JÜLICH
FORSCHUNGSZENTRUM

JURECA

Hardware and Best Practices

June 7, 2016 | Krause, Kondylis | HPS group @ JSC

JÜLICH
FORSCHUNGSZENTRUM

Contents

Introduction to JURECA

- **Jü**lich **R**esearch on **E**xascale **C**luster **A**rchitectures
- Project partners: T-Platforms, ParTec
- FZJ next-generation general purpose production system
  - NIC, VSR and commercial projects
  - Replaces the decomissioned JUROPA system
- Intended for mixed capacity and capability workloads
  - Designed with big-data science needs in mind
- **Cluster** architecture
  - Commodity hardware
  - Largely based on a open-source software stack

Contents

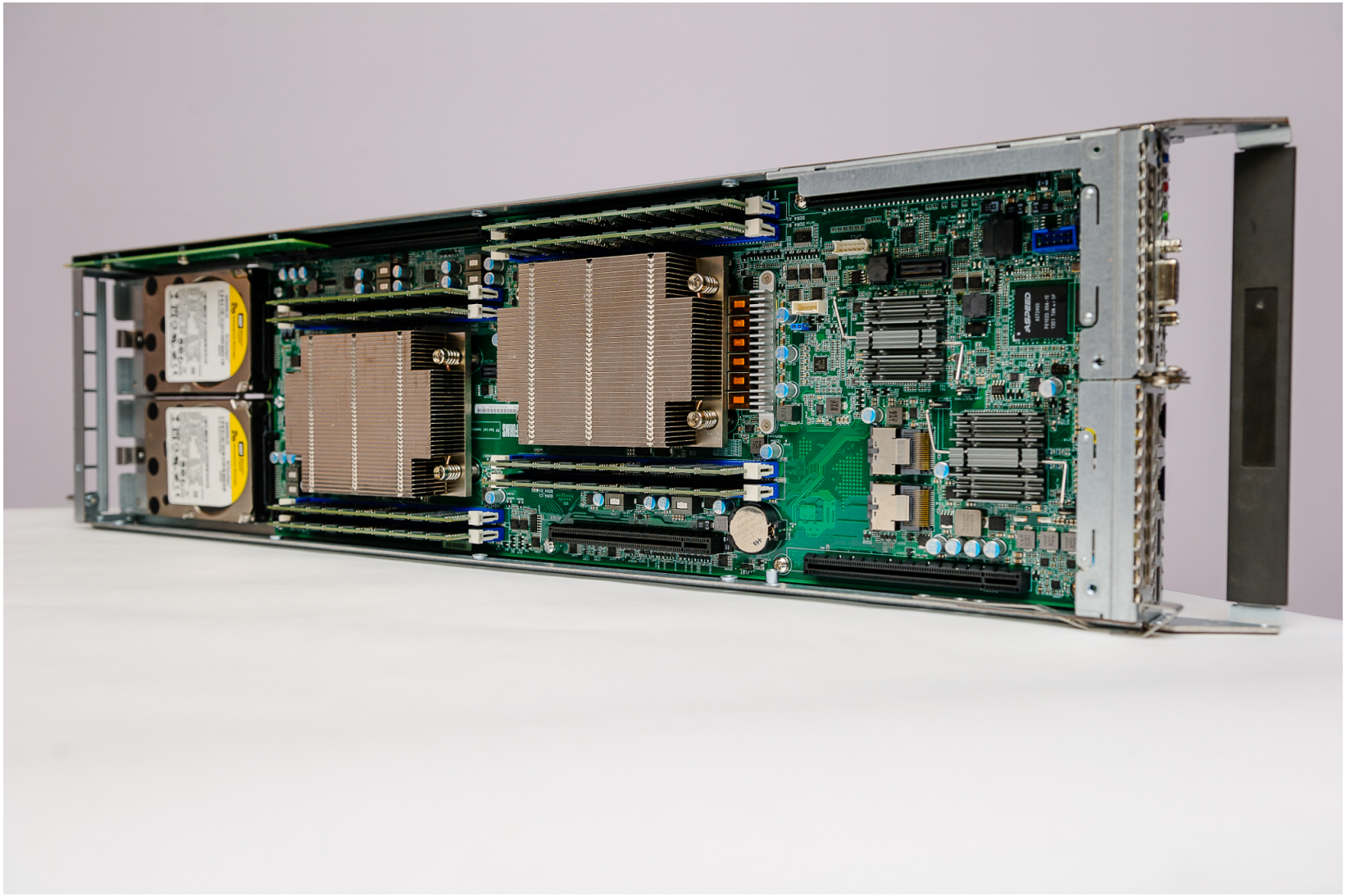
JURECA

JURECA: RACKS

JURECA: V-CLASS CHASSIS (FRONT)

JURECA: V-CLASS CHASSIS (BACK)

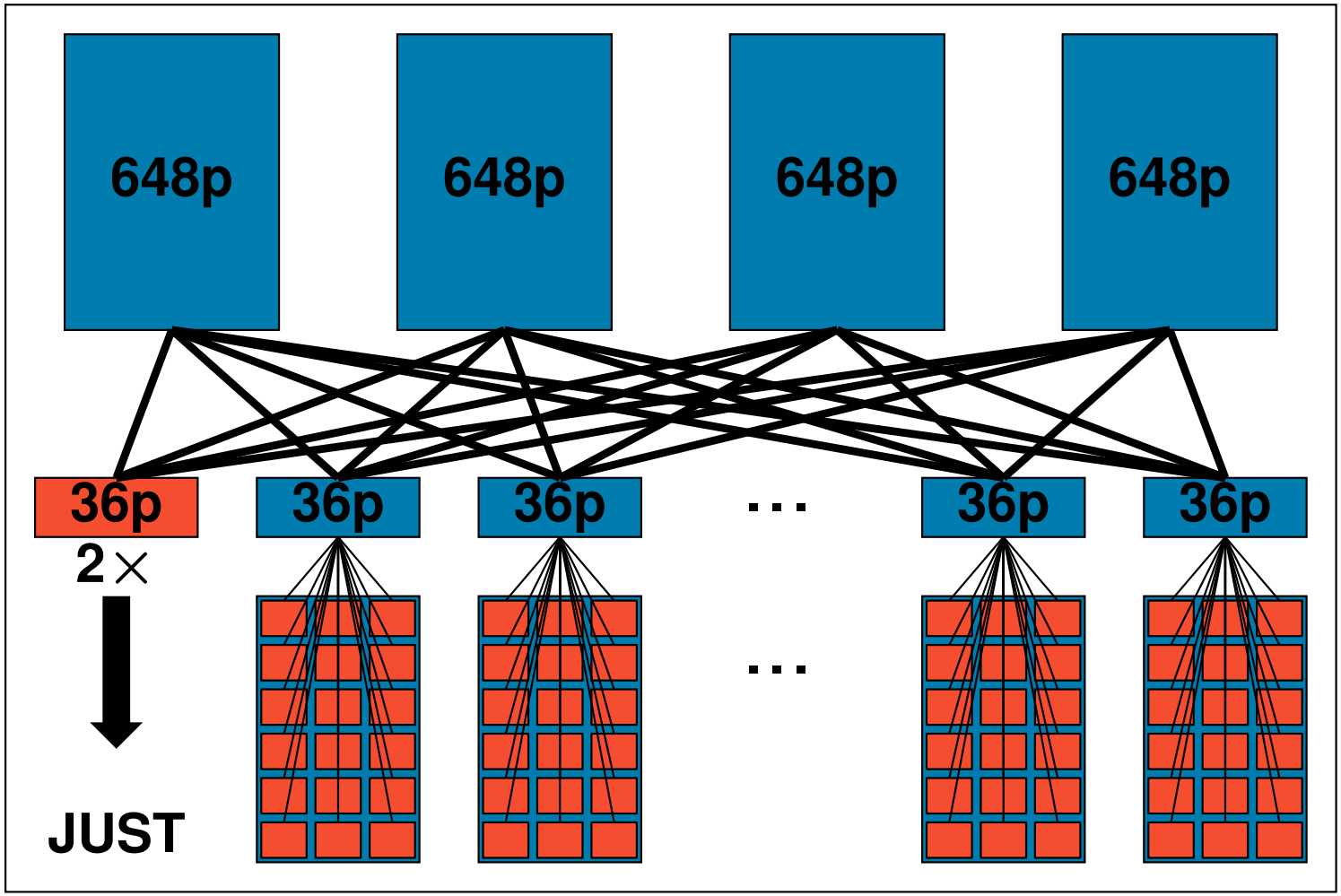
JURECA: V210S NODE



JURECA Node Overview

- Dual-socket Intel Xeon **E5-2680 v3** Haswell nodes
  - 24 cores @ 2.5 GHz
- NVIDIA K40 and K80 GPUs
- 128/256/512 GiB memory per node (DDR4 @ 2133 MHz)
- 1884 compute nodes ⇨ 45,216 cores
  - **1800** TFps + 430 TFps peak performance
- InfiniBand **EDR** (100 Gbps per link and direction)
  - Full fat tree topology
- 100 GiBps I/O bandwidth to central GPFS storage cluster

JURECA: FAT-TREE INFINIBAND TOPOLOGY



JURECA NODE TYPES

- **Login nodes**
  - 256 GiB memory
  - Intended for interactive work: development, compilation, interactive pre- and post-processing
  - CPU time limits (2 hours)
- **Standard/slim nodes**
  - 128 GiB memory
  - Default for batch jobs (**`batch`** partition)
  - Smallest allocation is one node, charge based on wall-clock time
  - No direct login ⇨ Interactive sessions with **`salloc`** and **`srun --forward-x --pty`**

JURECA NODE TYPES (2)

- **Fat (type 1): 256 GiB memory**
  - `--gres=mem256`
  - Included in `batch`
- **Fat (type 2): 512 GiB memory**
  - `-p mem512 --gres=mem512`
  - Currently in a separate `mem512` partition (lower performance)
- **Fat (type 3): 1 TiB memory**
  - `-p mem1024 --gres=mem1024`
  - Intended for memory-intense, lowly scalable pre- and post-processing tasks

JURECA NODE TYPES (3)

- **Visualization nodes**
  - $\geq$512 GiB memory (2 nodes with 1 TiB), 2$\times$ NVIDIA K40
  - `-p vis --gres=gpu:[1-2]`
  - `--gres=mem1024` for large memory nodes
  - Client-server visualization requires `ssh` tunneling
- **GPU nodes**
  - 128 GiB memory, 2$\times$ NVIDIA K80 (4 visible GPUs per host)
  - `-p gpus --gres=gpu:[1-4]`

JURECA NODE QUANTITIES

Node type	#	Characteristics
Standard/Slim	1605	24 cores, 128 GiB
Fat (type 1)	128	24 cores, 256 GiB
Fat (type 2)	64	24 cores, 512 GiB
Accelerated	75	24 cores, 128 GiB, 2× K80
Login	12	24 cores, 256 GiB
Visualization (type 1)	10	24 cores, 512 GiB, 2× K40
Visualization (type 2)	2	24 cores, 1 TiB, 2× K40

Contents

JURECA: Software Specifications

- **Operating system:** CentOS 7.X
- **Batch system** based on **Slurm/Parastation**
  - Workload management and UI ⇨ Slurm
  - Resource management ⇨ Parastation (**psid + psslurm**)
- **Programming environment:**
  - GNU Compilers, Intel Professional Fortran, C/C++ Compilers, OpenMP (Intel, GNU)
  - CUDA
  - Parastation MPI (based on **MPICH3**), Intel MPI, MVAPICH2-GDR
  - Optimized mathematical libraries (Intel Math Kernel Library, etc.) and applications (`/usr/local`)

JURECA: ACCESSING THE SYSTEM

```
$ ssh <user>@jureca.fz-juelich.de
```

```
$ ssh <user>@jureca[01-12].fz-juelich.de
```

- Access with SSH keys
  - Recommendation: 2048 bit RSA (**`ssh-keygen -t rsa -b 2048`**)
  - Protection of private key with non-trivial pass phrase is mandatory!
- CPU time limits apply
  - Soft limit: 2 hours

JURECA: Accessing software (hierarchical modules)

1. List available toolchains

```
$ module avail
```

2. Load the desired compiler and MPI

```
$ module load <Compiler> <MPI>
```

3. List availables packages based on current list of modules

```
$ module avail
```

4. Load additional applications/libraries

```
$ module load <module name>
```

Search for an application/library

```
$ module spider <name>
```

Contents

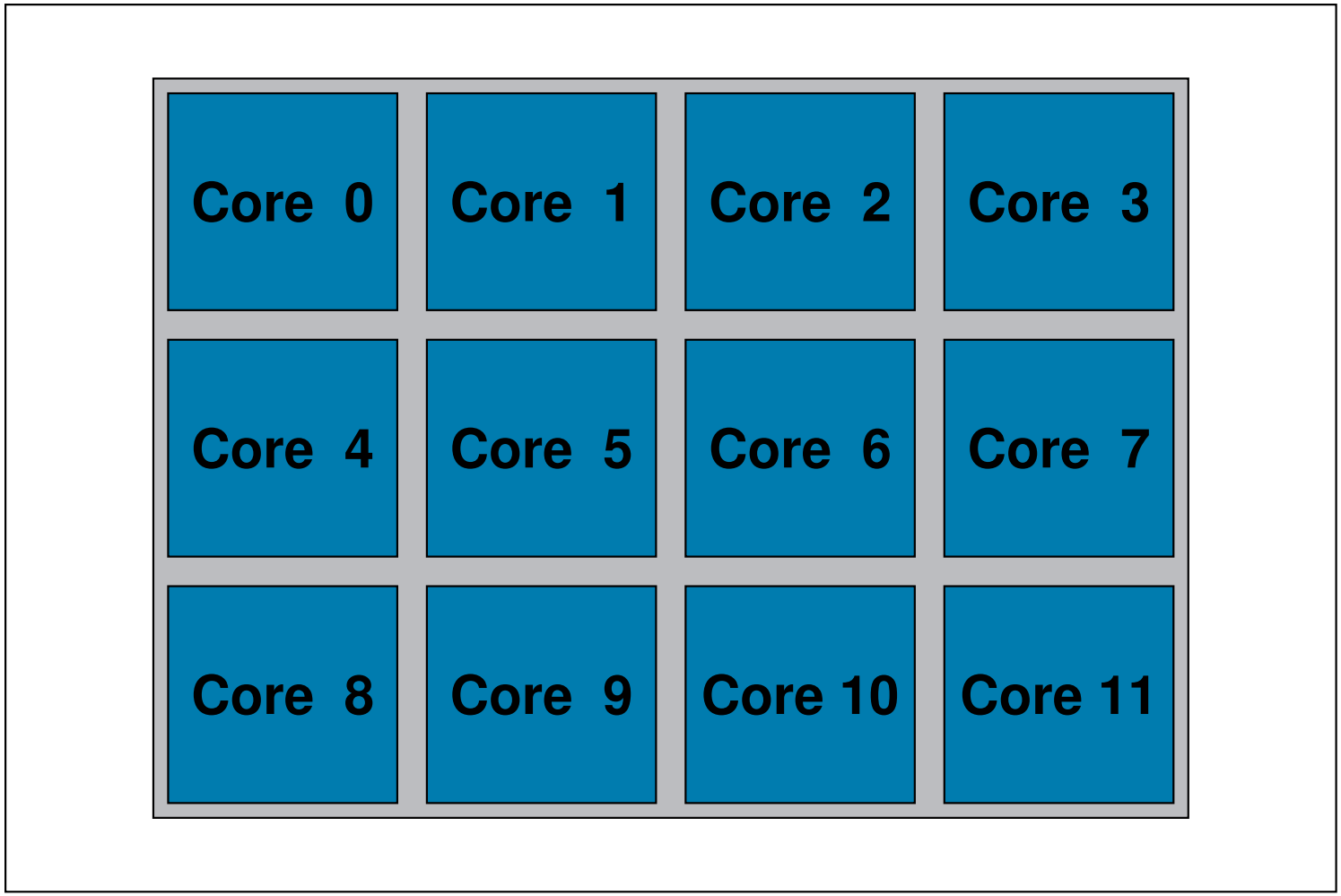
JURECA: Best Practices

- CPUs
- GPUs
- FS Usage and IO

JURECA: CPUs

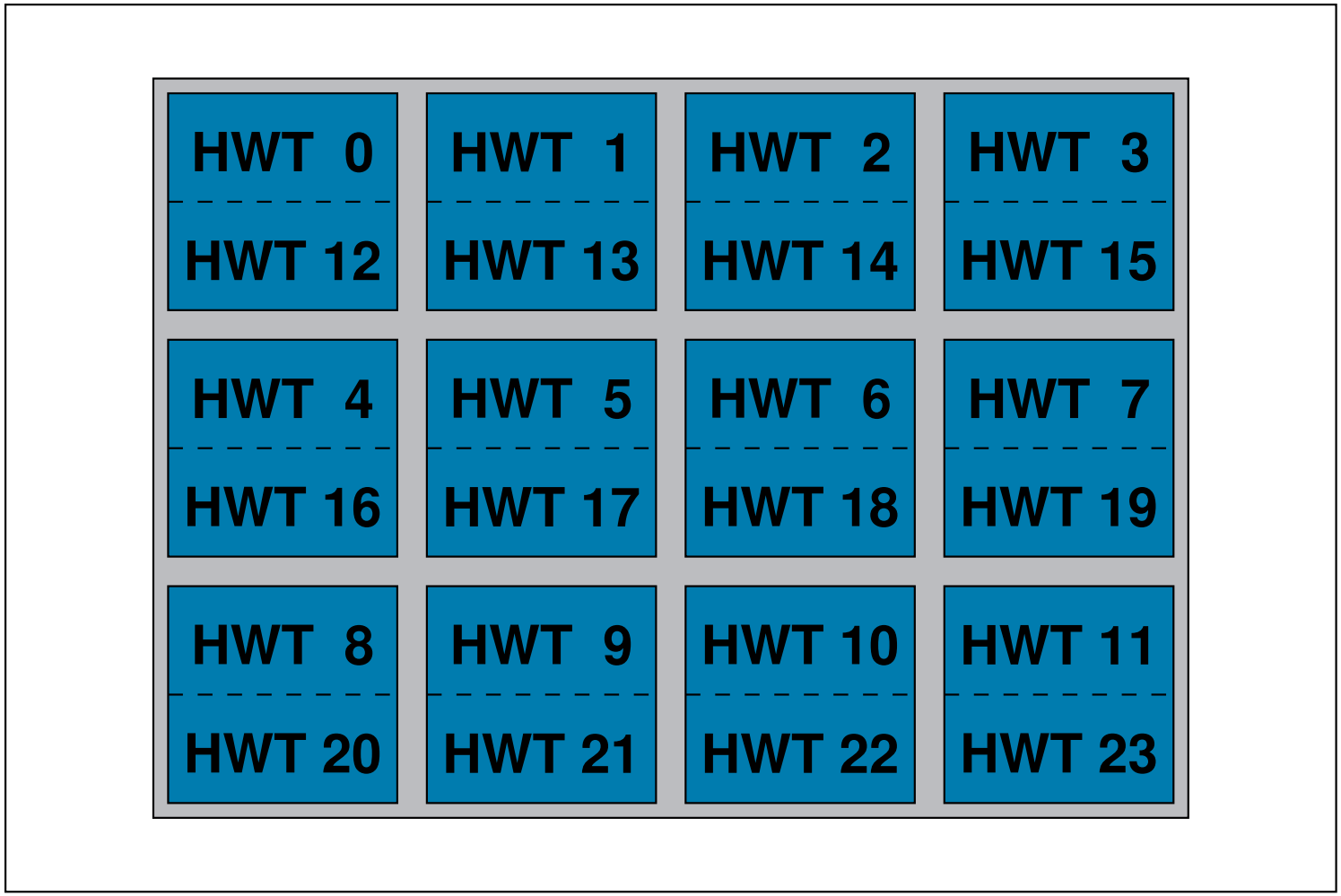
- 2 x **Intel Xeon E5-2680 v3 Haswell CPUs** / node
  - 12 cores, 2.5 GHz (x 2 = 24 Cores / node)
  - Intel Hyperthreading Technology (Simultaneous Multithreading)
  - AVX 2.0 ISA extension

JURECA: CPUs Multicore




```
$> lstopo -of txt
```

JURECA: CPUs Hyper-Threading Technology




```
$> lstopo -of txt
```

JURECA: CPUs Compilers & Modules

- Intra-Node
  - OpenMP
  - PThreads
  - MPI
- Inter-Node
  - MPI
- Available Compilers
  - GCC
  - Intel (First choice for Intel platforms)
  - PGI (For OpenACC & CUDA)

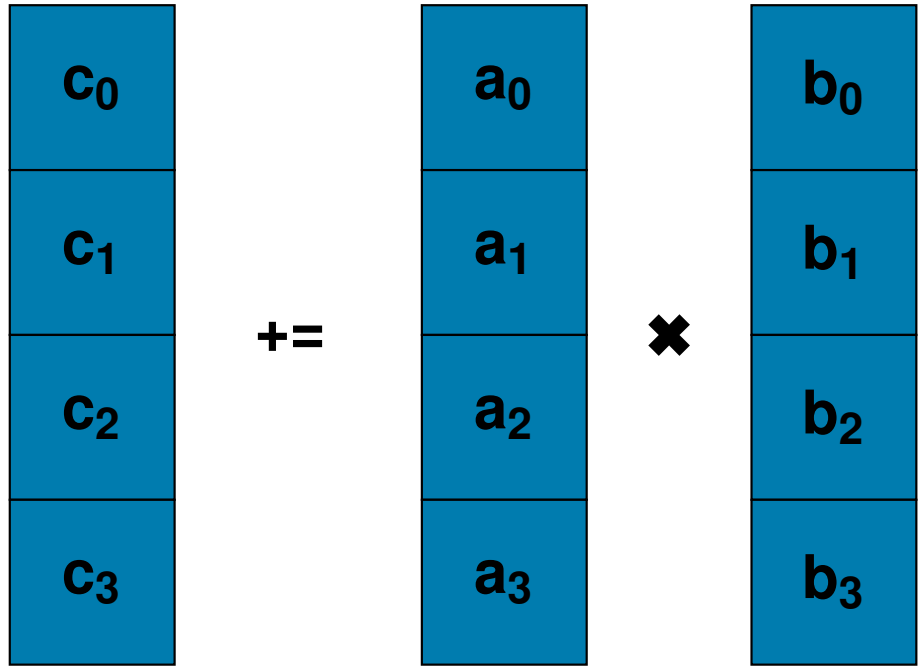
JURECA: CPUs Compilers & Modules

- MPIs
  - Parastation MPI (default)
    - Supports MPI-3 standard
    - based on MPICH
    - First choice on the system
    - Good performance for both, small and large, messages
  - Intel MPI
    - Supports MPI-3 standard
    - based on MPICH
    - ofa fabric usually is preferable over default dapl version (see JURECA FAQ entry)
  - MVAPICH2
    - CUDA-aware MPI: Allows to send device (GPU) memory directly and pipelines data transfer and sends to reach better performance)

JURECA: CPUs Workshops

- **9-12 Aug 2016** Introduction to parallel programming with MPI and OpenMP
- **28-30 Nov 2016** Advanced Parallel Programming with MPI and OpenMP

JURECA: CPUs AVX 2.0 ISA EXTENSION

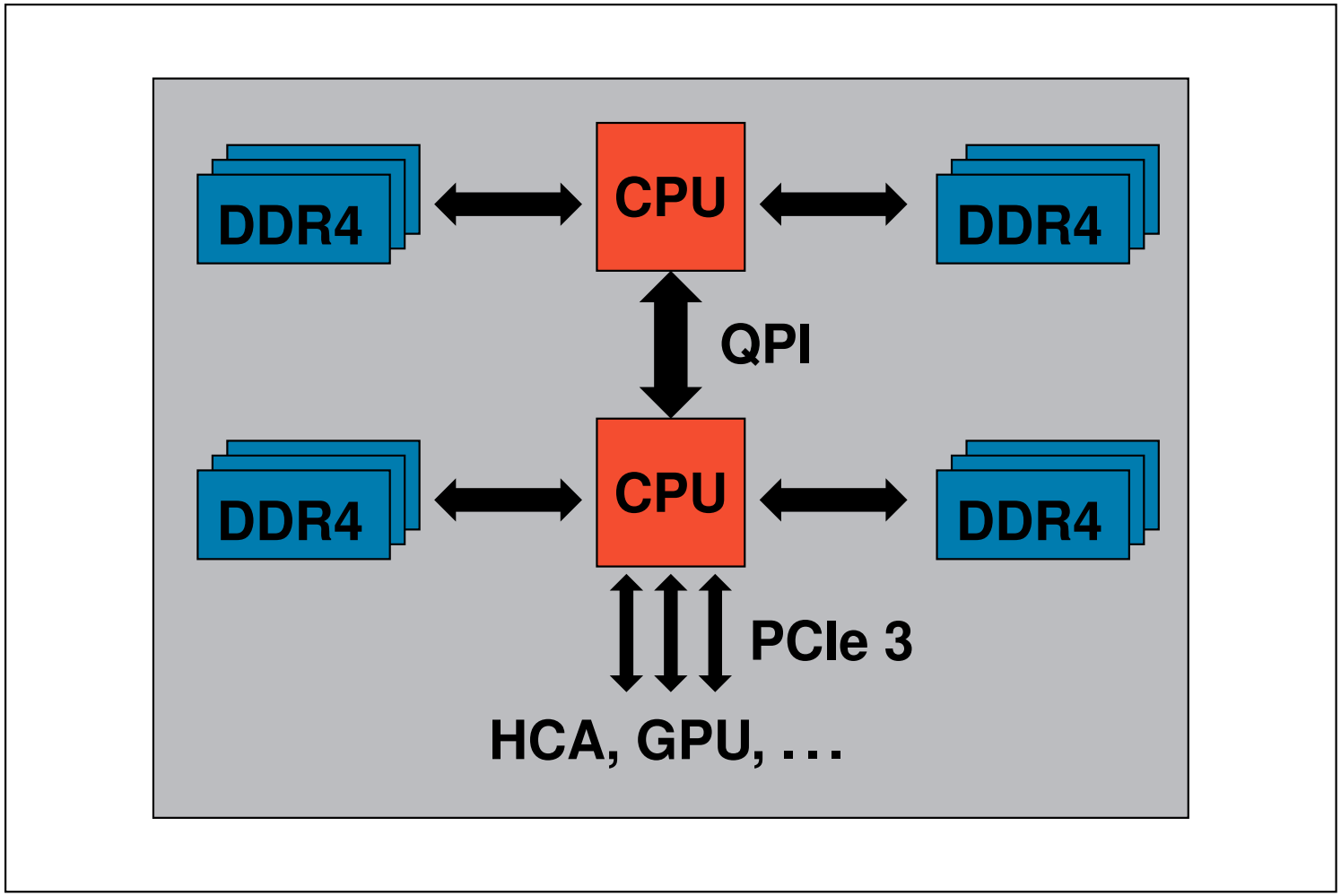


- **AVX 2.0** ISA extension ⇨ Two 256-bit wide multiply-adds per cycle **!**

JURECA: CPUs FLOPs Counter

$$
\begin{aligned}
&\#\ \textit{Cores}\ (24) \\
&*\ \textit{Frequency}\ (2,5\ \textit{GHz}) \\
&*\ 2\ \textit{fmadd} \\
&*\ 2\ (\textit{vector instructions / cycle}) \\
&*\ \textit{Vector length}\ (256\ \textit{bit} -> \ 4\ \textit{doubles}) \\
&=\ 960\ \textit{GFlops}
\end{aligned}
$$

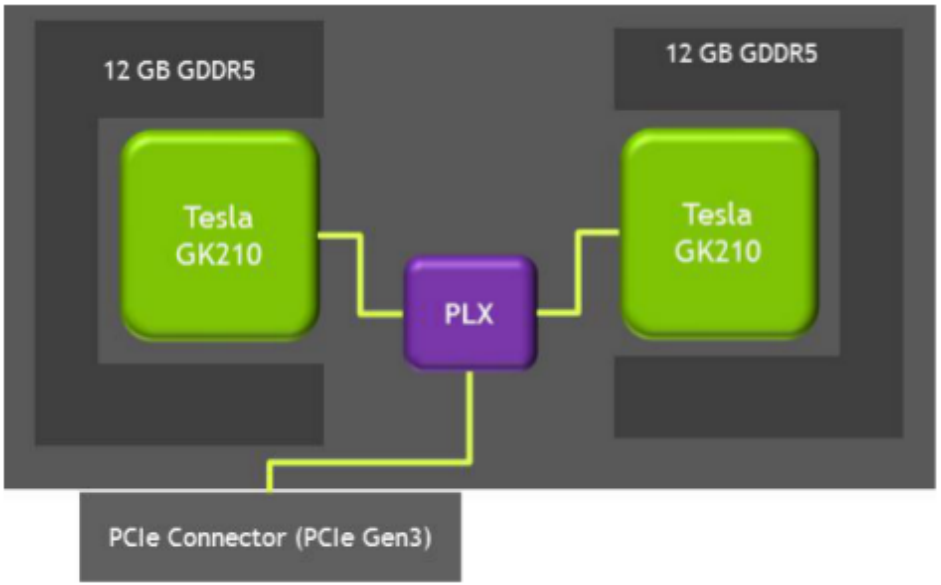
JURECA: CPUs NUMA ARCHITECTURE




```
numactl -hardware
```

JURECA: GPUs

- 2 x **NVIDIA Tesla K80** / node
  - dual GPU design 2 x **GK210 GPU**



  - 2 x 2496 Cores
  - 2 x **12GBs** GDDR5 Memory
  - 2 x 240 GB/s Memory Bandwidth

JURECA: GPUs

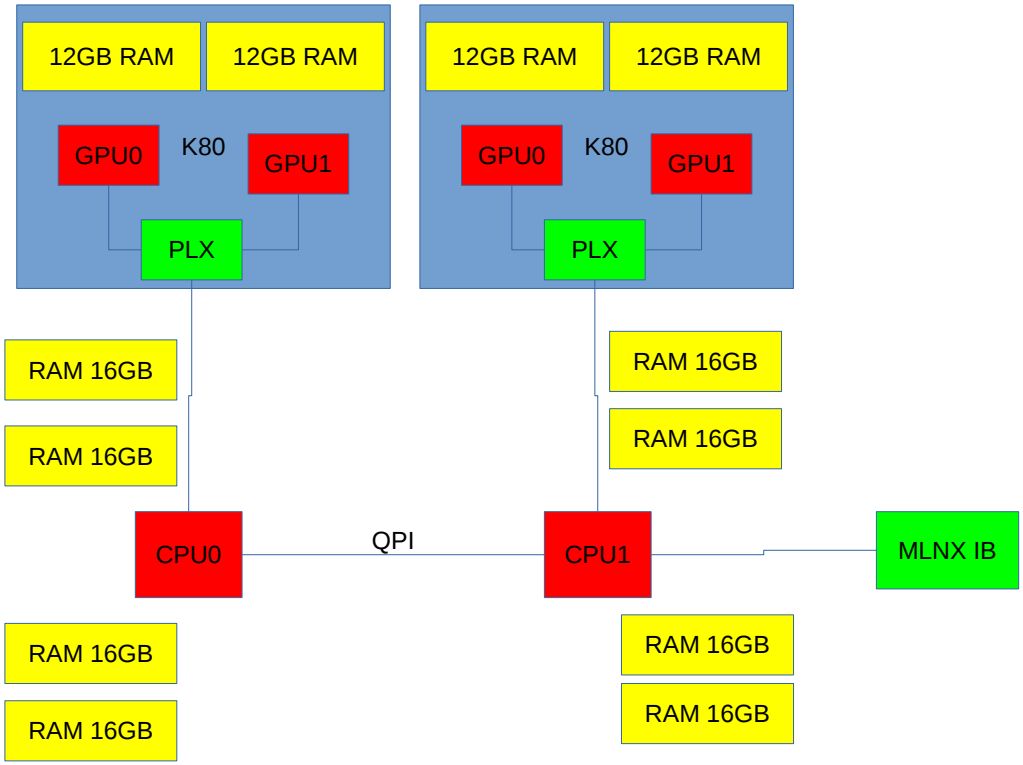


Figure 1: `$> nvidia-smi topo -m`

JURECA: GPUs Modules

- To use CUDA
  - `$> module load GCC/4.9.3-2.25`
  - `$> module load Intel/2015.3.187-GCC-4.9.3-2.25`
  - `$> module load PGI/16.3-GCC-4.9.3-2.25`
- To use OpenACC
  - `$> module load PGI/16.3-GCC-4.9.3-2.25`

JURECA: GPUs Workshops

- GPU Programming with OpenACC: A LBM Case Study, Jiri Kraus, NVIDIA
- **24-25 Oct 2016** Introduction to GPU programming using OpenACC

JURECA: FILESYSTEMS

- All user filesystems mounted from the central GPFS fileserver **Jü**lich **St**orage Cluster (JUST)
  - **Exception:** Node local `/tmp` filesystem (**ext4**), $\mathcal{O}$(10 GiB)

- `$HOME`
- `$WORK`
- `$ARCH`

JURECA: FILESYSTEMS ($HOME)

- **Purposes**
  - Storage of regularly used files and applications
  - Storage of smaller files used for current computation
- Daily backup
- **Quota:** Max. 10 TiB disk space and max. 3 mio. inodes per group

```
$ q_dataquota [-l]
```

- Careful with **`chmod -R`** !
  - **Safer alternative:** Access control lists (ACL)

JURECA: FILESYSTEMS ($WORK)

- **Purpose**
  - Storage of large files used or generated by the current computation
- Scratch filesystem with highest performance
- No backup
- Files will be deleted 90 days after last usage !
  - **atime** is not updated for performance reasons
- **Quota:** Max. 30 TiB disk space and max. 4 mio. inodes per group

```
$ q_dataquota [-l]
```

- Copy important files to **$HOME** or **$ARCH**

JURECA: FILESYSTEMS (`$ARCH`)

- **Purpose**
  - Storage of large, not recently used, files
- Not available on compute nodes **!**
- Daily backup
- Files migrated to tapes
- **Quota:** No space quota and max. 2 mio. inodes per group
- **Usage recommendations**
  - **`tar`**/**`zip`** many small files
  - Do not touch/move files

JURECA: FILESYSTEMS & IO USAGE

- Parallel I/O, Wolfgang Frings, JSC
- **Unknown Date**, Parallel I/O Workshop

CONTACT

sc@fz-juelich.de

QUESTIONS ?

Thank You !!