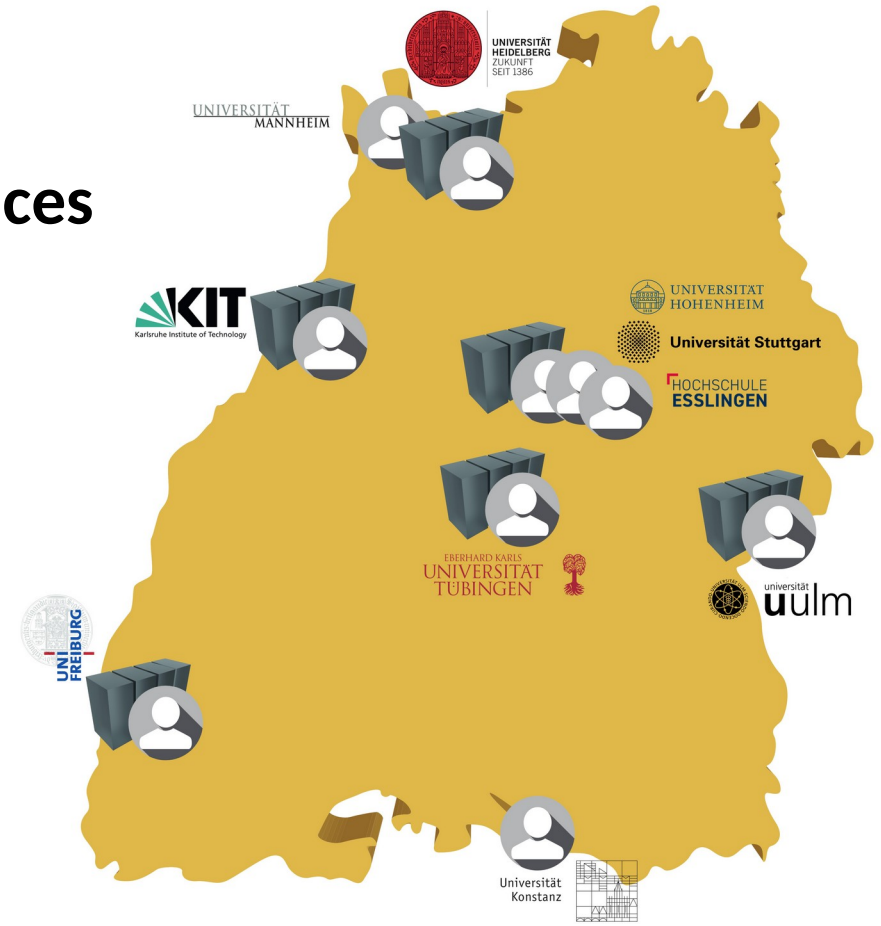
bw|HPC

KIT | NHR
National High-Performance Computing Center

Batch System – Best Practices

Andreas Baer, SCC, KIT



Funding:
Baden-Württemberg
MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST
Bundesministerium für Bildung und Forschung

How to read the following slides

Abbreviation/Colour code	Full meaning
`$ command -opt value`	`$` = prompt of the interactive shell The full prompt may look like: `user@machine:path $` The command has been entered in the interactive shell session
`<integer>` `<string>`	`<>` = Placeholder for integer, string etc
`foo, bar`	Metasyntactic variables
`${WORKSHOP}`	`/opt/bwhpc/common/workshops/2024-04-10/` (bwUniCluster) `/software/all/workshops/2024-04-10/` (HoreKa)

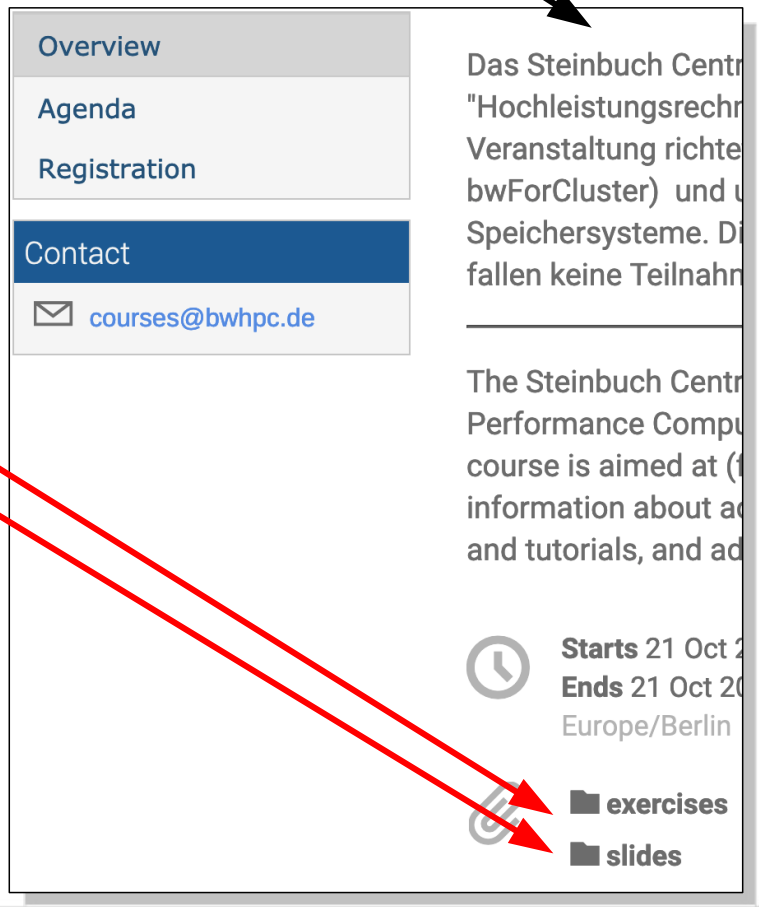
- Slurm: more info can be found in article *Batch Jobs* at:
  https://wiki.bwhpc.de/e/Category:BwUniCluster_2.0
  https://www.nhr.kit.edu/userdocs/horeka/

Where to get the slides/exercises/reservation?

- https://indico.scc.kit.edu/e/hpc_course_2024-04-10 or

  bwUniCluster: /opt/bwhpc/common/workshops/2024-04-10
  HoreKa: /software/all/workshops/2024-04-10

  - exercises
  - slides



- Workshop reservation:
  bwUniCluster:

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

```
sbatch -p multiple --reservation=ws-m ...
```

- HoreKa:

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

Outline

- Job preparation
- Job submission
- Job execution (monitoring)
- Parallel jobs

Job Preparation

Sbatch Directives

- Write SBATCH defaults in your script, overwrite time interactively via CLI, e.g.: **sbatch -J newname <script>**

```
#!/bin/bash

#SBATCH --ntasks=1
#SBATCH --time=00:01:00
#SBATCH --mem=500mb
#SBATCH --partition=single
#SBATCH --job-name=serial-test
#SBATCH --mail-type=BEGIN,FAIL,END
#SBATCH --mail-user=<my_email_address>


printenv
```

Header (Interpreter)

Sbatch Directives (Resource requirements, Notification options...)

Main section (Execution part)

Ressource specifications (1)

- @ SLURM (bwUniCluster 2.0 / HoreKa)
  - `node` = compute server
  - `task` = number of „instances“ of your command to be executed,
    default: max tasks per node is equal max cores (exception: overcommit)
  - `cpu` = CPU Cores
  - `hyperthreading` = 2 processing units on each core for threads

Examples:

- *Usually: don’t use neither* --overcommit *nor hyperthreading*

1) Addressing 2 (virtual) CPU cores (@1 task) on 1 node:

```
#SBATCH --nodes=1 --ntasks=1 --cpus-per-task=2
```

2) 80 Tasks on 1 node (on 40 physical cores):

```
#SBATCH -N 1 -n 80 --overcommit
```

Hyperthreading

- Hyperthreading is activated on bwUniCluster 2.0 and HoreKa!
- Using OpenMP hyperthreading can be used
  - Usually: don’t use it, i.e. don’t use more threads than physical cores are available
  - Check that you aren’t using hyperthreading
  - If you albeit want to use it, do performance tests before usage
- You can’t use it for MPI-programs
- For hybrid programs (MPI + OpenMP) the same holds as for OpenMP

Ressource specifications (2)

- @ SLURM (bwUniCluster 2.0, HoreKa)
  - `--mem` = Memory per node (Check defaults: „scontrol show config“)
  - `--mem-per-cpu` = Minimum memory per allocated cpu, if exceeding MaxMemPerCPU, do workaround with mem, ntasks and cpus-per-task
  - Node access policies
    - No sharing with other users‘ jobs *on* bwUniCluster 2.0 and HoreKa
    - Except in developer queues

Job Script: Templates

- Use templates for job scripts
  - Provided by many installed software packages or help description of software

Example:

- How to get? Search in description for example directory, e.g. Turbomole

```
$ module show chem/turbomole 2>&1 | grep "EXA_DIR"
/opt/bwhpc/common/chem/turbomole/7.4.1_tmolex452/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
...
...
```

Best Practises – Job setup (1)

- Directory:
  - Don’t run your code/application/job in ${HOME}, if you are working on large datasets

    Valid destinations are:

    1. Workspaces (`ws_allocate`)
    2. $TMPDIR/BeeOND (limited capacity, limited to single job)

  - Issues:
    - Workspaces:
      - Does not handle well codes producing Tbyte of scratch files and more then 10000 files. Solution: Change your application code, Apply for Tiger Team Support.
    - $TMPDIR:
      - Requires job script setup to handle data transfer

```
#!/bin/bash
#SBATCH ...
cp -pr ${SLURM_SUBMIT_DIR}/<file> ${TMPDIR}
...
cp -pr ${TMPDIR}/<results>* ${SLURM_SUBMIT_DIR}
```

Best Practises – Job setup (2)

- Directory: (cont.)
  - Potential problems:
    - During job run, binaries/inputfiles etc not found

      → give full path to binaries/inputfiles

      → change DIR in jobscript

```
#!/bin/bash
#SBATCH ...
# Change to job submission directory
cd ${SLURM_SUBMIT_DIR}
```

- Record resource usage to optimise resource requests
  - Walltime:

```
#!/bin/bash
#SBATCH ...

# Record runtime of executed program
SECONDS=0
time ./program
...
echo $SECONDS
```

Best Practises – Job setup (3)

- Check resource usage to optimize resource requests (job output)
  - Wallclock, CPU, memory & consumed energy

```
============================== 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 / 19.0861111111111 Watthours
Average node power draw: 1164.57627118644 Watt
```

Job submission

Job Submit: passing arguments to job script

- MOAB: Not allowed:

```
$ msub job.sh -x argument
```

→ msub will interprete -x as an own option

- SLURM: allowed:

```
$ sbatch job.sh -x argument
```

→ sbatch assumes that everything after stating the script name are user‘s arguments

- Solution:

  Submit wrapper script:

```
#!/bin/bash
your_script -x argument
```

- Export your script options and arguments to environment variable; read in that variable during runtime of script

```
if [ -n "${SCRIPT_FLAGS}" ] ; then
   if [ -z "${*}"  ] ; then
      set -- ${SCRIPT_FLAGS}
   fi
fi
```

Job execution

Slurm: Best Practices – Job „Observation“

- Do NOT run script that submits **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

Parallel Jobs

How to do exercises?

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

```
$ ws_allocate bwhpc-course 30
Workspace created. Duration is 720 hours.
Further extensions available: 3
/pfs/work2/workspace/scratch/xy1234-bwhpc-course-0
```

- Copy examples to your workspace

```
$ WORKSHOP=/opt/bwhpc/common/workshops/2024-04-10   # bwUniCluster
$ WROKSHOP=/software/all/workshops/2024-04-10       # HoreKa
$ cd $(ws_find bwhpc-course)
$ mkdir -v 2024-04-10; cd 2024-04-10
$ cp -pr ${WORKSHOP}/exercises/04 ./
```

- Submit jobs from your workspace

```
$ cd $(ws_find bwhpc-course)/2024-04-10/04
$ sbatch -p {single|cpuonly} --reservation=ws [--exclusive] <jobscript>
```

Compile executables

- We need executables „pi_omp”, “parmmul” and “parmmul_omp”
- Open files “omp.README” and “parmmul.README”
  => execute commands in these files
- Hints:
  - You can use cat for an easy copy paste

```
$ cat omp.README
# To get binary for ${WORKSHOP}/exercises/04/parallel/omp.sh
module load compiler/intel
...
# pi_omp
# first compile seconds.c
icc -DFTNLINKSUFFIX -O -c seconds.c -o seconds.o
ifort -O -qopenmp pi.f90 seconds.o -o pi_omp
```

  - You can parse the whole file with bash to execute all commands subsequently

```
$ bash omp.README
$ bash parmmul.README
```

OpenMP parallel jobs (1)

- OpenMP = Open Multi-Processing

  → compiler directives, lib routines, environment variables to enable multithreading on shared-memory multiprocessor platforms

  - https://www.openmp.org

- Courses:
  - Next courses are held in 2023: HLRS (Stuttgart), JSC (Jülich)

- Typical issues:
  - Number of threads not matching given resources

    → Normally 1 thread should be mapped to 1 core
  - Thread binding and mapping

OpenMP parallel jobs (2)

Example: `${WORKSHOP}/exercises/04/omp.sh`

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --time=00:05:00

# Set executable name to variable
# ifort -O -qopenmp -o pi_omp pi_f90 seconds.o
exe=./pi_omp
# Load modules
module load compiler/intel
# Setup OpenMP environment variable
export OMP_NUM_THREADS=${SLURM_NPROCS}
# Printout number of threads
echo "No.threads = ${OMP_NUM_THREADS}"
# Execute program
${exe}
```

- Shared memory is restricted to 1 node.
- Use the precompiled pi_omp or compile pi.f90 and store the executable in pi_omp
- Do not define number of threads explicitely. Use environment variables.

OpenMP parallel jobs: Pinning

Using Intel OpenMP Thread Affinity for Pinning Threads differently

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=10
#SBATCH --time=00:01:00

# Set executable name to variable
exe=./pi_omp
# Load modules
module load compiler/intel
# Setup OpenMP environment variable
export OMP_NUM_THREADS=${SLURM_NPROCS}
# Use different pinning: none, scatter, compact
export KMP_AFFINITY=verbose,scatter
# Printout number of threads
echo "No.threads = ${OMP_NUM_THREADS}"
# Execute program
${exe}
```

`${WORKSHOP}/exercises/04/omp_v2.sh`

- Submit script with different pinnings
- Also try *verbose,compact,1,0*
- Compare results

MPI parallel jobs (1)

- MPI = Message Passing Interface
  → To enable programs parallelly running on a distributed memory system
  - MPI tutorial from Livermore Computing Center (https://computing.llnl.gov/tutorials/mpi/)
  - MPI Standards on http://www.mpi-forum.org
- Variants @ Clusters
  - Intel-MPI (impi → modules: mpi/impi/<version>)
  - OpenMPI (openmpi → modules: mpi/openmpi/<version>
  - Dependencies?
    - **Versions depend on compilers!**
- Courses:
  - Next courses will be held in 2023: HLRS (Stuttgart), JSC (Jülich), LRZ (Munich)

MPI process binding

- Compute-bound MPI
  - As many MPI tasks per node as cores available
- Memory-bound MPI
  - One MPI task per socket/node
- Hybrid MPI + OpenMP
  - One MPI task per „socket (domain)“ or „node“
  - OpenMP multithreaded process over the whole socket (domain) or node

Compute-bound: MPI parallel jobs (OpenMPI)

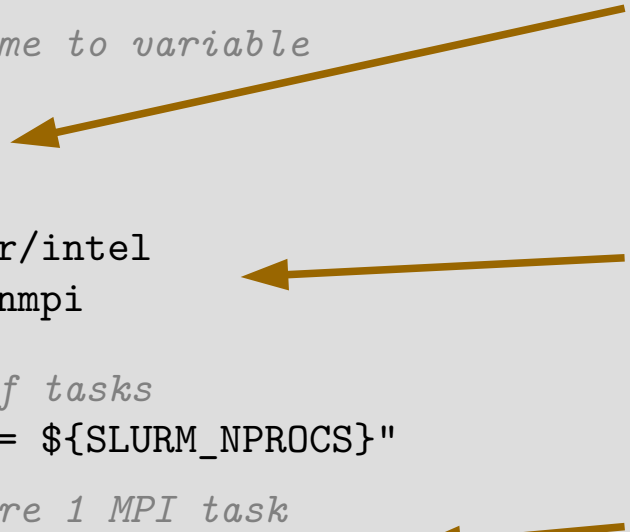
Example: `${WORKSHOP}/exercises/04/openmpi.sh`

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --ntasks=20
#SBATCH --time=00:05:00

# Set executable name to variable
exe=./parmmul
# Load modules
module purge
module load compiler/intel
module load mpi/openmpi

# Printout number of tasks
echo "No.MPI tasks = ${SLURM_NPROCS}"
# Spawn for each core 1 MPI task
mpirun --bind-to core --report-bindings ${exe}
```



- For testing example, copy the executable to submit directory
- Load Intel compiler module and OpenMPI module
- Use mpirun to execute the binary, print details of task pinning.

Memory-bound: MPI parallel jobs (OpenMPI)

```
${WORKSHOP}/exercises/04/openmpi_v2.sh
```

Spawning only 1 MPI task per socket (i.e. 2 MPI-tasks per node)

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --time=00:05:00


# Set executable name to variable
exe=./parmmul
# Load modules
module purge
module load compiler/intel
module load mpi/openmpi

# Printout number of MPI tasks
echo "No.MPI tasks = ${SLURM_NTASKS}"

# Spawn only one MPI task per socket
mpirun --bind-to core --map-by socket --report-bindings ${exe}
```

- 2 tasks per node → each task can consume 45 GB (120 GB on HoreKa)
- Loading the needed environment
- Map each MPI task to one socket

Hybrid parallel jobs: MPI + OpenMP (OpenMPI)

Spawning 1 MPI task per socket, and n(=20) OpenMP threads per MPI task

`${WORKSHOP}/exercises/04/hybrid_openmpi_omp.sh`

```
#!/bin/bash

#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=20
#SBATCH --time=00:05:00

# Set executable name to variable
exe=./parmmul_omp
# Load modules
module load compiler/intel mpi/openmpi
# Setup OpenMP env variable
export OMP_NUM_THREADS=$((${SLURM_CPUS_PER_TASK}))
export KMP_AFFINITY=verbose,scatter
# Printout number of nodes = MPI tasks
echo "No. MPI tasks (nodes) = ${SLURM_NTASKS}"
echo "No. threads per node  = ${OMP_NUM_THREADS}"
# Spawn only one MPI task per socket
mpirun -n ${SLURM_NTASKS} --bind-to core --map-by socket:PE=${OMP_NUM_THREADS}
       --report-bindings ${exe}
```

Set KMP_AFFINITY to bind and map threads to cores!

Thank you for your attention!

Questions?