JURECA CLUSTER+BOOSTER
JURECA IN A NUTSHELL

JURECA Cluster
• 1882 compute nodes based on dual-Socket Intel Xeon Haswell
• Mellanox InfiniBand EDR100 Gb/s network
• Full fat-tree topology
• 2.2 PF/s

JURECA Booster
• 1640 compute nodes based on Intel Xeon Phi 7250-F
• Intel Omni-Path Architecture 100 Gb/s network
• Full fat-tree topology
• 5 PF/s
JURECA CLUSTER
JURECA CLUSTER
JURECA CLUSTER

- Jülich Research on Exascale Cluster Architectures
  - Cluster project partners: T-Platforms, ParTec
- FZJ next-generation general purpose production system
  - NIC, VSR and commercial projects
- Replaces the decommissioned 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
JURECA: V210S NODE
JURECA: V-CLASS CHASSIS (FRONT)
JURECA: V-CLASS CHASSIS (BACK)
JURECA: RACKS
JURECA CLUSTER HARDWARE 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 TF/s (CPU) + 430 TF/s (GPU) peak performance
• InfiniBand EDR (100 Gbps per link and direction)
  • Full fat tree topology
• 100 GB/s I/O bandwidth to central GPFS storage cluster
JURECA SOFTWARE OVERVIEW

- 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 CLUSTER NODE TYPES (1/3)

• 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, charged based on wall-clock time
  • No direct login ⇒ Interactive sessions with salloc and srun --forward-x --pty
JURECA CLUSTER NODE TYPES (2/3)

- Fat (type 1): 256 GiB memory
  - `-p mem256 (--gres=mem256)`
  - Overlaps with batch partition
- Fat (type 2): 512 GiB memory
  - `-p mem512 (--gres=mem512)`
  - In a separate `mem512` partition due to lower node performance
- Fat (type 3): 1 TiB memory
  - `-p mem1024 (--gres=mem1024)`
  - Intended for memory-intense, lowly scalable pre- and post-processing tasks
• Visualization nodes
  • \( \geq 512 \text{ GiB memory (2 nodes with 1 TiB), 2} \times \text{NVIDIA K40} \)
  • `-p vis --gres=gpu:[1-2]`
  • `--gres=mem1024` for large memory nodes
• Client-server visualization requires ssh tunneling
• Six nodes directly accessible as visualization login nodes
• GPU nodes
  • 128 GiB memory, \( 2 \times \text{NVIDIA K80 (4 visible GPUs per host)} \)
  • `-p gpus --gres=gpu:[1-4]`
# JURECA Cluster Node Quantities

<table>
<thead>
<tr>
<th>Node type</th>
<th>#</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard/Slim</td>
<td>1605</td>
<td>24 cores, 128 GiB</td>
</tr>
<tr>
<td>Fat (type 1)</td>
<td>128</td>
<td>24 cores, 256 GiB</td>
</tr>
<tr>
<td>Fat (type 2)</td>
<td>64</td>
<td>24 cores, 512 GiB</td>
</tr>
<tr>
<td>Accelerated</td>
<td>75</td>
<td>24 cores, 128 GiB, 2× K80</td>
</tr>
<tr>
<td>Login</td>
<td>12</td>
<td>24 cores, 256 GiB</td>
</tr>
<tr>
<td>Visualization Login</td>
<td>6</td>
<td>24 cores, 512 GiB, 2× K40</td>
</tr>
<tr>
<td>Visualization (type 1)</td>
<td>4</td>
<td>24 cores, 512 GiB, 2× K40</td>
</tr>
<tr>
<td>Visualization (type 2)</td>
<td>2</td>
<td>24 cores, 1 TiB, 2× K40</td>
</tr>
</tbody>
</table>
JURECA: ACCESSING THE SYSTEM

- 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

```bash
$ ssh <user>@jureca.fz-juelich.de
$ ssh <user>@jureca[01-12].fz-juelich.de
$ ssh <user>@jurecavis.fz-juelich.de
$ ssh <user>@jurecavis[01-06].fz-juelich.de
```
1. List available toolchains
   
   \$ \texttt{module avail}

2. Load compiler and MPI
   
   \$ \texttt{module load <Compiler> <MPI>}

3. List available packages
   
   \$ \texttt{module avail}

4. Load additional applications and libraries
   
   \$ \texttt{module load <module name>}

5. Search for an application/library
   
   \$ \texttt{module spider <name>
JURECA CLUSTER: SKETCH
JURECA CLUSTER: FAT-TREE IB TOPOLOGY
JURECA CLUSTER: NUMA ARCHITECTURE
JURECA CLUSTER: MULTICORE

Core 0  Core 1  Core 2  Core 3
Core 4  Core 5  Core 6  Core 7
Core 8  Core 9  Core 10 Core 11
JURECA CLUSTER: HYPER-THREADING
JURECA CLUSTER: AVX 2.0 ISA EXTENSION

- AVX 2.0 ISA extension ⇒ Two 256-bit wide multiply-adds per cycle!
JURECA BOOSTER
JURECA BOOSTER HARDWARE

• Extension of JURECA
  • Deployed in 2017
  • Augments Cluster module with a highly-scalable component
• Designed for capability workloads
  • System integrators: Intel with Dell
• Compute time allocation
  • Primarily for scientists from Jülich and Aachen
  • Available for admissible researchers at German universities for a two-year interim period via NIC
• First implementation of a Modular Supercomputer at Petascale
JURECA BOOSTER 101

- Same login nodes and file systems as JURECA Cluster module
- Same system software environment as JURECA Cluster module
  - CentOS 7.X
  - GNU, Intel Compiler
  - ParaStation MPI, Intel MPI
- One workload management system: Slurm/ParaStation
  - Separate partitions for Booster nodes
  - Similar to handling of e.g., GPU-equipped nodes
JURECA BOOSTER NODE ARCHITECTURE

Dell PowerEdge C6320P specifications

- Dell PowerEdge C6320P solution
- Intel Xeon Phi “Knights Landing” 7250-F
  - 68 cores @ 1.4 GHz
  - 96 GiB main memory, 16 GiB MCDRAM
- On-package Intel Omni-Path Architecture network interface
JURECA BOOSTER CHASSIS ARCHITECTURE

- 1 Chassis houses 4 KNL blades
- 1 Rack houses 18 chassis
- 23 Racks equipped with KNL chassis
- 1640 Booster KNL systems
- 157 TiB main memory
  - + 26 TiB MCDRAM
- Peak performance
  - 5 PF/s
JURECA BOOSTER OPA INTERCONNECT

- Intel Omni-Path Architecture network
  - 100 Gb/s per link and direction
  - Full fat-tree topology
- Design for 200 GB/s storage bandwidth
INTEL KNIGHTS LANDING ARCHITECTURE

- 36 tiles, 2-dim mesh
- tile = 2 cores + 2 VPU/core + 1 MB L2
- 4 threads per core
- AVX-512 ISA extension
- 16 GiB MCDRAM
  - High bandwidth
  - Ca. 500 GB/s
- 6 DDR4 channels
  - Ca. 100 GB/s
The Booster has essentially the same software environment than JURECA.

Key differences:
- Less software, due to its more specialized nature.
- Slightly different ISA (AVX-512): incompatible with Haswell nodes in most cases.
- Interactive sessions need extra care:
  - `srun --pty --cpu_bind=none --mpi=none /bin/bash {\-l\-i}`
- Test and compile partition in Slurm is `develbooster`.
- To browse the Booster SW from the login nodes:
  - `ml Architecture/KNL`
JURECA BOOSTER ENVIRONMENT (2/3)

• Option 1: Compilation on KNL nodes
  • Get an interactive session on a Booster node, with -l or -i
  • Will load the Booster SW environment
  • Set your flags correctly to enable AVX-512 (Intel: -xHost, GNU: -march=native)

• Option 2: Cross-compilation
  • Load the Architecture/KNL module
  • Set your flags correctly to enable AVX-512 (Intel: -xMIC-AVX512
    GNU: -march=knl -mtune=knl)
  • Can fail if the build process requires to execute binaries compiled
    with these flags
Job submission

- Use booster partitions: booster, develbooster, modetestbooster, largebooster

Make sure you have the right environment for the job:

- either when submitting the job (Architecture/KNL loaded)
- or inside the job (load Architecture/KNL in the script submitted)

Support for different KNL NUMA (snc4flat, snc4cache) modes at modetestbooster and (quadcache) mode available at booster and develbooster partition

Information about partitions published online at JSC ⇒ Expertise ⇒ Supercomputers ⇒ JURECA ⇒ UserInfo ⇒ QuickIntroduction
JURECA CLUSTER+BOOSTER
JURECA CLUSTER+BOOSTER ARCHITECTURE

- **Bisection Bw:** 94 Tb/s
- **198 bridge nodes**
  - Capacity: 20 Tb/s
- **26 router nodes**
  - Capacity: 2 Tb/s
- **2x SX6036G**
  - Capacity: 1.4 Tb/s
- **Bisection Bw:** 82 Tb/s
JURECA: FILESYSTEMS

• All user filesystems mounted from the central GPFS fileserver Jülich Storage Cluster (JUST)
  • Exception: Node local /tmp filesystem (ext4), O(10 GiB)
• $PROJECT
• $HOME
• $SCRATCH
• $FASTDATA
• $DATA
• $ARCHIVE
POC: FULL-SYSTEM LINPACK ON JURECA (NOV 2017)

<table>
<thead>
<tr>
<th>T/V</th>
<th>N</th>
<th>NB</th>
<th>P</th>
<th>Q</th>
<th>Time</th>
<th>Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHC00L2L4</td>
<td>5321904</td>
<td>336</td>
<td>40</td>
<td>84</td>
<td>26565.78</td>
<td>3.78257e+06</td>
</tr>
</tbody>
</table>

HPL_pdgesv() start time Sun Nov 5 00:23:35 2017

HPL_pdgesv() end time Sun Nov 5 07:46:21 2017

HPL Efficiency by CPU Cycle 5328300.353%
HPL Efficiency by BUS Cycle 9446281.578%

\[ \|Ax-b\|_{\infty} / (\epsilon \cdot (\|A\|_{\infty} \cdot \|x\|_{\infty} + \|b\|_{\infty}) \cdot N) = 0.0030562 \quad \ldots \quad \text{PASSED} \]

1760 Cluster nodes + 1600 Booster nodes + 120 bridge nodes
• Cluster + Booster operated as **one system**
  • One management domain
  • Common **slurmctld** + ParaStation Modulo
  • CentOS 7.X
    • We try to keeping major versions in sync
• Slurm 19.05 + ParaStation Modulo psslurm plugin
  • Support for heterogeneous jobs with common MPI_COMM_WORLD
• ParaStation Modulo: Gateway (GW) protocol
  • ParaStation psgwda gateway daemons launched on bridge nodes
  • GW protocol integrated in ParaStation pscom layer ⇒
    Transparent for MPI application using ParaStation MPI
• Software-defined/adaptable routing for traffic
• Job submission via new Slurm heterogeneous job feature
  • multiple components with individual resource specifications (partition, gres, etc.) supported
• Please contact sc@fz-juelich.de if you need support for this feature
CLUSTER+BOOSTER JOBS (2/5)

Slurm Hetjobs

• Hetjobs are co-scheduled jobs
• Users can spawn a job across multiple partitions of a cluster

salloc/srun example:

```
$ salloc -A <budget account> -N 1 -p batch : -N 10 -p booster
$ srun ./prog1 : ./prog2
```

sbatch example:

```
#!/bin/bash
#SBATCH -A <budget account> -N 1 -p batch
#SBATCH packjob
#SBATCH -N 10 -p booster

srun ./prog1 : ./prog2
```
• A tool called xenv was implemented to ease the task of loading modules for heterogeneous jobs.

Example jobscript:

```bash
#!/bin/bash
#SBATCH -A <budget account> -N 1 -p batch
#SBATCH packjob
#SBATCH -N 10 -p booster

srun ./prog1 : ./prog2
srun -p batch xenv -L intel-para ./app-cluster : -p booster xenv -L
Architecture/KNL intel-para ./app-booster
```
When the nodes of a job belong to different interconnects and MPI communication is used, bridging has to take place. To solve this problem a Gateway Daemon (psgwd) was implemented.

The psgw plugin for the ParaStation management daemon extends the Slurm commands salloc, srun and sbatch with the following options:

- `--gw_num=number` Number of gateway nodes
- `--gw_file=path` Path to the gateway routing file
- `--gw_plugin=string` Name of the route plugin
CLUSTER+BOOSTER JOBS (5/5)

MPI Traffic Across Modules

Usage example:

```
$ salloc --gw_num=2 -N 1 hostname : -N 2 hostname
$ srun xenv [-L pscom-gateway] ./prog1 : xenv [-L pscom-gateway] [msa_fix_ld] ./prog2
```

- pscom-gateway needs to be loaded on Cluster Module

- For an up to date documentation please visit:
  - JSC gitlab documentation resource
FURTHER INFORMATION

• motd: Message of the day
  • Information about preventive and emergency maintenances
  • Information about system configuration changes
• https://dispatch.fz-juelich.de:8812/HIGHMESSAGES#jureca
• https://gitlab.version.fz-juelich.de/hps-public/changelog/wikis/JURECA
• During offline maintenance you are able to access your files via JUDAC
• On-line documentation
  • http://www.fz-juelich.de/ias/jsc/jureca
• User support at FZJ
  • sc@fz-juelich.de
  • Phone: 02461 61-2828