



DISTRIBUTED GPU PROGRAMMING FOR EXASCALE

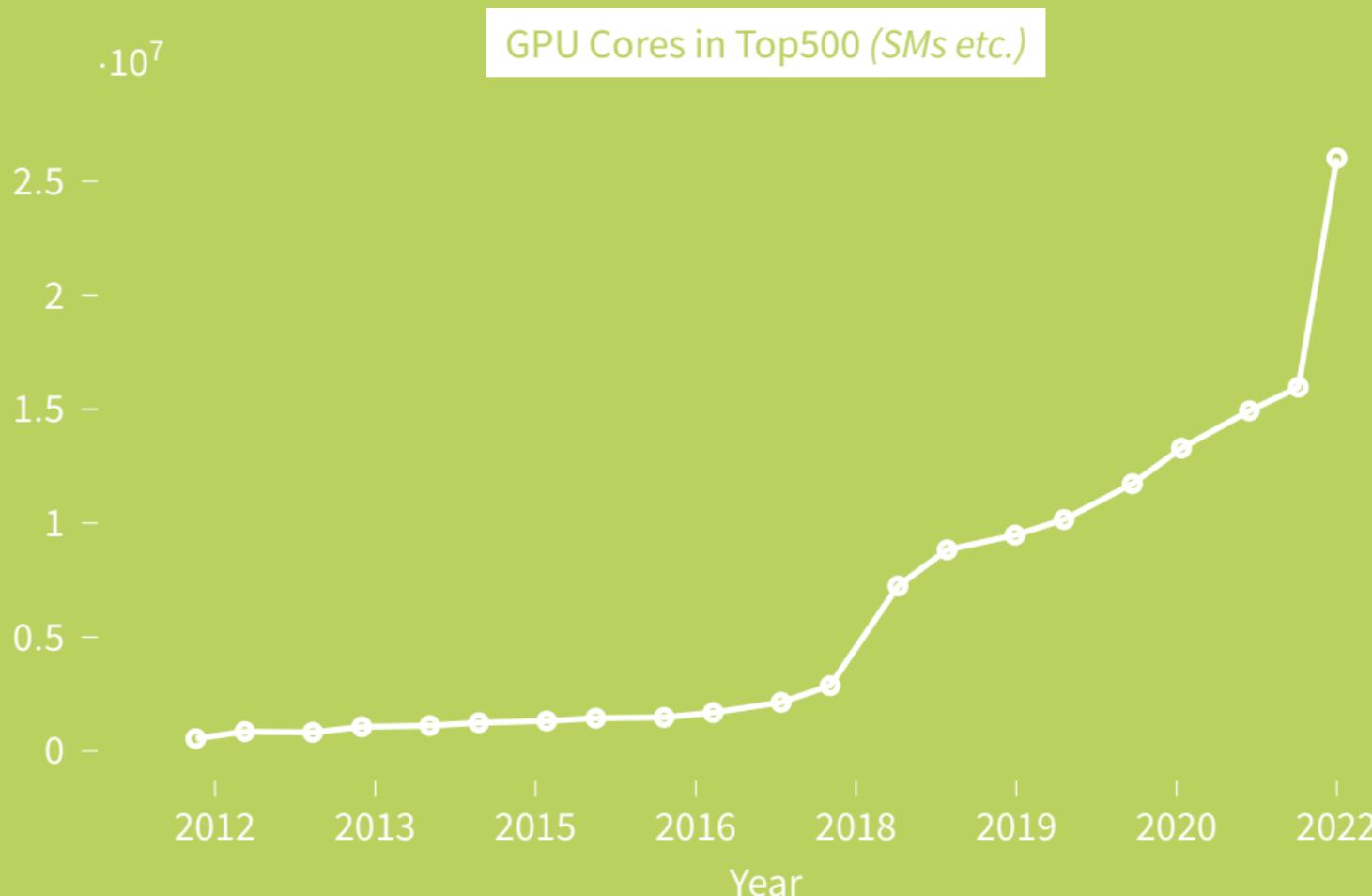
SC22 TUTORIAL *SESSION 1*

14 November 2022 | Andreas Herten | Jülich Supercomputing Centre, Forschungszentrum Jülich

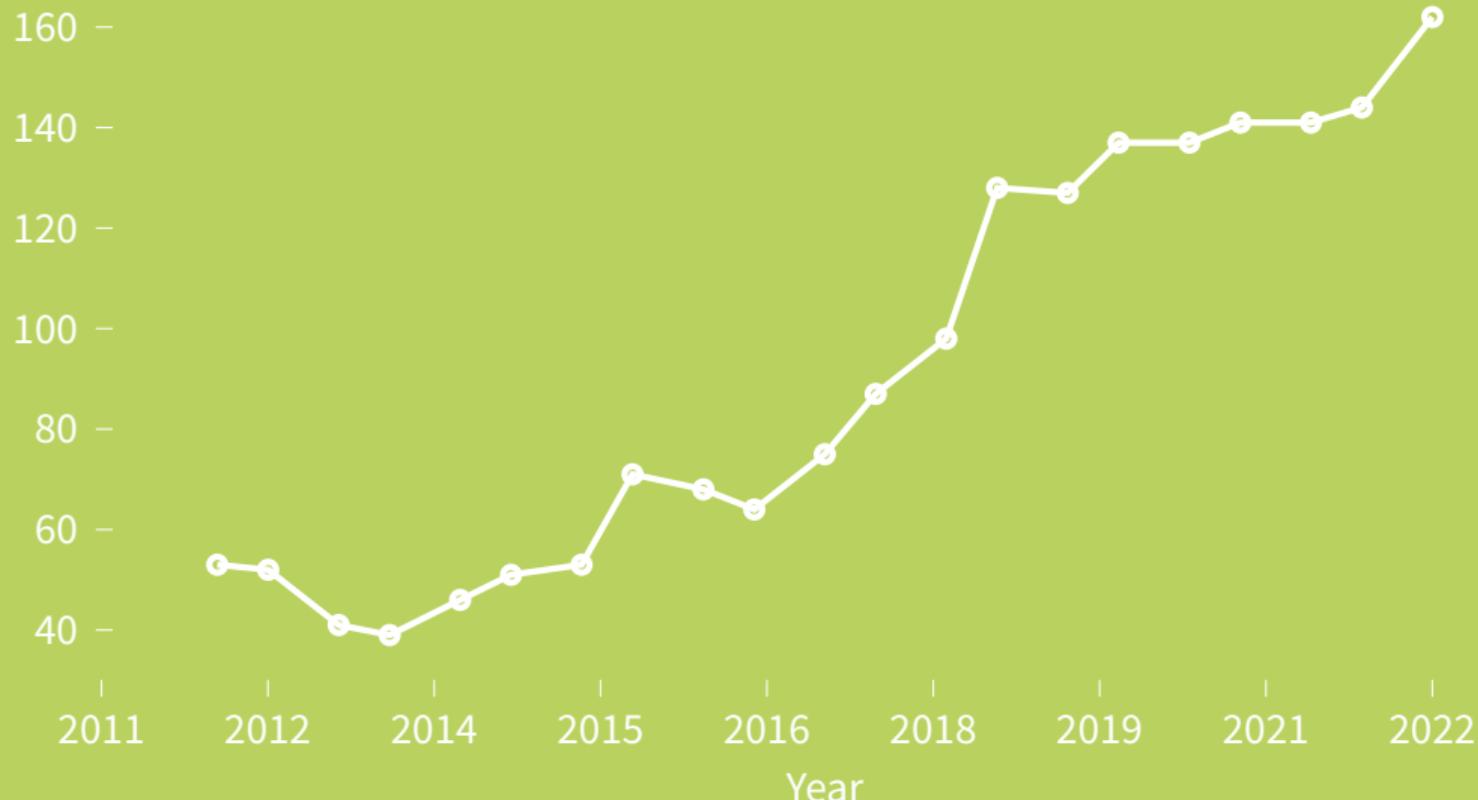
Welcome to

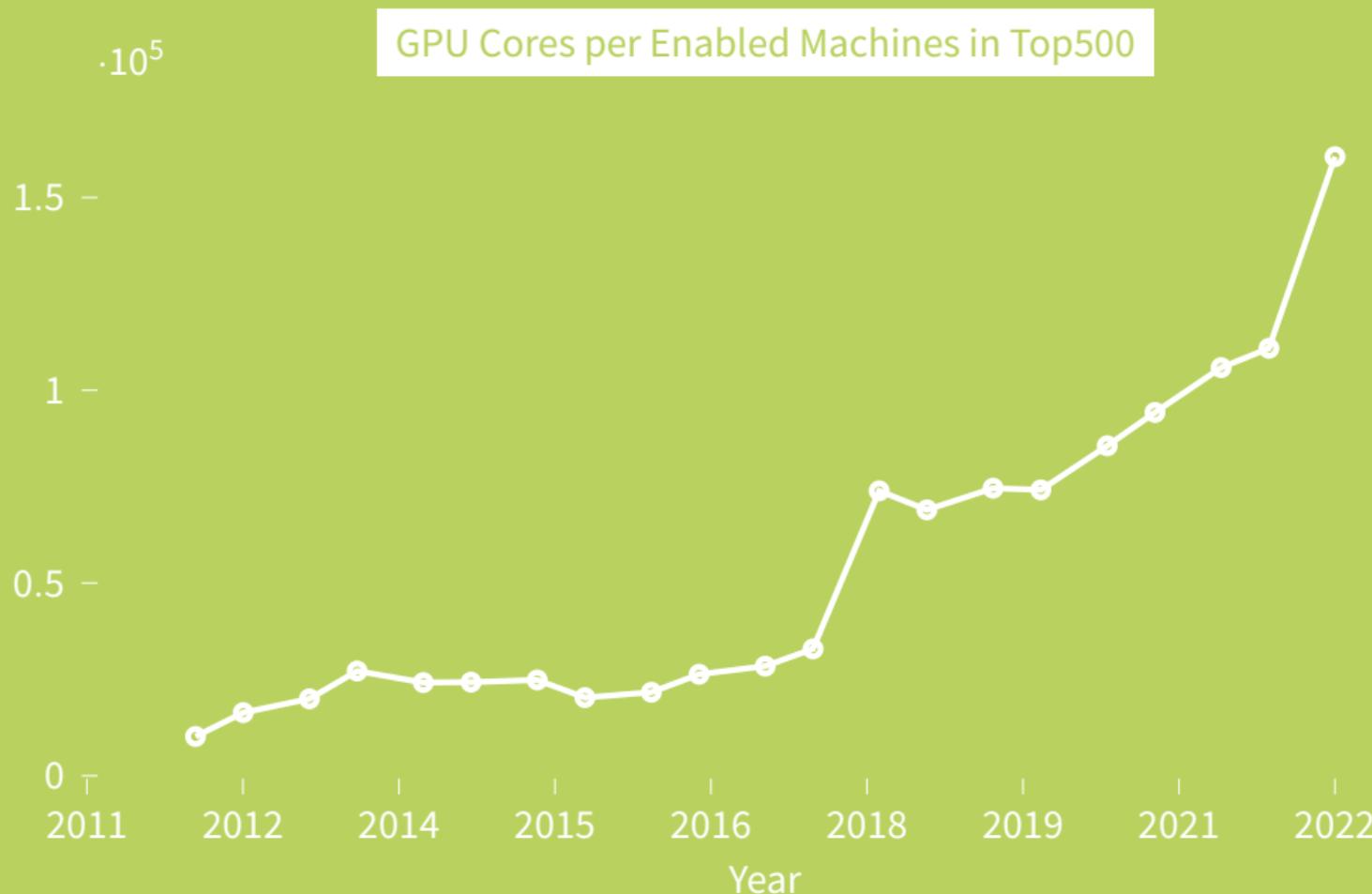
Efficient Distributed GPU Programming for Exascale,

an SC22 Tutorial



GPU-enabled Machines in Top500





State of the GPUUnion

Landscape Overview

- Last 10 years
 - More and more GPUs installed in HPC machines
 - More and more HPC machines with GPUs
 - More and more GPUs in each machine

State of the GPUUnion

Landscape Overview

- Last 10 years
 - More and more GPUs installed in HPC machines
 - More and more HPC machines with GPUs
 - More and more GPUs in each machine
- Future
 - GPUs selected as technology for enabling Exascale
→ Even larger GPU machines with larger GPUs
 - Pre-Exascale systems: LUMI^A, Leonardo^N; JUWELS Booster^N, Perlmutter^N
 - Exascale systems: Frontier^A, El Capitan^A, Aurora^I; JUPITER[?]



ORNL's Frontier: > 35 000 GPUs, 1.5 EFLOP/s, 2022

32 % of all installed GPU cores in Frontier!
Rendering by ORNL

State of the GPUUnion

Landscape Overview

- Last 10 years
 - More and more GPUs installed in HPC machines
 - More and more HPC machines with GPUs
 - More and more GPUs in each machine
 - Future
 - GPUs
 - Even longer list of machines in longer list
- We help to Tame the Beasts!*
- Pre-Exascale systems: LUMI^A, Leonardo^N;
 - JUWELS Booster^N, Perlmutter^N
 - Exascale systems: Frontier^A, El Capitan^A,
 - Aurora^I; JUPITER[?]



ORNL's Frontier: > 35 000 GPUs, 1.5 EFLOP/s,
2022

32 % of all installed GPU cores in Frontier!
Rendering by ORNL

About Tutorial

Goals

- Prepare for large-scale GPU systems
- Learn GPU+MPI basics
- Apply optimization, analysis techniques
- Study CPU-less GPU+MPI
- Use advanced libraries to improve scaling efficiency

About Tutorial

Goals

- Prepare for large-scale GPU systems
- Learn GPU+MPI basics
- Apply optimization, analysis techniques
- Study CPU-less GPU+MPI
- Use advanced libraries to improve scaling efficiency
- Learn interactively!

About Tutorial

Goals

- Prepare for large-scale GPU systems
- Learn GPU+MPI basics
- Apply optimization, analysis techniques
- Study CPU-less GPU+MPI
- Use advanced libraries to improve scaling efficiency
- Learn interactively!

Non-Goals

- Optimize your GPU application; we teach tools and techniques, you need to apply!
- Learn MPI; we expect base-level knowledge of MPI, you don't need more.
- Discuss general scalability; GPU-independent features (like load balancing) are too broad a topic
- Learn CUDA; we expect principle knowledge of GPU programming
- Showcase all GPU platforms; we use an NVIDIA system and teach NVIDIA libraries and tools, other platforms (AMD) follow along (see last lecture)

About Tutorial

Goals

- Prepare for large-scale GPU systems
- Learn GPU+MPI basics
- Apply optimization, analysis techniques
- Study CPU-less GPU+MPI
- Use advanced libraries to improve scaling efficiency
- Learn interactively!

Non-Goals

- Optimize your GPU application; we teach tools and techniques, you need to apply!
- Learn MPI; we expect base-level knowledge of MPI, you don't need more.
- Discuss general scalability; GPU-independent features (like load balancing) are too broad a topic
- Learn CUDA; we expect principle knowledge of GPU programming
- Showcase all GPU platforms; we use an NVIDIA system and teach NVIDIA libraries and tools, other platforms (AMD) follow along (see last lecture)

Curriculum

1L	Lecture <i>Onboarding</i>	Tutorial Overview, Introduction to System <i>Accessing JUWELS Booster</i>
2L	Lecture	Introduction to MPI-Distributed Computing with GPUs
3H	Hands-On	Multi-GPU Parallelization <i>10:30 - 11:00: Coffee Break</i>
4L	Lecture	Performance and Debugging Tools
5L	Lecture	Optimization Techniques for Multi-GPU Applications
6H	Hands-On	Overlap Communication and Computation with MPI <i>12:30 - 13:30: Lunch Break</i>
7L	Lecture	Overview of NCCL and NVSHMEM in MPI Programs
8H	Hands-On	Using NCCL and NVSHMEM <i>15:00 - 15:30: Coffee Break</i>
9L	Lecture	Device-initiated Communication with NVSHMEM
10H	Hands-On	Device-initiated Communication with NVSHMEM
11L	Lecture	Outline of Advanced Topics and Conclusion

Tutorial Team

Hello from Europe



Simon Garcia

Acc. and Comm. for HPC
Barcelona Supercomputing Center



Jiri Kraus

DevTech Compute
NVIDIA



Andreas Herten

Accelerating Devices Lab
Jülich Supercomputing Centre



Lena Oden

Prof. for Computer Engineering
University Hagen



Markus Hrywniak

DevTech Compute
NVIDIA

System: JUWELS Booster

JUWELS Overall Architecture

JUWELS Cluster (2018)

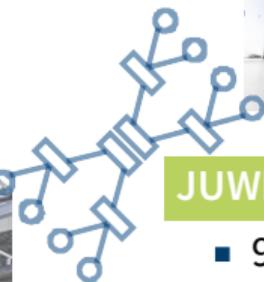
- 2511 compute nodes ($2 \times$ Skylake)
- 48 GPU nodes ($4 \times$ V100 w/ NVLink2)
- Mellanox EDR 100 Gbit/s network,
fat-tree topology (1:2@L1)
- 12 PFLOP/s



JUWELS Overall Architecture

JUWELS Cluster (2018)

- 2511 compute nodes ($2 \times$ Skylake)
- 48 GPU nodes ($4 \times$ V100 w/ NVLink2)
- Mellanox EDR 100 Gbit/s network,
fat-tree topology (1:2@L1)
- 12 PFLOP/s



JUWELS Booster (2020)

- 936 compute nodes ($2 \times$ AMD Rome,
 $4 \times$ A100 w/ NVLink3)
- Mellanox HDR 200 Gbit/s network,
DragonFly+ topology
- 73 PFLOP/s

JUWELS Overall Architecture

JUWELS Cluster (2018)

- 251
- 48
- Mell
- fat-
- 12



- Top500 Nov-2020:**
- #1 Europe
 - #7 World
 - #1* GreenTop500



JUWELS Booster (2020)

- 936 compute nodes (2× AMD Rome, 4× A100 w/ NVLink3)
- Mellanox HDR 200 Gbit/s network, DragonFly+ topology
- 73 PFLOP/s

JUWELS Booster Overview

Node Configuration

Arch Atos Bull Sequana XH2000

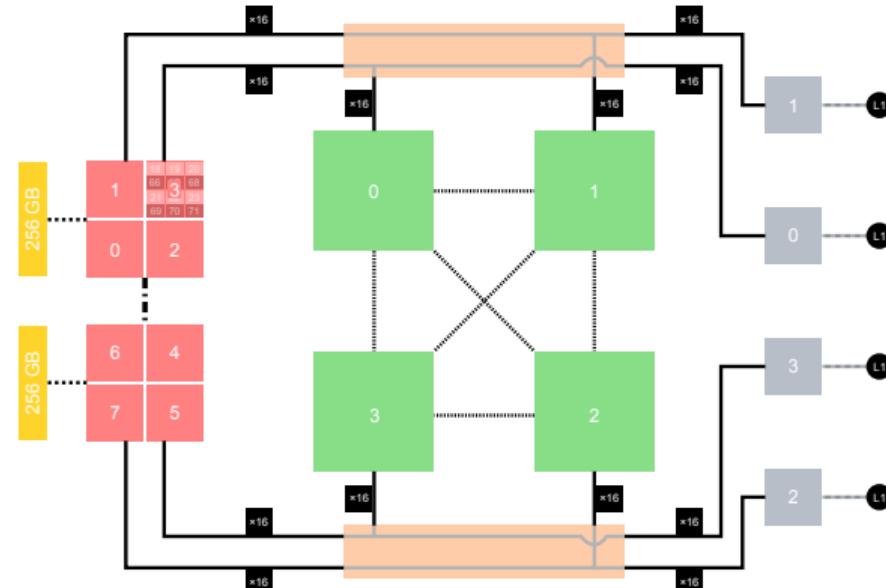
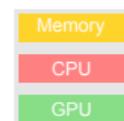
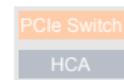
CPU 2 × AMD EPYC 7402:

2_{Socket} × 24_{Core} × 2_{SMT},
2 × 256 GB DDR4-3200 RAM;
NPS-4

GPU 4 × NVIDIA A100 40 GB, NVLink3

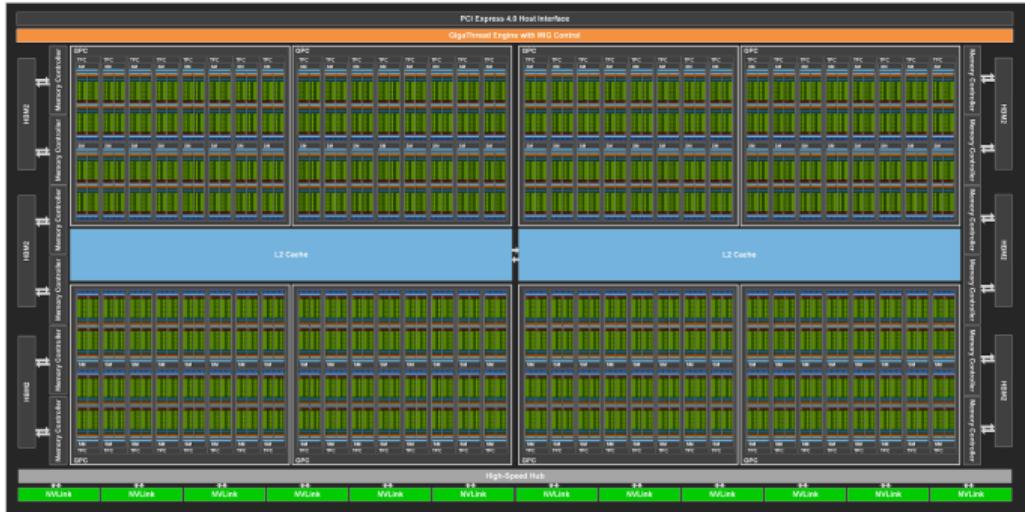
HCA 4 × Mellanox HDR200
(200 Gbit/s) InfiniBand
ConnectX 6

etc 2 × PCIe Gen 4 switch



NVIDIA A100 Tensor Core GPU

- JUWELS Booster:
3744 A100 GPUs
- Vs. V100: More SMs, larger/faster L1, larger/faster L2, faster memory, faster NVLink, faster PCIe, ...
- New features: MIG, async copy to/barrier in shared mem



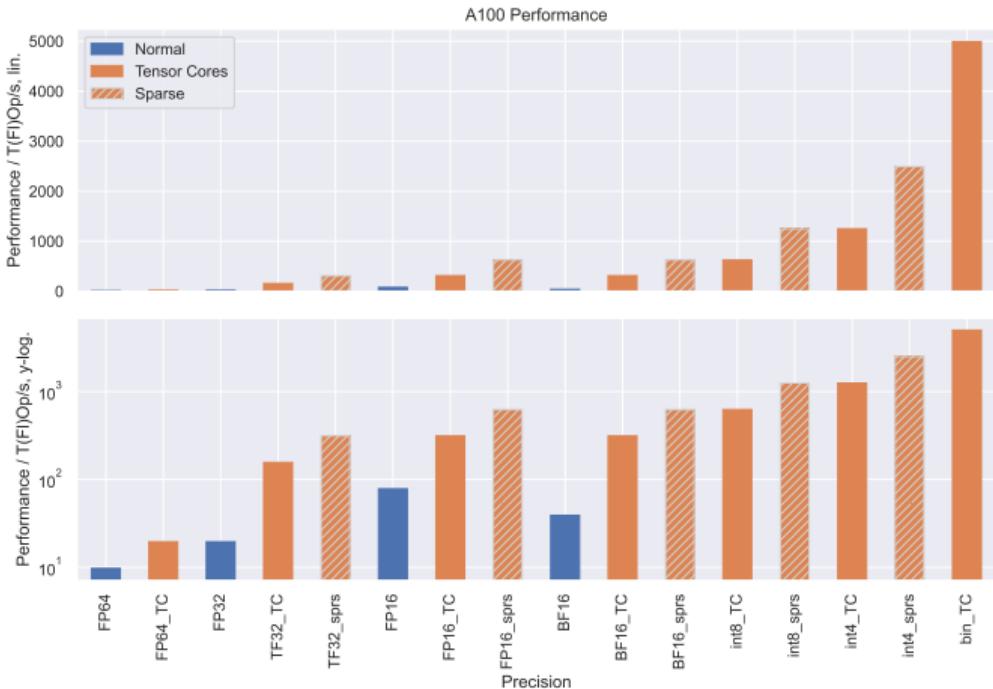
NVIDIA A100 Tensor Core GPU

- JUWELS Booster:
3744 A100 GPUs
- Vs. V100: More SMs, larger/faster L1, larger/faster L2, faster memory, faster NVLink, faster PCIe, ...
- New features: MIG, async copy to/barrier in shared mem
- New Tensor Cores



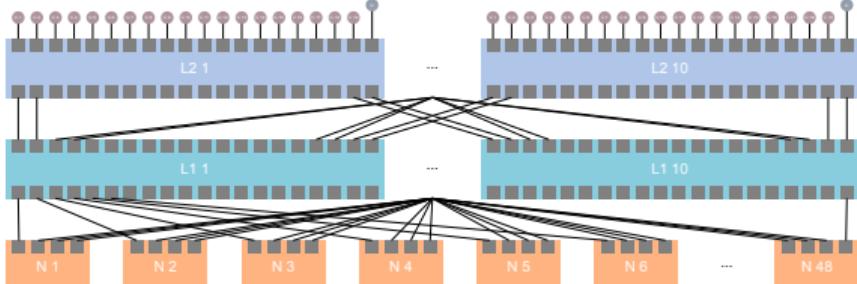
NVIDIA A100 Tensor Core GPU

- JUWELS Booster:
3744 A100 GPUs
- Vs. V100: More SMs, larger/faster L1, larger/faster L2, faster memory, faster NVLink, faster PCIe, ...
- New features: MIG, async copy to/barrier in shared mem
- New Tensor Cores, new performances:
 $73 \text{ PFLOP}/\text{s}_{\text{FP64TC}}$ $584 \text{ PFLOP}/\text{s}_{\text{FP32TC}}$
 $1.16 \text{ EFLOP}/\text{s}_{\text{FP16TC}}$ $18.7 \text{ EOP}/\text{s}_{\text{BinTC}}$

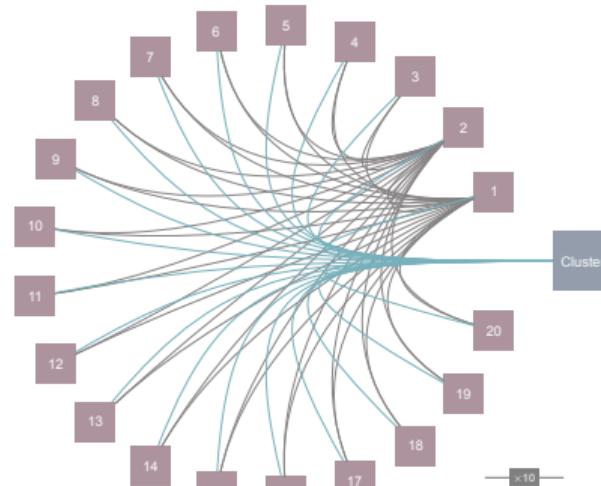


JUWELS Booster Overview

Network Configuration: DragonFly+ Network



In-Cell (48 nodes): Full fat-tree in 2 levels



Inter-Cell (20 cells): 10 links between each pair of cells

JUWELS

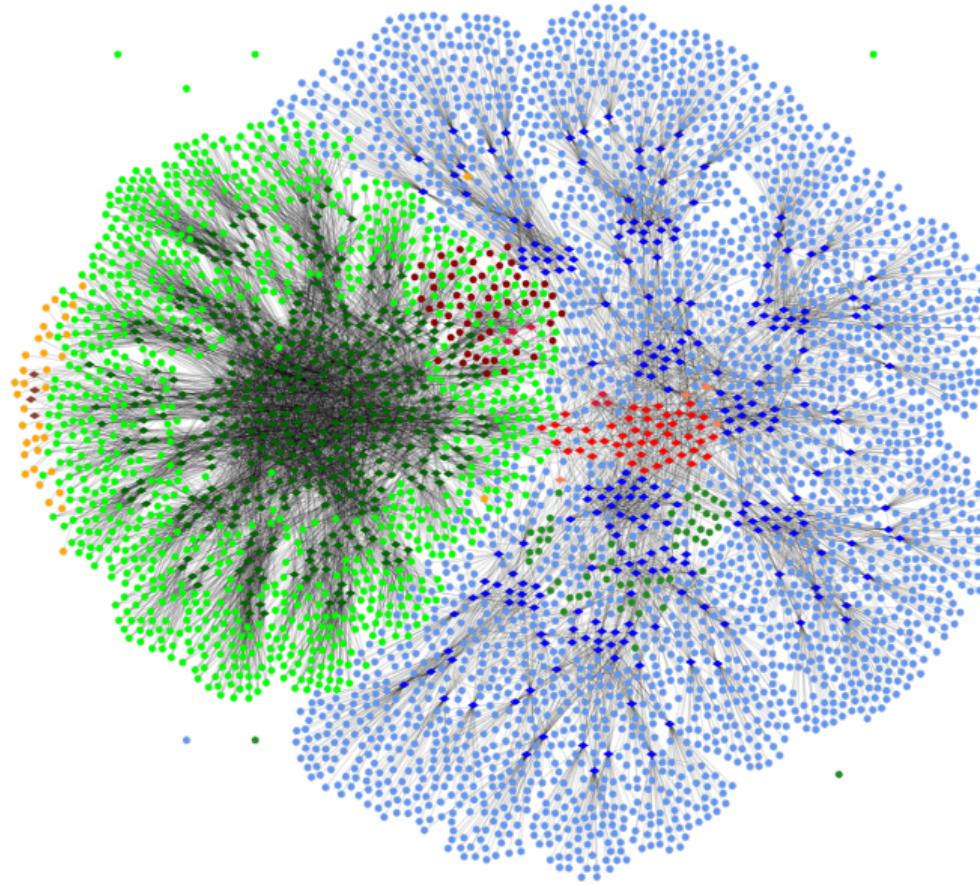
Cluster Booster Integration

Fully integrated system: **JUWELS** with Cluster and Booster modules

- File system: GPFS
- Network: InfiniBand
- Workload management: Slurm
- Resource management: ParaStation / ParaStation Slurm

Picture legend:

- | | |
|---|---|
| ■ Cluster CPU node | ■ Booster node |
| ■ Cluster GPU node | ■ Booster switch |
| ■ Cluster switch | ■ Booster gateway |
| ■ Cluster gateway | ■ JUST |
| ■ Top-level switch | ■ Service node |



AMD EPYC Rome

Some Processor Basics

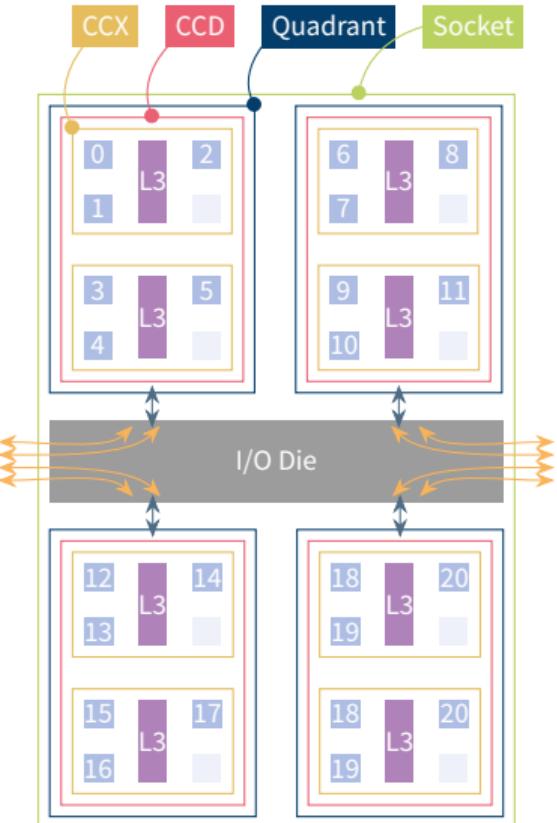
- JUWELS Booster: AMD EPYC Rome 7402 (24 core (SMT-2) × 2 sockets)
- EPYC processor: Built as Multi-Chip Module
→ Many building blocks, hierarchies

AMD EPYC Rome

Some Processor Basics

- JUWELS Booster: AMD EPYC Rome 7402 (24 core (SMT-2) \times 2 sockets)
- EPYC processor: Built as Multi-Chip Module
→ Many building blocks, hierarchies

3 cores	Core-Complex (CCX), shared L3 (<i>max 4 cores</i>)
2 CCXs	Core Complex Die (CCD)
1 CCD	1 quadrant (<i>max 2 CCDs per quadrant</i>)
4 quadrants	1 socket; NPS-4: 4 NUMA domains per socket
2 sockets	1 node (<i>only 1 shown</i>)
I/O Die	Connections between quadrants and outside
RAM	8 memory channels, 2 per quadrant



AMD EPYC Rome

Some Processor Basics

- JUWELS Booster: AMD EPYC Rome 7402 (24 core (SMT-2) \times 2 sockets)
- EPYC processor: Built as Multi-Chip Module
→ Many building blocks, hierarchies

3 cores Core-Complex (CCX), shared L3 (*max 4 cores*)

2 CCXs Core Complex Die (CCD)

1 CCD 1 quadrant (*max 2 CCDs per quadrant*)

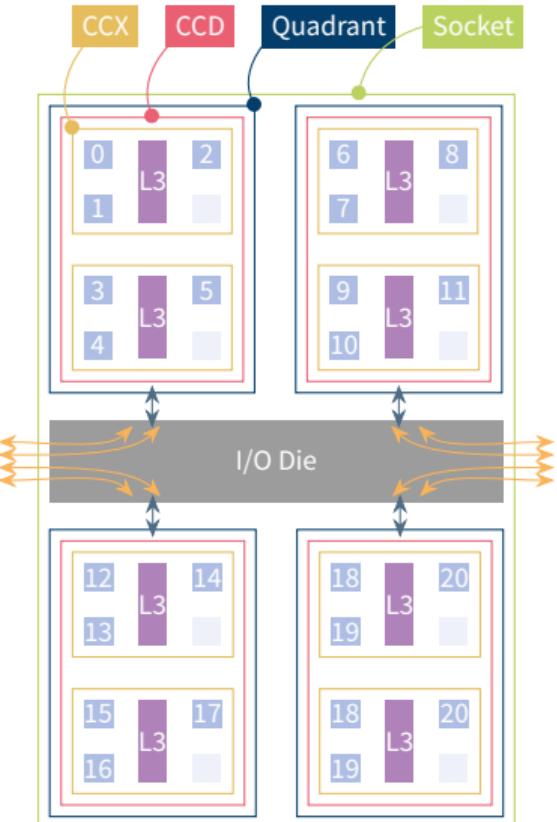
4 quadrants 1 socket; NPS-4: 4 NUMA domains per socket

2 sockets 1 node (*only 1 shown*)

I/O Die Connections between quadrants and outside

RAM 8 memory channels, 2 per quadrant

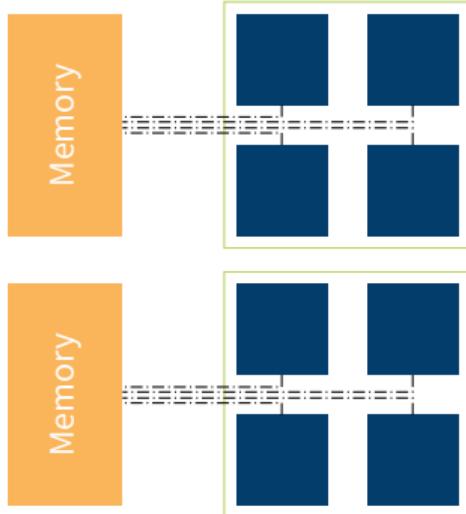
⇒ Complex topology!



PCIe Affinity

3 C 2 CCX 1 CCD 4 Q 2 S

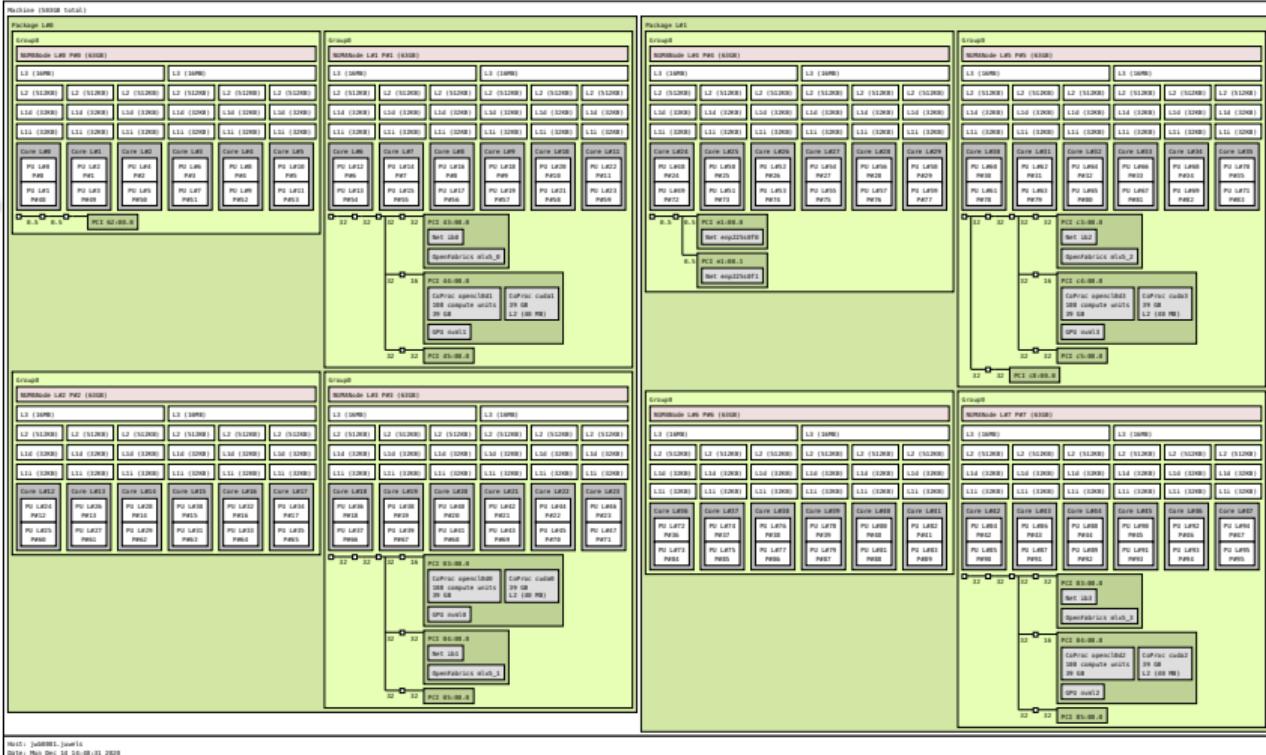
- PCIe via I/O die → also hierarchy
- GPUs connected via PCIe (through PCIe switch)



PCIe Affinity

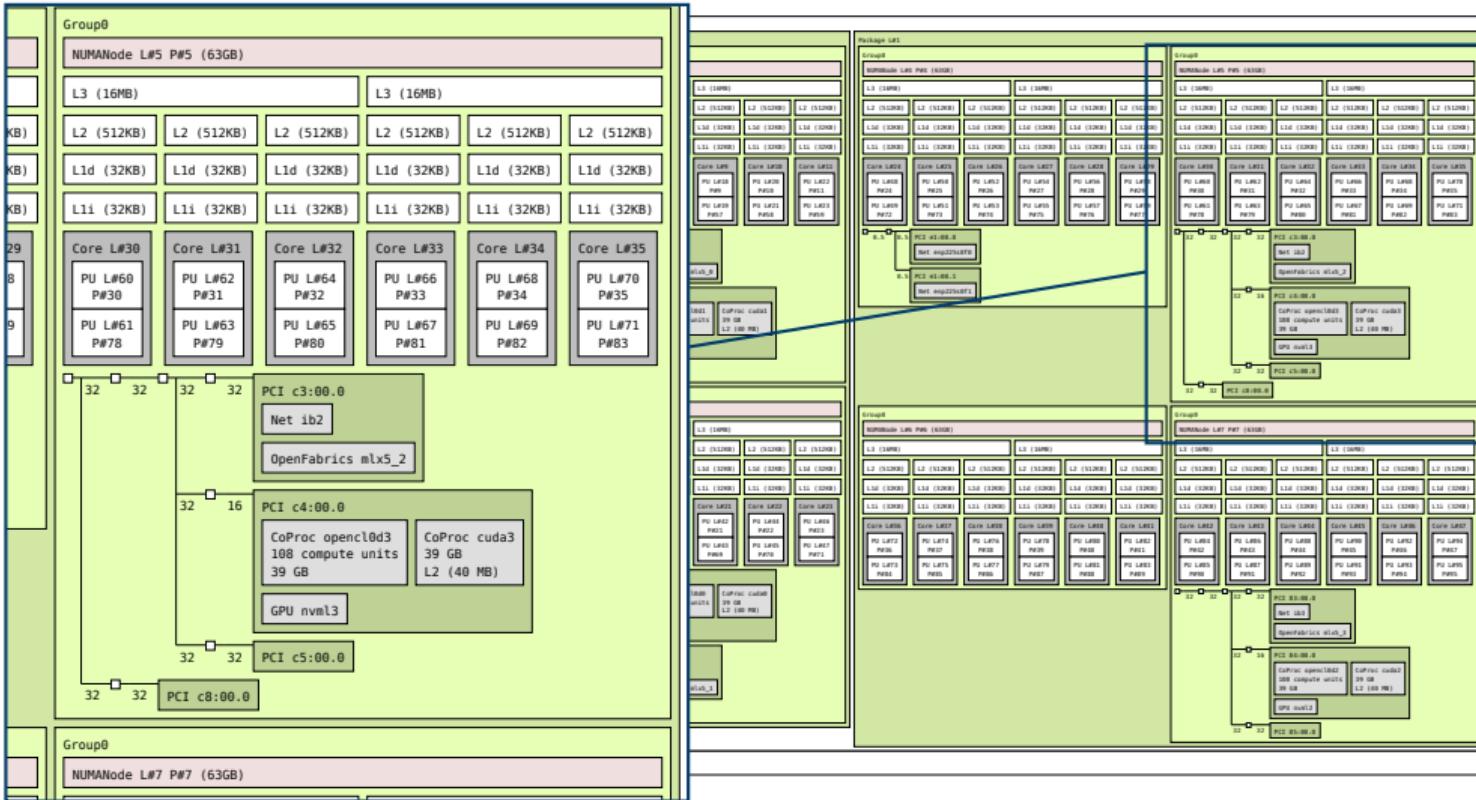
3C 2CCX 1CCD 4Q 25

- PCIe via GPUs
- GPUs config



PCIe Affinity

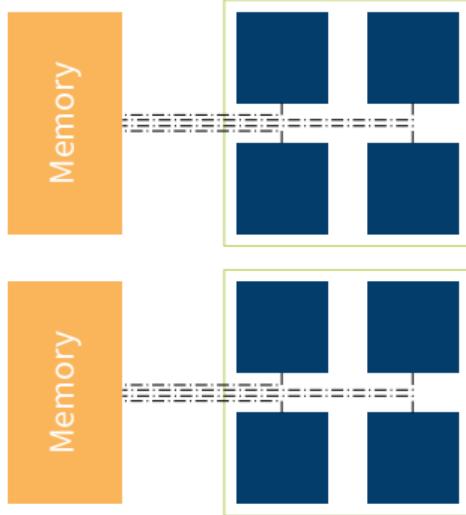
3C 2CCX 1CCD 4Q 25



PCIe Affinity

3 C 2 CCX 1 CCD 4 Q 2 S

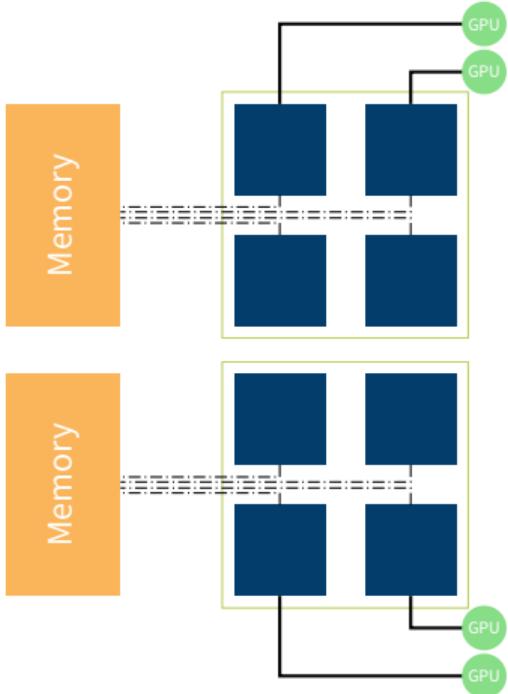
- PCIe via I/O die → also hierarchy
- GPUs connected via PCIe (through PCIe switch)
- PCIe 4.0 lanes: 2×16 , each 16 connected to 1 quadrant



PCIe Affinity



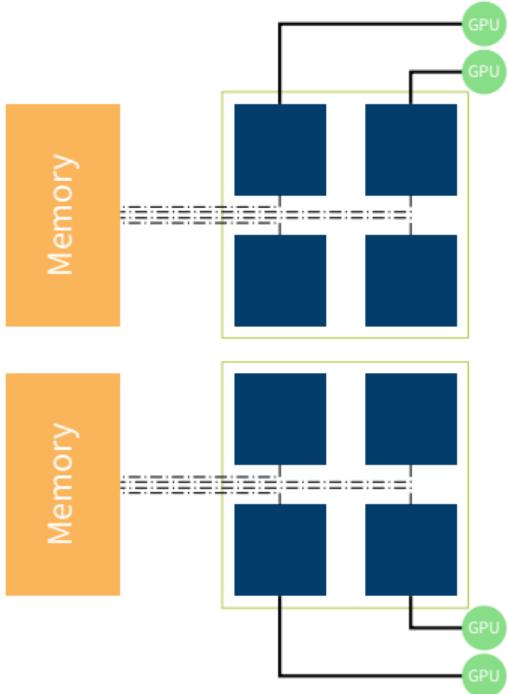
- PCIe via I/O die → also hierarchy
 - GPUs connected via PCIe (through PCIe switch)
 - PCIe 4.0 lanes: 2×16 , each 16 connected to 1 quadrant
- *True GPU affinity only by half of chiplets*
But impact application-dependent and possibly quite low



PCIe Affinity

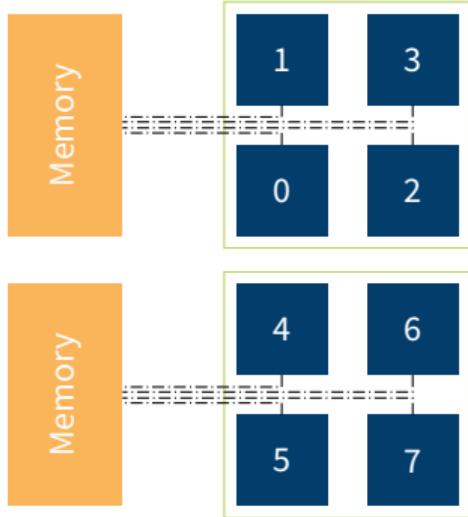


- PCIe via I/O die → also hierarchy
 - GPUs connected via PCIe (through PCIe switch)
 - PCIe 4.0 lanes: 2×16 , each 16 connected to 1 quadrant
- *True GPU affinity only by half of chiplets*
- But impact application-dependent and possibly quite low*
- InfiniBand HCA adapters also via PCIe switch
 - Same affinity considerations
 - 1:1 relation between HCA and GPU



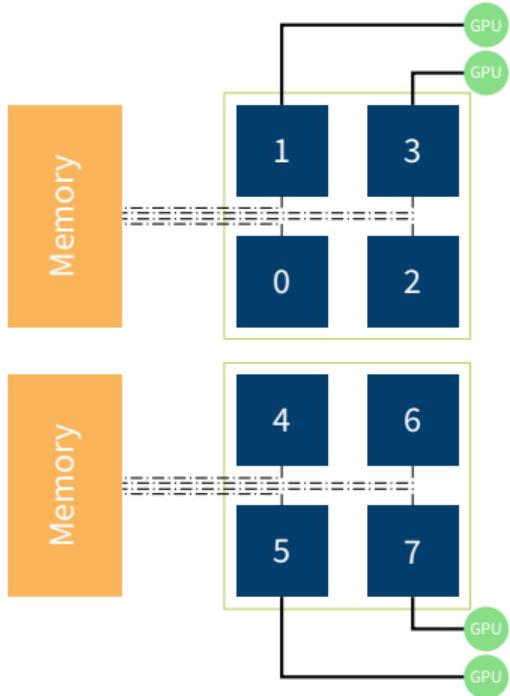
NUMA Numbering

- NUMA domains (= quadrants) are numbered



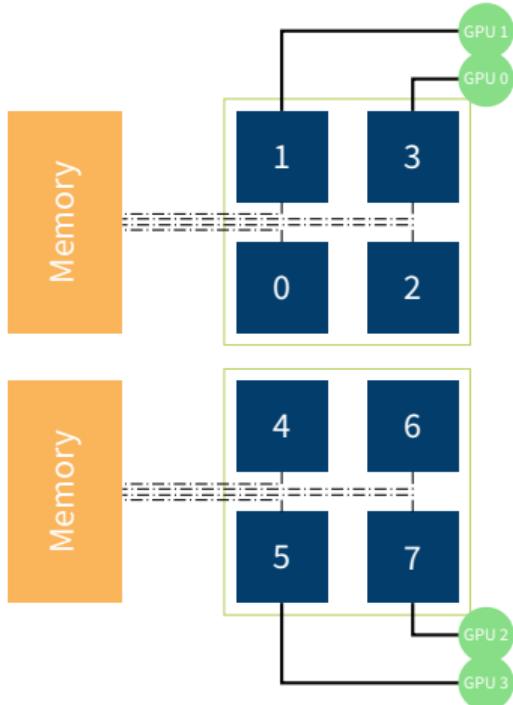
NUMA Numbering

- NUMA domains (= quadrants) are numbered
- Not all domains have GPU/HCA affinity, only 1, 3, 5, 7



NUMA Numbering

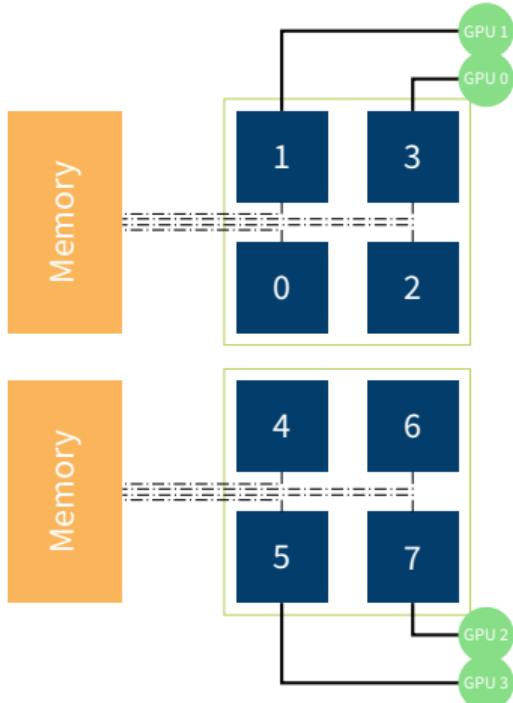
- NUMA domains (= quadrants) are numbered
- Not all domains have GPU/HCA affinity, only 1, 3, 5, 7
- *But GPU 0 is not attached to NUMA domain 0...*



NUMA Numbering

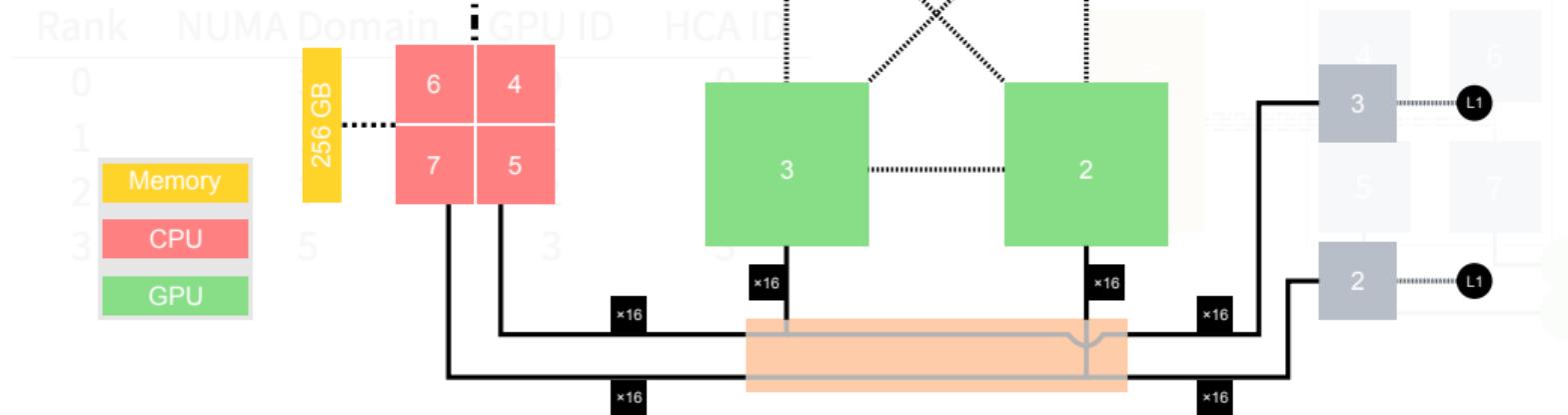
- NUMA domains (= quadrants) are numbered
- Not all domains have GPU/HCA affinity, only 1, 3, 5, 7
- *But GPU 0 is not attached to NUMA domain 0...*
- Complete affinity table

Rank	NUMA Domain	GPU ID	HCA ID
0	3	0	0
1	1	1	1
2	7	2	2
3	5	3	3



NUMA Numbering

- NUMA domains (= quadrants) are numbered 0, 1, 2, 3
- Not all domains have an HCA
- But GPU 0 is not in domain 0
- Complete affinity table



Affinity in Practice

CPU and GPU Affinity

- We configured *PSSlurm* to use best topology **as default**

Affinity in Practice

CPU and GPU Affinity

- We configured *PSSlurm* to use best topology **as default**
- *PSSlurm* sets CPU affinity masks with *GPU-first-approach*: Fill cores with GPU affinity first

Affinity in Practice

CPU and GPU Affinity

- We configured *PSSlurm* to use best topology **as default**
- *PSSlurm* sets CPU affinity masks with *GPU-first-approach*: Fill cores with GPU affinity first
- *PSSlurm* sets `$CUDA_VISIBLE_DEVICES` to associated GPU

Affinity in Practice

CPU and GPU Affinity

- We configured PSSlurm to use best topology **as default**
- PSSlurm sets CPU affinity masks with *GPU-first-approach*: Fill cores with GPU affinity first
- PSSlurm sets `$CUDA_VISIBLE_DEVICES` to associated GPU
- Example GPU affinity for 2 tasks:

```
$ srun -n 2 --cpu-bind=verbose env | grep 'PMI_RANK\|CUDA_VIS'  
cpu_bind=THREADS - jwb0001, task 0 0 [18307]: mask 0x40000 set  
cpu_bind=THREADS - jwb0001, task 1 1 [18309]: mask 0x40 set  
PMI_RANK=0  
CUDA_VISIBLE_DEVICES=0  
PMI_RANK=1  
CUDA_VISIBLE_DEVICES=1
```

DragonFly+ Topology

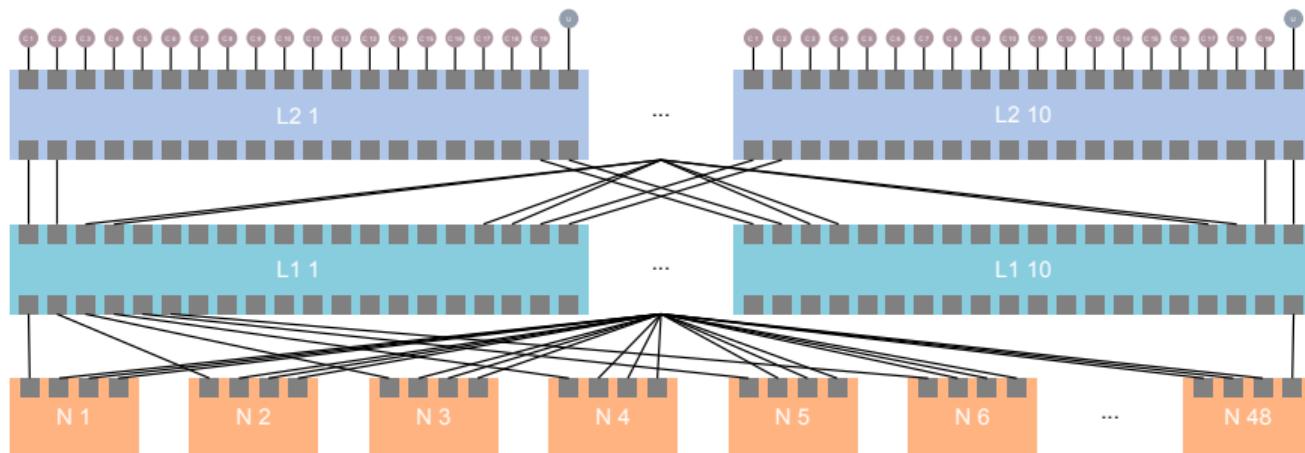
Overview

- 3744 end points; adaptive routing
- HDR200: 200 Gbit/s per link per direction
- Dragonfly+ Topology selected as good way to balance factors (space, investment, performance)
- Two-level topology: local (in-cell), non-local (inter-cell)
- Still learning practical implications of topology

DragonFly+ Topology

In-Cell Topology

- In-cell: full fat-tree in 2 levels
 - 48 nodes, each 4 links (*strided to 4 L1 switches*)
 - 10 L1 switches, 40 port (each L1 switch: 2 links to L2, strided to 10 L2 switches)
 - 10 L2 switches, 40 port (each L2 switch: 1 link to L2 switch of all other cells, 1 link to cluster)



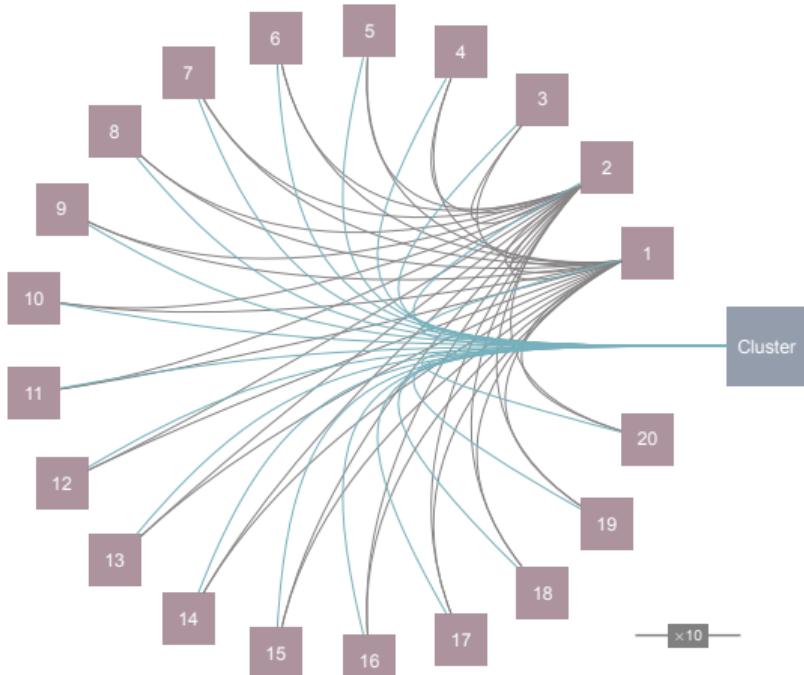
DragonFly+ Topology

Intra-Cell Topology

- Inter-cell: 20 cells

- 10 links between each pair of cells
- 10 links per cell to JUWELS Cluster
- Filesystem access via cell 20
- Bi-section bandwidth between N cells:

$$\mathcal{B}(N) = \lfloor (N/2)^2 \rfloor \times (10 \times bw_1)$$



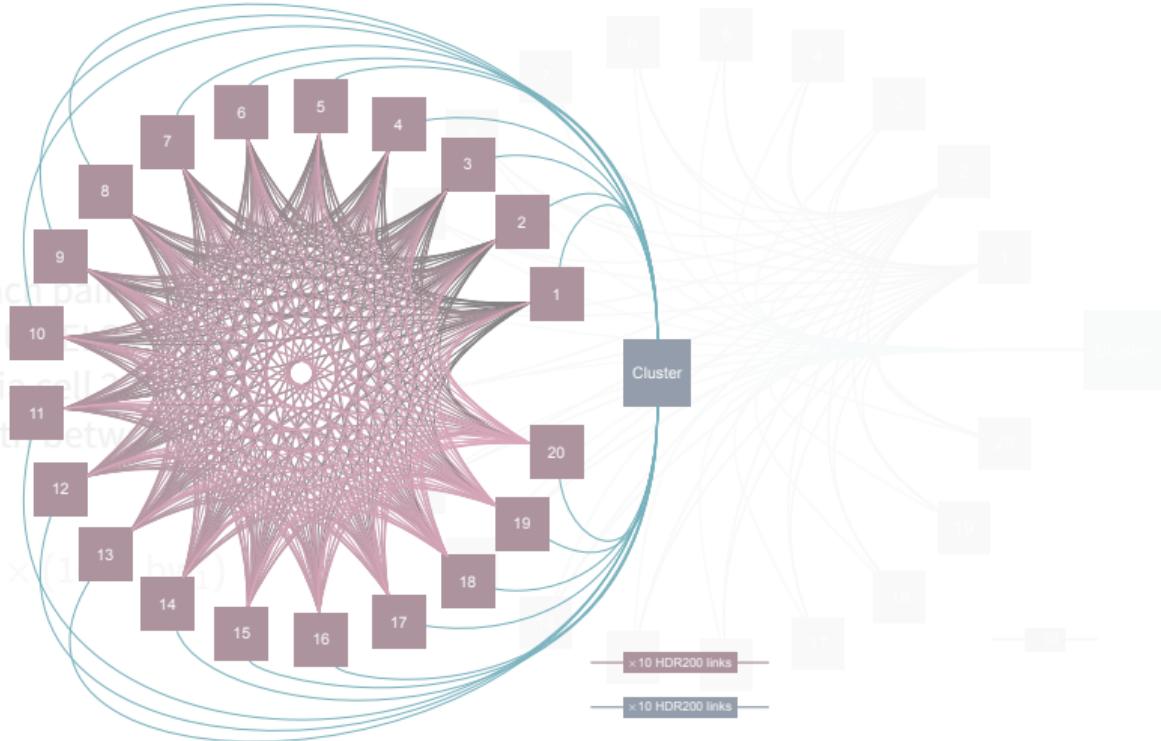
DragonFly+ Topology

Intra-Cell Topology

- Inter-cell: 20 cells

- 10 links between each pair of cells
- 10 links per cell to JSC Cluster
- Filesystem access via cell-to-cell links
- Bi-section bandwidth between two cells:

$$\mathcal{B}(N) = \lfloor (N/2)^2 \rfloor \times 10 \text{ Gbps}$$



Summary and Conclusions

Summary and Conclusions

- Exascale and Pre-Exascale systems mainly based on GPUs, with thousands of devices
- Many advanced technologies in place to enable large-scale GPU applications
- Tutorial with team experienced in distributed GPU workloads
- Supercomputer of tutorial: JUWELS Booster, European flagship system based on A100 GPUs and HDR200 InfiniBand network

Summary and Conclusions

- Exascale and Pre-Exascale systems mainly based on GPUs, with thousands of devices
- Many advanced technologies in place to enable large-scale GPU applications
- Tutorial with team experienced in distributed GPU workloads
- Supercomputer of tutorial: JUWELS Booster, European flagship system based on A100 GPUs and HDR200 InfiniBand network

Thank you
for your attention!
a.herten@fz-juelich.de