



bw | HPC - S5



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386



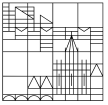
universität
uulm

universität freiburg



University of Stuttgart
Germany

Universität
Konstanz



UNIVERSITY OF
HOHENHEIM

ESSLINGEN
UNIVERSITY

Batch system

- Best Practices & Running parallel jobs -

Samuel Braun, KIT, SCC

■ Reference: bwHPC Wiki

- Most information given by this talk can be found at <https://wiki.bwhpc.de>

- Select cluster
- Select "Batch System" or "Running Batch Jobs"

The screenshot shows the 'BwUniCluster3.0' page on the bwHPC Wiki. A red circle highlights the 'BwUniCluster 3.0' link in the left sidebar. Another red circle highlights the 'Running Batch Jobs' link in the 'User Documentation' section. The main content area includes a description of the system, a transition notice from version 2.0 to 3.0, and sections for registration, changes, migration, training, and user documentation. A photograph of server racks is also visible.

BwUniCluster3.0

The **bwUniCluster 3.0-KIT-GFA-HPC 3** is the joint high-performance computer system of Baden-Württemberg's Universities and Universities of Applied Sciences for **general purp** (University of Applied Sciences KIT). The bwUniCluster 3.0 complements the four bwForClusters and their dedicated scientific areas.

Transition bwUniCluster 2.0 -- bwUniCluster 3.0

The HPC cluster bwUniCluster 3.0 is the successor of bwUniCluster 2.0. It features accelerated and CPU-only nodes, with the host system of both node types consisting of classic x86_64 architecture. To ensure that you can use the new system successfully and set up your working environment with ease, the following points should be noted.

Registration

All users who already have an entitlement on bwUniCluster 2.0 are authorized to access bwUniCluster 3.0. The user only needs to **register for the new service** at <https://bwidm.s>

Changes

Hardware, software and the operating system have been updated and adapted to the latest standards. We would like to draw your attention in particular to the changes in policy. Changes to hardware, software and policy can be looked up here: [Summary of Changes](#)

Migration

bwUniCluster 3.0 features a completely new file system. **There is no automatic migration of user data!** The file systems of the old system and the login nodes will remain in operation for a period of **3 months** after the new system goes live (till July 6, 2025). In order to move data that is still needed, user software, and user specific settings from the old HOME directory to the new HOME directory, or to new workspaces, instructions are available at [Migration Guide](#)

Training & Support

- Getting Started
- E-Learning Courses (2)
- Support
- FAQ
- Send Feedback about Wiki pages

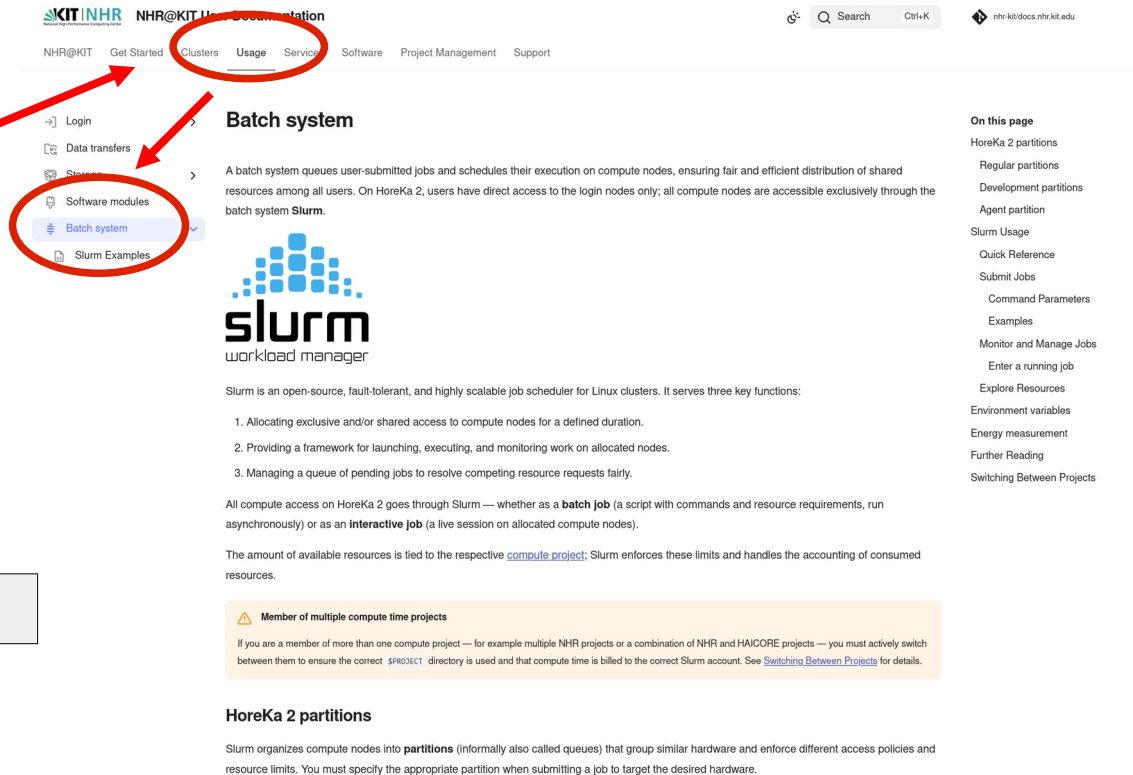
User Documentation

- Access: Registration, Deregistration
- Login
 - SSH Clients
 - Data Transfer
- Hardware and Architecture
 - Compute Resources
 - File Systems
- Cluster Specific Software
 - Using containers
 - Running Jobs
 - Running Batch Jobs
 - Running Interactive Jobs
 - Interactive Computing with Jupyter

Reference: NHR@KIT User Documentation

Most information given by this talk can be found at <https://docs.nhr.kit.edu>

Select “Usage” → “Batch System”



The screenshot shows the NHR@KIT User Documentation website. The navigation menu at the top includes 'NHR@KIT', 'Get Started', 'Clusters', 'Usage', 'Services', 'Software', 'Project Management', and 'Support'. The 'Usage' menu is expanded, showing 'Login', 'Data transfers', 'Storage', 'Software modules', 'Batch system', and 'Slurm Examples'. The 'Batch system' option is highlighted. The main content area is titled 'Batch system' and describes a batch system that queues user-submitted jobs and schedules their execution on compute nodes. It mentions HoreKa 2 and Slurm. The Slurm logo and 'workload manager' text are displayed. A list of three functions is provided: 1. Allocating exclusive and/or shared access to compute nodes for a defined duration. 2. Providing a framework for launching, executing, and monitoring work on allocated nodes. 3. Managing a queue of pending jobs to resolve competing resource requests fairly. A note states that all compute access on HoreKa 2 goes through Slurm, either as a batch job or an interactive job. A warning box titled 'Member of multiple compute time projects' advises users to ensure the correct PROJECT directory is used and that compute time is billed to the correct Slurm account. The 'HoreKa 2 partitions' section explains that Slurm organizes compute nodes into partitions, which group similar hardware and enforce different access policies and resource limits.

Workshop reservation bwUniCluster3.0 / HoreKa / HoreKa 2:

```
sbatch -p cpu --reservation=ws ...
```

■ Material: Slides & Scripts

- <https://indico.kit.edu/event/5464/>
- bwUniCluster 3.0: /opt/bwhpc/common/workshops/2026-04-15/
- HoreKa 2: /software/all/workshop/2026-04-15/

How to read the following slides

Abbreviation	Full meaning
<code>\$ command -option value</code>	<code>\$</code> = prompt of the interactive shell The full prompt may look like: <code>user@machine:path\$</code> The command has been entered in the interactive shell session
<code><integer></code> <code><string></code>	<code><></code> = Placeholder for integer, string etc
<code>foo, bar</code>	Metasyntactic variables

■ Outline

■ Best Practices

■ Parallel jobs

■ OpenMP

■ MPI

■ Hybrid (MPI + OpenMP)

Best practices

■ Best practices for job submission

■ SLURM long options for the batch script, short options for the command line

■ Set default values in the script, overwrite in command line if needed

■ Pass arguments to batch job on submission via command line: `$ sbatch job.sh -x argument`

■ When using shared nodes:

■ Only requested resources available (e.g. memory; defaults apply though)

■ SSD storage is shared **without** resource control!

■ Environment and data paths

■ Always ensure a well-defined environment

```
module purge  
module load foo bar
```

```
#SBATCH --export=NONE # do not export env
```

■ Change to absolute path, Slurm submit directory or similar

■ Job script templates

- Many software packages provide help description, example cases or job templates
- How to get it? => Search in description for example directory, example Turbomole

```
$ module show chem/turbomole 2>&1 | grep "EXA_DIR"  
/opt/bwhpc/common/chem/turbomole/TmoleX2025/bwhpc-examples
```

bwHPC_turbomole_single-node_tmpdir_example.sh

```
#!/bin/bash  
  
## Purpose: Turbomole JOB example script for bwHPC, such as bw{For,Uni}  
Cluster  
##           for   S I N G L E   N O D E   runs   O N L Y  
...  

```

■ Job monitoring

- Do NOT run scripts that submit **every second** commands like:

- `squeue`
- `scontrol show job <JOB_ID>`
- `tail -f <Global_file_system>/<file>`
 - Change to „`tail -f -s 10`“ etc.

- How to follow **live** the job progress on compute node?

- `srun --jobid=<JOB_ID> [--overlap] --pty /usr/bin/bash`
- `htop`
- monitor local storage

■ Job monitoring – job report

■ Check out resource usage for optimization

■ Wallclock time

■ CPU

■ Memory

■ Energy

■ Further monitoring => Talk later

===== JOB FEEDBACK =====

Job ID: 23482310

Cluster: hk

User/Group: ab1234/my-project

Account: my-project

State: COMPLETED (exit code 0)

Partition: cpuonly

Nodes: 4

Cores per node: 152

Nodelist: hkn[0201-0202,1603-1604]

CPU Utilized: 00:21:29

CPU Efficiency: 3.59% of 09:57:52 core-walltime

Job Wall-clock time: 00:00:59

Starttime: Mon Mar 25 13:35:31 2024

Endtime: Mon Mar 25 13:36:30 2024

Memory Utilized: 25.92 GB

Memory Efficiency: 2.73% of 950.00 GB

Energy Consumed: 68710 Joule / 19.086111111111111 Watthours

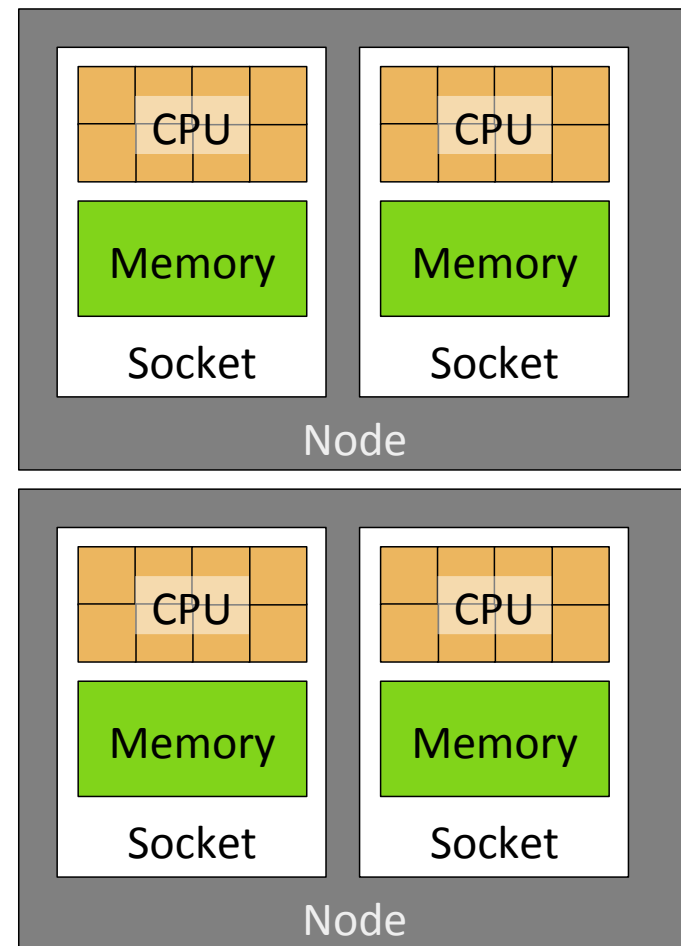
Average node power draw: 1164.57627118644 Watt

Parallel Jobs

■ Resource specifications

■ Computing resources are clustered in several layers:

- node = compute server
(sharing main memory, mostly 2 sockets per node)
 - socket = set of cpu unit (with several cores) + main memory
 - cpu = CPU Core (with two threads each)
 - task = instance of your program
 - thread = sequence for instructions
- ### ■ Hyperthreading:
- Use of both threads per core
 - Only with OpenMP, not MPI
 - Only in some cases beneficial => test if it improves performance



■ How to do the exercises?

■ Login to cluster & Generate workspace „bwhpc-course“

```
$ ws_allocate bwhpc-course 30  
Info: creating workspace.  
/pfs/work9/workspace/scratch/ka_ej4555-bwhpc-course  
remaining extensions : 3  
remaining time in days: 30
```

■ Clone examples to your workspace

```
$ cd $(ws_find bwhpc-course)  
$ git clone git@gitlab.kit.edu:kit/scc/scs/job-examples-nhr-and-bwhpc.git  
or (publicly available)  
$ git clone git@gitlab.kit.edu:kit/scs-public/job-examples-nhr-and-bwhpc.git
```

■ Submit jobs from your workspace

```
$ cd job-examples-nhr-and-bwhpc/openmp  
$ sbatch --reservation=ws <jobscript>
```

■ Compile executables

- Compilation is done when the job runs
- Typically, compilation/installation is done in advance, separately from the job

■ OpenMP

■ OpenMP = Open Multi-Processing

- Compiler directives, library routines, environment variables to enable multi-threading on **shared-memory** multiprocessor platforms (e.g. on single node)

- <https://www.openmp.org>

■ Single task is split into multiple threads for parallel execution

- E.g. `for loop` for a matrix multiplication

■ Typical issues:

- Number of spawned threads should match resources: typically 1 thread per core
- Proper thread binding and mapping

■ OpenMP parallel example

```
#!/bin/bash
#SBATCH --partition=cpu
#SBATCH --nodes=1
#SBATCH --cpus-per-task=48
#SBATCH --time=00:02
#SBATCH --output="hello-world_omp_%j.out"

# Load modules
source /opt/bwhpc/common/etc/easybuild/enable_eb_modules
module load GCC/15.2.0

# Compile source code
g++ -fopenmp -lhwlloc hello-world_omp.cpp -o hello-world_omp

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
export OMP_PLACES=cores
export OMP_PROC_BIND=TRUE

./hello-world_omp
```

Example:

```
openmp/uc3_openmp.sh
openmp/hk2_openmp.sh
```

- Shared memory is restricted to 1 node.
- Load compiler and compile
- Do not define number of threads explicitly. Use environment variables.

■ MPI

■ MPI = Message Passing Interface

- Open standard for parallelization on **distributed memory** systems, e.g. multiple nodes
- Program is started by wrapper (e.g. `mpirun`, `srun`, ...), spawning several tasks (on multiple nodes)
 - Each task executes the same binary
 - Differences between the tasks execution depend on the rank (task id) and total number of tasks
=> relevant for the programmer
- Standard on <https://www.mpi-forum.org>

■ Variants on the clusters

- Intel-MPI
- OpenMPI

=> different versions, depending on different compilers

■ Typical issues: task assignment and binding

■ MPI process binding

■ Compute-bound job

- One MPI task per available core

■ Memory-bound job

- One/few MPI tasks per socket or node

■ Hybrid jobs (MPI + OpenMP)

- One MPI task per socket or node
- OpenMP multithreading over the whole socket/node

■ MPI parallel jobs: compute-bound

```
#!/bin/bash
#SBATCH --partition=cpu
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=96
#SBATCH --threads-per-core=1
#SBATCH --time=00:02
#SBATCH --output="hello-world_mpi+openmpi_%j.out"
```

```
# Load modules
source /opt/bwhpc/common/etc/easybuild/enable_eh_modules
module load OpenMPI
```

```
# Compile source code
mpic++ hello-world_mpi.cpp -o hello-world_mpi
```

```
mpirun ./hello-world_mpi
```

Example:

```
mpi/uc3_mpi.sh
mpi/hk2_mpi.sh
```

- Load OpenMPI module and compile
- Use mpirun to execute the binary

■ MPI parallel jobs: memory-bound

```
#!/bin/bash
#SBATCH --partition=cpu
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --time=00:02
#SBATCH --output="hello-world_mpi+openmpi_%j.out"

# Load modules
source /opt/bwhpc/common/etc/easybuild/enable_eb_modules
module load OpenMPI

# Compile source code
mpic++ hello-world_mpi.cpp -o hello-world_mpi

mpirun --bind-to core \
       --map-by socket ./hello-world_mpi
```

Example (modified):

```
mpi/uc3_mpi.sh
mpi/hk2_mpi.sh
```

■ 2 tasks per node
→ each task can consume 190 GB on UC3

■ Map each MPI task to one socket

■ MPI + OpenMP hybrid job

```
#!/bin/bash
#SBATCH --partition=cpu
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --ntasks-per-socket=1
#SBATCH --cpus-per-task=48
#SBATCH --threads-per-core=1
#SBATCH --time=00:02
#SBATCH --mem=5000
#SBATCH --output="hello-world_mpi+omp_openmpi_%j.out"

# Load modules
source /opt/bwhpc/common/etc/easybuild/enable_eb_modules
module load OpenMPI

# Compile source code
mpic++ -fopenmp -lhwloc hello-world_mpi+omp.cpp -o hello-world_mpi+omp_openmpi

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
export OMP_PLACES=cores
export OMP_PROC_BIND=TRUE

mpirun --map-by package --bind-to package ./hello-world_mpi+omp_openmpi
```

```
mpi+openmp/uc3_one-mpi-task-per-socket.sh
mpi+openmp/hk2_one-mpi-task-per-socket.sh
```

Spawn 1 MPI task per socket and
48 OpenMP threads per MPI task (UC3)
96 OpenMP threads per MPI task (HK2)

Bind and map
threads to cores!

Thank you for your attention.

Questions?