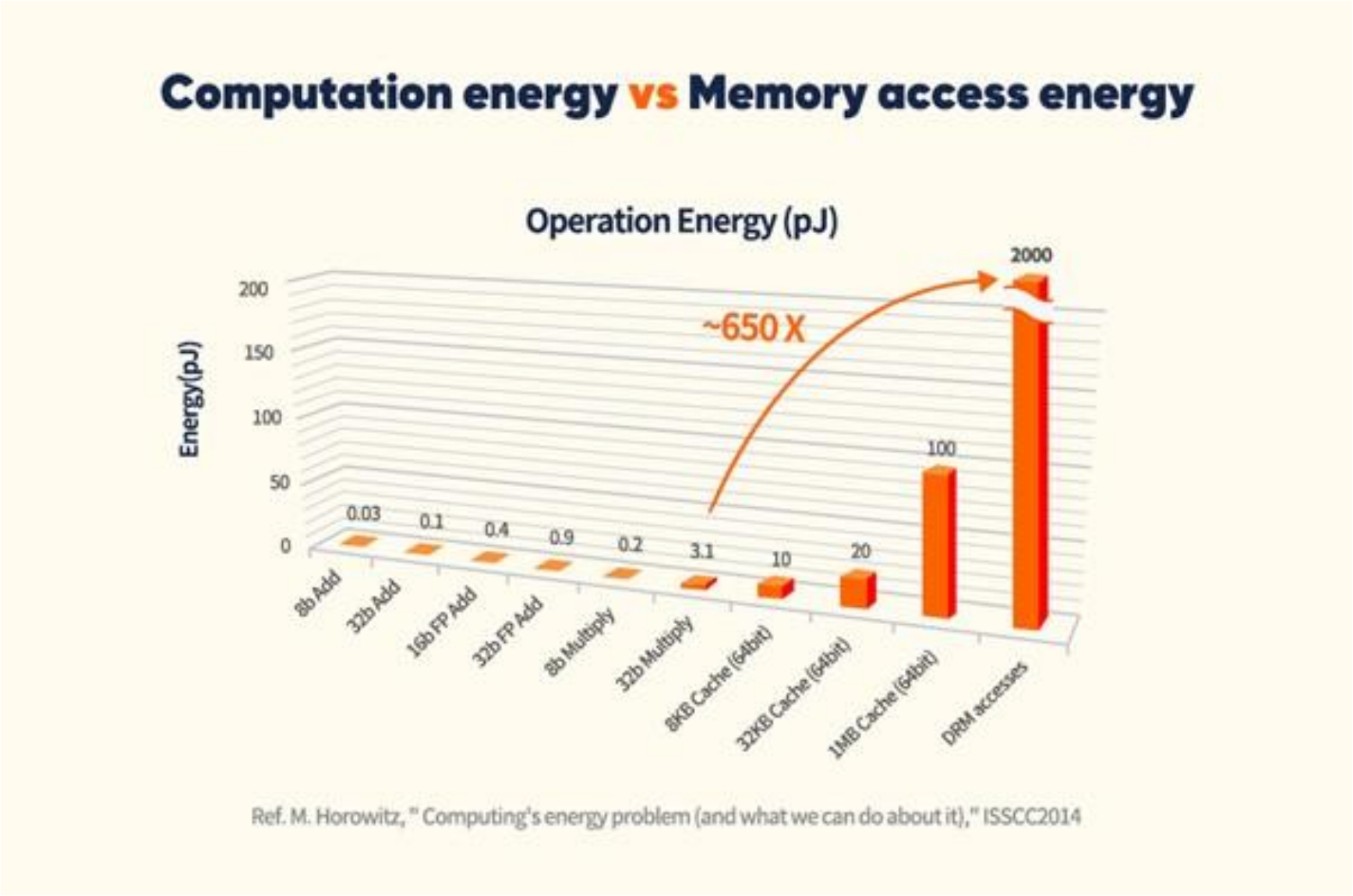
von Neumann bottleneck

- A von Neumann machine executes one instruction at a time, each operating on a few pieces of data (**stored in registers**)
- CPU can **read** (fetch) data from memory, or **write** (store) data to it
- Separation of CPU and memory is known as **von Neumann bottleneck**
- The interconnect determines the rate at which data is transferred

How much does that cost?



Memory Access Cost



https://www.eetimes.com/the-prospect-of-processing-in-memory-pim-in-memory-systems-for-ai-applications/

Tension between usability and specialization/efficiency

General-purpose
(Easier to program)

Single-core CPU | Multi-core CPU | GPU | FPGA | ASIC

Bitcoin mining

TPUs

Specialized

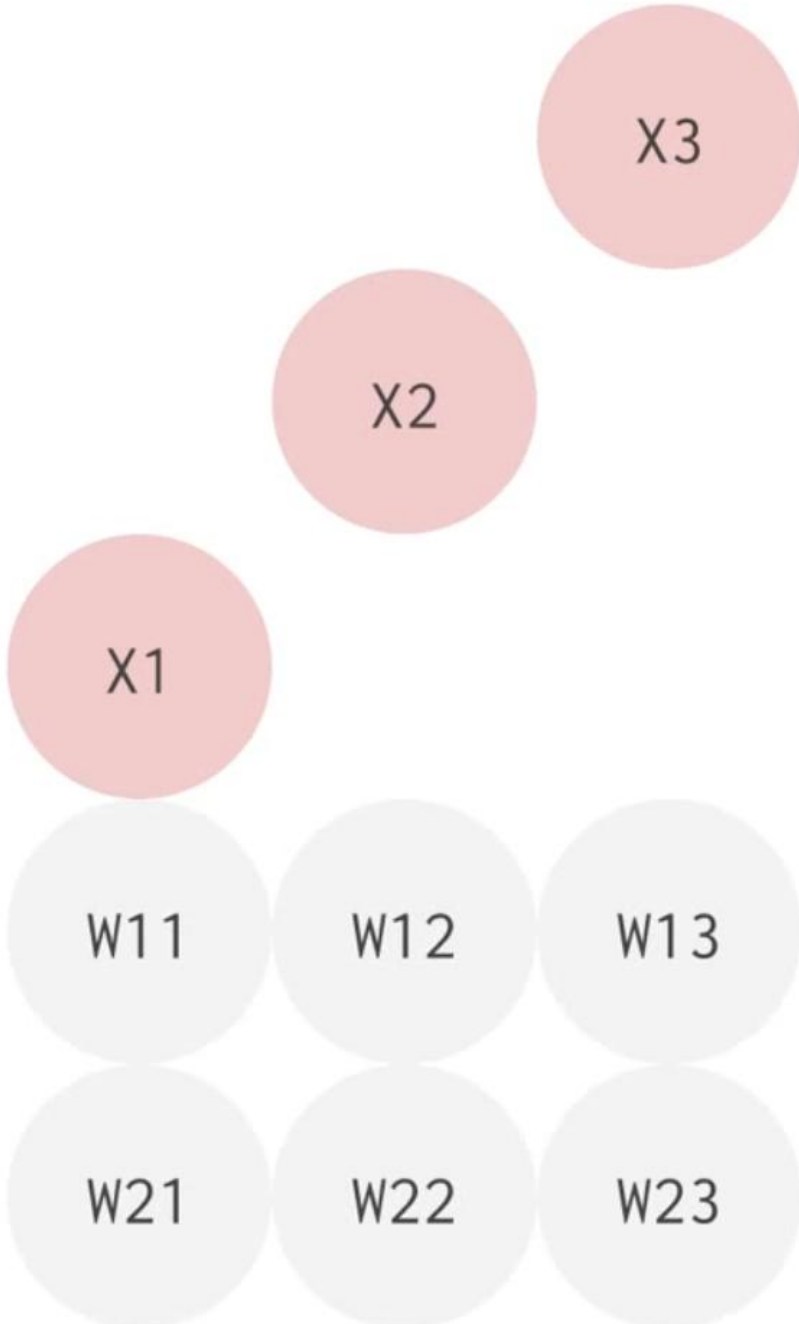
Energy-efficient

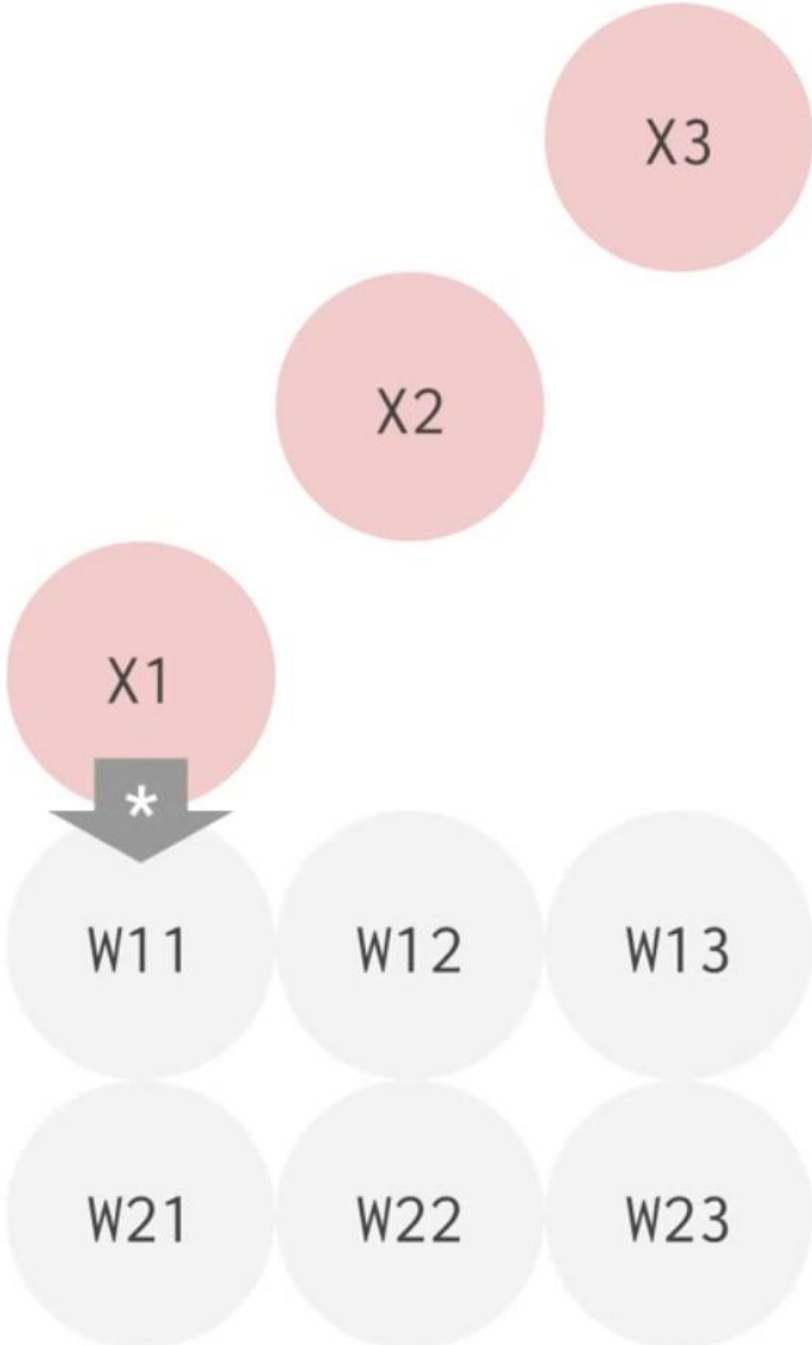
Tensor Cores

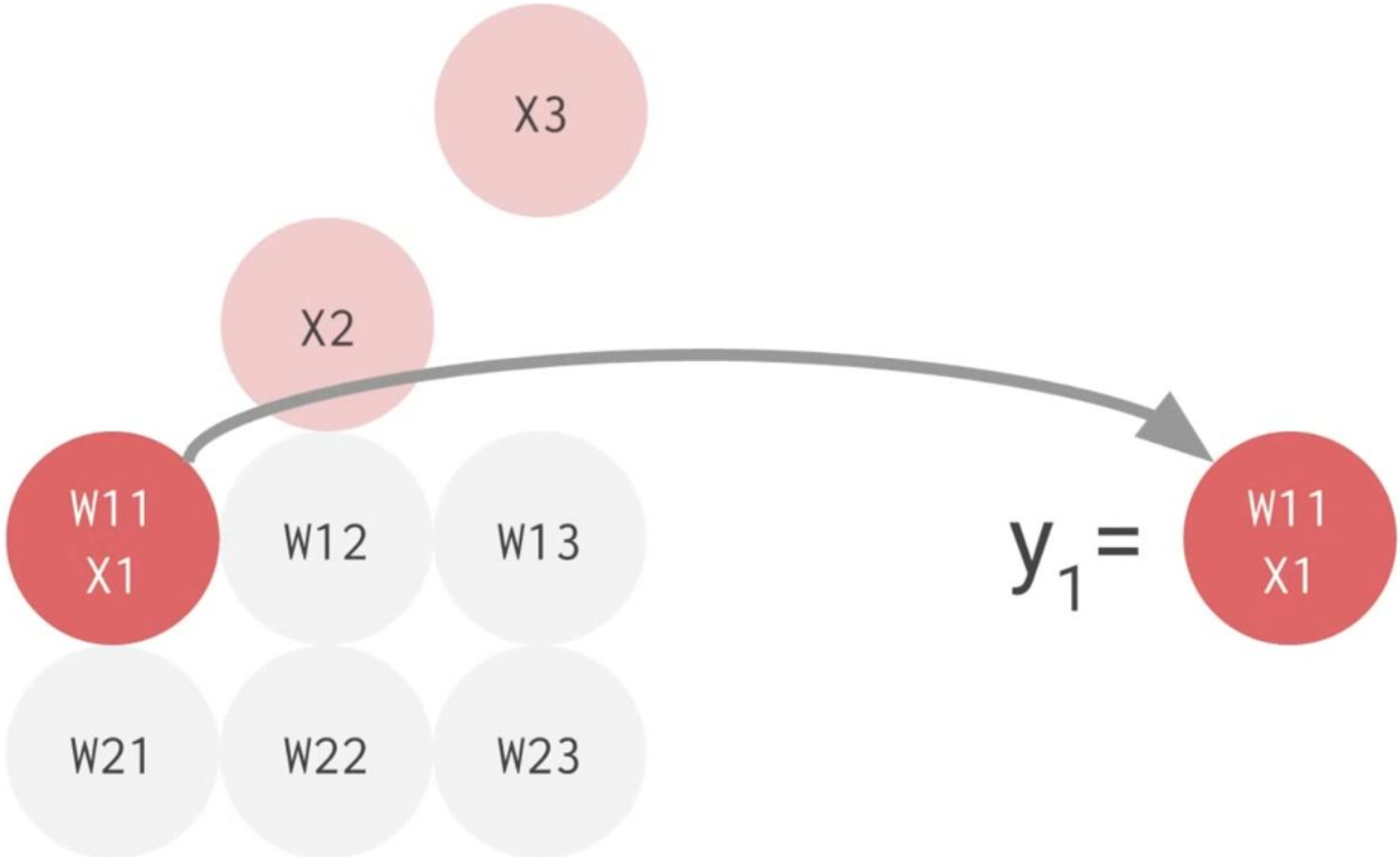
- Introduced starting from Volta generation (2017)
- Hardware unit/core optimized for matrix-matrix multiply
- Exploit data reuse
- (Supposedly) based on systolic array architecture
- Tensor = Multidimensional array

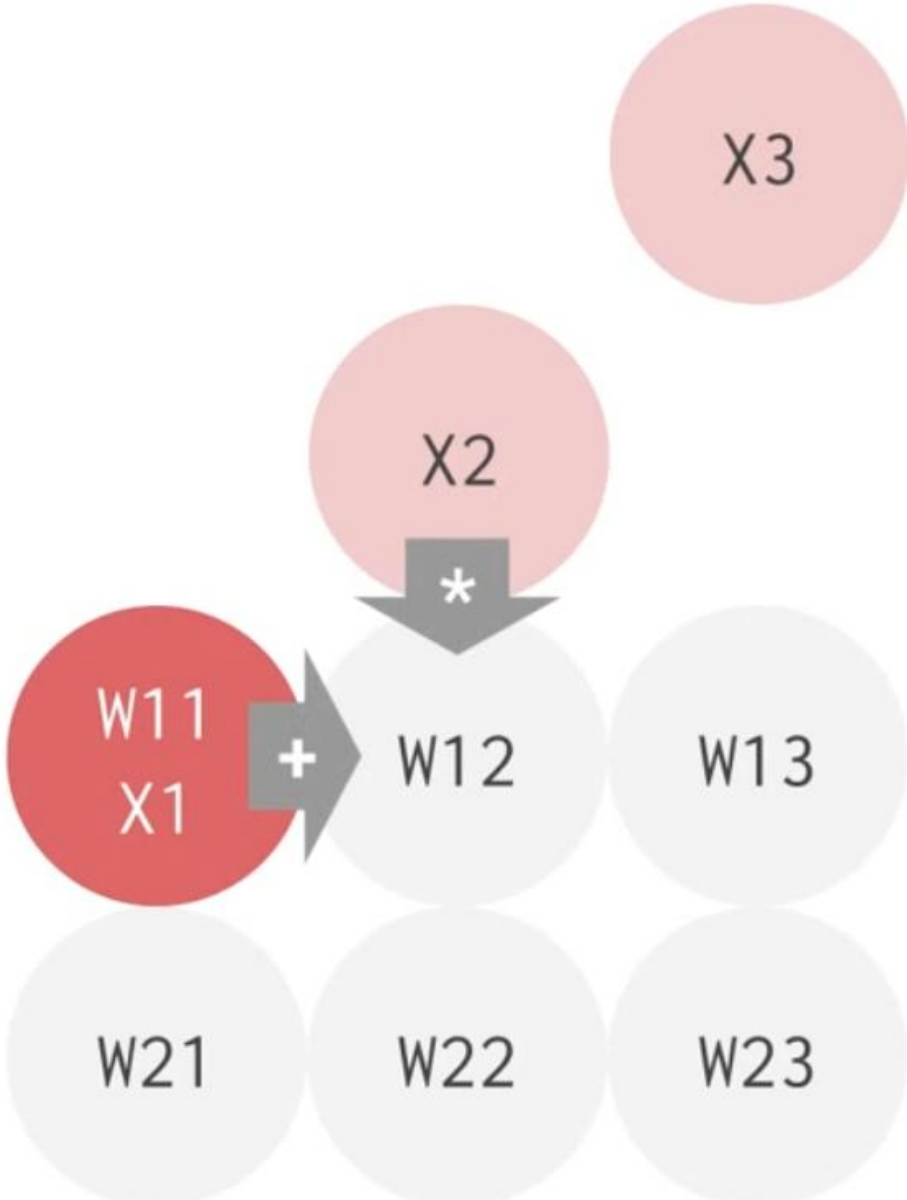
Systolic Array
Matrix-Vector Multiply

https://www.youtube.com/watch?v=nzVyQzv8FG8

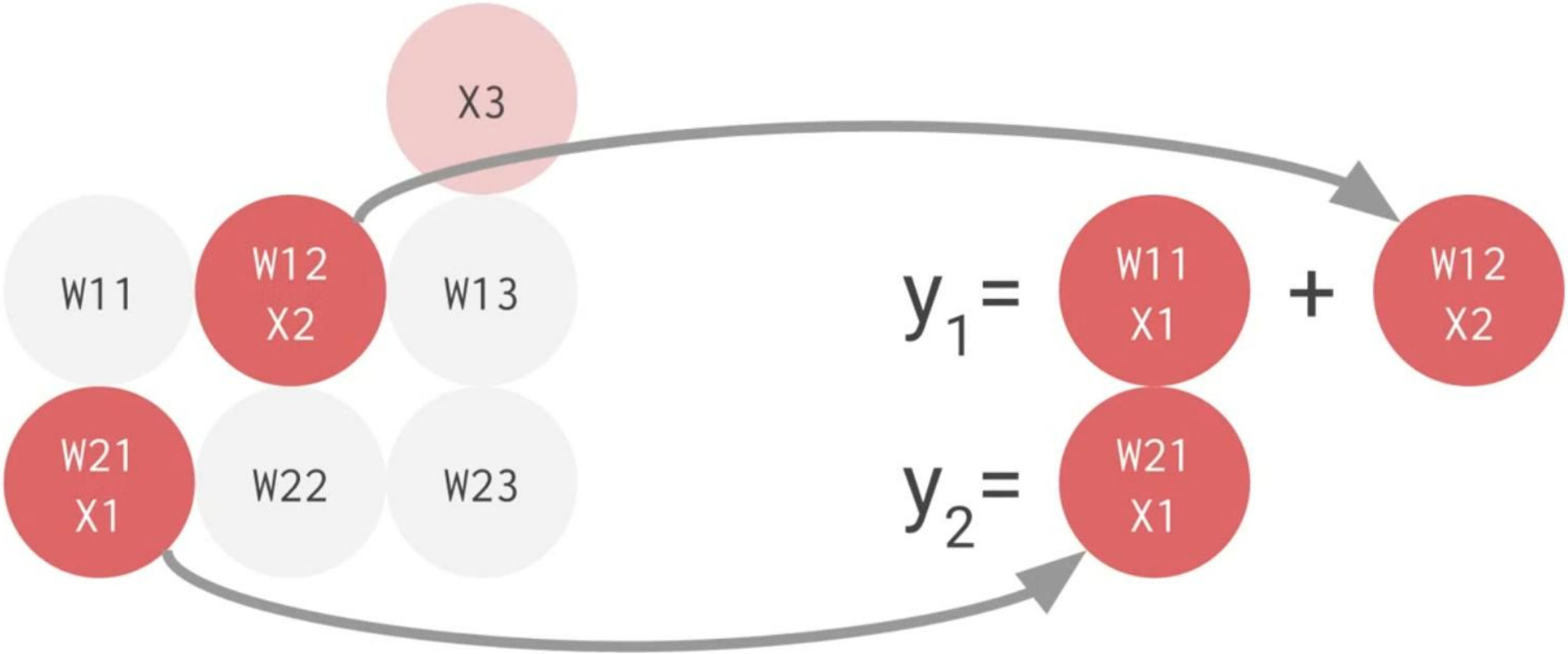

X3
X2
X1
W11
W12
W13
W21
W22
W23

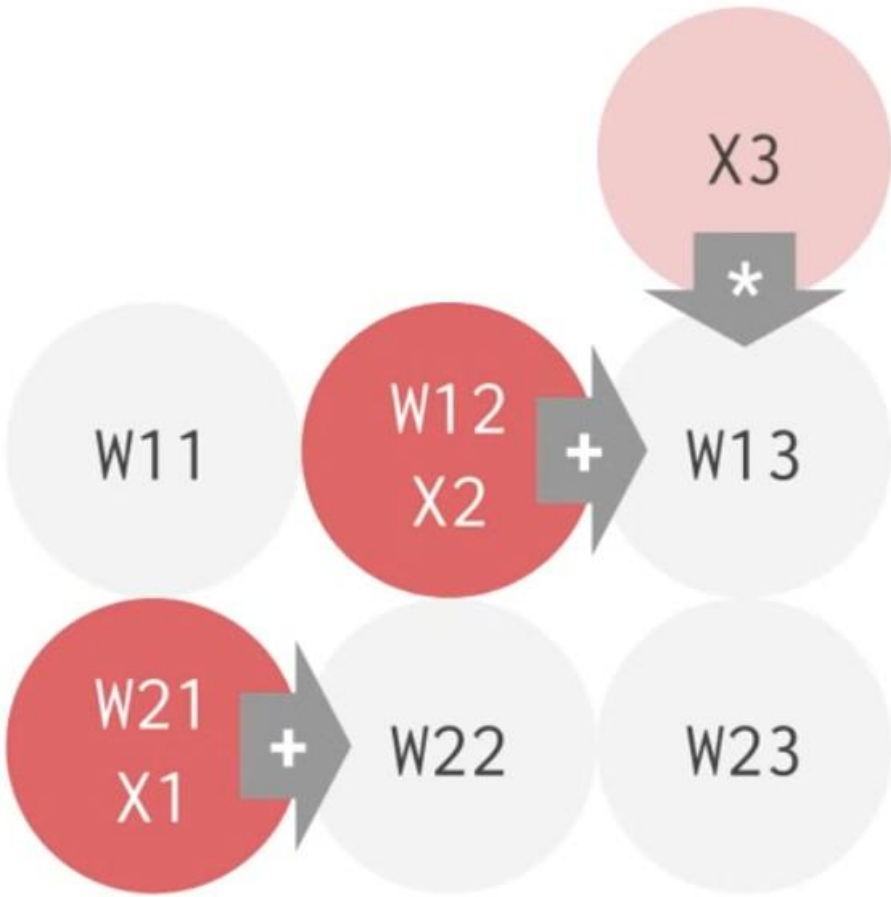

X3
X2
X1
*
W11
W12
W13
W21
W22
W23


X3
X2
W11
X1
W12
W13
W21
W22
W23
y1 =
W11
X1



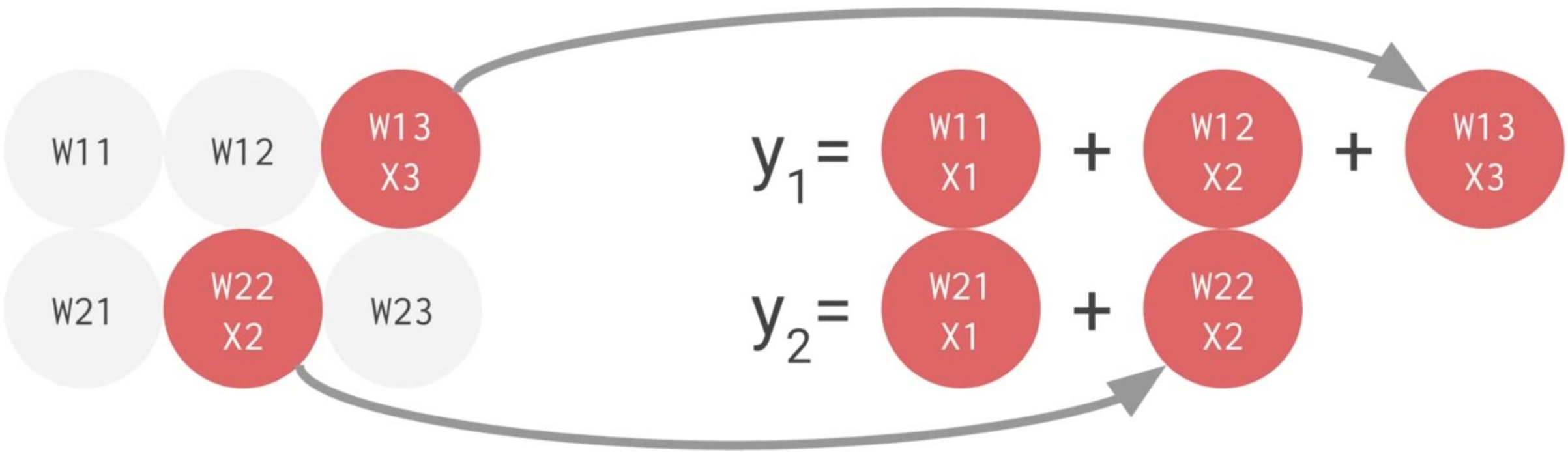
$y_1 =$ W11 X1


X3
W11
W12
X2
W13
W21
X1
W22
W23
y1=
W11
X1
+
W12
X2
y2=
W21
X1



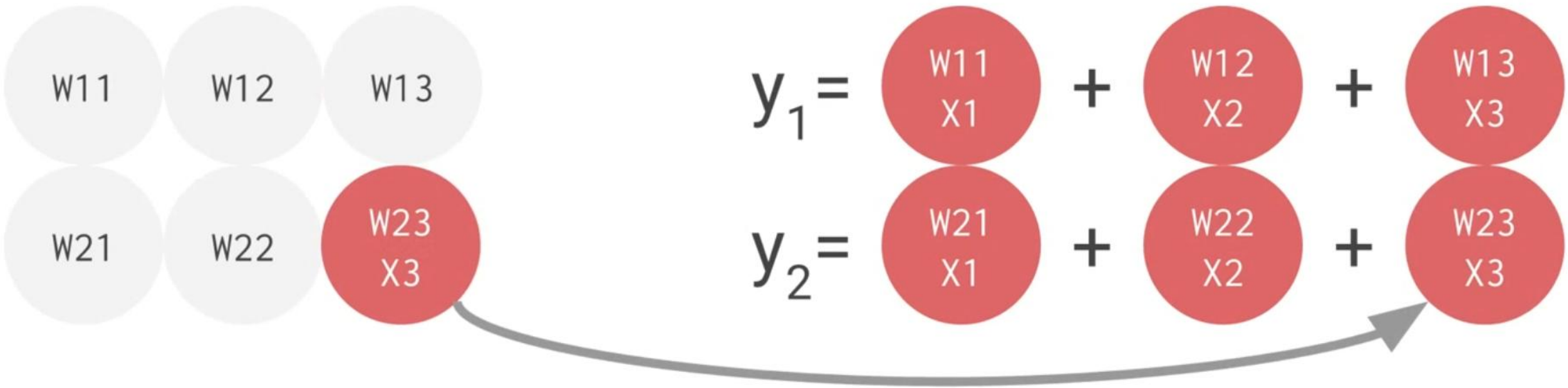
$$y_1 = W11\ X1 + W12\ X2$$

$$y_2 = W21\ X1$$


W11
W12
W13
X3
W21
W22
X2
W23
y1 =
W11
X1
+
W12
X2
+
W13
X3
y2 =
W21
X1
+
W22
X2

$$y_1 = W11\,X1 + W12\,X2 + W13\,X3$$

$$y_2 = W21\,X1 + W22\,X2$$


W11
W12
W13
W21
W22
W23
X3
y_1=
W11
X1
+
W12
X2
+
W13
X3
y_2=
W21
X1
+
W22
X2
+
W23
X3

$$y_1 = W11\,X1 + W12\,X2 + W13\,X3$$

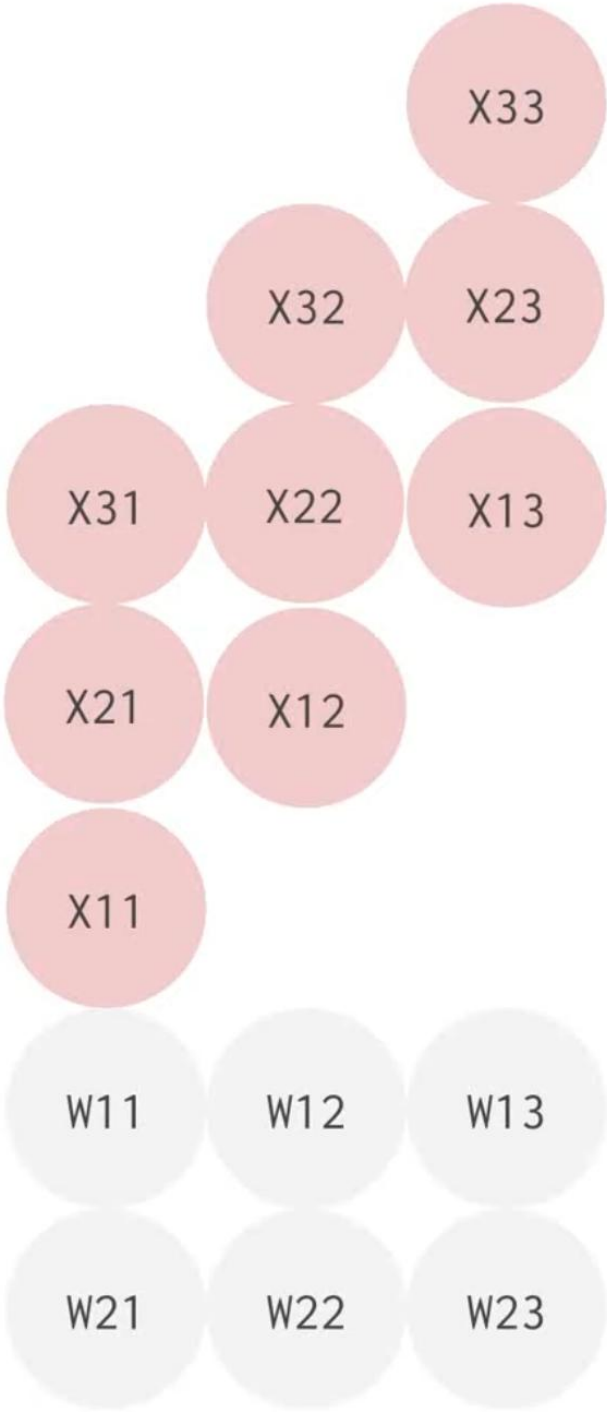
$$y_2 = W21\,X1 + W22\,X2 + W23\,X3$$

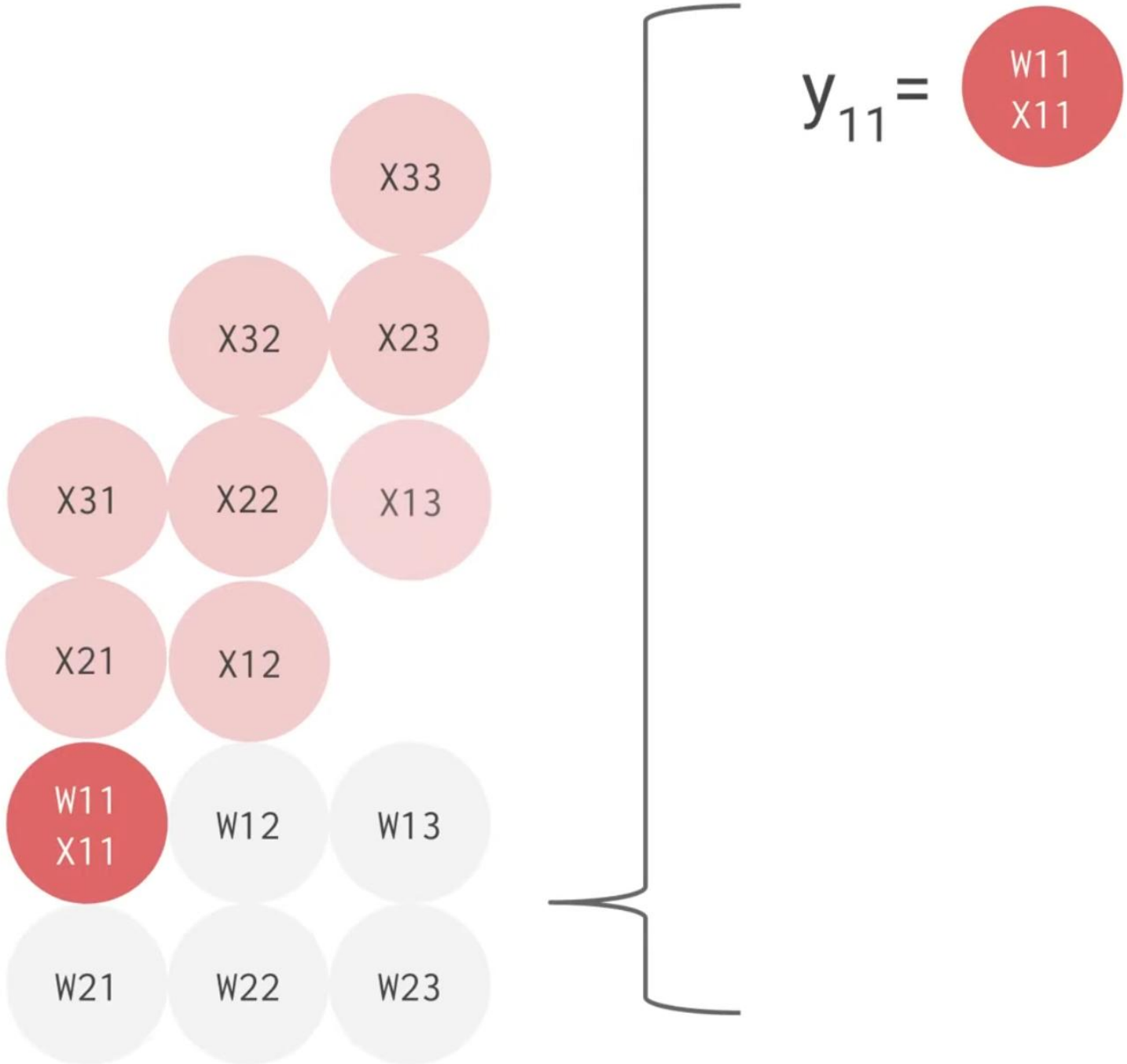
Performance

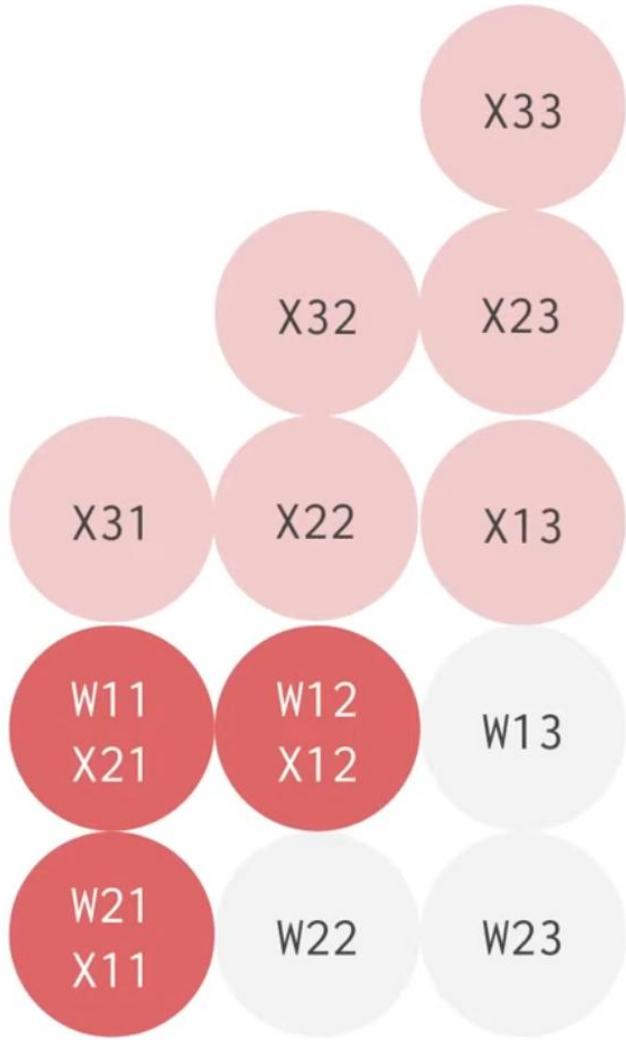
- 4x FMADD
- You would need 6x FMADD on a traditional architecture (+LOADS/STORES)

Systolic Array
Matrix-Matrix Multiply

https://www.youtube.com/watch?v=eK5fjuEFJuO


X33
X32
X23
X31
X22
X13
X21
X12
X11
W11
W12
W13
W21
W22
W23

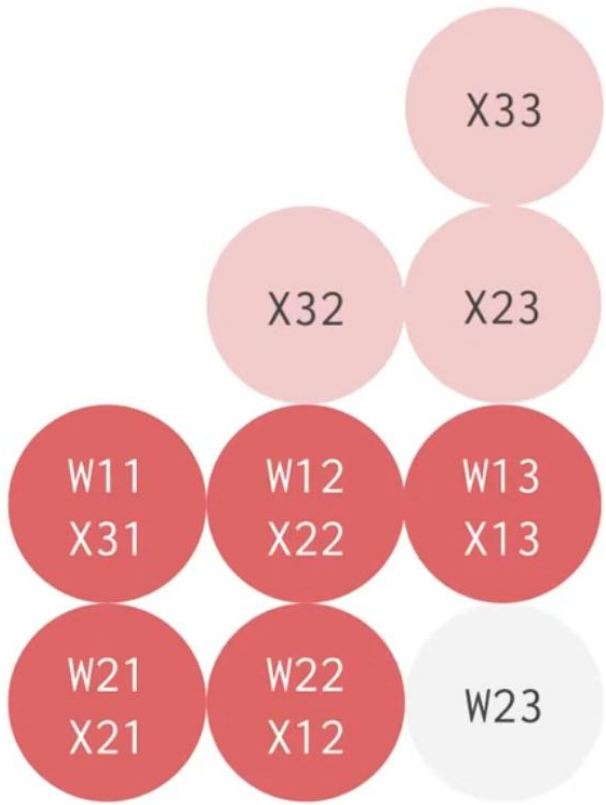

X33
X32
X23
X31
X22
X13
X21
X12
W11
X11
W12
W13
W21
W22
W23
y11 =
W11
X11



$y_{11} = W11\ X11 + W12\ X12$

$y_{12} = W21\ X11$

$y_{21} = W11\ X21$



$y_{11} =$ W11 X11 + W12 X12 + W13 X13

$y_{12} =$ W21 X11 + W22 X12

$y_{21} =$ W11 X21 + W12 X22

$y_{22} =$ W21 X21

$y_{31} =$ W11 X31

$$y_{11} = W11\,X11 + W12\,X12 + W13\,X13$$

$$y_{12} = W21\,X11 + W22\,X12 + W23\,X13$$

$$y_{21} = W11\,X21 + W12\,X22 + W13\,X23$$

$$y_{22} = W21\,X21 + W22\,X22$$

$$y_{31} = W11\,X31 + W12\,X32$$

$$y_{32} = W21\,X31$$

$$y_{11} = W11\,X11 + W12\,X12 + W13\,X13$$

$$y_{12} = W21\,X11 + W22\,X12 + W23\,X13$$

$$y_{21} = W11\,X21 + W12\,X22 + W13\,X23$$

$$y_{22} = W21\,X21 + W22\,X22 + W23\,X23$$

$$y_{31} = W11\,X31 + W12\,X32 + W13\,X33$$

$$y_{32} = W21\,X31 + W22\,X32$$

$$y_{11} = W11\,X11 + W12\,X12 + W13\,X13$$

$$y_{12} = W21\,X11 + W22\,X12 + W23\,X13$$

$$y_{21} = W11\,X21 + W12\,X22 + W13\,X23$$

$$y_{22} = W21\,X21 + W22\,X22 + W23\,X23$$

$$y_{31} = W11\,X31 + W12\,X32 + W13\,X33$$

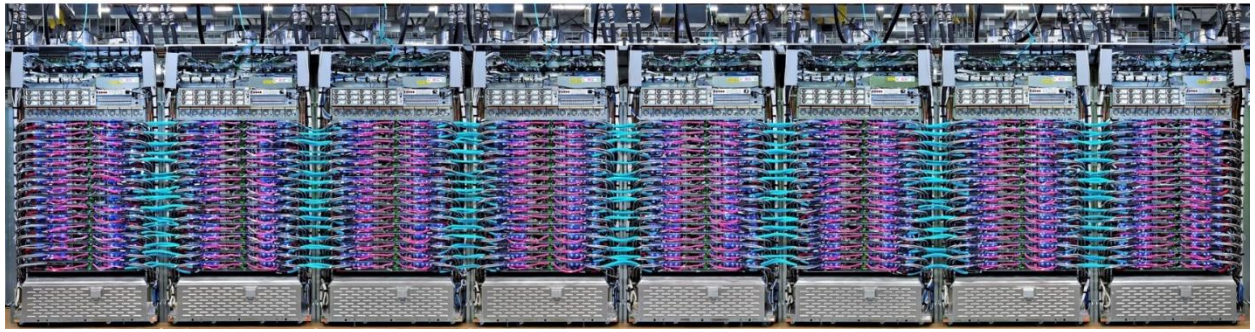
$$y_{32} = W21\,X31 + W22\,X32 + W23\,X33$$

Performance

- 6x FMADD
- You would need 18x FMADD on a traditional architecture (+LOADS/STORES)
- Performance gain comes from both parallelism when doing multiple FMADDs at the same time but, most importantly, from avoiding to LOAD/STORE data from global memory. Data is streamed directly between FMADD units

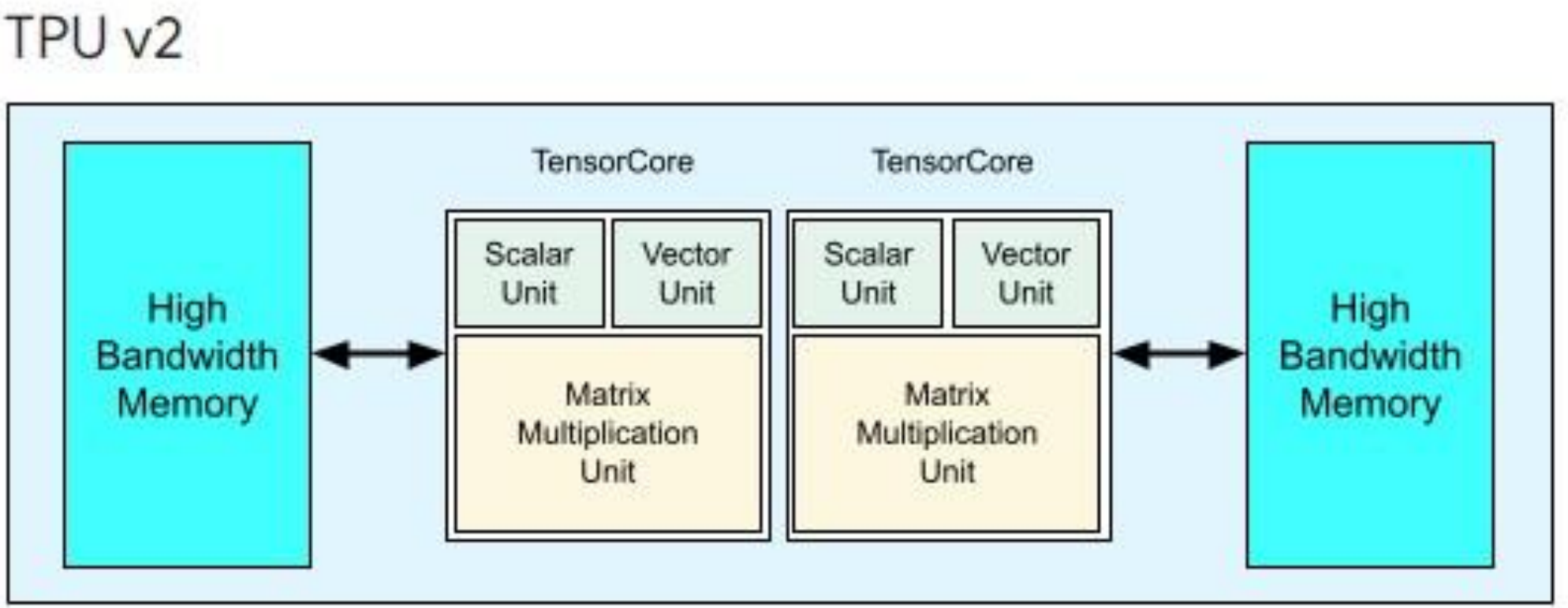
Questions?

TPUs



- It's an accelerator, similar to a GPU (CPU feeds the data etc..)
- Domain-specific architecture (DSA)
- TPU stands for *Tensor Processing Units*
- Introduced by Google in 2015, made available to generic public in 2018
- Largely based on the *systolic array* architecture (1979)

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

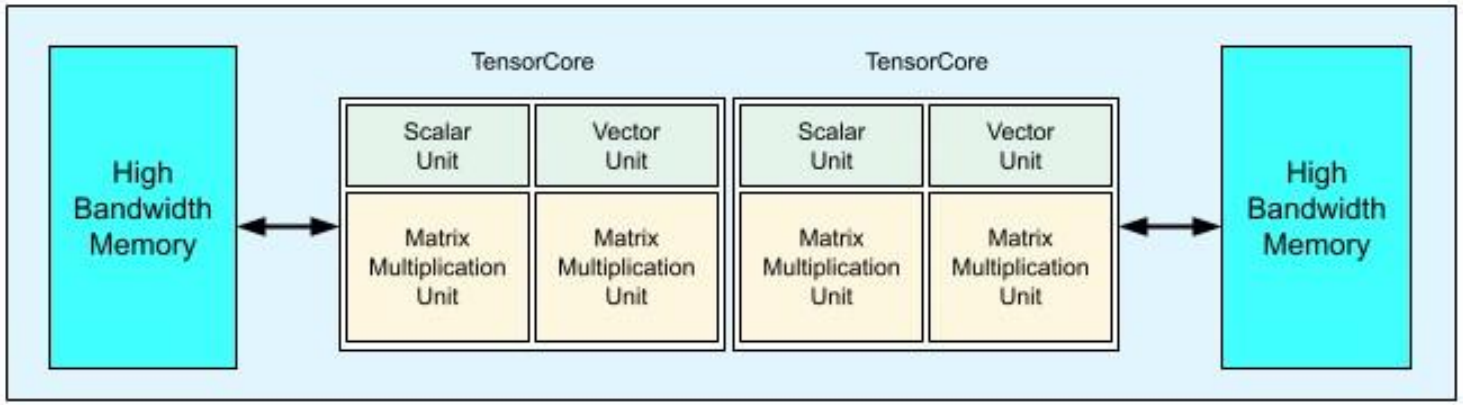


- Has scalar units and vector units
- Key feature is the Matrix Multiplication Unit (similar to tensor cores)

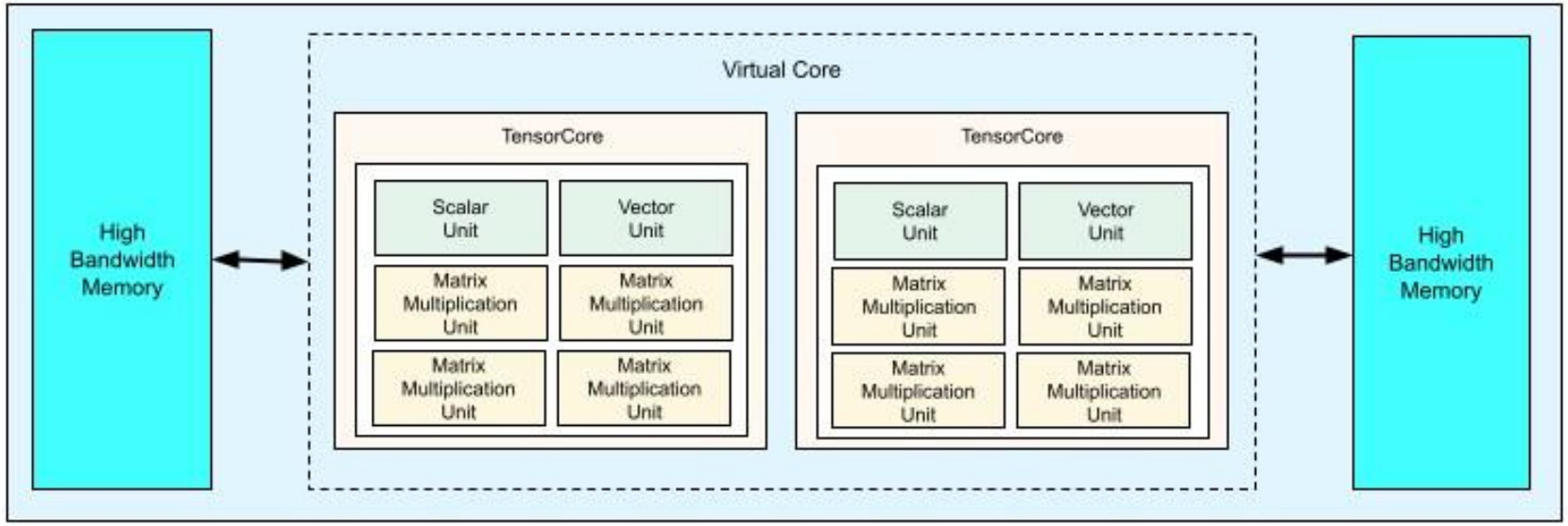
https://www.nextplatform.com/2022/10/11/deep-dive-on-googles-exascale-tpuv4-ai-systems/

TPUs

TPU v3



TPU v4

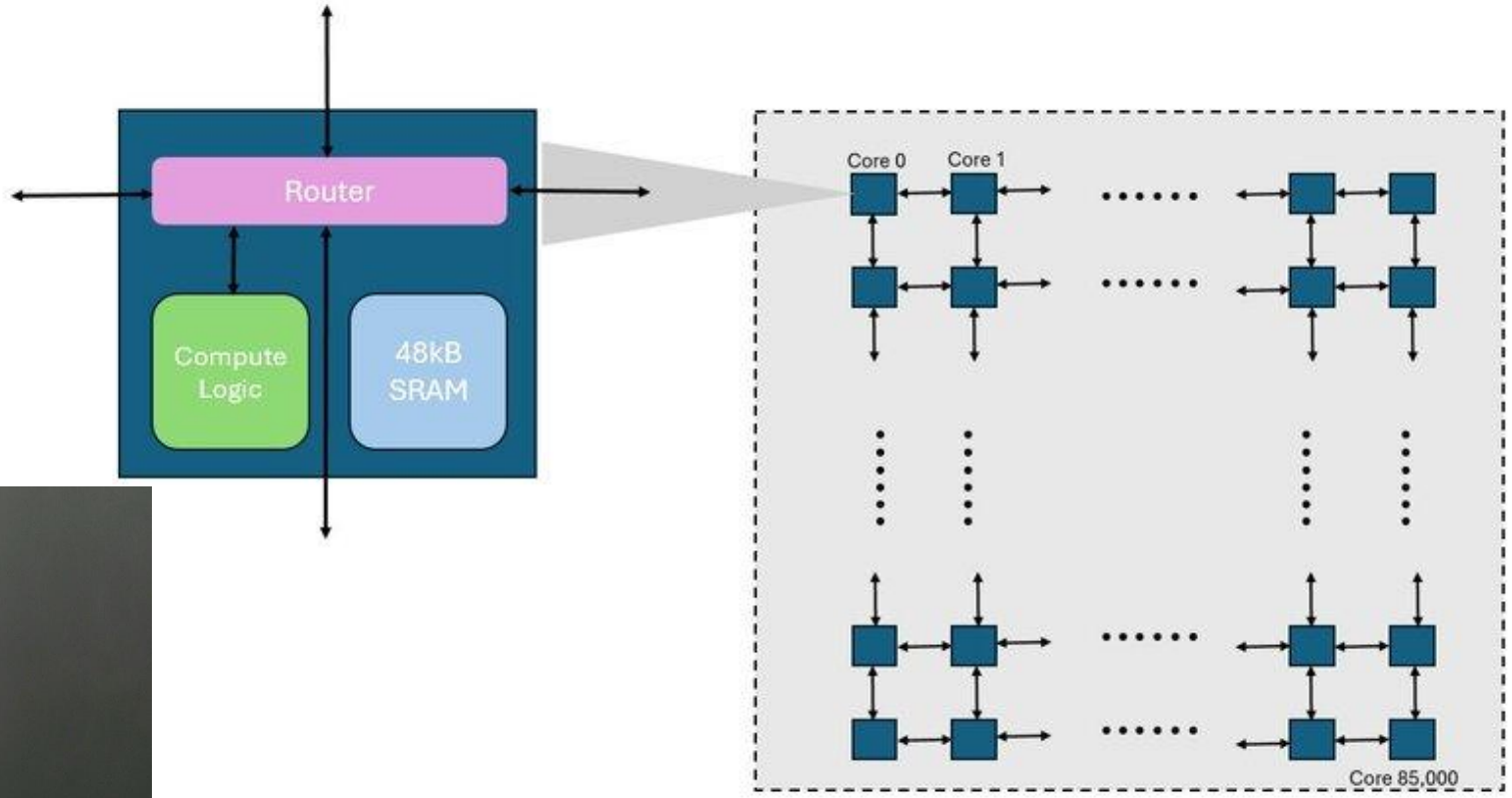


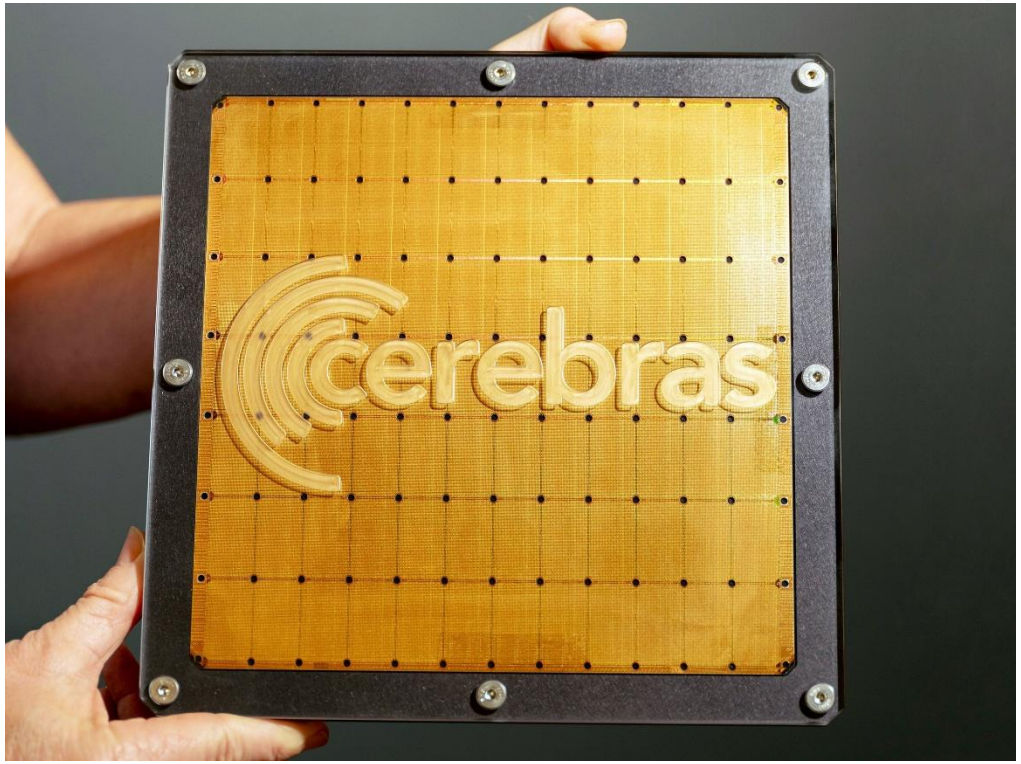
https://www.nextplatform.com/2022/10/11/deep-dive-on-googles-exascale-tpuv4-ai-systems/

Matrix Multiplication Unit (MXU)

- Architecture optimized for tensor operations (e.g., matrix multiplication)
- It does not work according to the standard Von-Neumann architecture we have in CPUs and GPUs
- Is a spatial/dataflow/non-Von-Neumann architecture

Spatial/Dataflow Architectures – Cerebras WSE





- 800,000 cores
- Local (small) SRAM on each core
- Cooperation by exchanging data between cores
- Cores connected in a 2D mesh

Other emerging architectures

Lot of startups coming out, most of them will fail, a few will thrive.
Some recent/future architectures:

- Tenstorrent
- Graphcore
- Habana Gaudi
- SambaNova
- Tesla Dojo
- IBM's NorthPole
- …

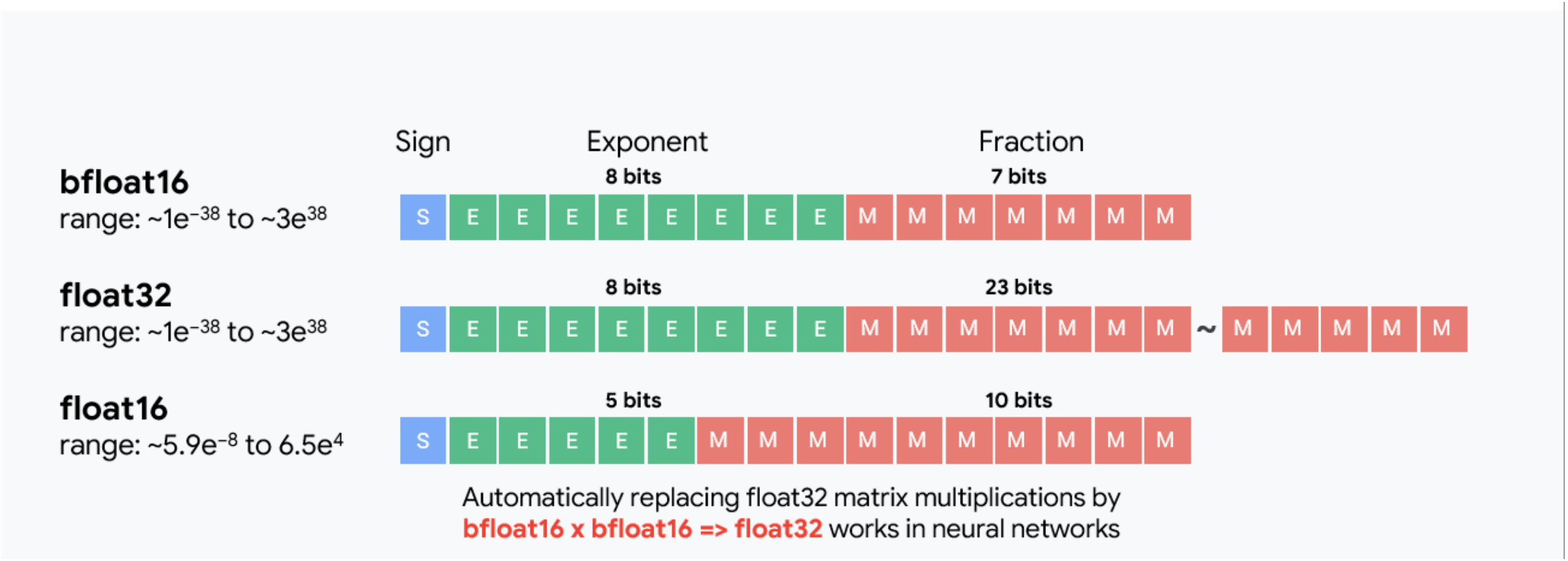
HPC Systems Today

HPCG lower efficiency caused by data movements

Different ways to improve systems efficiency:

- Increase data movement efficiency within the node
  - Traditional von-Neumann architectures
  - Non-traditional architectures
  - Reduced-precision Data Types
- Increase data movement efficiency between the nodes
  - Scale-out Network
  - Scale-up Network
  - Programming models/APIs

Reduced Precision



- Are these new formats acceptable for traditional HPC applications?
  - Some researchers complain that even 64 bits are not enough
- Chip design is driven by money (i.e., nowadays design heavily influenced by AI, which works as fine with lower precision)

HPC Systems Today

HPCG lower efficiency caused by data movements

Different ways to improve systems efficiency:

- Increase data movement efficiency within the node
  - Traditional von-Neumann architectures
  - Non-traditional architectures
  - Reduced-precision Data Types
- Increase data movement efficiency between the nodes
  - Scale-out Network
  - Scale-up Network
  - Programming models/APIs

HPC Systems Today – Top500

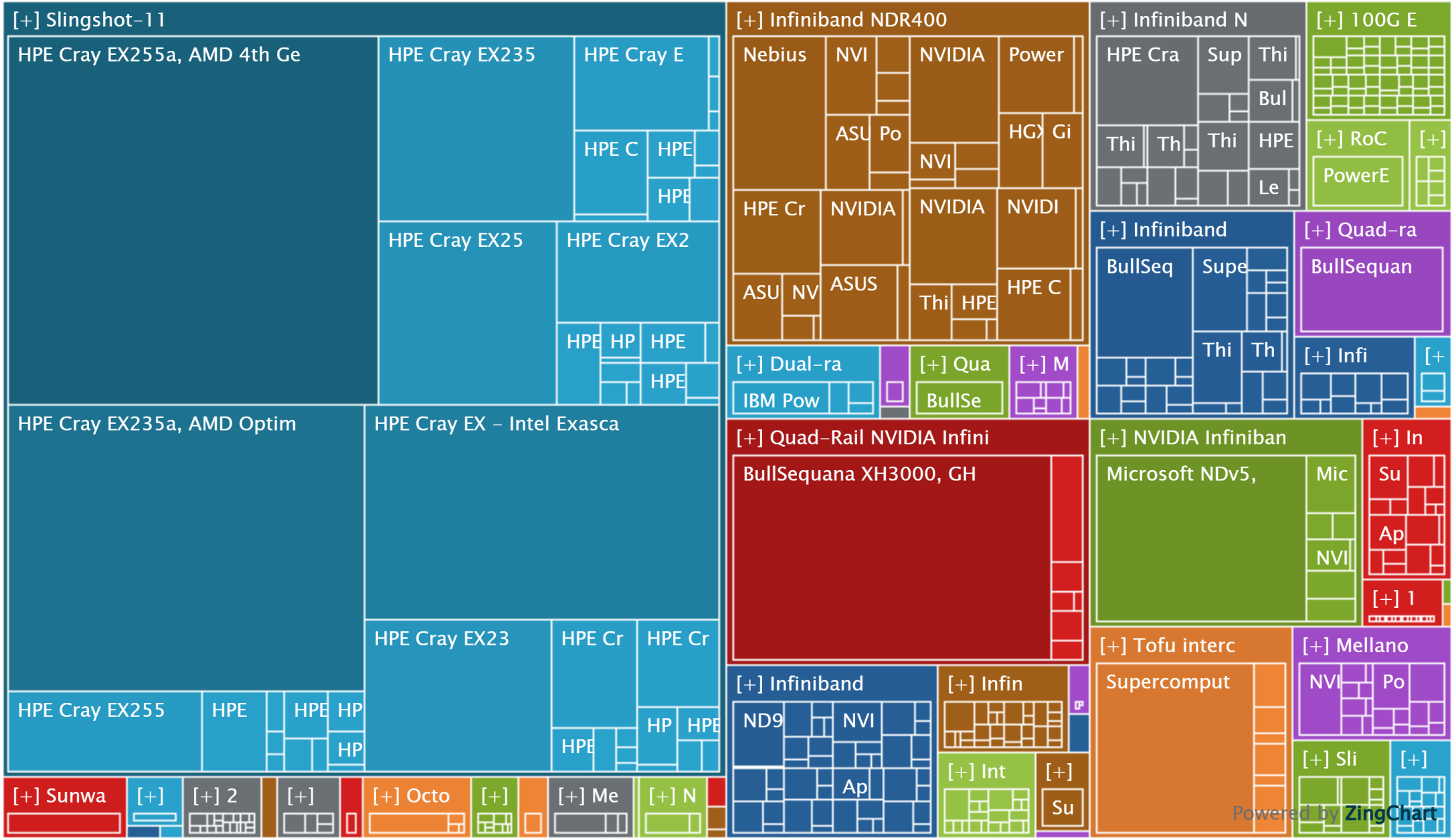
Network Speed Acceleration over the years

Ethernet (1979 -)	10 Mbit/sec
Fast Ethernet (1993 -)	100 Mbit/sec
Gigabit Ethernet (1995 -)	1000 Mbit /sec
ATM (1995 -)	155/622/1024 Mbit/sec
Myrinet (1993 -)	1 Gbit/sec
Fibre Channel (1994 -)	1 Gbit/sec
InfiniBand (2001 -)	2 Gbit/sec (1X SDR)
10-Gigabit Ethernet (2001 -)	10 Gbit/sec
InfiniBand (2003 -)	8 Gbit/sec (4X SDR)
InfiniBand (2005 -)	16 Gbit/sec (4X DDR)
	24 Gbit/sec (12X SDR)
InfiniBand (2007 -)	32 Gbit/sec (4X QDR)
40-Gigabit Ethernet (2010 -)	40 Gbit/sec
InfiniBand (2011 -)	54.6 Gbit/sec (4X FDR)
InfiniBand (2012 -)	2 x 54.6 Gbit/sec (4X Dual-FDR)
25-/50-Gigabit Ethernet (2014 -)	25/50 Gbit/sec
100-Gigabit Ethernet (2015 -)	100 Gbit/sec
Omni-Path (2015 -)	100 Gbit/sec
InfiniBand (2015 -)	100 Gbit/sec (4X EDR)
InfiniBand (2017 -)	200 Gbit/sec (4X HDR)
Slingshot10/11 (2021 -)	200 Gbit/sec
Omni-Path-Express (2021 -)	100 Gbit/sec
Google Aquila (2021 -)	100 Gbit/sec
InfiniBand (2022 -)	400 Gbit/sec (4X NDR)
Omni-Path-Express (2024 -)	400 Gbit/sec (CN5000)

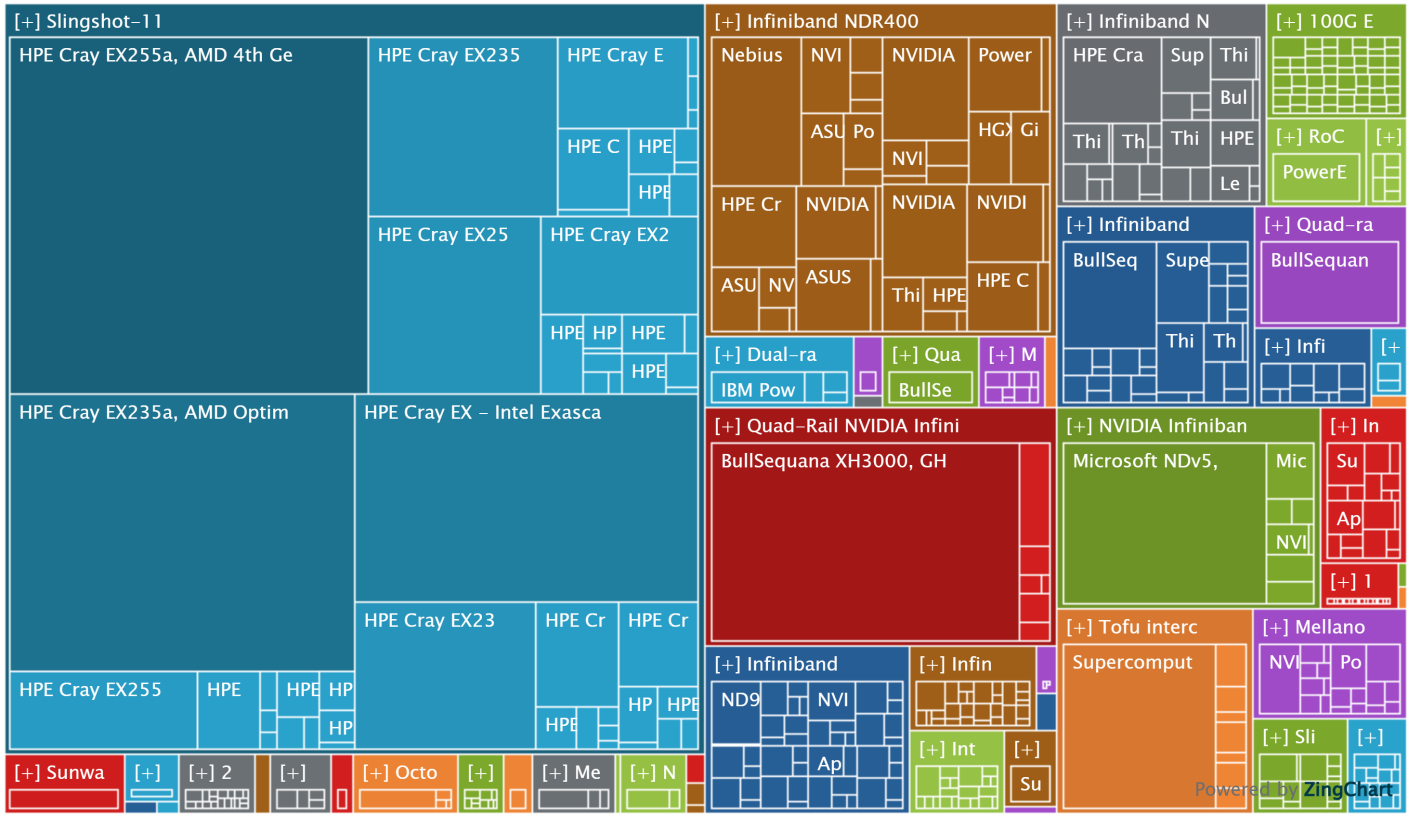
200 times in the last 24 years!!

https://hoti.org/assets/slides/2025_08_22_day3_Tutorial_Track1_HPNetworks.pdf

HPC Systems Today - Interconnect



HPC Systems Today - Interconnect

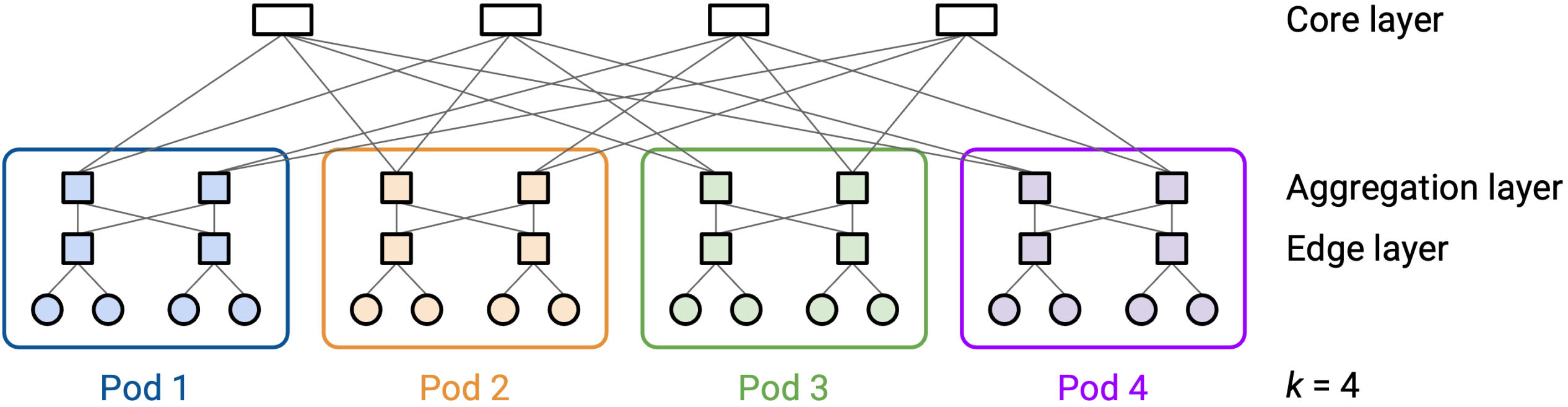


Almost all are RDMA networks

Most Widely Used Scale-Out Networks

- Infiniband/RoCE
- Slingshot
- OmniPath
- UltraEthernet (coming soon)

How to connect the compute nodes?



Other alternatives: Dragonfly, Dragonfly+, torus, HxMesh, etc…

https://textbook.cs168.io/datacenter/topology.html

HPC Systems Today

HPCG lower efficiency caused by data movements

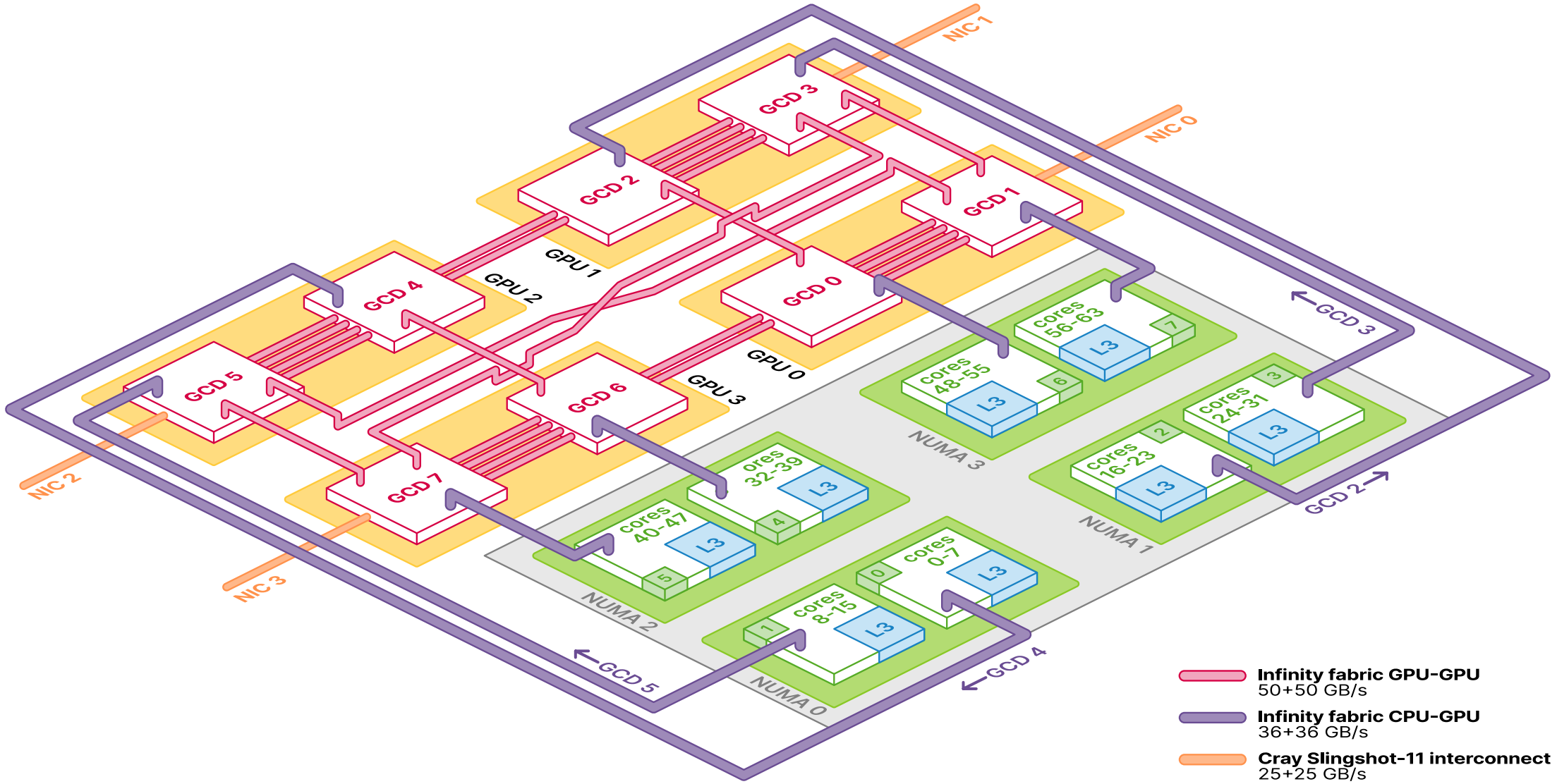
Different ways to improve systems efficiency:

- Increase data movement efficiency within the node
  - Traditional von-Neumann architectures
  - Non-traditional architectures
  - Reduced-precision Data Types
- Increase data movement efficiency between the nodes
  - Scale-out Network
  - Scale-up Network
  - Programming models/APIs

Existing Scale-Up Networks

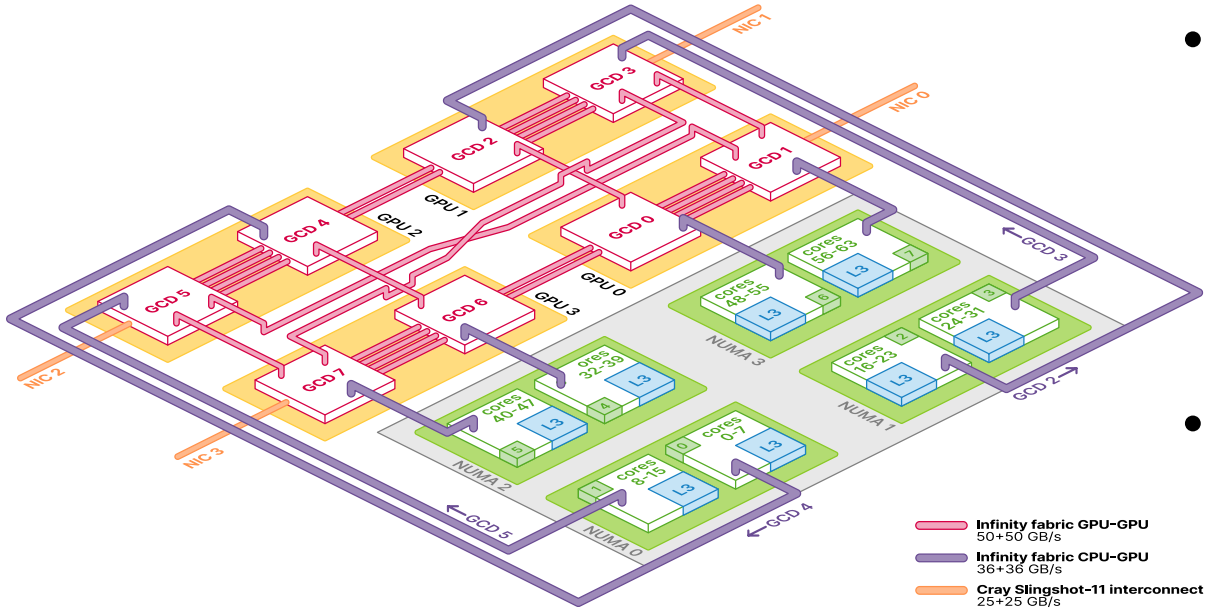
- NVLink (NVIDIA)
- InfinityFabric (AMD)
- Scale-Up Ethernet (open standard, backed by Broadcom)
- UALink (open standard, backed by AMD)

Scale-up Networks - MI250X/LUMI



https://docs.lumi-supercomputer.eu/hardware/lumig/

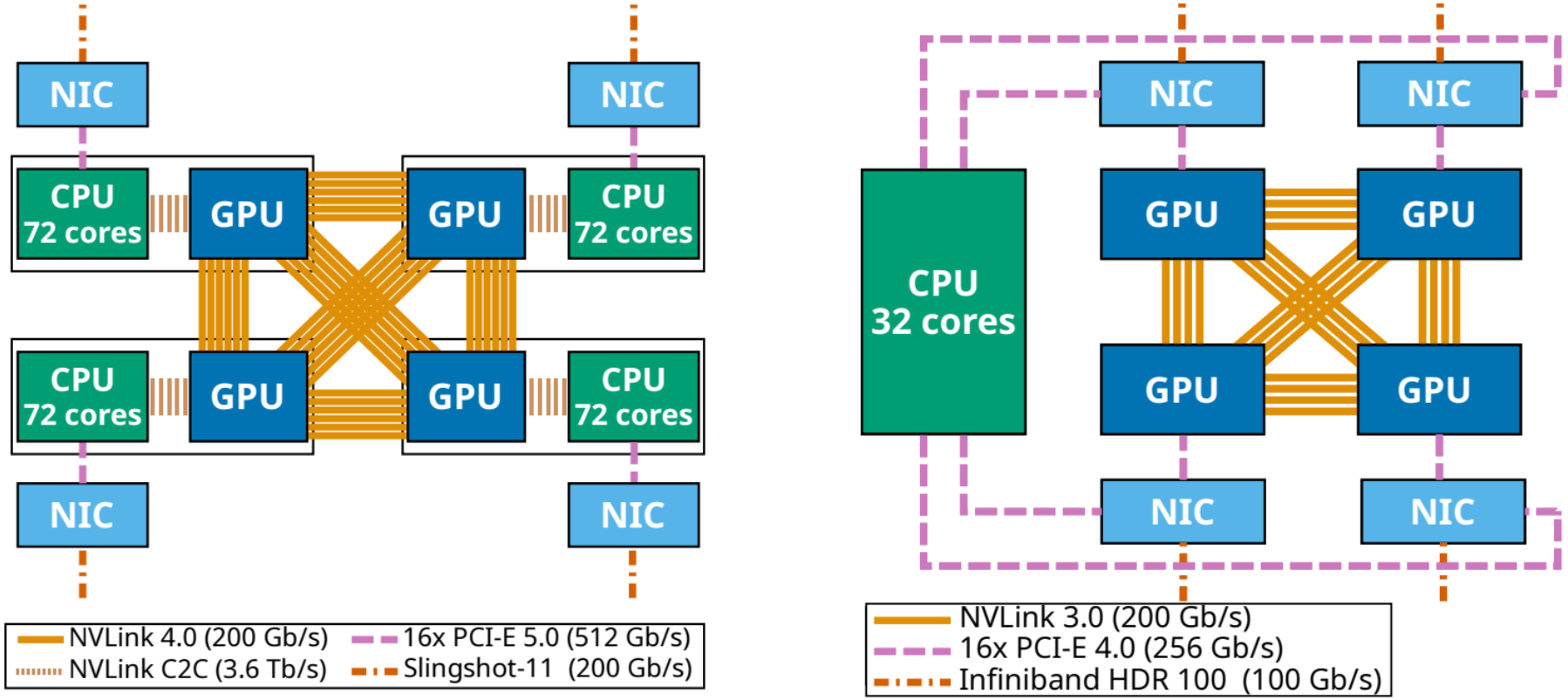
Scale-up Networks – MI250X/LUMI



- Communication between GPUs on the same server has a higher bandwidth compared to communication between GPUs on different servers
- This also depends on the specific GPU location (e.g., GPU 0 and 1 are connected with three links, but GPU 0 and 6 with only two links)

https://docs.lumi-supercomputer.eu/hardware/lumig/

Scale-up Networks – Leonardo and LUMI

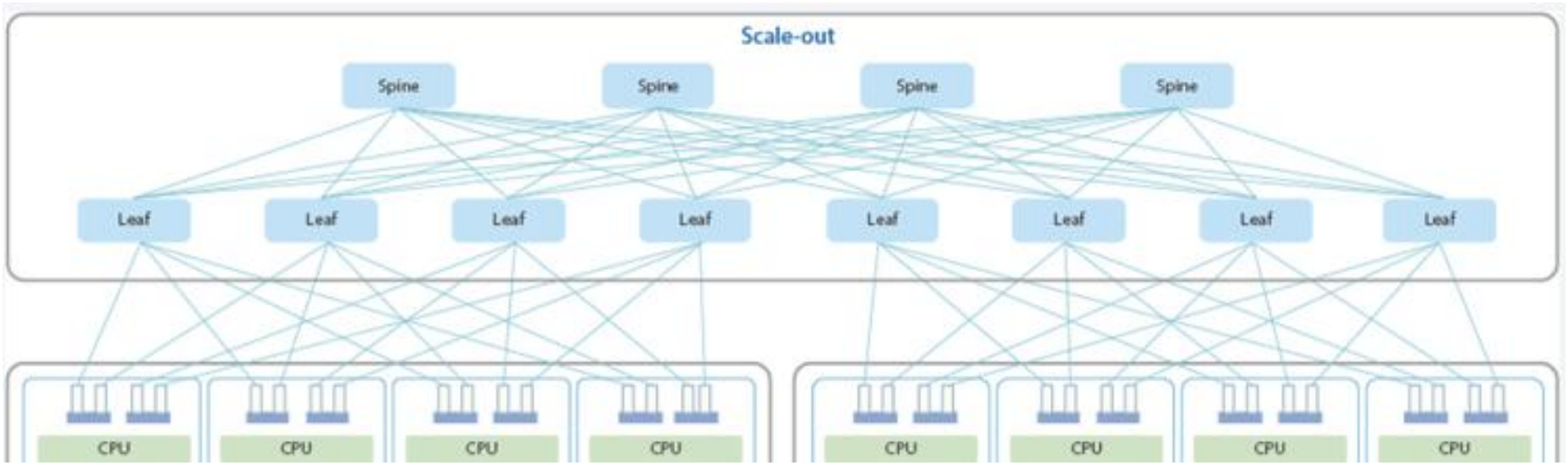


(a) Alps. (b) Leonardo.

Fig. 1: Alps and Leonardo (Booster) node architectures.

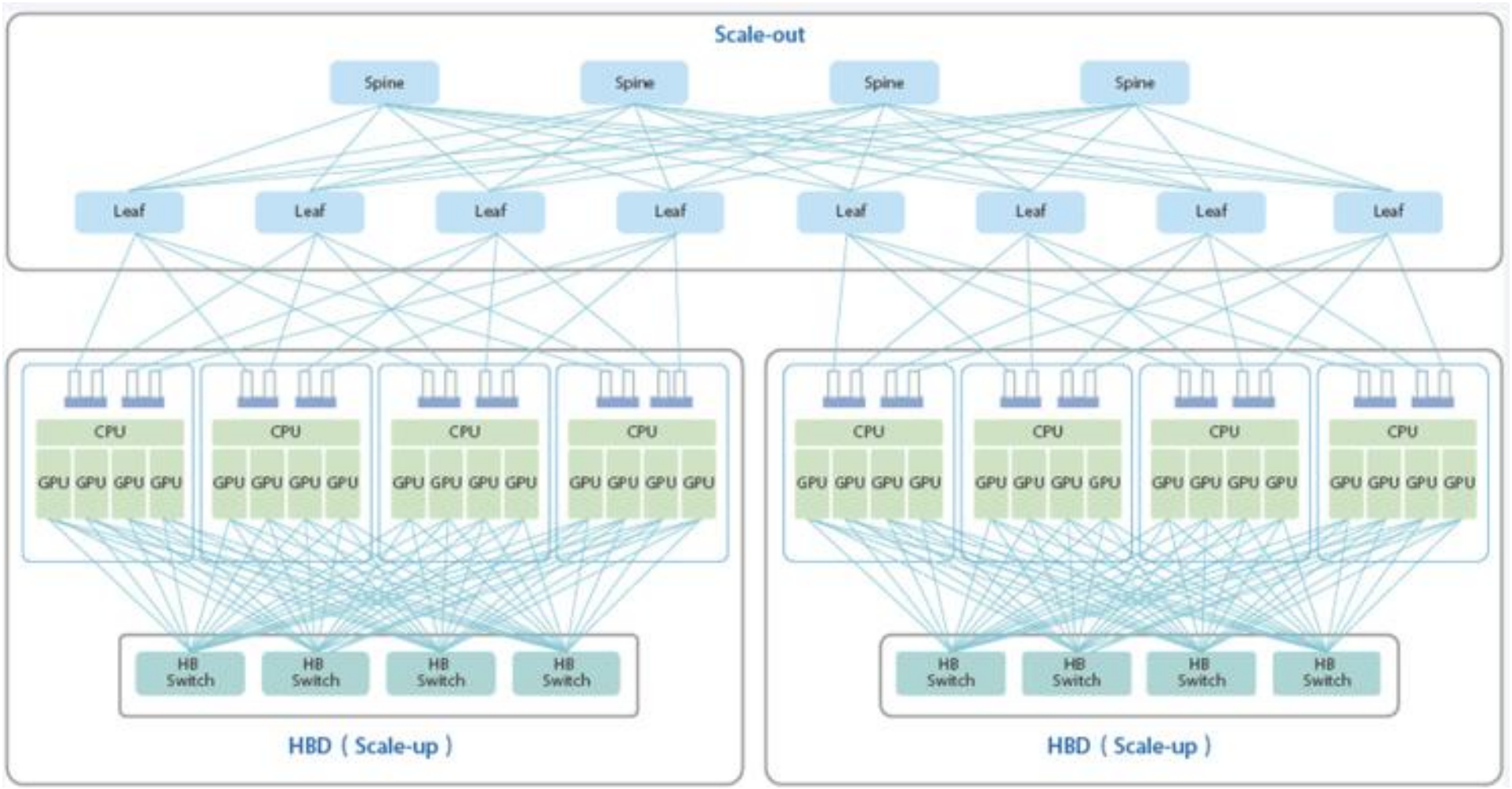
https://arxiv.org/pdf/2408.14090

Scale-up+Scale-out



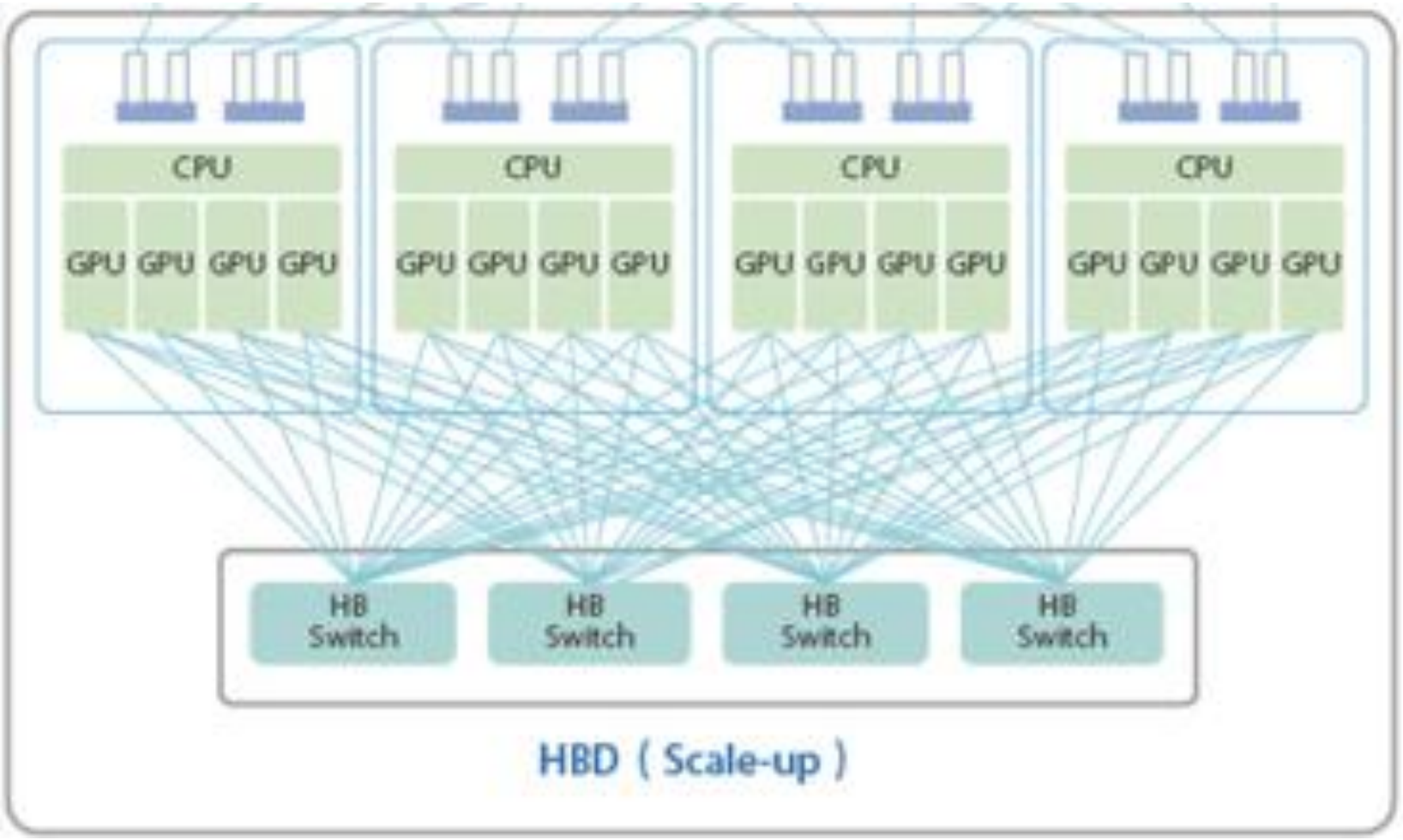
https://www.linkedin.com/pulse/understanding-scale-up-vs-scale-out-ai-infrastructure-ibelc

Scale-up+Scale-out



https://www.linkedin.com/pulse/understanding-scale-up-vs-scale-out-ai-infrastructure-ibelc

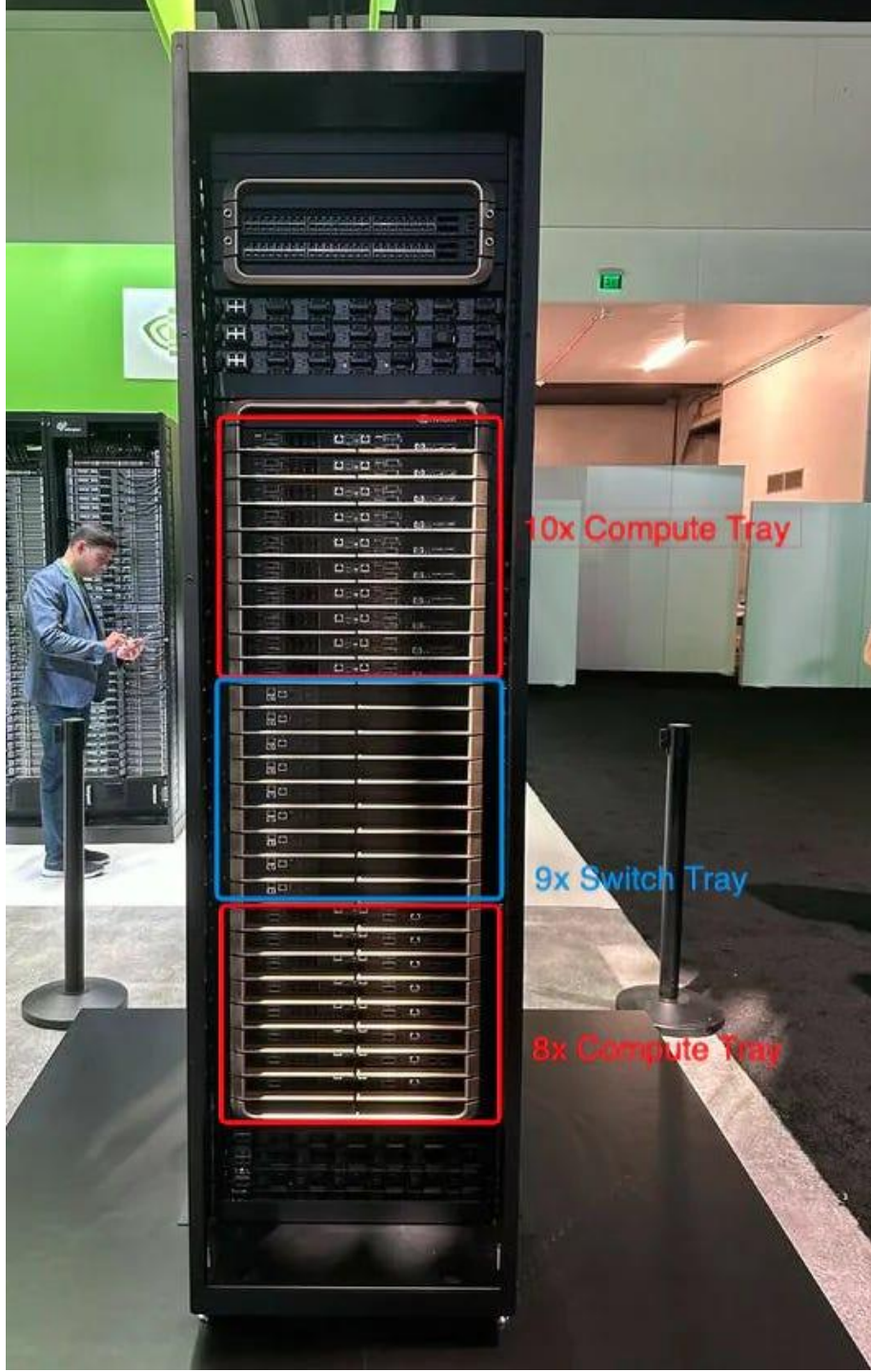
Scale-up – Multiple Planes



https://www.linkedin.com/pulse/understanding-scale-up-vs-scale-out-ai-infrastructure-ibelc

Scale-up – NVL72

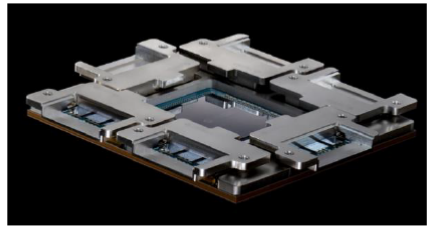
- 5000 copper links
- 1,36 tonnes (1360 kg)
- Cabling weight: 25%
- 150 KW



Slide courtesy: Fabrizio Petrini (Intel)
https://naddod.medium.com/nvidia-gb200-interconnect-architecture-analysis-nvlink-infiniband-and-future-trends-91dc6ba49bf3

NVL576 power estimates

Quantum-X 144 ports



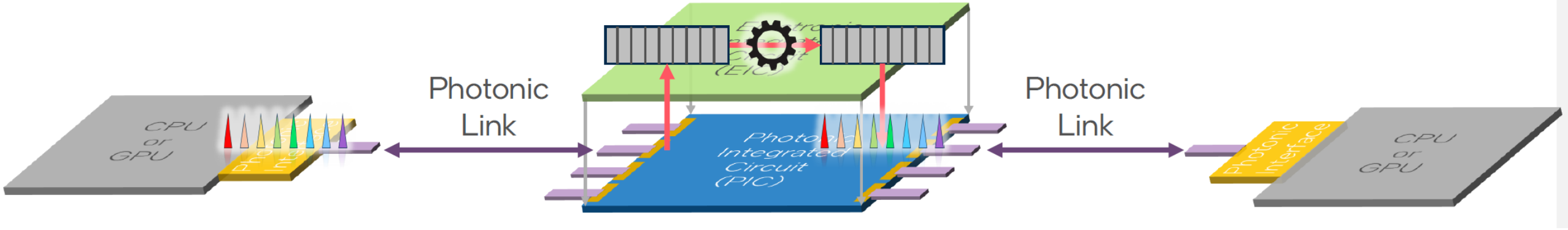
NVswitch 288 ports

Metric	Value
R400 Power (W)	3200
Quantum-X 144 ports power (W)	633
NVL576 GPU power	460,800
NVL576 NVswitch power (W)	**182,304**
Total NVL576 power (W)	643,104
Ratio switch/compute	**0.395625**
Ratio switch/system	**0.28346**

Slide courtesy: Fabrizio Petrini (Intel)

What's Next

- In the old days, the general rule was that the network cost had to be around 5% of the overall system cost
- Nowadays, the power consumption (and likely the cost) is around 30% (and even more)
- Optical networks can improve the range, simplify the design, and lower the power consumption of the network (not necessarily the cost, optic cables are expensive)



Slide courtesy: Fabrizio Petrini (Intel)

HPC Systems Today

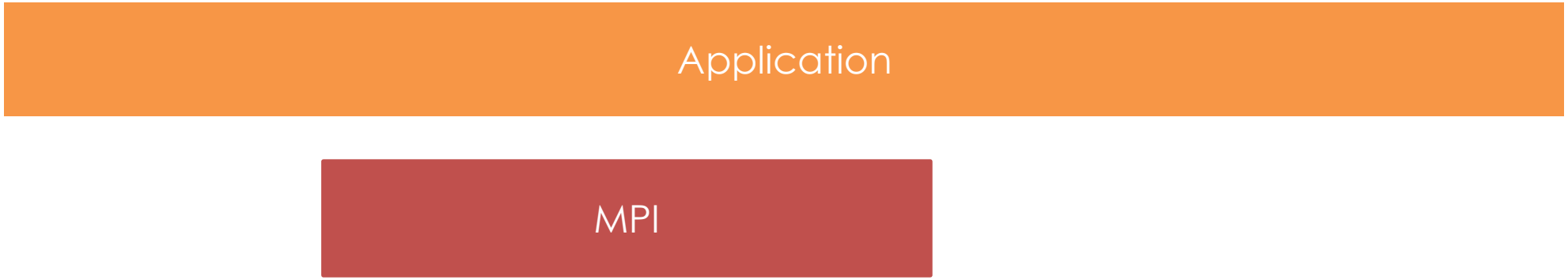
HPCG lower efficiency caused by data movements

Different ways to improve systems efficiency:

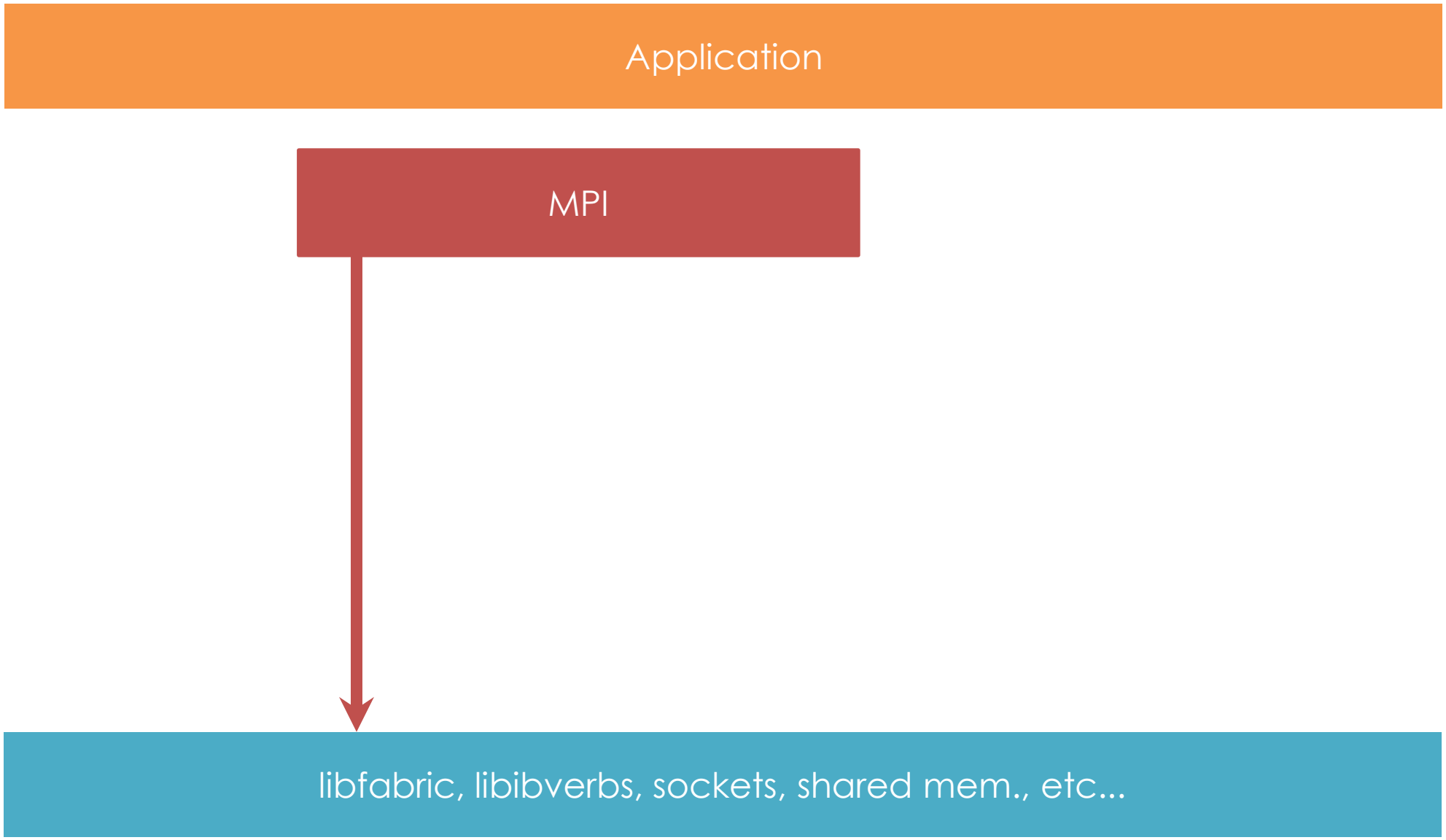
- Increase data movement efficiency within the node
  - Traditional von-Neumann architectures
  - Non-traditional architectures
  - Reduced-precision Data Types
- Increase data movement efficiency between the nodes
  - Scale-out Network
  - Scale-up Network
  - Programming models/APIs

Software Stack

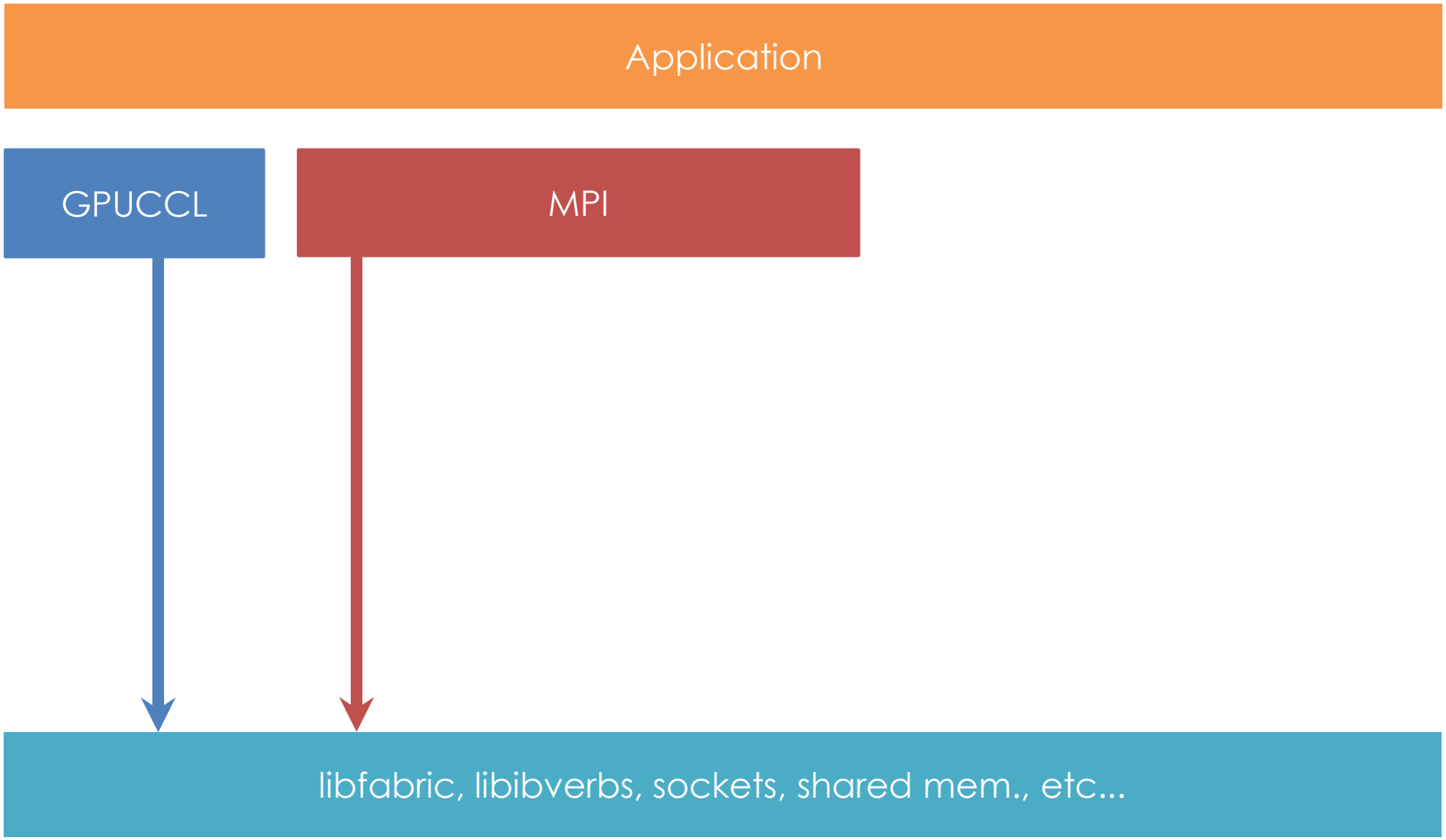
Software Stack

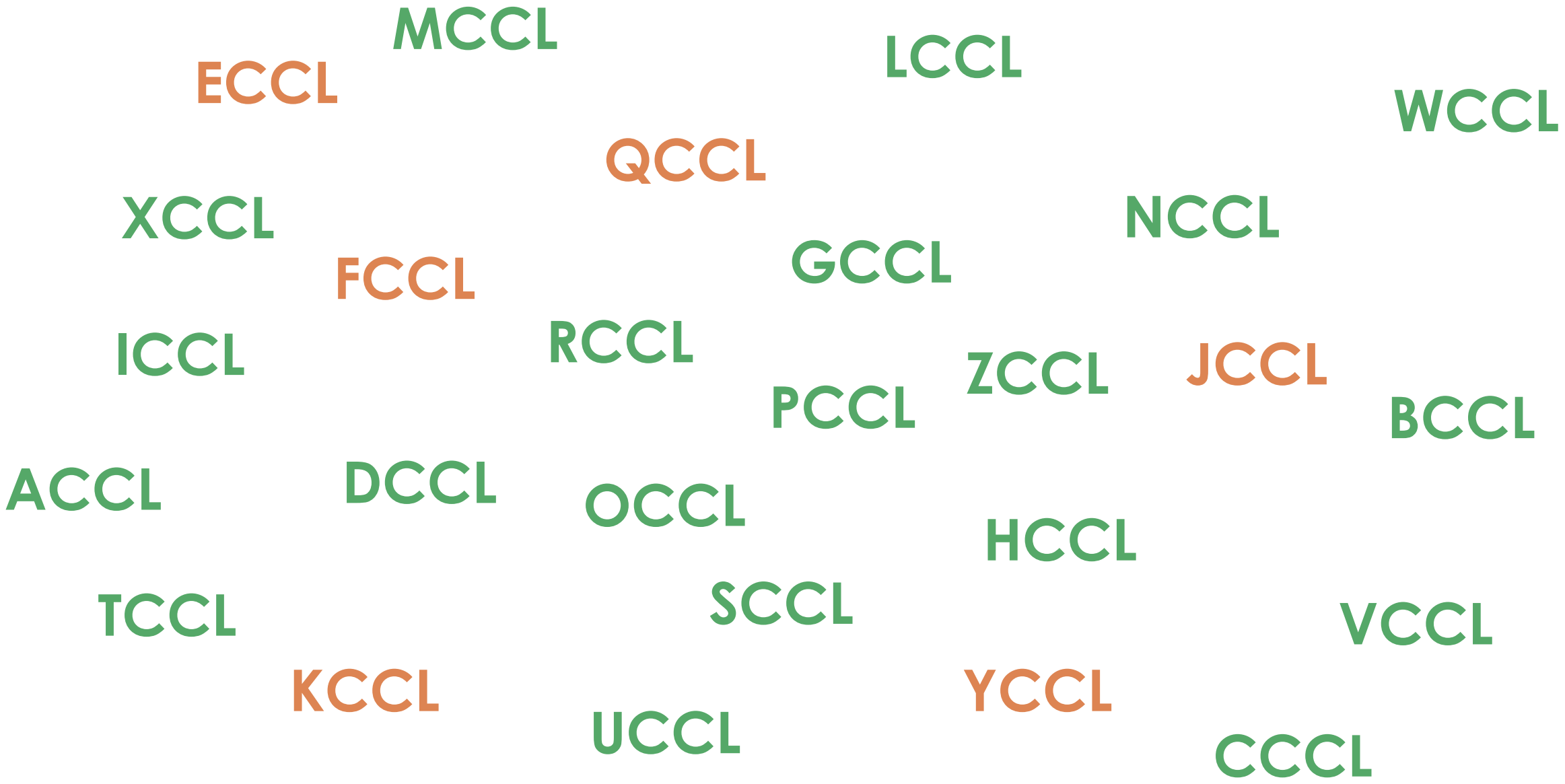


Software Stack



Software Stack




MCCL
ECCL
LCCL
WCCL
QCCL
XCCL
NCCL
FCCL
GCCL
ICCL
RCCL
ZCCL
JCCL
PCCL
BCCL
ACCL
DCCL
OCCL
HCCL
TCCL
SCCL
VCCL
KCCL
YCCL
UCCL
CCCL

MPI vs NCCL

The most obvious difference is stream support, but there are more differences:

- MPI allows underflow: send_count < recv_count.
- MPI datatypes allow arbitrary, nested, heterogeneous, noncontiguous data.
- MPI supports tags, which distinguish messages within a communicator.
- MPI supports “wildcards” (e.g. ANY_SOURCE), which make message matching hard.
- MPI supports multiple ranks per GPU.

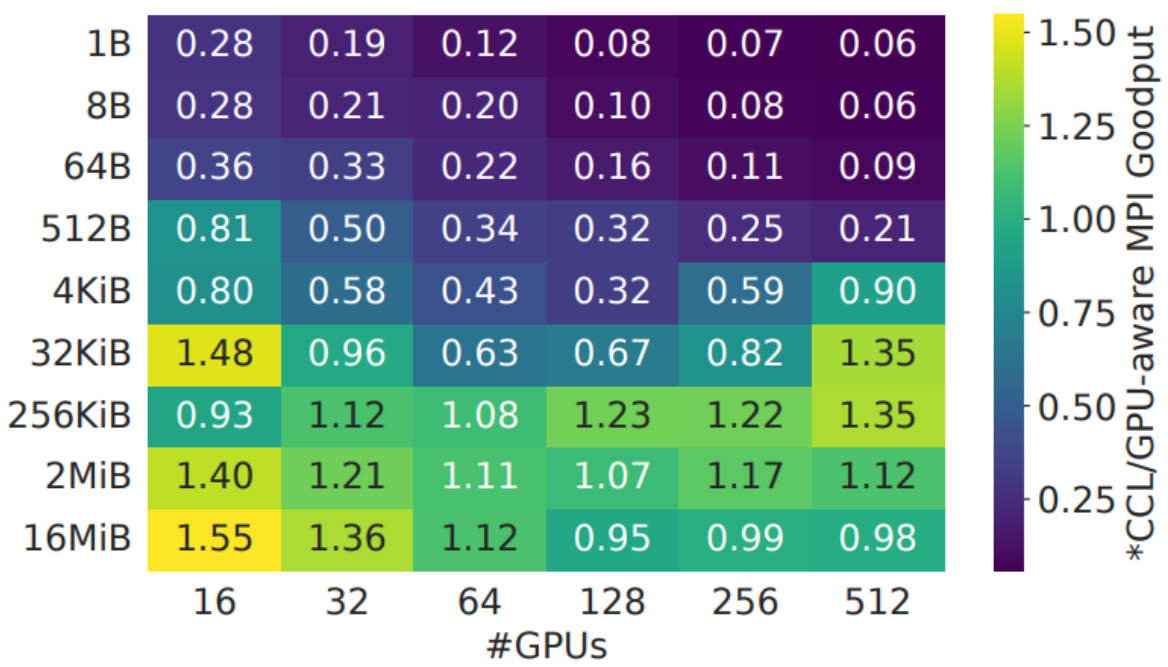
NCCL supports none of these features. MPI 4.0 allows some of them to be disabled, but as they are on by default, little to no effort has gone into optimizing for these scenarios.

NCCL also supports multiple ranks (i.e. GPUs) per process, which some AI workloads require.

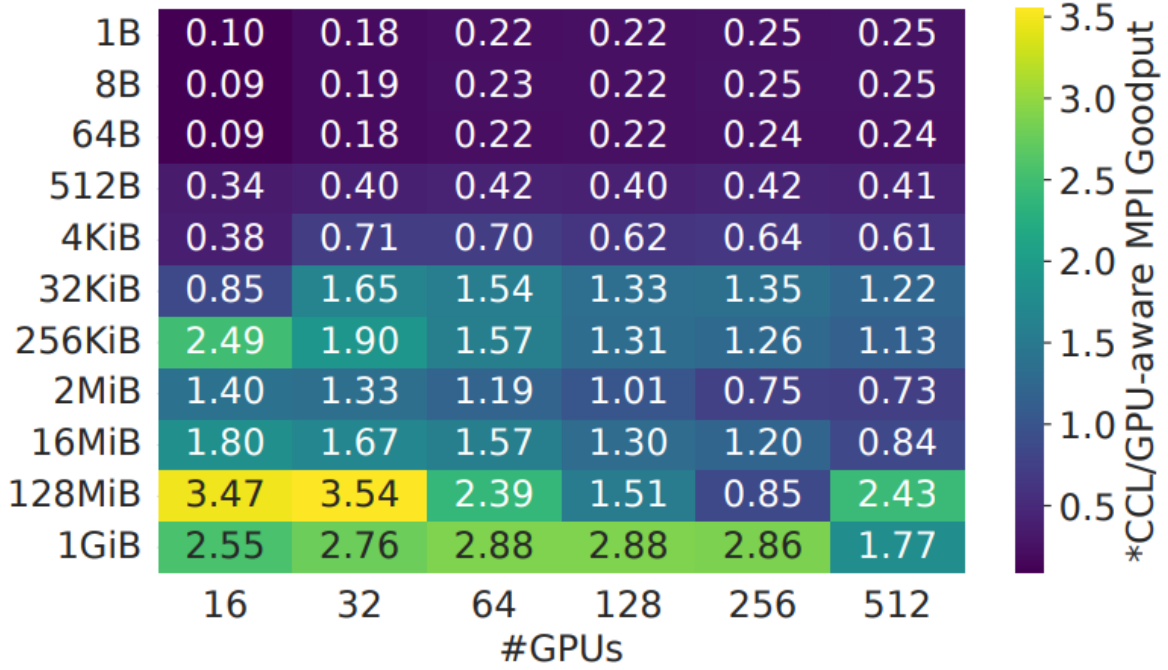
Mitigating MPI Message Matching Misery. ISC 2016. https://doi.org/10.1007/978-3-319-41321-1_15

Slide courtesy: Jeff Hammond (NVIDIA)

MPI vs. GPUCCL



(a) Alltoall



(b) Allreduce

Fig. 11: Ratio between RCCL and GPU-Aware MPI goodput for different collective sizes and nodes count on LUMI.

https://danieledesensi.github.io/assets/pdf/2024_GPUGPU.pdf

MPI vs. GPUCCL

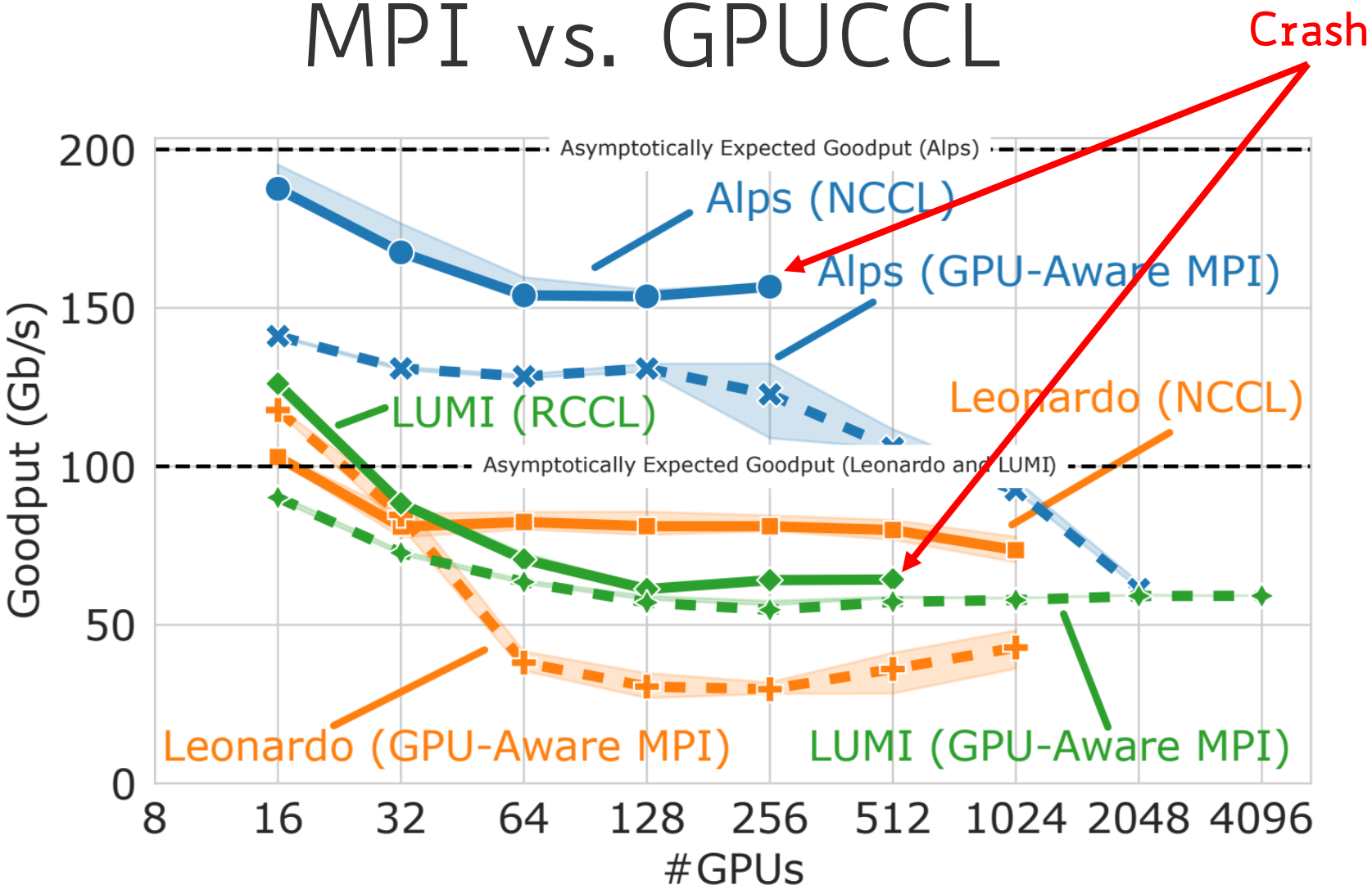


Fig. 9: 2 MiB alltoall scalability.

https://danieledesensi.github.io/assets/pdf/2024_GPUGPU.pdf

MPI vs. GPUCCL

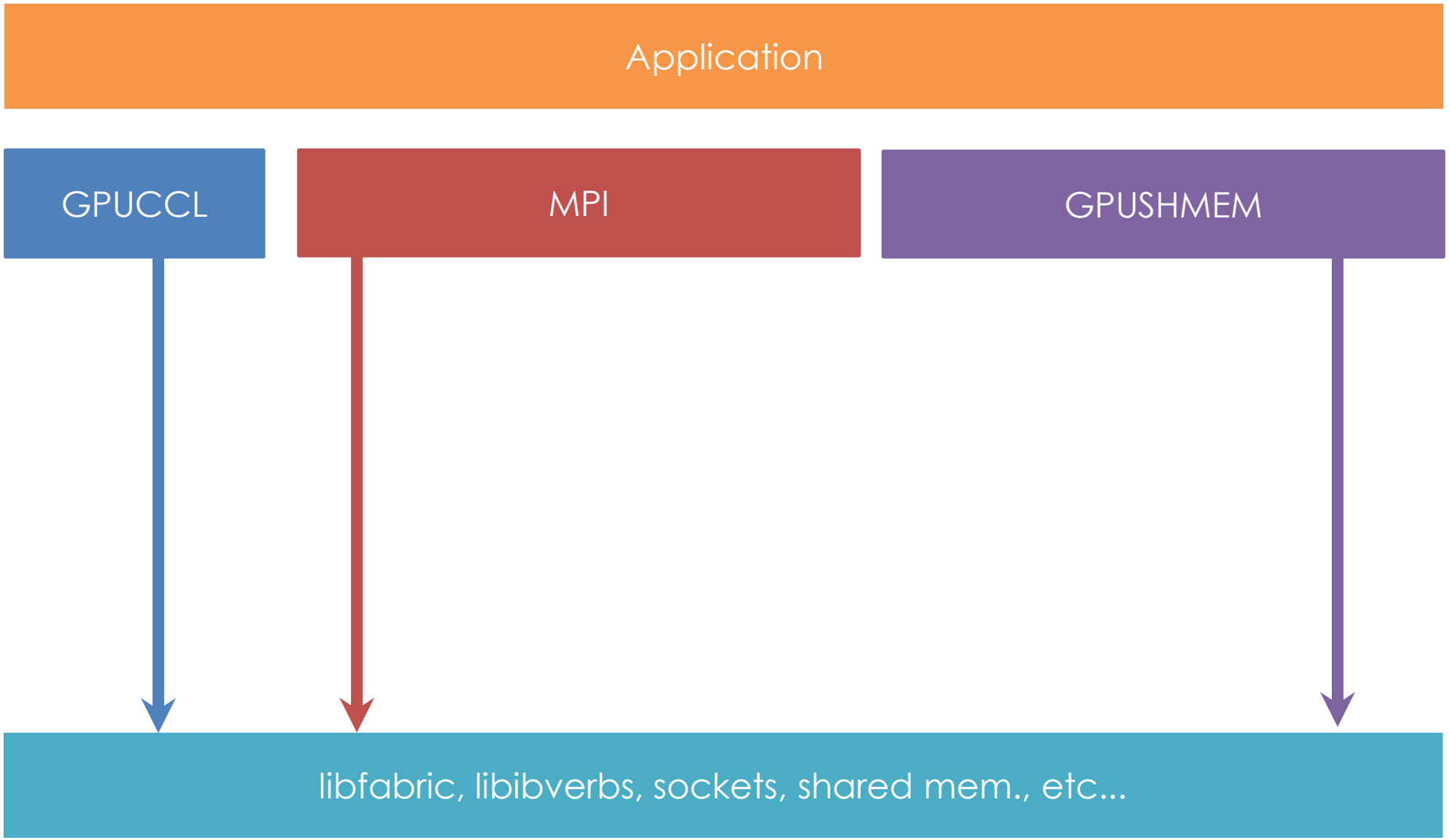
MPI Pros:

- MPI is a standard, NCCL/RCCL/etc.. are not.
- I.e., you can't run NCCL code on AMD GPUs, and vice-versa
- More stable/reliable at large scale

NCCL Pros:

- Simpler/More efficient (no support for tags, custom datatypes, wildcards, etc...)
- Network-aware (scale-up/scale-out) collectives
- Native supports for streams (we will see next week)
- GPU-Initiated communication (just added a few weeks ago)

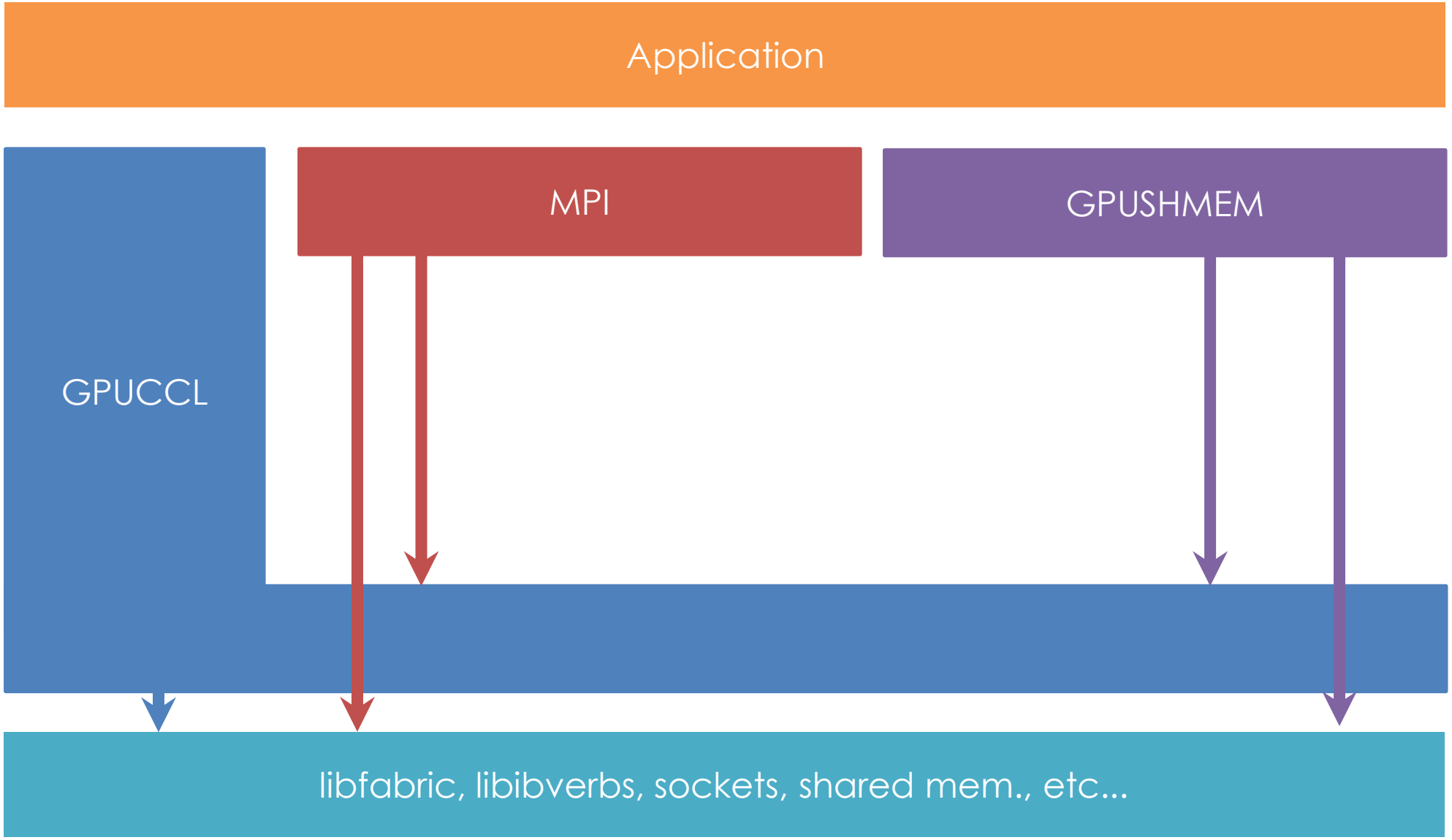
Software Stack



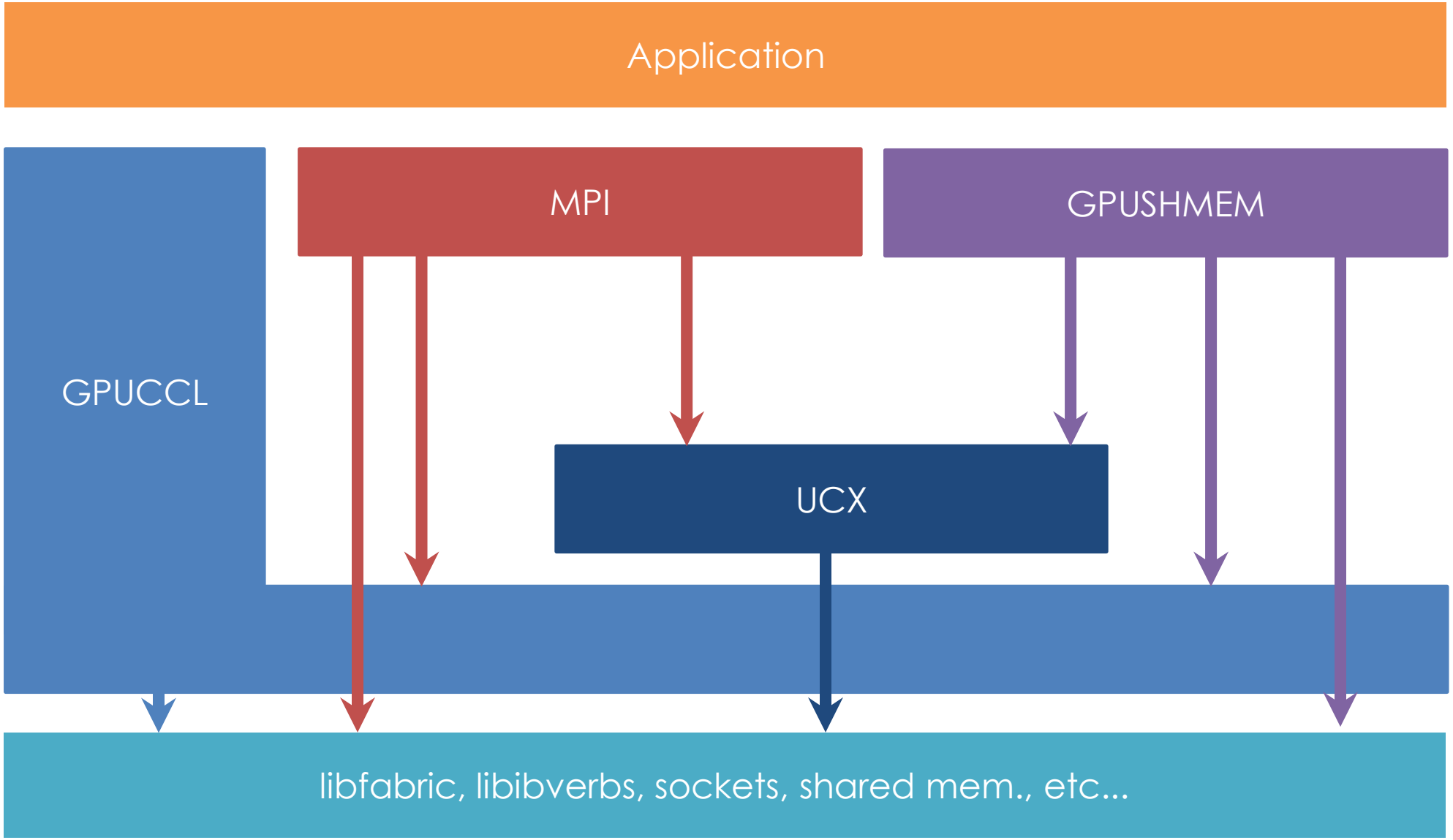
GPUSHMEM

- PGAS (Partitioned Global Address Space) libraries implementing OpenSHMEM for GPUs
- Conceptually GPUs have access to a (partitioned) globally «shared» memory space
- Accesses through one-sided put/get operations
- Support for point-to-point and collective communication across GPUs
- Host-side and device-side API
- Stream-aware
- Gained traction because of MoEs

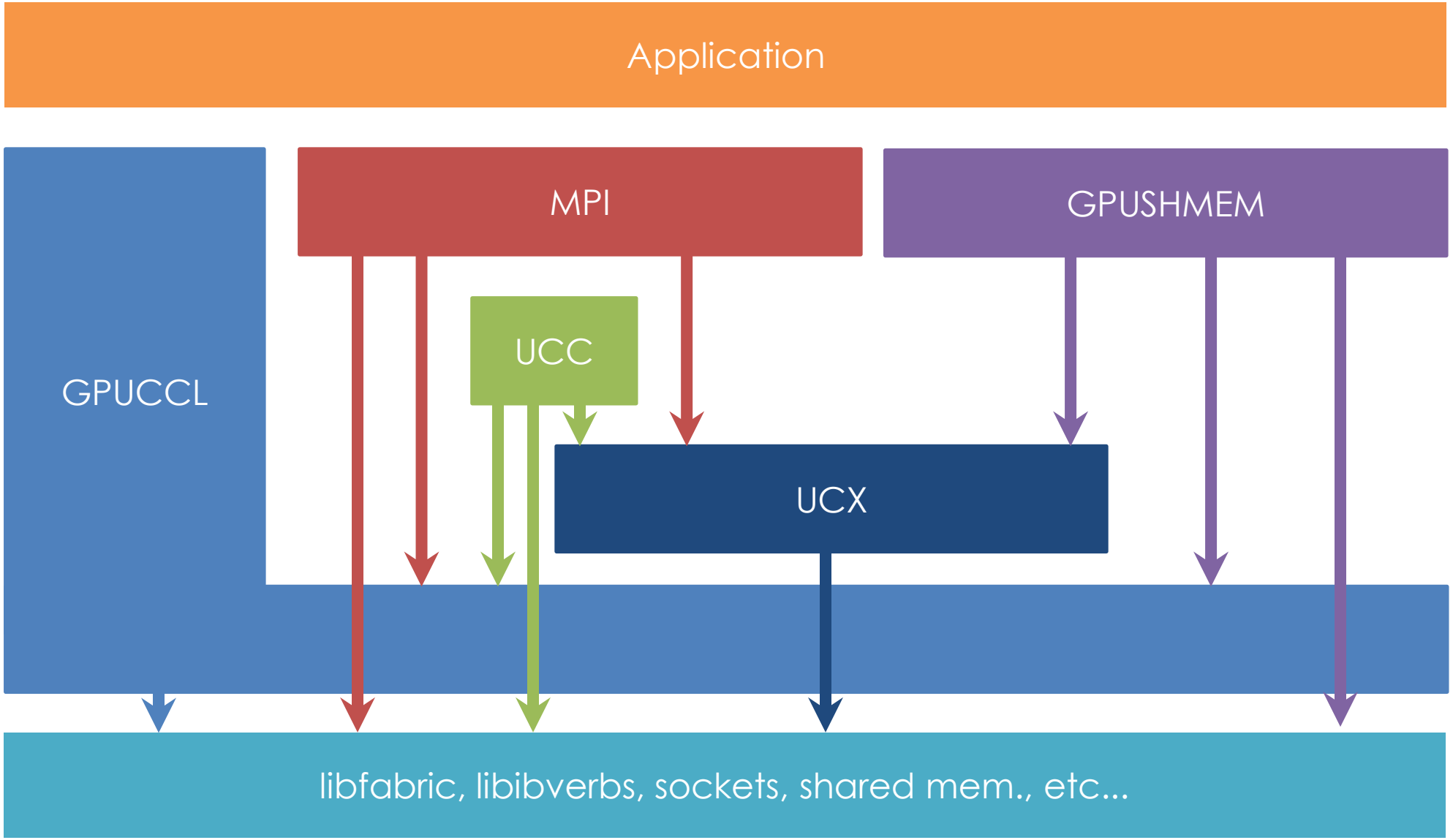
Software Stack



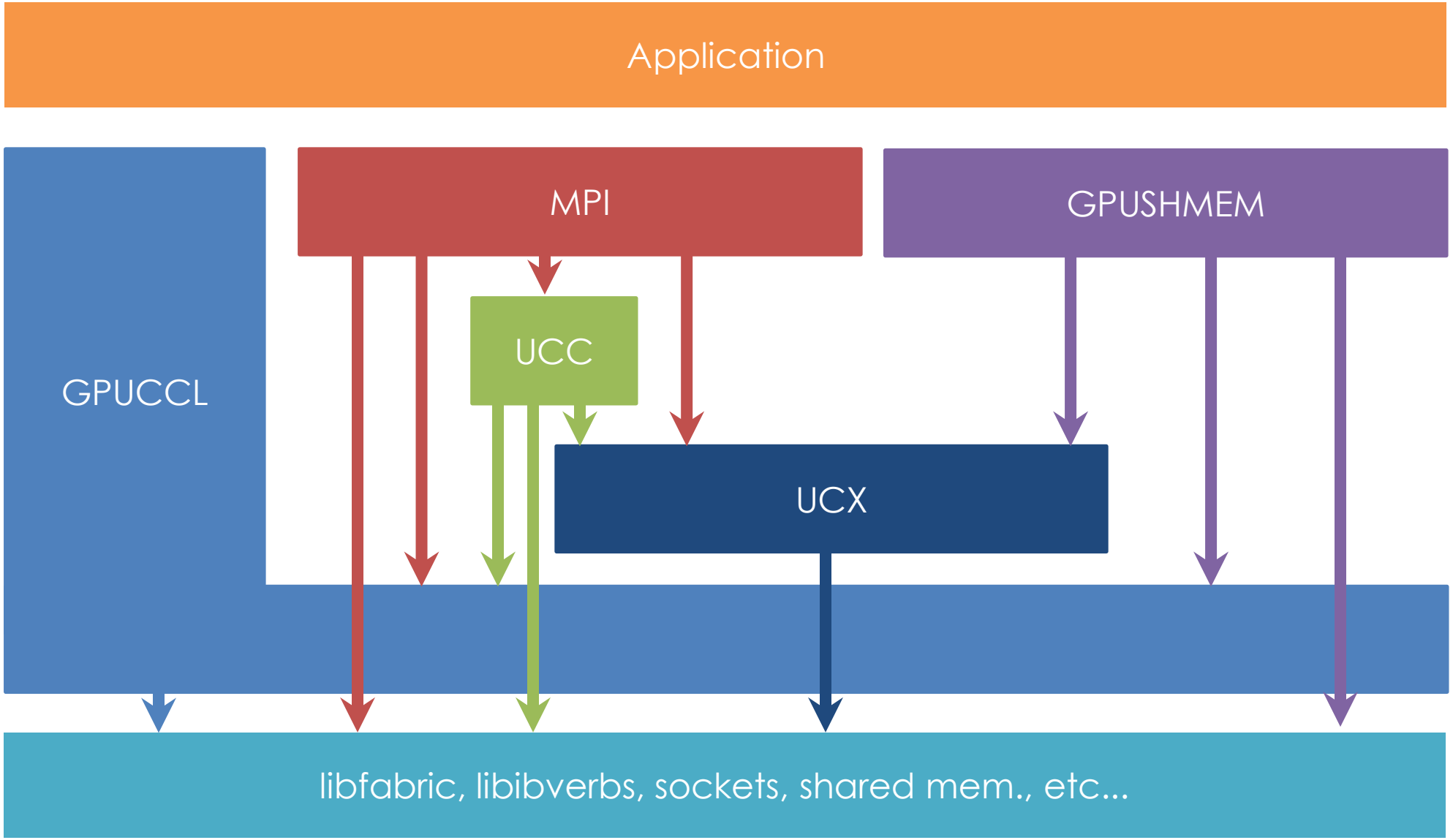
Software Stack



Software Stack



Software Stack



Additional References

Non Von-Neumann Architectures:

- *"A Comprehensive Evaluation of Novel AI Accelerators for Deep Learning Workloads"* – Emani et al.

Scale-out/up Networks:

- *"An In-Depth Analysis of the Slingshot Interconnect"* – De Sensi et al.
- *"Survey of Intra-Node GPU Interconnection in Scale-Up Network: Challenges, Status, Insights, and Future Directions"* – Song et al.

GPU-GPU Communication:

- *"Exploring GPU-to-GPU Communication: Insights into Supercomputer Interconnects"* – De Sensi et al.
- *"The Landscape of GPU-Centric Communication"* – Unat et al.
- *"Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms"* – Hu et al.
- *"GPU-Initiated Networking for NCCL"* – Hamidouche et al.