JUWELS BOOSTER 101

JUWELS BOOSTER TUNING & SCALING WORKSHOP

7 March 2022 | Andreas Herten | Jülich Supercomputing Centre, Forschungszentrum Jülich

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

Overview

JUWELS Bird's Eye View
- JUWELS Overall Architecture
- JUWELS Booster Architecture
- JUWELS Integration

Architecture Details
- Aside: AMD EPYC Core Design
- Intra-Node Affinities
- Inter-Node Affinity
- Job Reportings

Summary and Conclusions

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

JUWELS Overall Architecture

JUWELS Cluster (2018)

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

Top500 Nov-2021:

- #1 Europe
- #8 World
- #4* Top/Green500

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

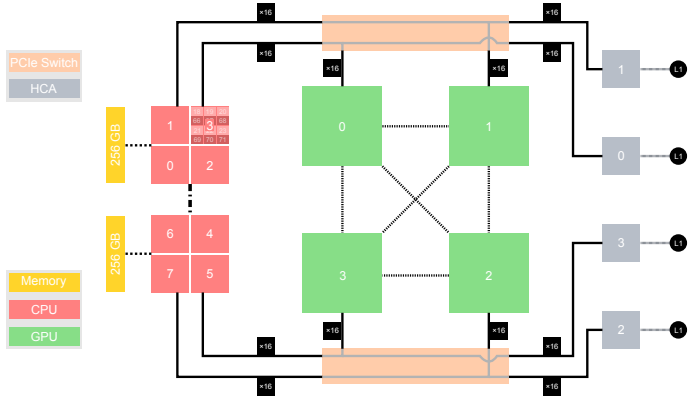
Arch Atos Bull Sequana XH2000

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

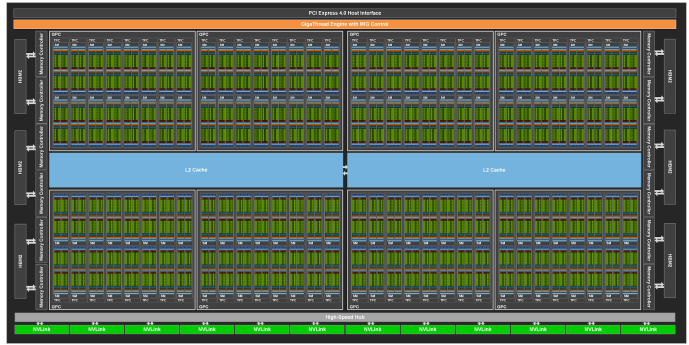
etc 2 × PCIe Gen 4 switch



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

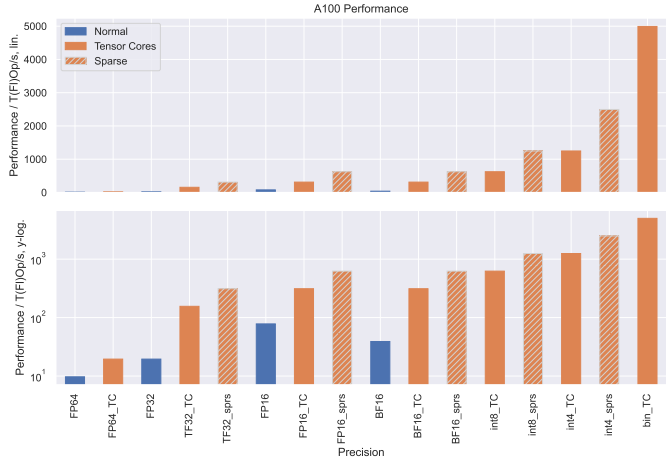
- JUWELS Booster: 3744 A100 GPUs
- Vs. V100: More SMs, larger/faster L1, larger/faster L2, faster memory, faster NVLink, faster PCIe, …
- New features: MIG, async copy to/barrier in shared mem
- New Tensor Cores

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

NVIDIA A100 Tensor Core GPU

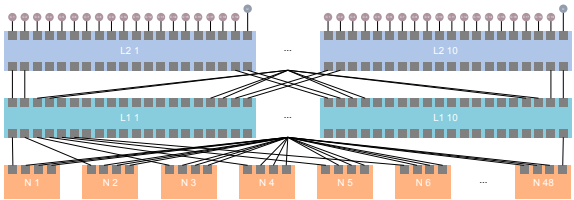
- JUWELS Booster: 3744 A100 GPUs
- Vs. V100: More SMs, larger/faster L1, larger/faster L2, faster memory, faster NVLink, faster PCIe, …
- New features: MIG, async copy to/barrier in shared mem
- New Tensor Cores, new performances:

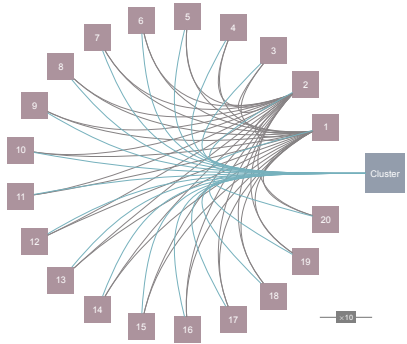
  73 PFLOP/s_{FP64TC} 584 PFLOP/s_{FP32TC}
  1.16 EFLOP/s_{FP16TC} 18.7 EOP/s_{BinTC}



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

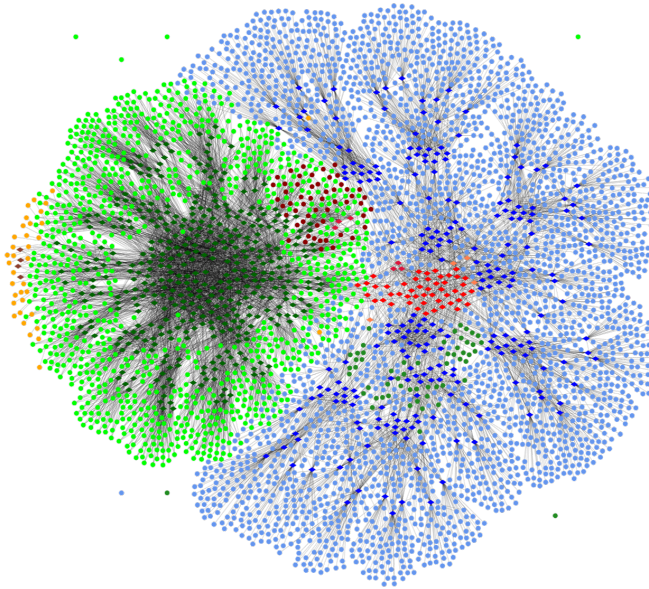
Cluster Booster Integration

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

JUWELS Software Stack

- Software
  - Software management: EasyBuild, LMod
  - Compilers: GCC, Intel, NVHPC
  - GPU-aware MPIs (ParaStationMPI, OpenMPI; via UCX)
  - New Stage: 2022

  → `https://apps.fz-juelich.de/jsc/llview/juwels_modules_booster/`
- Operation
  - Operation System: RockyLinux 8.5
  - Provisioning: Ansible

Architecture Details

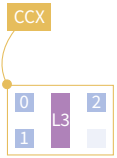
AMD EPYC Rome

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

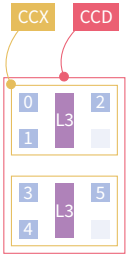
  3 cores Core-Complex (CCX), shared L3 *(max 4 cores)*

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)

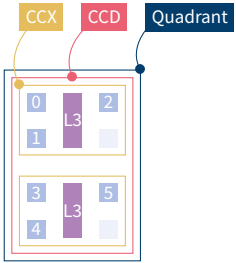


AMD EPYC Rome

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)*

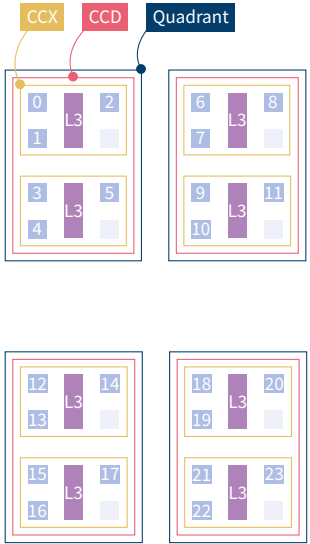


AMD EPYC Rome

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



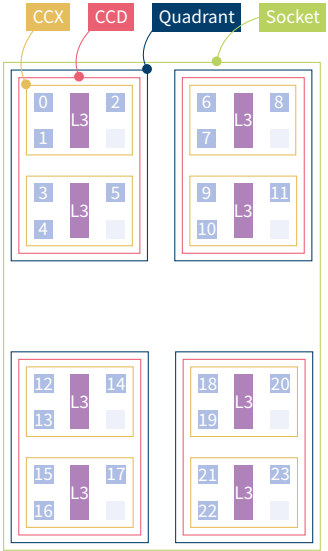
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

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)*



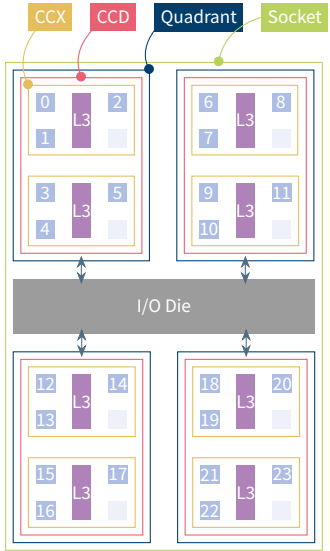
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

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



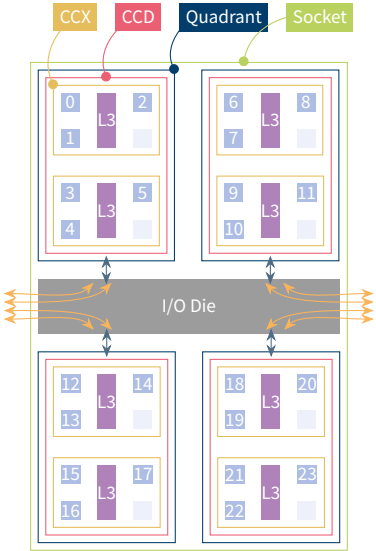
JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

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



JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

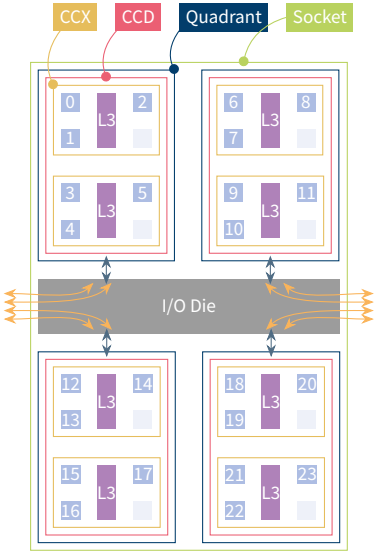
AMD EPYC Rome

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

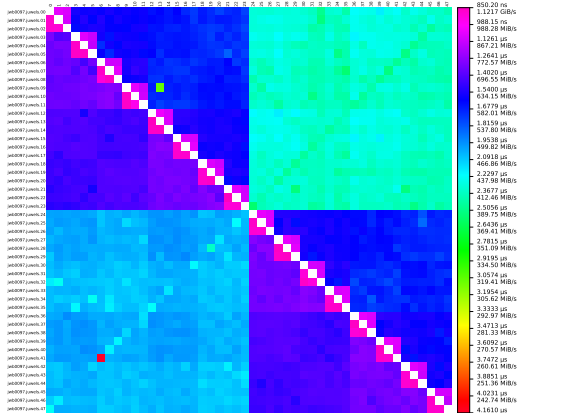
EYPC Core-to-Core

Linktest

- Test core-to-core connections with JSC's Linktest
- Linktest: MPI Ping Pong usually for large machines
- Here: Within a node
- EPYC structure clearly visible

⇒ Thread placement very important, especially for L3-using application
We saw 30 % performance increase by proper placement

- *Configuration: GCC, ParaStationMPI, serial*



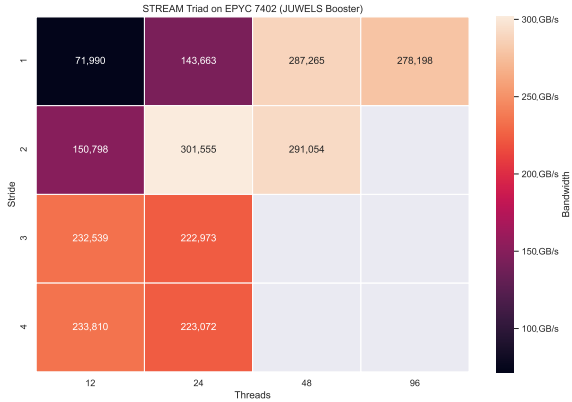
Graphic provided by M. Holicki

Memory

STREAM

- EPYC Rome: 8 memory channels, 2 per quadrant
- Thread placement important for memory performance
- STREAM benchmark! *all in Appendix*

⇒ Thread placement very important, also for memory-intensive application

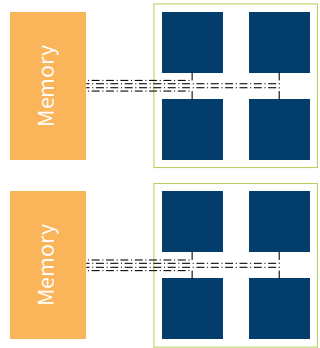


Stride: `OMP_PLACES={0}:24:2` → stride 2

STREAM config: AOCC 3.0.0, `-O3 -mcmodel=large -ffp-contract=fast -fnt-store`, array size: 2500000000, best of 200

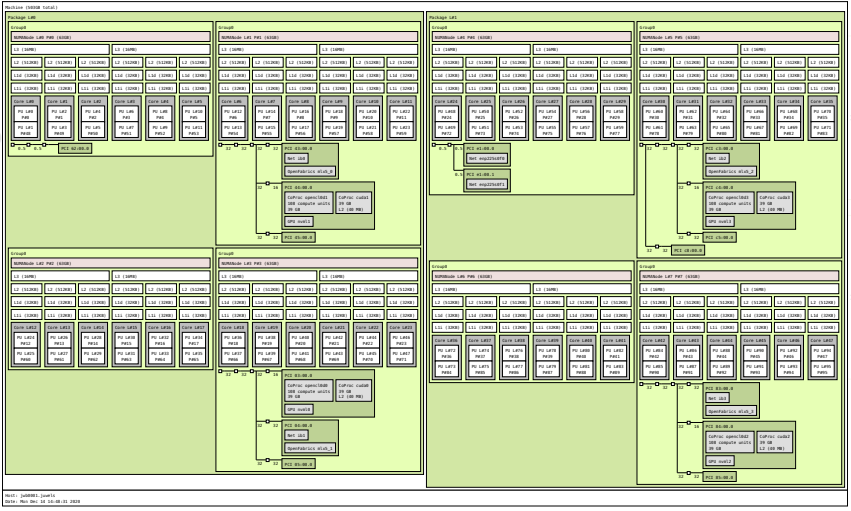
PCIe Affinity

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

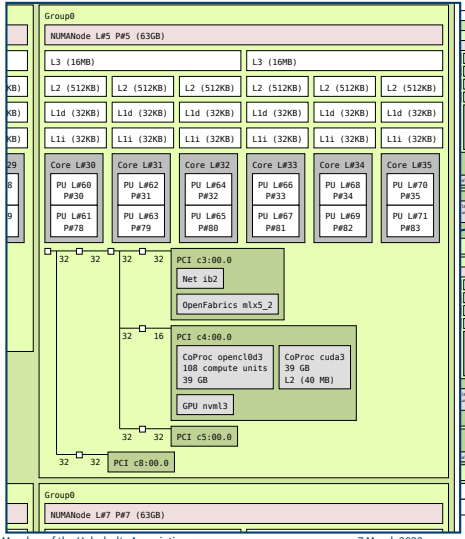


PCIe Affinity

- PCIe via
- GPUs co

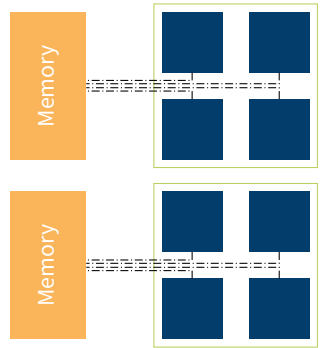


PCIe Affinity



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

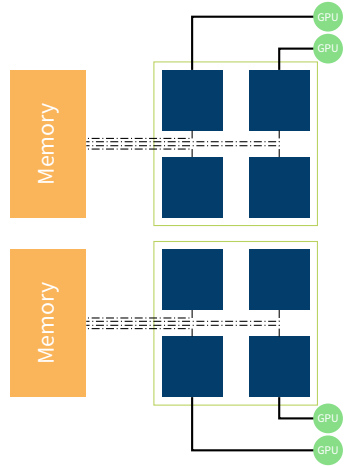


PCIe Affinity

- 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



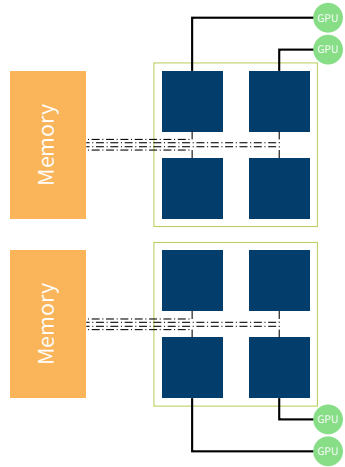
PCIe Affinity

- 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

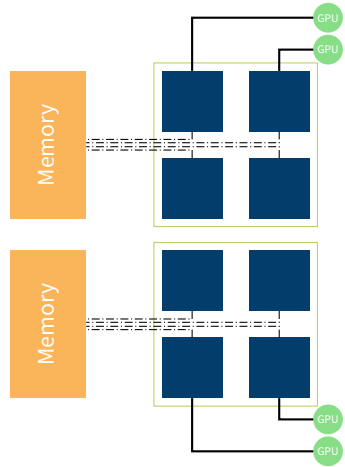
But impact application-dependent and possibly quite low

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



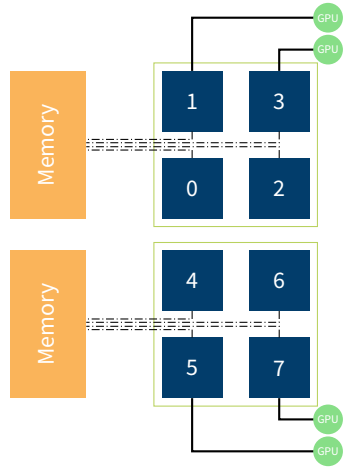
NUMA Numbering

- NUMA domains (= quadrants) are numbered
- Not all domains have GPU/HCA affinity
- Odd-numbered NUMA domains have affinity



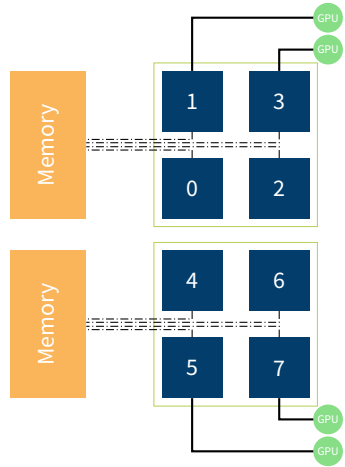
NUMA Numbering

- NUMA domains (= quadrants) are numbered
- Not all domains have GPU/HCA affinity
- Odd-numbered NUMA domains have affinity



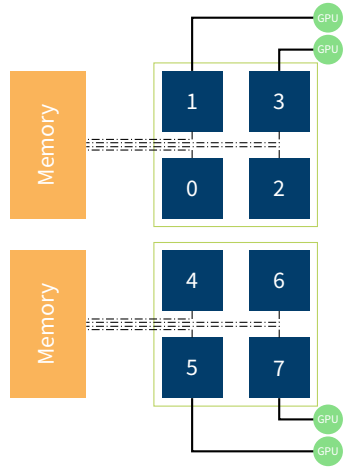
NUMA Numbering

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



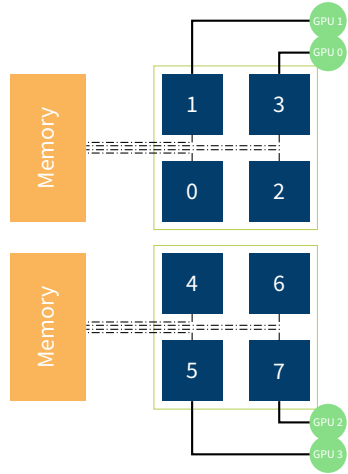
NUMA Numbering

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

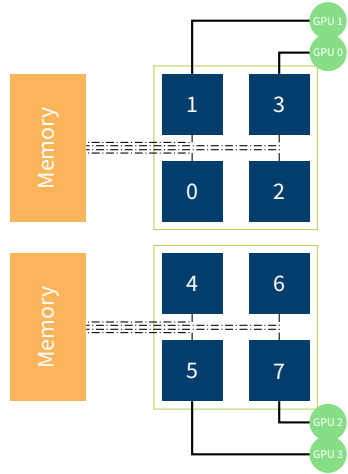


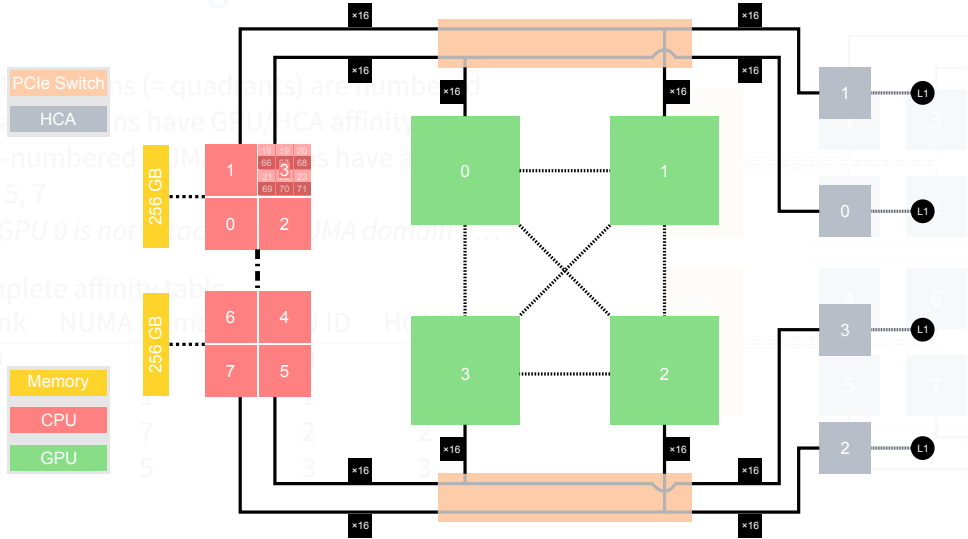
NUMA Numbering

- NUMA domains (= quadrants) are numbered
- Not all domains have GPU/HCA affinity
- Odd-numbered NUMA domains have affinity
- 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




PCIe Switch
HCA
Memory
CPU
GPU
256 GB
256 GB
×16
L3
1
0
2
6
4
7
5
0
1
3
2
L1

Affinity

- Modification of Slurm to match node topology
- Recipe: GPU first, CPU second

Affinity

- Modification of Slurm to match node topology
- Recipe: GPU first, CPU second
- Example Slurm task distribution for 5 tasks, each 1 core:

  Task 1 First core of NUMA domain 3
  Task 2 First core of NUMA domain 1
  Task 3 First core of NUMA domain 7
  Task 4 First core of NUMA domain 5
  Task 5 First core of NUMA domain 2

Affinity

- Modification of Slurm to match node topology
- Recipe: GPU first, CPU second
- Example Slurm task distribution for 5 tasks, each 1 core:

  Task 1 First core of NUMA domain 3
  Task 2 First core of NUMA domain 1
  Task 3 First core of NUMA domain 7
  Task 4 First core of NUMA domain 5
  Task 5 First core of NUMA domain 2

- Fill up NUMA domain with `--cpus-per-task N`

Affinity

- Modification of Slurm to match node topology
- Recipe: GPU first, CPU second
- Example Slurm task distribution for 5 tasks, each 1 core:

  Task 1 First core of NUMA domain 3
  Task 2 First core of NUMA domain 1
  Task 3 First core of NUMA domain 7
  Task 4 First core of NUMA domain 5
  Task 5 First core of NUMA domain 2

- Fill up NUMA domain with `--cpus-per-task N`
- Realized by CPU affinity masks, environment variables (GPU, HCA)

Affinity

- Modification of Slurm to match node topology
- Recipe: GPU first, CPU second
- Example Slurm task distribution for 5 tasks, each 1 core:

  Task 1 First core of NUMA domain 3
  Task 2 First core of NUMA domain 1
  Task 3 First core of NUMA domain 7
  Task 4 First core of NUMA domain 5
  Task 5 First core of NUMA domain 2

- Fill up NUMA domain with `--cpus-per-task N`
- Realized by **CPU affinity masks**, environment variables (GPU, HCA)

Affinity

CPU Affinity Mask I

- *PS*Slurm sets CPU affinity masks with *GPU first*

Affinity

CPU Affinity Mask I

- *PS*Slurm sets CPU affinity masks with *GPU first*
- CPU mask example for 5 tasks, each 1 core:

```
$ srun --cpu-bind=verbose -n 5 bash -c "" |& sort
cpu_bind=THREADS - jwb0001, task  0  0 [18049]: mask 0x40000 set
cpu_bind=THREADS - jwb0001, task  1  1 [18050]: mask 0x40 set
cpu_bind=THREADS - jwb0001, task  2  2 [18055]: mask 0x40000000000 set
cpu_bind=THREADS - jwb0001, task  3  3 [18059]: mask 0x40000000 set
cpu_bind=THREADS - jwb0001, task  4  4 [18061]: mask 0x1000 set
```

Affinity

CPU Affinity Mask I

- *PS*Slurm sets CPU affinity masks with *GPU first*
- CPU mask example for 5 tasks, each 1 core:

```
$ srun --cpu-bind=verbose -n 5 bash -c "" |& sort
cpu_bind=THREADS - jwb0001, task  0  0 [18049]: mask 0x40000 set
cpu_bind=THREADS - jwb0001, task  1  1 [18050]: mask 0x40 set
cpu_bind=THREADS - jwb0001, task  2  2 [18055]: mask 0x40000000000 set
cpu_bind=THREADS - jwb0001, task  3  3 [18059]: mask 0x40000000 set
cpu_bind=THREADS - jwb0001, task  4  4 [18061]: mask 0x1000 set
```

- Visualize masks in binary representation *(SMT-2 omitted)*

hex2bin

```
000000 000000 000000 100000 000000 000000 000000 000000  # 0x40000, GPU 0
000000 100000 000000 000000 000000 000000 000000 000000  # 0x40, GPU 1
000000 000000 000000 000000 000000 000000 000000 100000  # 0x40000000000, GPU 2
000000 000000 000000 000000 000000 100000 000000 000000  # 0x40000000, GPU 3
000000 000000 100000 000000 000000 000000 000000 000000  # 0x1000, no GPU
```

Affinity

CPU Affinity Mask II

- *PS*Slurm sets CPU affinity masks with *GPU first*
- CPU mask example for 2 tasks, each 2 cores:

```
$ srun --cpu-bind=verbose -n 2 --cpus-per-task 2 bash -c "" |& sort
cpu_bind=THREADS - jwb0001, task  0  0 [18402]: mask 0xc0000 set
cpu_bind=THREADS - jwb0001, task  1  1 [18403]: mask 0xc0 set
```

hex2bin

```
000000 000000 000000 110000 000000 000000 000000 000000  # 0xc0000, GPU 0
000000 110000 000000 000000 000000 000000 000000 000000  # 0xc0, GPU 1
```

Affinity

GPU Affinity

- *PS*Slurm sets `$CUDA_VISIBLE_DEVICES` to associated GPU

Affinity

GPU Affinity

- *PS*Slurm sets `$CUDA_VISIBLE_DEVICES` to associated GPU
- 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
```

Affinity

GPU Affinity

- *PS*Slurm sets `$CUDA_VISIBLE_DEVICES` to associated GPU
- GPU affinity for 2 tasks:

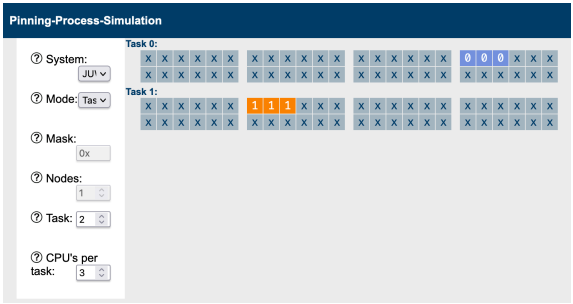
```
$ srun -n 2 --cpu-bind=verbose env | grep 'PMI_RANK\|CUDA_VIS'
cpu_bind=THREADS - jwb0001, task  0  0 [18307]: mask 0x40000 set
cpu_bind=THREADS - jwb0001, task  1  1 [18309]: mask 0x40 set
PMI_RANK=0
CUDA_VISIBLE_DEVICES=0
PMI_RANK=1
CUDA_VISIBLE_DEVICES=1
```

- Special case: GPU affinity for exactly 1 task:

```
$ srun -n 1 env | grep 'PMI_RANK\|CUDA_VIS'
PMI_RANK=0
CUDA_VISIBLE_DEVICES=0,1,2,3
```

Handling Affinity Handling

- Node affinity now more important than ever! *(maybe…)*
- JSC provides tools and documentation as first-level support!
- Mask generator
- CLI mask tool
- **Extensive system documentation**



apps.fz-juelich.de/jsc/llview/pinning/

DragonFly+ Topology

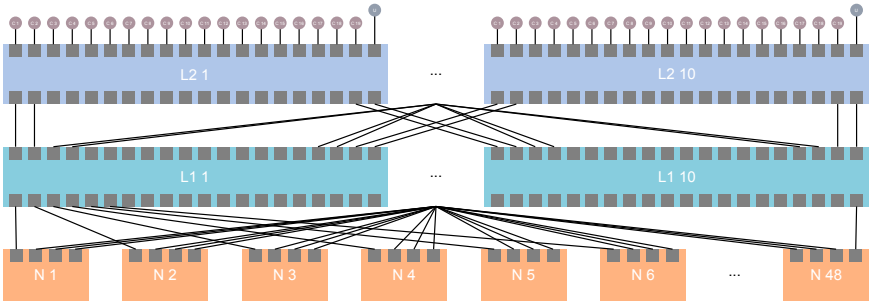
Overview

- 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)*

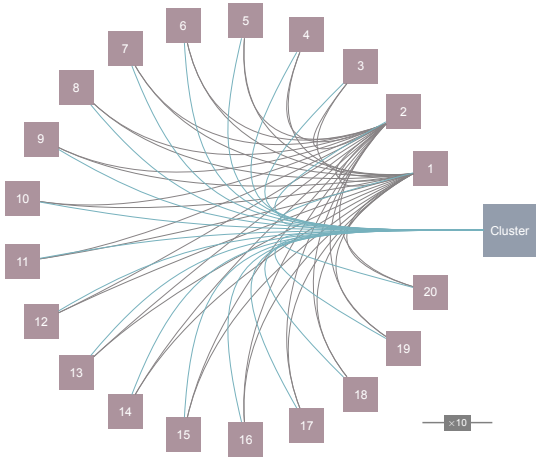


DragonFly+ Topology

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:

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

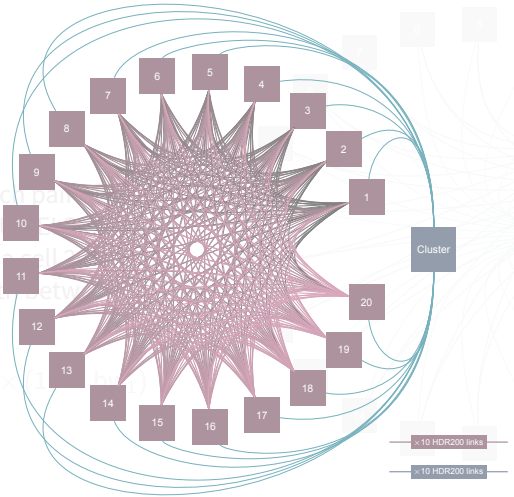


DragonFly+ Topology

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:

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



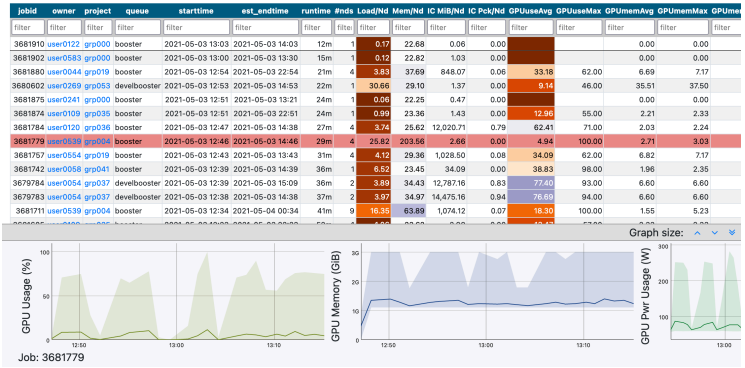
Interactive Job Reporting

LLview

- LLview: JSC's online tool for scheduler statistics
- Already around for some time, now extended with GPU-related info
- System overview (nodes, GPUs; timeline)
- Job focus (many metrics)

→ `llview.fz-juelich.de`

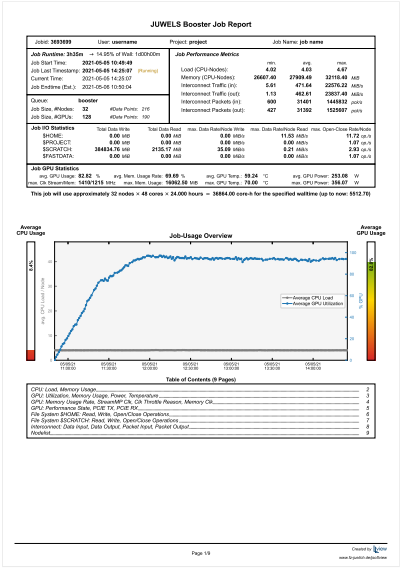
- JSC: Access via JuDoor



PDF Job Reports

From LLview

- **New:** Detailed job reports in HTML
- Automatically generated
- Many metrics, graphs
- Via LLview job view, for jobs $>$ 5 min



JUWELS Booster Job Report

Jobid: 3033699 User: username Project: project Job Name: job name

Job Runtime: 3h26m → 14.06% of Wall: 1d20h00m

Job Start Time: 2021-05-05 10:49:49
Job Last Timestamp: 2021-05-05 14:25:07 (Running)
Current Time: 2021-05-05 14:25:07
Job Endtime (Est.): 2021-05-06 10:53:04

Queue: booster
Job Size, #Nodes: 32
Job Size, #CPUs: 128
#Data Points: 216
#Data Points: 130

Job Performance Metrics

	min.	avg.	max.	
Load (CPU-Nodes):	4.02	4.05	4.07	
Memory (CPU-Nodes):	26607.40	27909.49	32110.40	MB
Interconnect. Traffic (in):	5.01	471.64	22576.22	MB/s
Interconnect. Traffic (out):	1.13	462.61	23837.40	MB/s
Interconnect. Packets (in):	600	31401	1445832	pckt/s
Interconnect. Packets (out):	427	31302	1525697	pckt/s

Job I/O Statistics

	Total Data Written	Total Data Read	max. Data Rate/node Write	max. Data Rate/node Read	max. Open-Close Rate/node
$HOME:	0.00 MiB	0.00 MiB	0.00 MiB/s	11.53 MiB/s	11.72 op/s
$PROJECT:	0.00 MiB	0.00 MiB	0.00 MiB/s	0.00 MiB/s	1.07 op/s
$SCRATCH:	38433.41 MiB	2135.17 MiB	35.09 MiB/s	0.21 MiB/s	2.93 op/s
$FASTDATA:	0.00 MiB	0.00 MiB	0.00 MiB/s	0.00 MiB/s	1.07 op/s

Job GPU Statistics

avg. GPU Usage: 62.82 %	avg. Mem. Usage Rate: 69.69 %	avg. GPU Temp.: 59.24 °C	avg. GPU Power: 253.08 W
max. Clk Streamfilters: 1410/1215 MHz	max. Mem. Usage: 18092.50 MiB	max. GPU Temp.: 70.00 °C	max. GPU Power: 338.07 W

This job will use approximately 32 nodes × 48 cores × 24.000 hours = 36864.00 core-h for the specified walltime (up to now: 5512.76)

Average CPU Usage

Job-Usage Overview

Average GPU Usage

Table of Contents (9 Pages)

CPU Load, Memory Usage 2
GPU Utilization, Memory Usage, Power, Temperature 3
GPU Memory Usage Rate, StreamMP Clk, Clk Throttle Reason, Memory Clk 4
GPU Performance State, PCIE TX, PCIE RX 5
File System $HOME: Read, Write, Open/Close Operations 6
File System $SCRATCH: Read, Write, Open/Close Operations 7
Interconnect. Data Input, Data Output, Packet Input, Packet Output 8
Hostlist 9

Page 1/9

Created by LLview

www.fz-juelich.de/jsc/llview

JÜLICH Forschungszentrum

JÜLICH SUPERCOMPUTING CENTRE

Jobid: **3693699** User: **username** Project: **project** Job Name: **job name**

Job Runtime: **3h35m** → 14.95% of Wall: 1d00h00m

Job Start Time: **2021-05-05 10:49:49**

Job Last Timestamp: **2021-05-05 14:25:07** (Running)

Current Time: 2021-05-05 14:25:07

Job Endtime (Est.): 2021-05-06 10:50:04

Queue: **booster**

Job Size, #Nodes: **32** *#Data Points: 216*

Job Size, #GPUs: **128** *#Data Points: 190*

Job Performance Metrics

	min.	avg.	max.	
Load (CPU-Nodes):	**4.02**	**4.03**	**4.67**	
Memory (CPU-Nodes):	**26607.40**	**27909.49**	**32118.40**	*MiB*
Interconnect Traffic (in):	**5.61**	**471.64**	**22576.22**	*MiB/s*
Interconnect Traffic (out):	**1.13**	**462.61**	**23837.40**	*MiB/s*
Interconnect Packets (in):	**600**	**31401**	**1445832**	*pck/s*
Interconnect Packets (out):	**427**	**31392**	**1525607**	*pck/s*

Job I/O Statistics

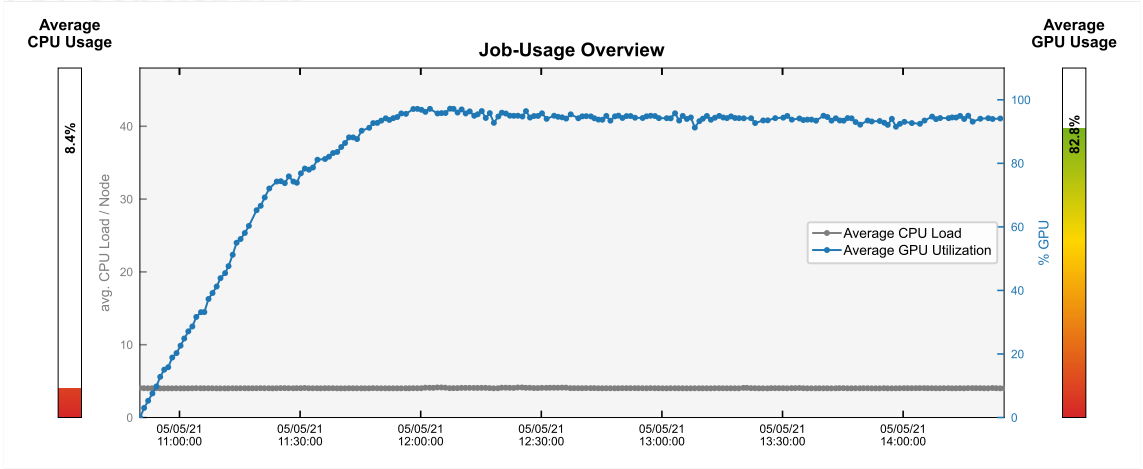
	Total Data Write	Total Data Read	max. Data Rate/Node Write	max. Data Rate/Node Read	max. Open-Close Rate/Node
$HOME:	**0.00** *MiB*	**0.00** *MiB*	**0.00** *MiB/s*	**11.53** *MiB/s*	**11.72** *op./s*
$PROJECT:	**0.00** *MiB*	**0.00** *MiB*	**0.00** *MiB/s*	**0.00** *MiB/s*	**1.07** *op./s*
$SCRATCH:	**384834.76** *MiB*	**2135.17** *MiB*	**35.09** *MiB/s*	**0.21** *MiB/s*	**2.93** *op./s*
$FASTDATA:	**0.00** *MiB*	**0.00** *MiB*	**0.00** *MiB/s*	**0.00** *MiB/s*	**1.07** *op./s*

Job GPU Statistics

avg. GPU Usage:	**82.82** %	avg. Mem. Usage Rate:	**69.69** %	avg. GPU Temp.:	**59.24** °C	avg. GPU Power:	**253.08** W
max. Clk Stream/Mem:	**1410/1215** MHz	max. Mem. Usage:	**16062.50** MiB	max. GPU Temp.:	**70.00** °C	max. GPU Power:	**356.07** W

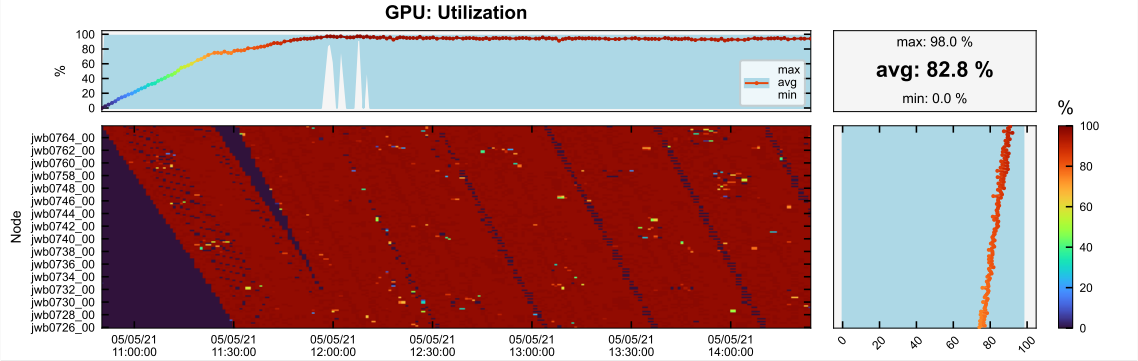
This job will use approximately 32 nodes × 48 cores × 24.000 hours = 36864.00 core-h for the specified walltime (up to now: 5512.70)

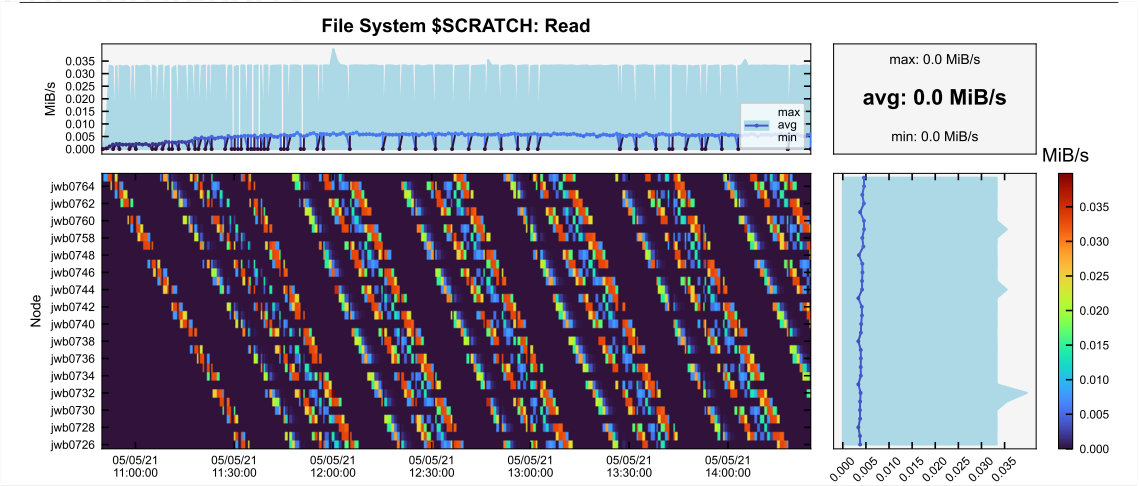
PDF Job Reports



PDF Job Reports

From LLview




File System $SCRATCH: Read
MiB/s
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
max
avg
min
max: 0.0 MiB/s
avg: 0.0 MiB/s
min: 0.0 MiB/s
MiB/s
Node
jwb0764
jwb0762
jwb0760
jwb0758
jwb0748
jwb0746
jwb0744
jwb0742
jwb0740
jwb0738
jwb0736
jwb0734
jwb0732
jwb0730
jwb0728
jwb0726
05/05/21 11:00:00
05/05/21 11:30:00
05/05/21 12:00:00
05/05/21 12:30:00
05/05/21 13:00:00
05/05/21 13:30:00
05/05/21 14:00:00
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035

Nodelist

1	jwb0726	2	jwb0727	3	jwb0728	4	jwb0729	5	jwb0730	6	jwb0731
1	**GPU0:** 74.1% 14.5GiB 1332MHz	5	**GPU0:** 74.5% 14.5GiB 1332MHz	9	**GPU0:** 74.8% 14.5GiB 1336MHz	13	**GPU0:** 76.9% 14.5GiB 1347MHz	17	**GPU0:** 76.1% 14.5GiB 1342MHz	21	**GPU0:** 77.5% 14.5GiB 1342MHz
2	**GPU1:** 77.1% 14.5GiB 1341MHz	6	**GPU1:** 76.8% 14.5GiB 1344MHz	10	**GPU1:** 78.3% 14.5GiB 1346MHz	14	**GPU1:** 77.3% 14.5GiB 1346MHz	18	**GPU1:** 78.2% 14.5GiB 1346MHz	22	**GPU1:** 78.2% 14.5GiB 1346MHz
3	**GPU2:** 74.5% 14.5GiB 1334MHz	7	**GPU2:** 75.2% 14.5GiB 1339MHz	11	**GPU2:** 75.0% 14.5GiB 1336MHz	15	**GPU2:** 77.6% 14.5GiB 1346MHz	19	**GPU2:** 78.7% 14.5GiB 1347MHz	23	**GPU2:** 78.8% 14.5GiB 1347MHz
4	**GPU3:** 76.5% 14.5GiB 1339MHz	8	**GPU3:** 77.1% 14.5GiB 1345MHz	12	**GPU3:** 77.4% 14.5GiB 1344MHz	16	**GPU3:** 77.8% 14.5GiB 1346MHz	20	**GPU3:** 77.0% 14.5GiB 1342MHz	24	**GPU3:** 79.1% 14.5GiB 1349MHz
	Interconnect group: 11		Interconnect group: 11		Interconnect group: 11		Interconnect group: 11		Interconnect group: 11		Interconnect group: 11

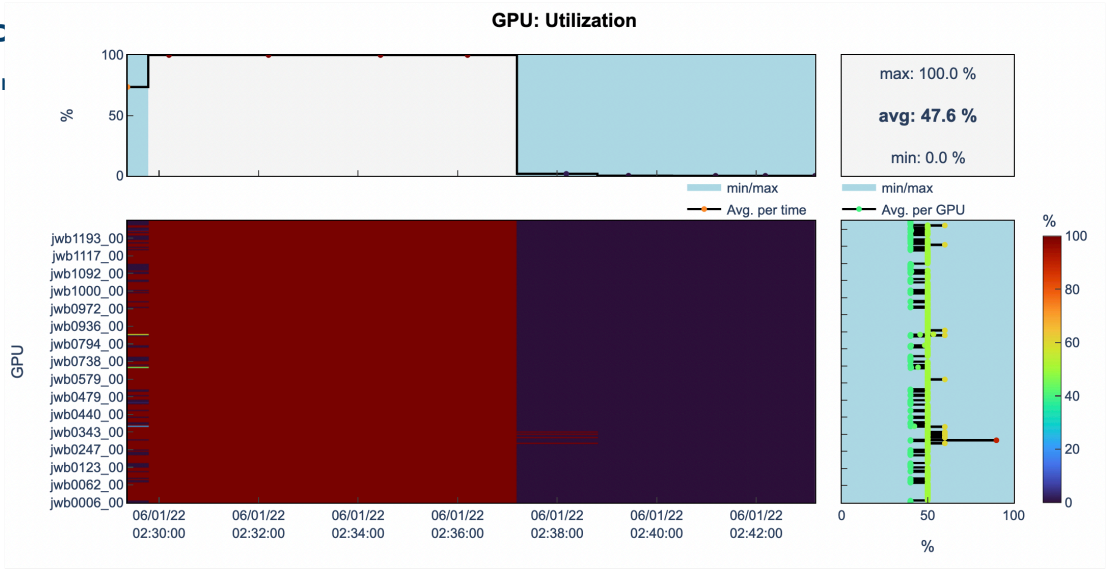
7	jwb0732	8	jwb0733	9	jwb0734	10	jwb0735	11	jwb0736	12	jwb0737
25	**GPU0:** 78.6% 14.5GiB 1349MHz	29	**GPU0:** 78.7% 14.5GiB 1350MHz	33	**GPU0:** 78.4% 14.5GiB 1346MHz	37	**GPU0:** 79.7% 14.5GiB 1351MHz	41	**GPU0:** 80.9% 14.5GiB 1354MHz	45	**GPU0:** 80.8% 14.5GiB 1354MHz
26	**GPU1:** 75.8% 14.5GiB 1342MHz	30	**GPU1:** 78.1% 14.5GiB 1344MHz	34	**GPU1:** 79.3% 14.5GiB 1347MHz	38	**GPU1:** 79.3% 14.5GiB 1347MHz	42	**GPU1:** 79.3% 14.5GiB 1347MHz	46	**GPU1:** 79.3% 14.5GiB 1347MHz
27	**GPU2:** 79.1% 14.5GiB 1349MHz	31	**GPU2:** 79.4% 14.5GiB 1351MHz	35	**GPU2:** 80.8% 14.5GiB 1352MHz	39	**GPU2:** 81.8% 14.5GiB 1356MHz	43	**GPU2:** 80.8% 14.5GiB 1352MHz	47	**GPU2:** 81.3% 14.5GiB 1354MHz
28	**GPU3:** 78.9% 14.5GiB 1349MHz	32	**GPU3:** 79.8% 14.5GiB 1349MHz	36	**GPU3:** 80.4% 14.5GiB 1352MHz	40	**GPU3:** 80.4% 14.5GiB 1351MHz	44	**GPU3:** 79.2% 14.5GiB 1349MHz	48	**GPU3:** 81.3% 14.5GiB 1354MHz
	Interconnect group: 11		Interconnect group: 11		Interconnect group: 11		Interconnect group: 11		Interconnect group: 11		Interconnect group: 12

13	jwb0738	14	jwb0739	15	jwb0740	16	jwb0741	17	jwb0742	18	jwb0743
49	**GPU0:** 81.2% 14.5GiB 1356MHz	53	**GPU0:** 81.6% 14.5GiB 1359MHz	57	**GPU0:** 81.6% 14.6GiB 1356MHz	61	**GPU0:** 82.1% 14.6GiB 1359MHz	65	**GPU0:** 84.3% 14.6GiB 1364MHz	69	**GPU0:** 82.1% 14.6GiB 1359MHz
50	**GPU1:** 79.2% 14.5GiB 1354MHz	54	**GPU1:** 80.3% 14.5GiB 1359MHz	58	**GPU1:** 81.1% 14.6GiB 1357MHz	62	**GPU1:** 83.0% 14.6GiB 1359MHz	66	**GPU1:** 83.7% 14.6GiB 1362MHz	70	**GPU1:** 83.6% 14.6GiB 1362MHz
51	**GPU2:** 80.9% 14.5GiB 1361MHz	55	**GPU2:** 81.3% 14.5GiB 1361MHz	59	**GPU2:** 81.8% 14.6GiB 1357MHz	63	**GPU2:** 82.3% 14.6GiB 1362MHz	67	**GPU2:** 83.9% 14.6GiB 1362MHz	71	**GPU2:** 81.4% 14.6GiB 1362MHz
52	**GPU3:** 82.2% 14.5GiB 1361MHz	56	**GPU3:** 82.4% 14.5GiB 1362MHz	60	**GPU3:** 82.4% 14.6GiB 1361MHz	64	**GPU3:** 83.6% 14.6GiB 1364MHz	68	**GPU3:** 84.4% 14.6GiB 1364MHz	72	**GPU3:** 84.4% 14.6GiB 1369MHz
	Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12

19	jwb0744	20	jwb0745	21	jwb0746	22	jwb0747	23	jwb0748	24	jwb0757
73	**GPU0:** 83.9% 14.6GiB 1362MHz	77	**GPU0:** 84.3% 14.6GiB 1366MHz	81	**GPU0:** 84.3% 14.6GiB 1367MHz	85	**GPU0:** 86.4% 14.6GiB 1371MHz	89	**GPU0:** 87.5% 14.6GiB 1374MHz	93	**GPU0:** 87.4% 14.6GiB 1374MHz
74	**GPU1:** 83.9% 14.6GiB 1364MHz	78	**GPU1:** 84.9% 14.6GiB 1366MHz	82	**GPU1:** 84.3% 14.6GiB 1366MHz	86	**GPU1:** 86.0% 14.6GiB 1369MHz	90	**GPU1:** 85.3% 14.6GiB 1369MHz	94	**GPU1:** 85.7% 14.6GiB 1371MHz
75	**GPU2:** 83.4% 14.6GiB 1362MHz	79	**GPU2:** 83.6% 14.6GiB 1367MHz	83	**GPU2:** 85.8% 14.6GiB 1372MHz	87	**GPU2:** 85.5% 14.6GiB 1367MHz	91	**GPU2:** 86.8% 14.6GiB 1371MHz	95	**GPU2:** 87.2% 14.6GiB 1376MHz
76	**GPU3:** 84.1% 14.6GiB 1366MHz	80	**GPU3:** 84.6% 14.6GiB 1367MHz	84	**GPU3:** 81.6% 14.6GiB 1361MHz	88	**GPU3:** 84.2% 14.6GiB 1366MHz	92	**GPU3:** 84.4% 14.6GiB 1369MHz	96	**GPU3:** 84.0% 14.6GiB 1372MHz
	Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12

25	jwb0758	26	jwb0759	27	jwb0760	28	jwb0761	29	jwb0762	30	jwb0763
97	**GPU0:** 87.8% 14.6GiB 1376MHz	101	**GPU0:** 89.6% 14.6GiB 1380MHz	105	**GPU0:** 89.4% 14.6GiB 1382MHz	109	**GPU0:** 89.5% 14.6GiB 1385MHz	113	**GPU0:** 88.8% 14.6GiB 1382MHz	117	**GPU0:** 88.6% 14.6GiB 1382MHz
98	**GPU1:** 86.0% 14.6GiB 1372MHz	102	**GPU1:** 85.6% 14.6GiB 1372MHz	106	**GPU1:** 85.9% 14.6GiB 1374MHz	110	**GPU1:** 86.6% 14.6GiB 1374MHz	114	**GPU1:** 86.9% 14.6GiB 1376MHz	118	**GPU1:** 89.5% 14.6GiB 1380MHz
99	**GPU2:** 88.7% 14.6GiB 1383MHz	103	**GPU2:** 89.3% 14.6GiB 1384MHz	107	**GPU2:** 88.7% 14.6GiB 1382MHz	111	**GPU2:** 90.1% 14.6GiB 1385MHz	115	**GPU2:** 87.6% 14.6GiB 1380MHz	119	**GPU2:** 88.3% 14.6GiB 1382MHz
100	**GPU3:** 83.8% 14.6GiB 1375MHz	104	**GPU3:** 86.6% 14.6GiB 1376MHz	108	**GPU3:** 87.7% 14.6GiB 1376MHz	112	**GPU3:** 86.8% 14.6GiB 1376MHz	116	**GPU3:** 90.0% 14.6GiB 1382MHz	120	**GPU3:** 90.6% 14.6GiB 1387MHz
	Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12		Interconnect group: 12

31	jwb0764	32	jwb0765
121	**GPU0:** 89.4% 14.6GiB 1384MHz	125	**GPU0:** 87.4% 14.6GiB 1380MHz
122	**GPU1:** 89.8% 14.6GiB 1382MHz	126	**GPU1:** 90.2% 14.6GiB 1385MHz
123	**GPU2:** 89.3% 14.6GiB 1383MHz	127	**GPU2:** 89.6% 14.6GiB 1385MHz
124	**GPU3:** 92.1% 14.6GiB 1389MHz	128	**GPU3:** 90.8% 15.6GiB 1387MHz
	Interconnect group: 12		Interconnect group: 12



Summary and Conclusions

Summary and Conclusions

- JUWELS Booster: European flagship system based on A100 GPUs and HDR200 InfiniBand network
- Highly scalable system design with > 70 PFLOP$/s_{FP64}$ compute performance and 749 Tbit$/$s acc. injection bandwidth
- Hierarchical design from core to system

Summary and Conclusions

- JUWELS Booster: European flagship system based on A100 GPUs and HDR200 InfiniBand network
- Highly scalable system design with > 70 PFLOP$/s_{FP64}$ compute performance and 749 Tbit$/$s acc. injection bandwidth
- Hierarchical design from core to system

Thank you for your attention!

a.herten@fz-juelich.de

Appendix

Appendix

- Network Performance
- STREAM Benchmark
- References

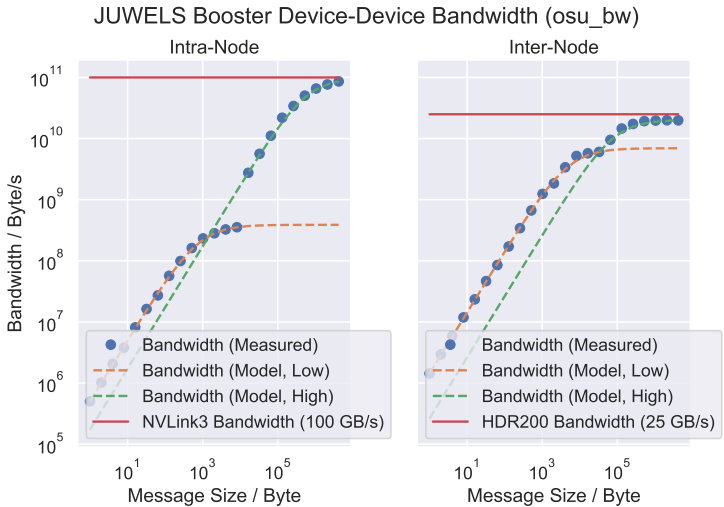
Appendix

Network Performance

Network Performance

OSU Micro-Benchmarks: Bandwidth

- OSU Microbenchmarks: device-device bandwidth (`osu_bw D D`)
- Good results, expected limiters
- Intra-node: NVLink3 bandwidth
- Inter-node: HDR200 bandwidth
- Model fits show 2 regimes (---/ ---)



Appendix

STREAM Benchmark

STREAM

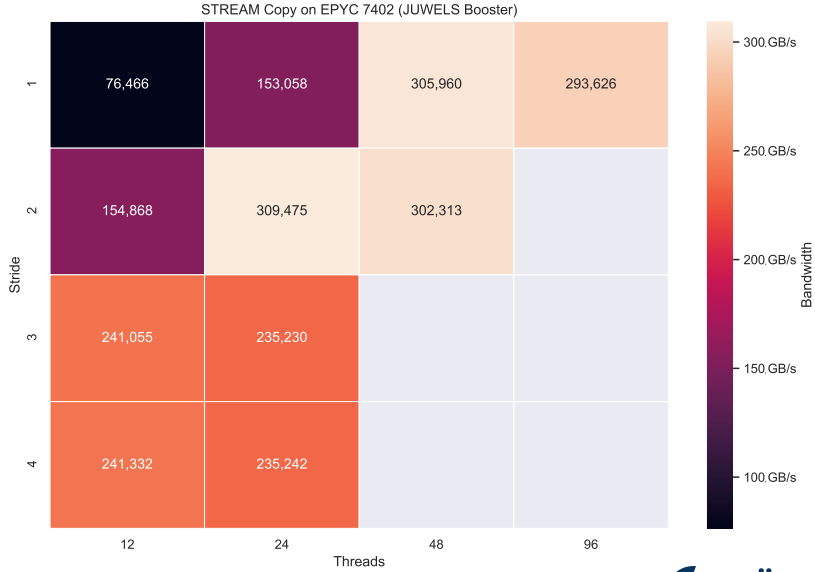
Config

AOCC 3.0.0 on EPYC 7402

```
$ clang -O3 -mcmodel=large -fopenmp -ffp-contract=fast -fnt-store -DSTREAM_ARRAY_SIZE=2500000000
↪  -DNTIMES=200 -DSTREAM_TYPE=double -o stream.exe stream.c
$ srun --cpu-bind=none \
    env \
        OMP_NUM_THREADS=24 \
        OMP_PLACES='{0}:24:2' \
        OMP_PLACES="cores" \
        OMP_PROC_BIND="true" \
        OMP_MAX_ACTIVE_LEVELS=1 \
        OMP_STACKSIZE=256M \
        OMP_SCHEDULE=static \
        OMP_DYNAMIC=false \
        OMP_THREAD_LIMIT=256 \
        OMP_DISPLAY_ENV=true \
    ./stream.exe
```

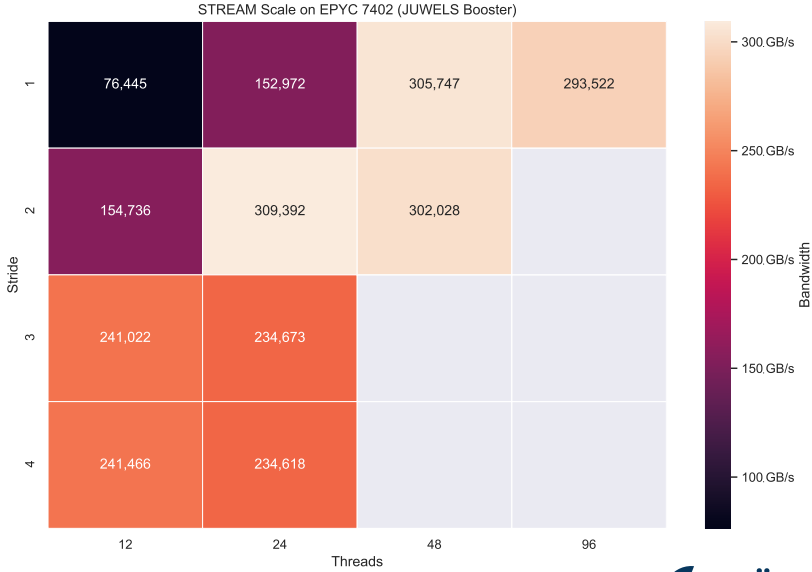
STREAM

Copy



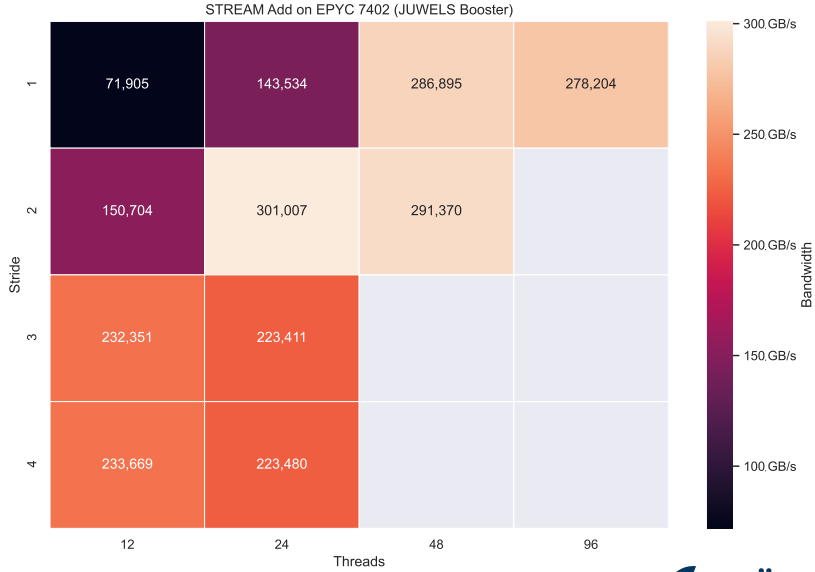
STREAM

Scale



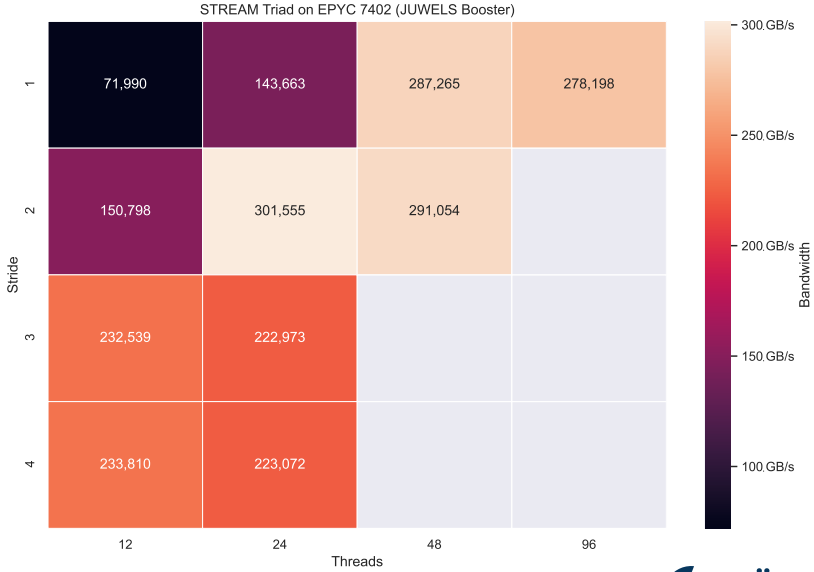
STREAM

Add



STREAM

Triad



Appendix

References

References I

References: Images, Graphics I

[1] Forschungszentrum Jülich GmbH (Ralf-Uwe Limbach). *JUWELS Cluster*.

[2] Forschungszentrum Jülich GmbH (Ralf-Uwe Limbach). *JUWELS Booster*.