DISTRIBUTED GPU PROGRAMMING FOR EXASCALE

ISC22 TUTORIAL *SESSION 1*

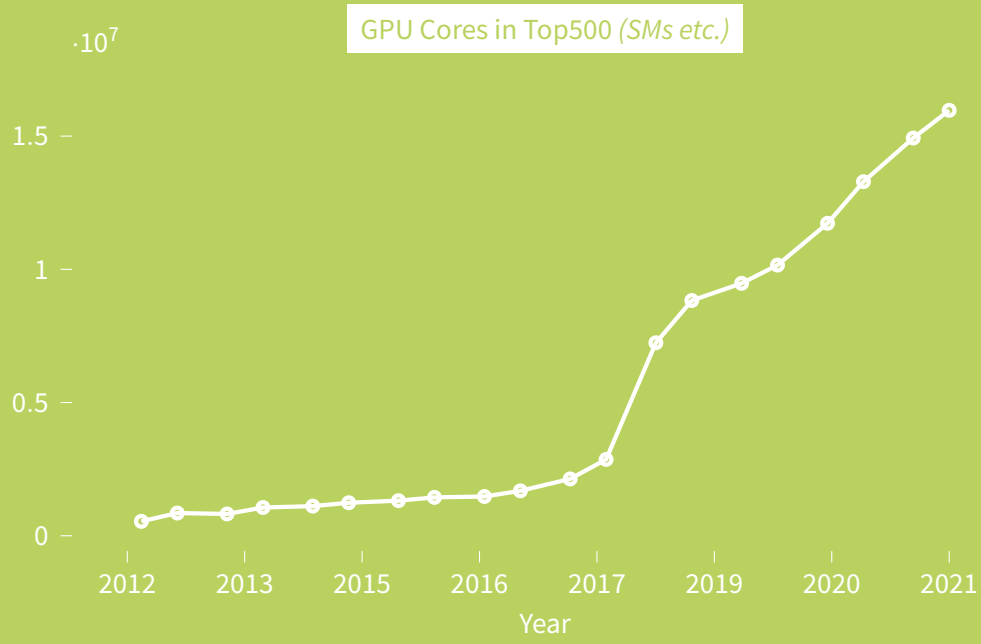
29 May 2022 | Andreas Herten | Jülich Supercomputing Centre, Forschungszentrum Jülich

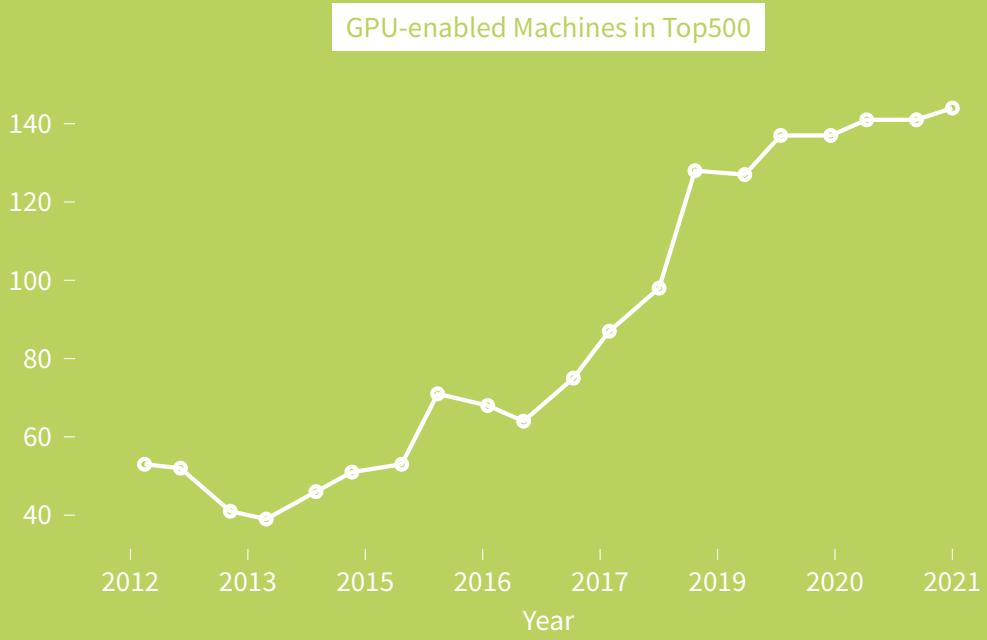
JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

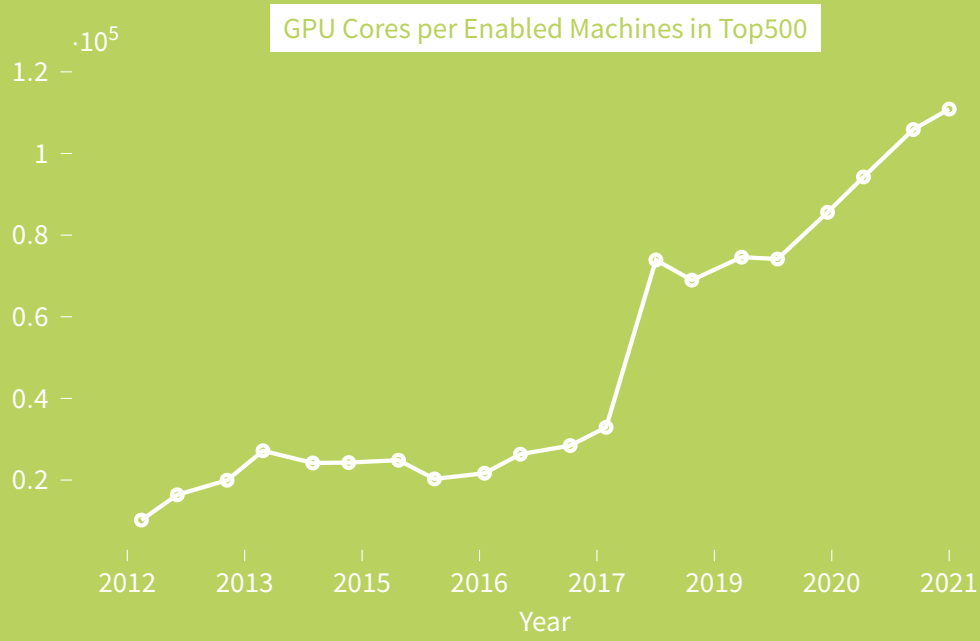
Welcome to

Efficient Distributed GPU Programming for Exascale,

an *ISC22 Tutorial*


GPU Cores in Top500 (SMs etc.)
·10^7
1.5
1
0.5
0
2012
2013
2015
2016
2017
2019
2020
2021
Year


GPU-enabled Machines in Top500
140
120
100
80
60
40
2012
2013
2015
2016
2017
2019
2020
2021
Year


GPU Cores per Enabled Machines in Top500
·10^5
1.2
1
0.8
0.6
0.4
0.2
2012
2013
2015
2016
2017
2019
2020
2021
Year

State of the GPUnion

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 GPUnion

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]

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

A: AMD, I: Intel, N: NVIDIA

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

State of the GPUnion

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 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]

We help to Tame the Beasts!

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

[A]: AMD, [I]: Intel, [N]: NVIDIA

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 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 follow along (see last lecture)

Curriculum

1L	Lecture	Tutorial Overview, Introduction to System
	Onboarding	*Accessing JUWELS Booster*
2L	Lecture	Introduction to MPI-Distributed Computing with GPUs
3H	Hands-On	Multi-GPU Parallelization
		Coffee Break
4L	Lecture	Performance and Debugging Tools
5L	Lecture	Optimization Techniques for Multi-GPU Applications
6H	Hands-On	Overlap Communication and Computation with MPI
		Lunch Break
7L	Lecture	Overview of NCCL and NVSHMEM in MPI Programs
8H	Hands-On	Using NCCL and NVSHMEM
		Coffee Break
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

Andreas Herten
Accelerating Devices Lab
Jülich Supercomputing Centre

Markus Hrywniak
DevTech Compute
NVIDIA

Jiri Kraus
DevTech Compute
NVIDIA

Lena Oden
Prof. for Computer Engineering
University Hagen

System: JUWELS Booster

JUWELS Overall Architecture

JUWELS Cluster (2018)

- 2511 compute nodes (2× Skylake)
- 48 GPU nodes (4× 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× Skylake)
- 48 GPU nodes (4× 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× AMD Rome, 4× A100 w/ NVLink3)
- Mellanox HDR 200 Gbit/s network, DragonFly+ topology
- 73 PFLOP/s

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

JUWELS Overall Architecture

JUWELS Cluster (2018)

- 251 ke)
- 48 nk2)
- Me rk,
  fat-
- 12

TOP 500 The List.

Top500 Jun-2021:

- #1 Europe
- #8 World
- #4* 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

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

JUWELS Booster Overview

Node Configuration

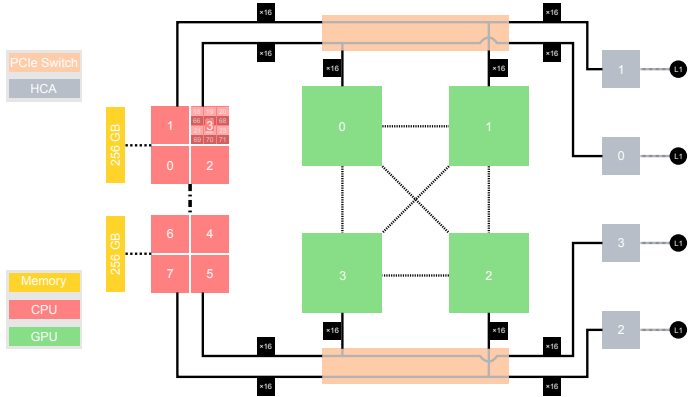
Arch Atos Bull Sequana XH2000

CPU 2 × AMD EPYC 7402:
$2_{\text{Socket}} \times 24_{\text{Core}} \times 2_{\text{SMT}}$,
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

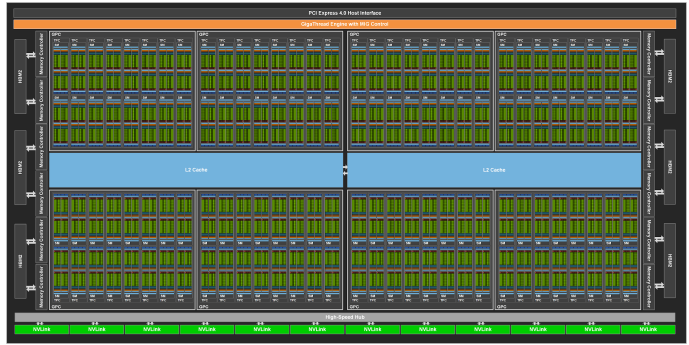
etc 2 × PCIe Gen 4 switch



JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

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



JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

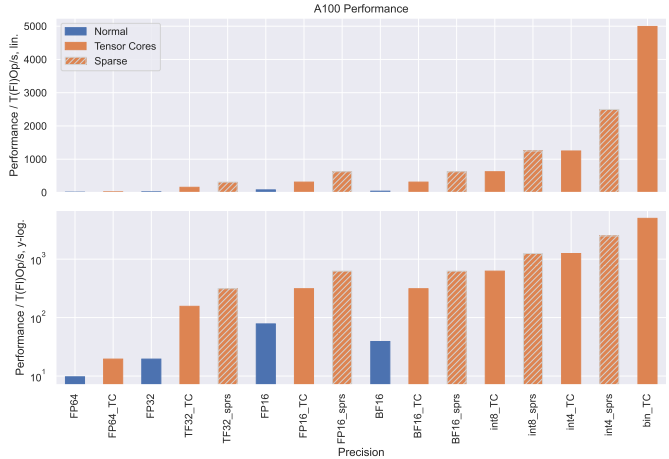
NVIDIA A100 Tensor Core GPU

- 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

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

NVIDIA A100 Tensor Core GPU

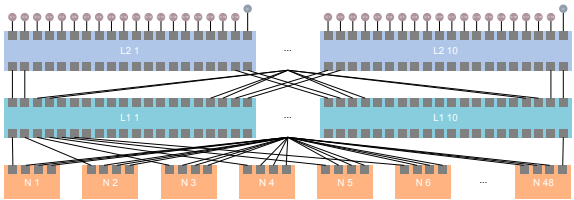
- JUWELS Booster:
  3744 A100 GPUs
- Vs. V100: More SMs, larger/faster L1, larger/faster L2, faster memory, faster NVLink, faster PCIe, …
- New features: MIG, async copy to/barrier in shared mem
- New Tensor Cores, new performances:
  73 PFLOP/s_{FP64TC} 584 PFLOP/s_{FP32TC}
  1.16 EFLOP/s_{FP16TC} 18.7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

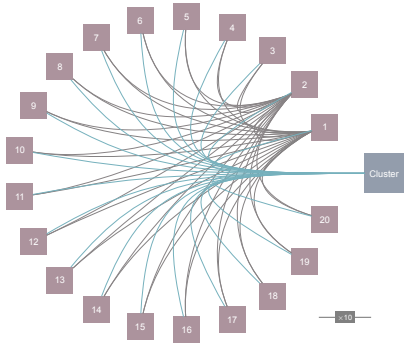


JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

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

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

JUWELS

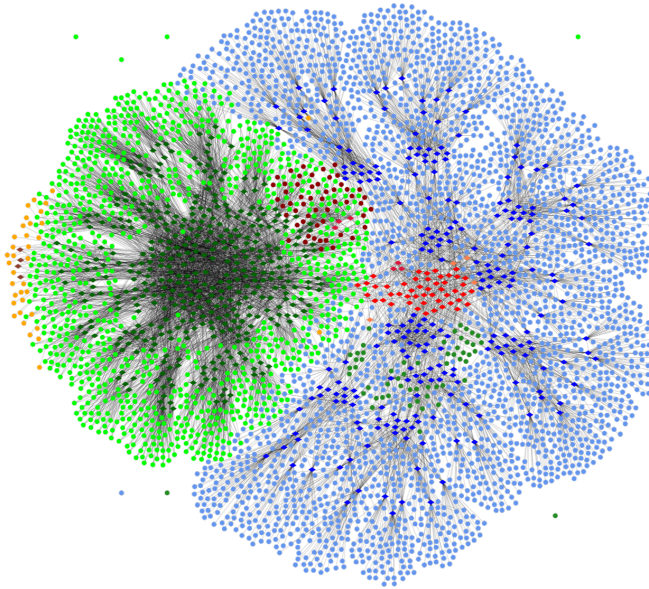
Cluster Booster Integration

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
- Cluster GPU node
- Cluster switch
- Cluster gateway
- Top-level switch
- Booster node
- Booster switch
- Booster gateway
- JUST
- Service node



Graphic provided by D. Alvarez

AMD EPYC Rome

Some Processor Basics

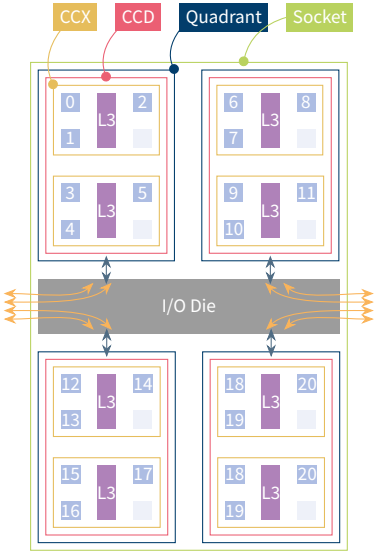
- JUWELS Booster: AMD EPYC Rome 7402 (24 core (SMT-2) $\times$ 2 sockets)
- EPYC processor: Built as Multi-Chip Module
  $\rightarrow$ Many building blocks, hierarchies

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
  $\rightarrow$ 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



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

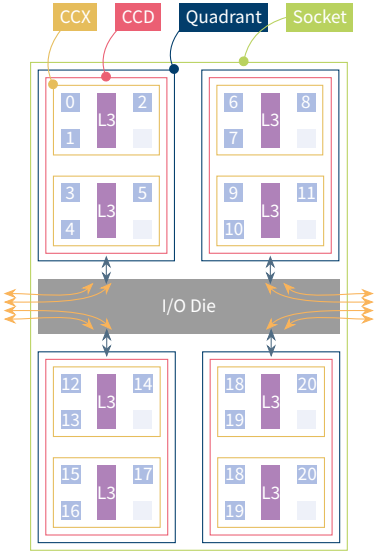
AMD EPYC Rome

Some Processor Basics

- JUWELS Booster: AMD EPYC Rome 7402 (24 core (SMT-2) $\times$ 2 sockets)
- EPYC processor: Built as Multi-Chip Module
  $\rightarrow$ 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

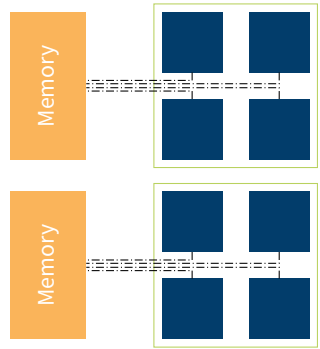
$\Rightarrow$ **Complex topology!**



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

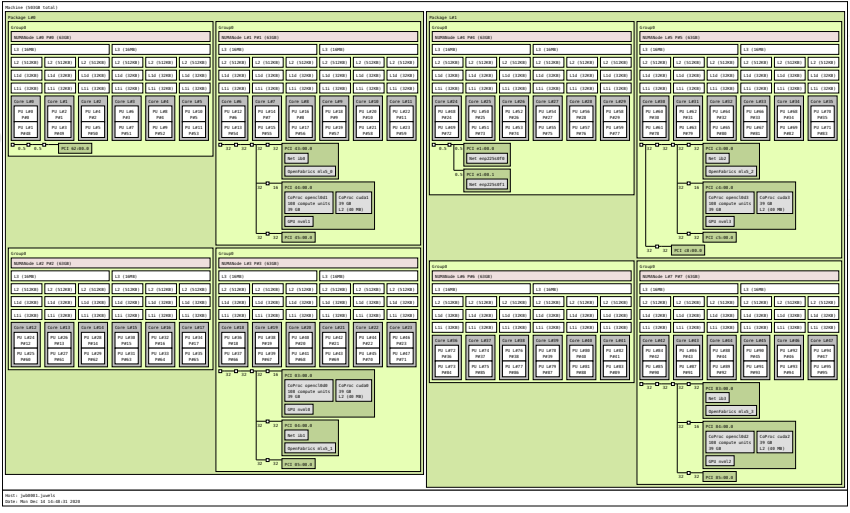
PCIe Affinity

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



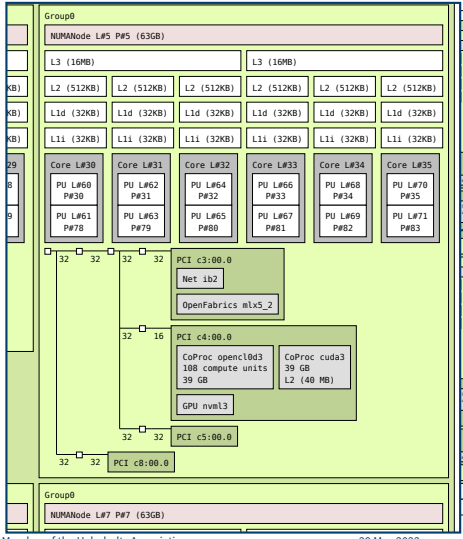
PCIe Affinity

- PCIe via
- GPUs co



lstopo --of pdf topo.pdf; or
lstopo-no-graphics out.xml; and lstopo -i out.xml --of svg > out.svg;

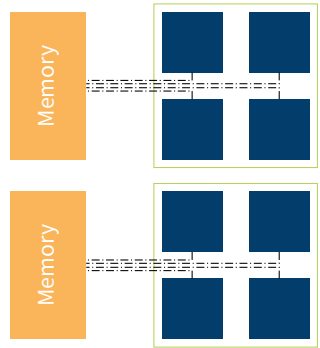
PCIe Affinity



lstopo --of pdf topo.pdf; or
lstopo-no-graphics out.xml; and lstopo -i out.xml --of svg > out.svg;

PCIe Affinity

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



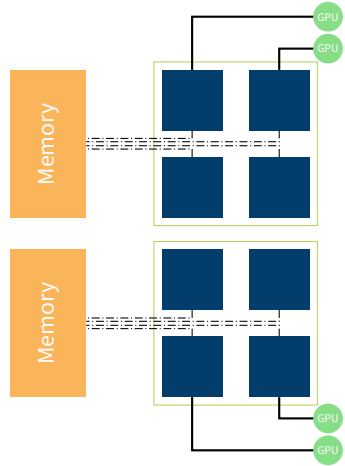
PCIe Affinity

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

→ *True* GPU affinity only by half of chiplets

But impact application-dependent and possibly quite low



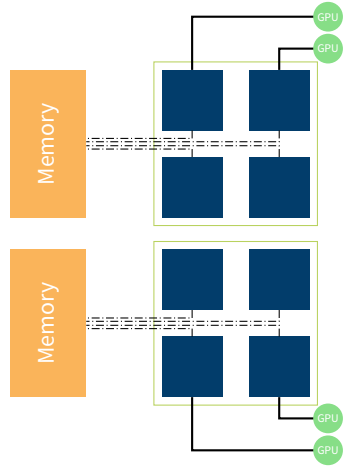
PCIe Affinity

- 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

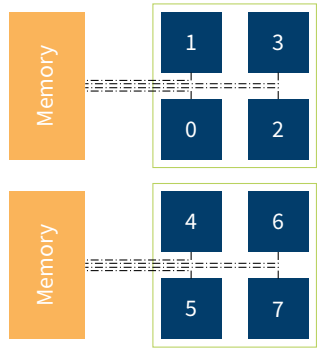
But impact application-dependent and possibly quite low

- InfiniBand HCA adapters also via PCIe switch
  - Same affinity considerations
  - 1:1 relation between HCA and GPU



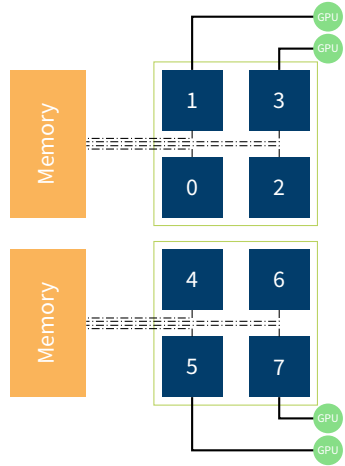
NUMA Numbering

- NUMA domains (= quadrants) are numbered



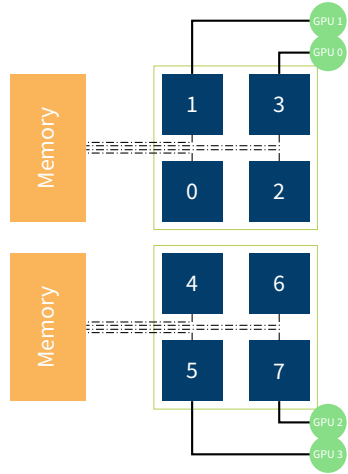
NUMA Numbering

- 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…*

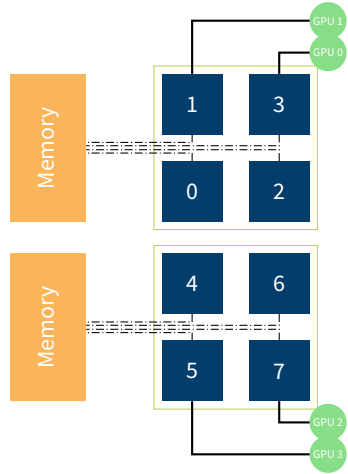


NUMA Numbering

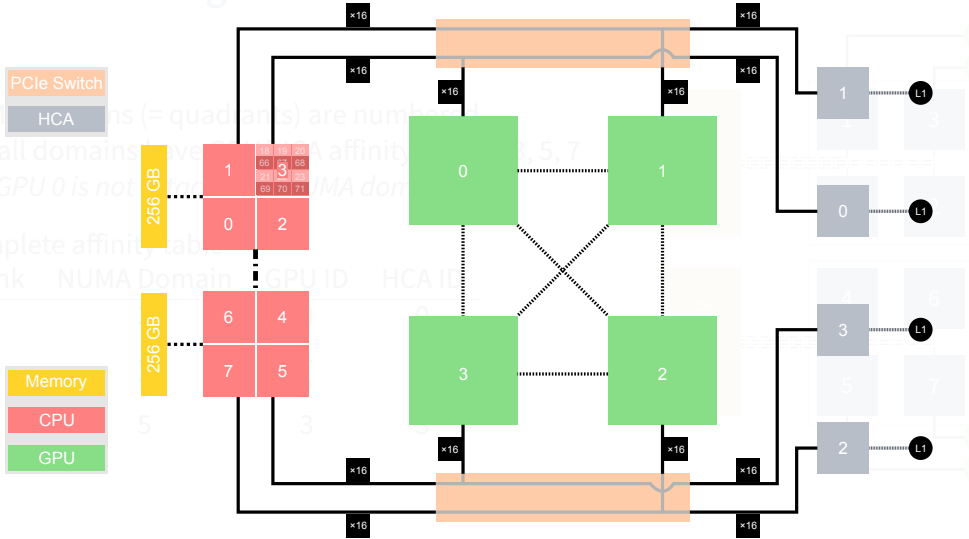
- NUMA domains (= quadrants) are numbered
- Not all domains have GPU/HCA affinity, only 1, 3, 5, 7
- *But GPU 0 is not attached to NUMA domain 0…*

- 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



Affinity in Practice

CPU and GPU Affinity

- We configured *PS*Slurm to use best topology **as default**

Affinity in Practice

CPU and GPU Affinity

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

Affinity in Practice

CPU and GPU Affinity

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

Affinity in Practice

CPU and GPU Affinity

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

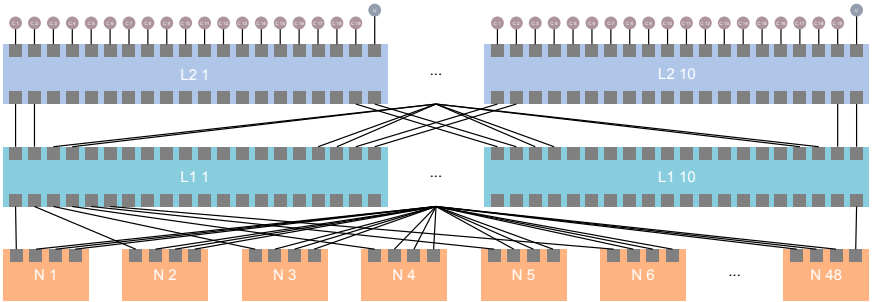
Overview

- 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, 48 port *(each L1 switch: 2 links to L2, strided to 10 L2 switches)*
  - 10 L2 switches, 48 port *(each L2 switch: 1 link to L2 switch of all other cells, 1 link to cluster)*

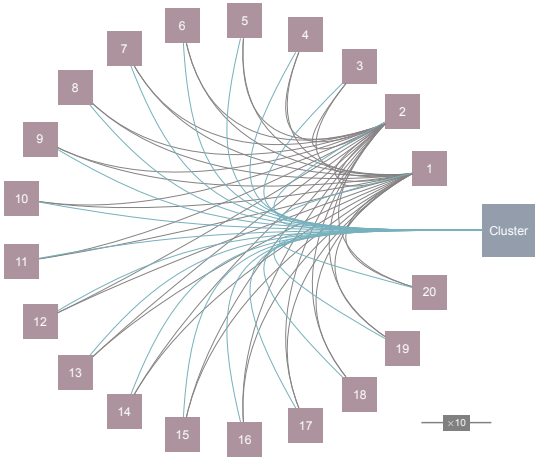


DragonFly+ Topology

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:

$$\mathcal{B}(N) = \lfloor (N/2)^2 \rfloor \times (10 \times \mathrm{bw}_1)$$

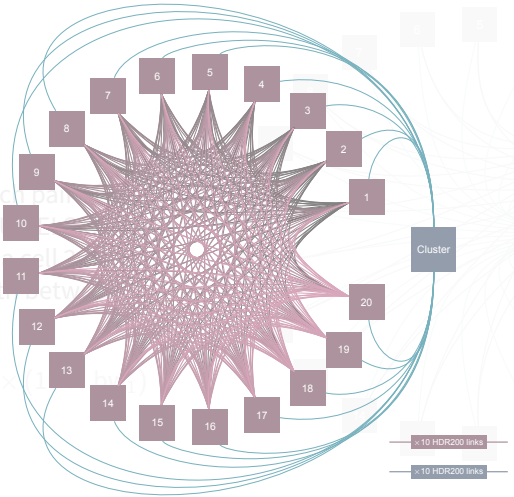


DragonFly+ Topology

Intra-Cell Topology

- Inter-cell: 20 cells
  - 10 links between each pa
  - 10 links per cell to J
  - Filesystem access via all
  - Bi-section bandwid
    cells:

$$B(N) = \lfloor (N/2)^2 \rfloor$$



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