

Introduction to Cluster Computing at Goethe-NHR & Fuchs

Anna Bartsch
Hans-Christian Jankowiak

Goethe Universität Frankfurt am Main
Center for Scientific Computing

24. March 2026

Outline

- 1 Cluster Basics
 - The CSC
 - Hardware
 - Software

- 2 SLURM
 - SLURM basics
 - Practical SLURM

- 3 Closing remarks

Outline

1 Cluster Basics

- The CSC
- Hardware
- Software

2 SLURM

- SLURM basics
- Practical SLURM

3 Closing remarks

The Center for Scientific Computing

The Center for Scientific Computing (CSC) is the central point of contact for researchers at Goethe University regarding digital services. These services cover all stages of the research cycle — from project conception and research design to data collection and analysis, publication, and knowledge transfer.

The Center for Scientific Computing

The CSC operates its own High Performance Computing (HPC) Cluster and expands its capabilities through collaborations such as with NHR South-West. It supports numerically intensive studies in a variety of research fields, ranging from economics over neuroscience to high-energy physics.

Clusters

The CSC operates two clusters

- Goethe-NHR
 - combined CPU-GPU cluster
 - available for users from universities in Germany
- FUCHS
 - CPU cluster
 - available for users from universities in the State of Hesse

Access to Goethe-NHR

Goethe-NHR

```
ssh <username>@goethe.hhlr-gu.de
```

Get an Account <https://wiki.csc.uni-frankfurt.de/>
→ Access/User Application

Prerequisites The project manager has to request for computing time for research projects at <https://jards.nhr-verein.de/>.

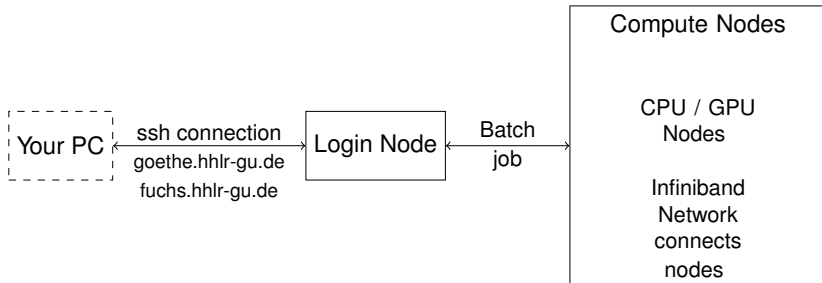
Application Please download the file Application Goethe-NHR & use a regular PDF viewer to open the forms.

FUCHS

```
ssh <username>@fuchs.hhlr-gu.de
```

Application Please download the file Application FUCHS & use a regular PDF viewer to open the forms.

Workflow



Outline

- 1 Cluster Basics
 - The CSC
 - Hardware
 - Software
- 2 SLURM
 - SLURM basics
 - Practical SLURM
- 3 Closing remarks

Hardware resources

	#nodes	CPU	# CPUs Cores	RAM
Goethe-NHR	410	Intel Xeon Skylake Gold 6148	2 / 40	192GB
	72	Intel Xeon Skylake Gold 6148	2 / 40	768GB
	139	Intel Xeon E5-2640v4 Broadwell	2 / 20	128GB
	106	AMD EPYC 7452 + 8x AMD Radeon Instinct MI210	2 / 64	512GB
FUCHS	198	Intel Xeon E5-2670v2 Ivy Bridge	2 / 20	128GB

Partitions

--partition	Max runtime	Max Nodes	Max NodesPU	Max JobsPU	Max SubmitPU	Type Nodes
general1	21d	474	150	40	50	Skylake
general2	21d	337	150	40	50	Broadwell
gpu	21d	106	20	20	30	AMD EPYC
test	1h	8	3	3	4	Skylake
gputest	8h	4 GPU	4 GPU	2	6	AMD EPYC
fuchs	21d	198	80	10	20	Ivy Bridge

Hyperthreading on Goethe-NHR

Node type	Sockets	# $\frac{\text{Cores}}{\text{Socket}}$	Threads	--extra- node-info	# $\frac{\text{Cores}}{\text{Node}}$	Default
Skylake	2	20	2	2:20:2	80	yes
	2	20	1	2:20:1	40	
AMD EPYC	2	32	2	2:32:2	128	yes
	2	32	1	2:32:1	64	
Broadwell Ivy Bridge	2	10	2	2:10:2	40	yes
	2	10	1	2:10:1	20	

Storage

mount point	/home	/work	/local	/arch0[1,2]
size	30 GB	5 TB / group member	Max. 1.4 TB	10 TB / project
access time	slow	fast	fast	slow
file system	NFS	PanFS	ext	NFS
network	ethernet	InfiniBand		ethernet

Storage

Note

- Use the /work-directory instead of /home to write out the standard output and error
- /work is a parallel filesystem & is available for huge data set during the computation
- /local is only available during computation. Files will be deleted when the job is finished.
- /arc0[1|2] is persistent & must be requested (only Frankfurt groups). They are only mounted on the login node, afterwards you can work on the node with rsync (slow).

Outline

- 1 Cluster Basics
 - The CSC
 - Hardware
 - Software
- 2 SLURM
 - SLURM basics
 - Practical SLURM
- 3 Closing remarks

Software Handling with Module Environment

- ① Basic Modules provided by CSC
 - ① Compiler
 - ② MPI
 - ③ MKL
- ② Modules provided by SPACK
- ③ Your own Modules

Environment Modules

Commands

Definition

Environments Modules provide software for specific purposes.

Syntax

```
module <command> [<modulename>]
```

<command>

avail	display all available modules
list	display all loaded modules
load or add <module>	load module
unload or rm <module>	unload module
purge	unload all currently loaded modules
switch or swap <old-module> <new-module>	unload old, load new module

Environment Modules

CSC provided modules

- Compiler
 - system: gcc 11.5.0
 - intel/oneapi/latest compiler/latest
 - intel/oneapi/2023.2.0 compiler/2023.2.1
- Libraries
 - intel/oneapi/latest mkl/latest
 - intel/oneapi/2023.2.0 mkl/2023.2.0
- MPI
 - INTEL
 - intel/oneapi/latest mpi/latest
 - intel/oneapi/2023.2.0 mpi/2021.10.0
 - OpenMPI
 - mpi/openmpi/5.0.9
 - mpi/openmpi/5.0.9-rocm-6.2.4
 - mpi/openmpi/5.0.5-rocm-7.2.0

Environment Modules

SPACK: A Package Manager for Supercomputers

- cluster user can install their research related software
- individual and modular
- Spack automates all package-related processes
 - 1 install the dependencies, if a `<package>` depends on other packages
 - 2 fetches the `<package>` 's tarball
 - 3 run `patch()` for software
 - 4 expands it
 - 5 verifies that it was downloaded without errors
 - 6 executing phase: configure
 - 7 executing phase: build
 - 8 executing phase: install
 - 9 installs it in its own directory
 - 10 makes module available

Environment Modules

Your own modules

- 1 writing a module file `~/privatemodules/modulename` in Tool Command Language (tcl) to set environment variables
- 2 `module use --append ~/privatemodules` enables you to load your own modules
- 3 `module load modulename`
- 4 use software provided by module

Outline

- 1 Cluster Basics
 - The CSC
 - Hardware
 - Software
- 2 SLURM
 - SLURM basics
 - Practical SLURM
- 3 Closing remarks

Batch System Concepts

Cluster

- consists of a set of tightly connected computers called nodes as a single system
- nodes are connected by high speed local network
- nodes have access to shared resources like shared file systems

Resource Manager

- responsible for managing the resources of a cluster like nodes, CPUs, memory, disk space and network
- manages the execution of jobs

Scheduler

- receives jobs from users
- control user jobs on the cluster
- handels job submission and put jobs into queues

Job

- Combination of a definition of needed resources and commands to be executed
- execution of user defined workflow by the batch system

SLURM

Cluster resource manager

- SLURM stands for **S**imple **L**inux **U**tility for **R**esource **M**anagement
- The user submits a job via `sbatch` to SLURM.
- SLURM calculates the work priority of each job.
- SLURM starts a job according to the priority & the resources availability.
- There is an exclusive node assignment per job.
- SLURM allocates resources of the jobs.
- SLURM provides a framework for starting & monitoring of the jobs.

SLURM commands

❶ job submission & execution

- `salloc` requests interactive jobs/allocations
- `sbatch` submits a batch script
- `srun` run jobs interactively (implicit resource allocation)

❷ job management

- `scancel` cancels a pending or running job
- `sinfo` shows information about nodes & partitions
- `squeue` shows a list of pending & running jobs
- `scontrol` shows detailed information about nodes & jobs

❸ accounting informations

- `sacct` displays accounting data for all jobs & job steps
- `sacctmgr` shows slurm account information

Job submission

1 Login to the cluster

```
ssh <username>@goethe.hhlr-gu.de
```

or

```
ssh <username>@fuchs.hhlr-gu.de
```

2 Create a job script e.g. with the .slurm extension

Example script name: `workshop_batch_script.slurm`

3 Submit this script to the cluster with `sbatch`

```
sbatch workshop_batch_script.slurm
```

4 Use of allocated resources

Job submission

- commands for job allocation

- `sbatch` is used to submit batch jobs to the queue
`sbatch [options] jobscript [args...]`
- `salloc` is used to allocate resource for interactive jobs
`salloc [options] [<command> [command args]]`

- command for job execution

- with `srun` the users can spawn any kind of application, process or task inside a job allocation
 - 1 Inside a job script submitted by `sbatch` (starts a job step)
 - 2 After calling `salloc` (execute programs interactively)

`srun [options] executable [args...]`

Indirect Job-Submission

sbatch

encapsulation of job parameters and user program call in a job script to the handover to submit command

Features:

- create prefabricated job scripts with important parameters
eliminates operator error
- simply add additional functionality
- allows transfer of additional parameters to submit command
- one-time additional expenses by draft the job-scripts

Direct Job-Submission

salloc

transfer of job parameters and user program to submit command

Features:

- allows simple, quick and flexible change of job parameters
- preferred in many of the same jobs that differ only in very few
- parameters (eg. benchmarks, the same process, different number of CPUs)
- prone to faulty operation
- additional functionality only via encapsulation in self-generated scripts (eg. load the library)

Job Execution

srun

- used to initiate job steps mainly within a job
- start an interactive job
- a job can contain multiple job steps
- executing sequentially or in parallel on independent nodes within the job's node allocation

Job-Submission

List of the submission/allocation options for sbatch and salloc:

-p, --partition	partition to be used from the job
-N, --nodes	compute nodes used by the job
-n, --ntasks	total No. of processes (MPI processes)
--ntasks-per-node	tasks per compute node
-c, --cpus-per-task	logical CPUs (hardware threads) per task
-B, --extra-node-info	Hyperthreading
-t, -time=day-hh:mm:ss	max wall-clock time of the job
-J, --job-name	set the name of the job
-o, --output	path to the job's standard output
-e, --error	path of the job's standard error
--mail-type	status information

Note

- --partition has to be set.
- srun accepts all options

Outline

- 1 Cluster Basics
 - The CSC
 - Hardware
 - Software
- 2 SLURM
 - SLURM basics
 - Practical SLURM
- 3 Closing remarks

Job Scripts Toy Examples

Listing 1: simple job starting one proces

```
#!/bin/bash
#SBATCH --job-name=TestJobSerial
#SBATCH --partition=fuchs
#SBATCH --nodes=1
#SBATCH --output=TestJobSerial-%j.out
#SBATCH --error=TestJobSerial-%j.err
#SBATCH --time=30
```

```
hostname
```

Job Scripts Toy Examples

Listing 2: Going parallel

```
#!/bin/bash
#SBATCH --job-name=TestJobParallel
#SBATCH --partition=fuchs
#SBATCH --nodes=1
#SBATCH --output=TestJobParallel-%j.out
#SBATCH --error=TestJobParallel-%j.err
#SBATCH --time=30

srun -n 2 hostname
```

Job Scripts Toy Examples

Listing 3: Going parallel across nodes

```
#!/bin/bash
#SBATCH --job-name=TestJobAcrossNodes
#SBATCH --partition=fuchs
#SBATCH --nodes=2
#SBATCH --output=TestJobAcrossNodes-%j.out
#SBATCH --error=TestJobAcrossNodes-%j.err
#SBATCH --time=60

srun --ntasks-per-node=3 hostname
```

Job Scripts Toy Examples

Listing 4: scriptexp.sh

```
#!/bin/bash
START=`date`
sleep 30
END=`date`
echo `hostname` $1 $START $END
exit 0
```

Listing 5: Serial Job, simpleloop.slurm

```
#!/bin/bash
#SBATCH --job-name=oneforloop
#SBATCH --output=expscript.out
#SBATCH --error=expscript.err
#SBATCH --partition=fuchs
#SBATCH --ntasks=20
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1200
#SBATCH --mail-type=FAIL

for N in `seq 1 20`; do
    ./scriptexp.sh $N &
done

wait
sleep 60
```

Creating a Parallel Job in SLURM

There are several ways a parallel job can be created:

- 1 by running several instances of multi-programs
- 2 by running a multi-process program (MPI)
- 3 by running a multi-threaded program (OpenMP or pthreads)

Creating a Parallel Job in SLURM

- In SLURM context, a task is to be understood as a process.
- A multi-threaded program consists of 1 task that uses several CPUs.
 - Option `--cpus-per-task` is defined for multi-threaded programs.
 - Multi-threaded jobs run on a single node, but use more than one processor on the node.
 - Tasks cannot be split across several compute nodes, so requesting several CPUs with the `--cpus-per-task` option will ensure all CPUs are allocated on the same compute node.
- A multi-process program is made of several tasks.
 - Option `--ntasks` is defined for multi-process programs.
 - By contrast, requesting the same amount of CPUs with the `--ntasks` option may lead to several CPUs being allocated on several, distinct compute nodes

Job Scripts Toy Examples

Listing 6: MPI Job, HT off

```
#!/bin/bash
#SBATCH -J TestMPI
#SBATCH --nodes=2
#SBATCH --ntasks=40
#SBATCH -o TestMPI-%j.out
#SBATCH -e TestMPI-%j.err
#SBATCH --time=0:15:00
#SBATCH --extra-node-info=2:10:1
#SBATCH --partition=fuchs

module load mpi/openmpi/5.0.9

mpirun ./mpi-prog
```

Job Scripts Toy Examples

Listing 7: MPI Job, HT on

```
#!/bin/bash
#SBATCH -J TestMPI
#SBATCH --nodes=2
#SBATCH --ntasks=80
#SBATCH -o TestMPI-7.out
#SBATCH -e TestMPI-7.err
#SBATCH --time=0:15:00
#SBATCH --extra-node-info=2:10:2
#SBATCH --partition=fuchs

module load mpi/openmpi/5.0.9

mpirun -n 80 --bind-to core:overload-allowed ./mpi-prog
```

Job Scripts Toy Examples

Listing 8: OpenMP Job

```
#!/bin/bash
#SBATCH --job-name=parallelopenmp
#SBATCH --output=expscript-%j.out
#SBATCH --error=expscript-%j.err
#SBATCH --partition=fuchs
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=20
#SBATCH --time=48:00:00
#SBATCH --mem-per-cpu=100

export OMP_NUM_THREADS=20

./helloworld
```

Job Scripts Toy Examples

Listing 9: Hybrid Job (MPI + OpenMP)

```
#!/bin/bash
#SBATCH -J TestHybrid
#SBATCH -p fuchs
#SBATCH -N 5
#SBATCH -o TestHybrid-%j.out
#SBATCH -e TestHybrid-%j.err
#SBATCH --time=2:00:00

export OMP_NUM_THREADS=5

module load mpi/openmpi/5.0.9

mpirun -n 20 ./helloworld
```

Job Array Support

- SLURM supports job arrays with option
`--array`
- Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily.
- Job arrays with many tasks can be submitted in milliseconds.
- All jobs have the same initial options (e.g. size, time, limit)
- Users may limit how many such jobs are running simultaneously.
- Job arrays are only supported for batch jobs.
- To address a job array, SLURM provides a base array ID & an arrayindex for each job, specify with
`<base job id>_<array index>`
- SLURM exports environment variables
- max array size is 1001

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:
`--ntasks=20 --ntasks-per-node=2`

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:
`--ntasks=20 --ntasks-per-node=2`
- 20 processes to stay on the same node:

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:
`--ntasks=20 --ntasks-per-node=2`
- 20 processes to stay on the same node:
`--ntasks=20 --ntasks-per-node=20`

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:
`--ntasks=20 --ntasks-per-node=2`
- 20 processes to stay on the same node:
`--ntasks=20 --ntasks-per-node=20`
- one process that can use 20 cores for multithreading (openmp):

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:
`--ntasks=20 --ntasks-per-node=2`
- 20 processes to stay on the same node:
`--ntasks=20 --ntasks-per-node=20`
- one process that can use 20 cores for multithreading (openmp):
`--ntasks=1 --cpus-per-task=20`

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:
`--ntasks=20 --ntasks-per-node=2`
- 20 processes to stay on the same node:
`--ntasks=20 --ntasks-per-node=20`
- one process that can use 20 cores for multithreading (openmp):
`--ntasks=1 --cpus-per-task=20`
- 4 processes that can use 5 cores each for multithreading (hybrid):

Submitting a Batch Script

Suppose you need 20 cores.

Example for cpus-per-task & ntasks-per-node

- 20 processes to spread across 10 nodes to have two processes per node:
`--ntasks=20 --ntasks-per-node=2`
- 20 processes to stay on the same node:
`--ntasks=20 --ntasks-per-node=20`
- one process that can use 20 cores for multithreading (openmp):
`--ntasks=1 --cpus-per-task=20`
- 4 processes that can use 5 cores each for multithreading (hybrid):
`--ntasks=4 --cpus-per-task=5`

Outline

- 1 Cluster Basics
 - The CSC
 - Hardware
 - Software
- 2 SLURM
 - SLURM basics
 - Practical SLURM
- 3 Closing remarks

Important Web Resources

- Center for Scientific Computing (CSC)
<https://csc.uni-frankfurt.de>
- Technical Documentation
<https://wiki.csc.uni-frankfurt.de>
- CSC Support
support@csc.uni-frankfurt.de
- NHR-SW
<https://nhrsw.de/>
- NHR
<https://www.nhr-verein.de/>

Upcoming events

- **asyncmd: Constructing Complex Molecular Dynamics Workflows in Python**
<https://indico.global/event/17114/>
Fr, 10.04.2026
- **Parallel Programming with MPI and OpenMP (4-Day Workshop)**
<https://indico.global/event/16369/>
Tu, 21.04.2026 to Fr, 24.04.2026
- **Introduction to Cluster Computing at Goethe-NHR & Fuchs**
<https://indico.global/event/17138/>
21.05.2026
- **NHR Workshop on Practical Quantum Computing 2026**
<https://veranstaltungen.hpc-in-deutschland.de/en-us/events/1715/>
Th, 21.05.2026

Event Calendar

- Simulation, Data Analysis and Machine Learning in Particle Physics
<https://indico.global/event/17091/>
Mo, 01.06.2026 to Tu, 02.06.2026
- AI for Molecular Mechanism Discovery - Introduction to trajectory-based enhanced sampling
<https://veranstaltungen.hpc-in-deutschland.de/en-us/events/1721/>
Mo, 13.07.2026 to We, 15.07.2026
- Monte Carlo Methods in High Performance Computing
<https://veranstaltungen.hpc-in-deutschland.de/en-us/events/1711/>
Mo, 13.07.2026

Event Calendar

- Particle Mesh Methods using HPC
<https://veranstaltungen.hpc-in-deutschland.de/en-us/events/1712/>
06.10.2026 to We, 07.10.2026
- cmake and Makefiles - An Introduction
<<https://veranstaltungen.hpc-in-deutschland.de/en-us/events/1714/>
Th, 08.10.2026
- Python for Scientists
<<https://veranstaltungen.hpc-in-deutschland.de/en-us/events/1713>
Tu, 17.11.2026 to Th, 19.11.2026

Event Calendar