

# High Performance Computing in a Nutshell

HPC Services, NHR@FAU

[hpc-support@fau.de](mailto:hpc-support@fau.de)

<https://doc.nhr.fau.de>

Which application area do you come from?

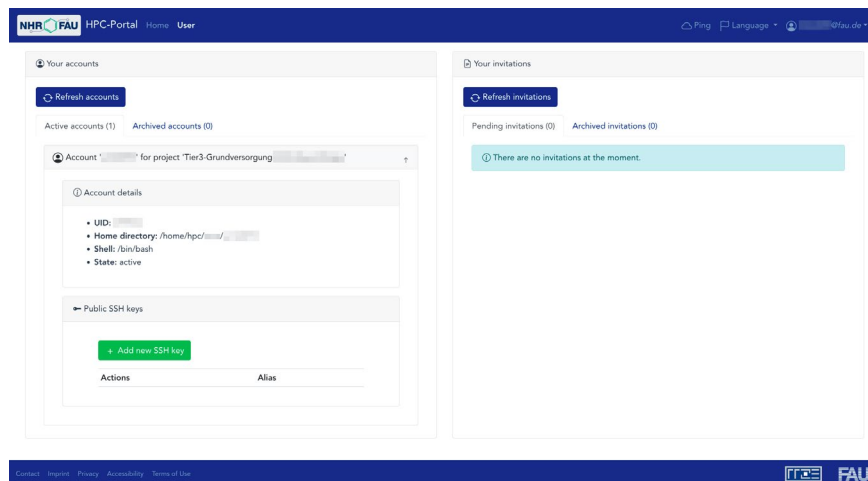


# Getting an HPC account

<https://doc.nhr.fau.de/account/>

# HPC account – HPC-portal

- Account via [portal.hpc.fau.de](https://portal.hpc.fau.de)
- Login to HPC portal with credentials of home institution via SSO
- Invitation has to be send by project PI/technical contact/contact person at chair
- If you change your affiliation or project, you need a new HPC account. Data migration may be required



The screenshot displays the HPC-portal user interface. The top navigation bar includes the NHR and FAU logos, the text 'HPC-Portal Home User', and utility links for 'Ping', 'Language', and a user profile icon. The main content area is split into two panels. The left panel, titled 'Your accounts', features a 'Refresh accounts' button and shows one active account for project 'Tier3-Grundversorgung'. Below this, the 'Account details' section lists: UID, Home directory (/home/hpc/...), Shell (/bin/bash), and State (active). A 'Public SSH keys' section contains an 'Add new SSH key' button and input fields for 'Actions' and 'Alias'. The right panel, titled 'Your invitations', has a 'Refresh invitations' button and shows zero pending or archived invitations, with a message: 'There are no invitations at the moment.' The footer contains links for 'Contact', 'Help', 'Privacy', 'Accessibility', and 'Terms of Use', along with the FAU logo.

<https://doc.nhr.fau.de/hpc-portal/>

# HPC account – HPC-portal

- No account passwords, cluster login only via SSH keys!
- SSH keys can be managed over HPC portal

The screenshot shows the HPC-Portal interface. At the top, there is a navigation bar with the NHR@FAU logo, 'HPC-Portal', 'Home', and a 'User' dropdown menu. Below this, the 'Your accounts' section is visible, featuring a 'Refresh accounts' button and tabs for 'Active accounts (1)' and 'Archived accounts (0)'. A single account is listed for the project 'Tier3-Grundversorgung'. Below the account list, the 'Account details' section shows fields for UID, Home directory, Shell, and State. At the bottom of the account list, there is a '+ Add new SSH key' button. A red box highlights this button, with an arrow pointing to a larger, detailed view of the 'Public SSH keys' form on the right.

project is either:

- "Tier3-Grundversorgung"
- name of an NHR project

The detailed view of the 'Public SSH keys' form shows a 'Back' button, 'Submit' button, and 'Reset' button. Below these is a section titled 'Add new SSH key'. It contains an 'Alias' input field with a placeholder text 'Please enter an alias for this SSH key.' and a 'Key Content' text area with a placeholder text 'Please enter the key content of this SSH key.' and a 'Hide advanced options' button.

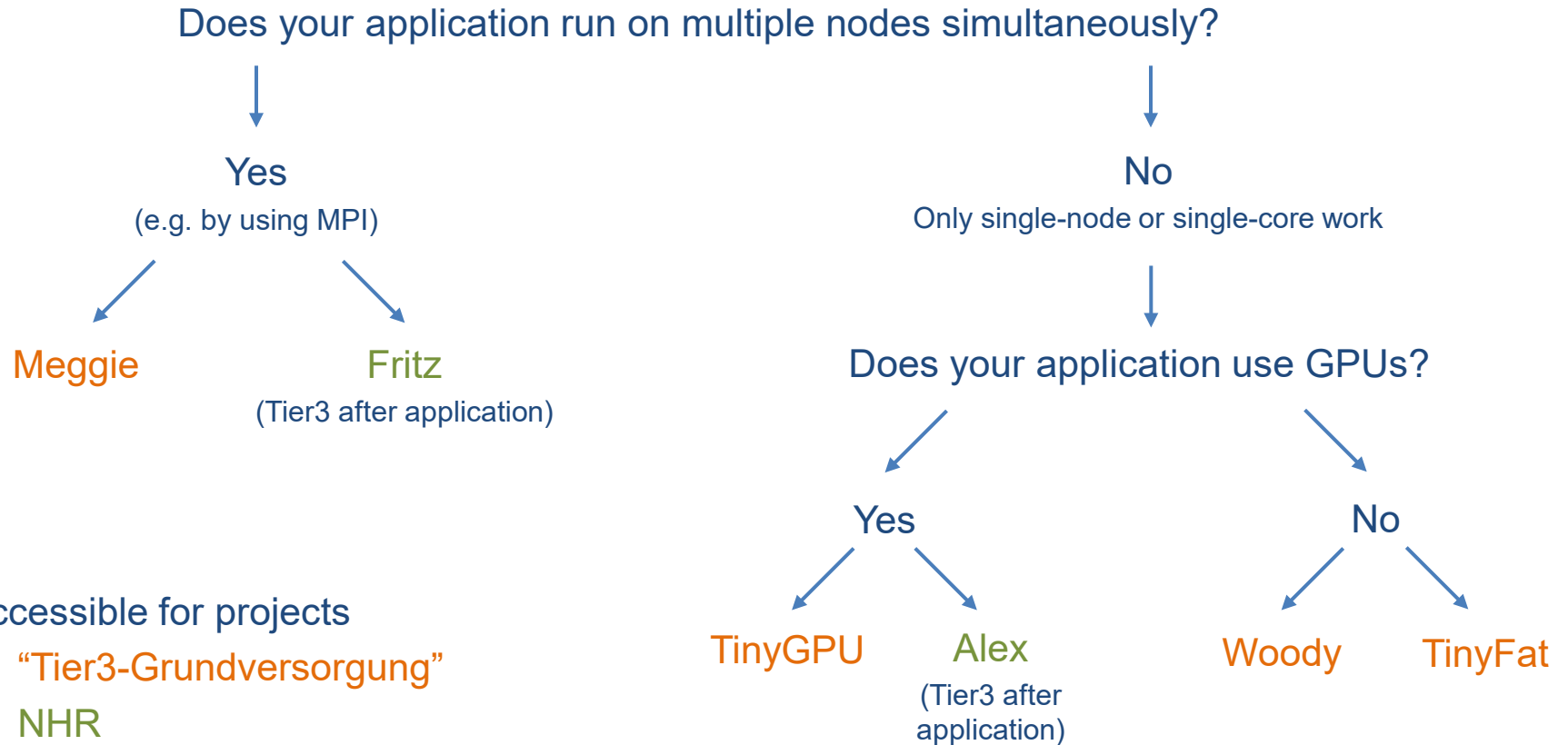
How to create SSH keys:

<https://doc.nhr.fau.de/access/ssh-command-line/>

# HPC systems at NHR@FAU

<https://doc.nhr.fau.de/clusters/overview/>

# Which cluster should I use?



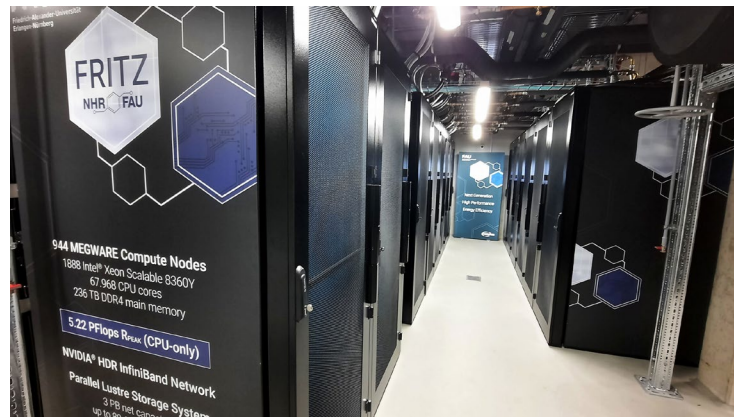
Accessible for projects

- “Tier3-Grundversorgung”
- NHR

# “Fritz” cluster

NHR parallel cluster, open for Tier3 users after application

- 992 compute nodes (71.424 cores)
  - 2 Intel Xeon Platinum 8360Y “Ice Lake” 2.4 GHz (36 cores)
  - 256 GB main memory per node
- 64 huge-memory nodes
  - 2 Intel Xeon Platinum 8470 “Sapphire Rapids” 2.0 GHz (52 cores)
  - 1 or 2 TB of main memory per node
- Blocking (1:4) HDR100 Infiniband network, up to 100 GBit/s
- Parallel file system with 3,5 PB capacity





# “Meggie” cluster

cluster for parallel jobs, intended for highly parallel jobs (Tier3)

- 728 Compute nodes (14.560 cores)
  - 2 Intel Xeon E5-2630 v4 (Broadwell) 2.2 GHz (10 cores)
  - 20 cores/node
  - 64 GB main memory
- No local disks
- Intel OmniPath network: Up to 100 Gbit/s
- To be decommissioned “soon”



# “Alex” cluster

NHR GPGPU cluster, open for Tier3 users after application

- 44 nodes with
  - 8x NVIDIA A100 (each 40 GB / 80GB HBM2)
  - 2x AMD EPYC 7713 “Milan” 2.0 GHz, 1024 GB / 2048 GB of main memory
  - 14TB local NVMe SSD
  - HDR200 Infiniband network
- 38 nodes with
  - 8x NVIDIA A40 (each with 48 GB DDR6)
  - 2x AMD EPYC 7713 “Milan” 2.0 GHz, 512 GB of main memory
  - 7 TB local NVMe SSD



# “TinyGPU” cluster

for GPU workloads – not all nodes always generally available (Tier3)

- 12 nodes with 2x “Skylake” @ 3.2 GHz, 96 GB RAM, 1.8 TB SSD, 4x RTX 2080Ti
- 4 nodes with 2x “Skylake” @3.2 GHz, 96 GB RAM, 2.9 TB SSD, 4x Tesla V100
- 7 nodes with 2x “Cascade Lake” @2.9 GHz, 384 GB RAM, 3.8 TB SSD, 8x RTX3080
- 8 nodes with 2x AMD Rome 7662 @2.0 GHz, 512 GB RAM, 5.8 TB SSD, 4x Volta A100



# “Woody” cluster

Main workhorse for throughput and single-node jobs (Tier3)

- 176 nodes with 4 cores and high clock frequency (3.5/3.7 GHz) Intel Xeon E3-1240 v? processors
  - 64x Intel Skylake, 32 GB RAM
  - 112x Intel Kaby Lake, 32 GB RAM
- 70 nodes with 2x Intel Xeon Gold 6326 (32 cores total @2.9 GHz, 256 GB RAM)
- at least 960 GB local HDD/SSD
- Gbit network only



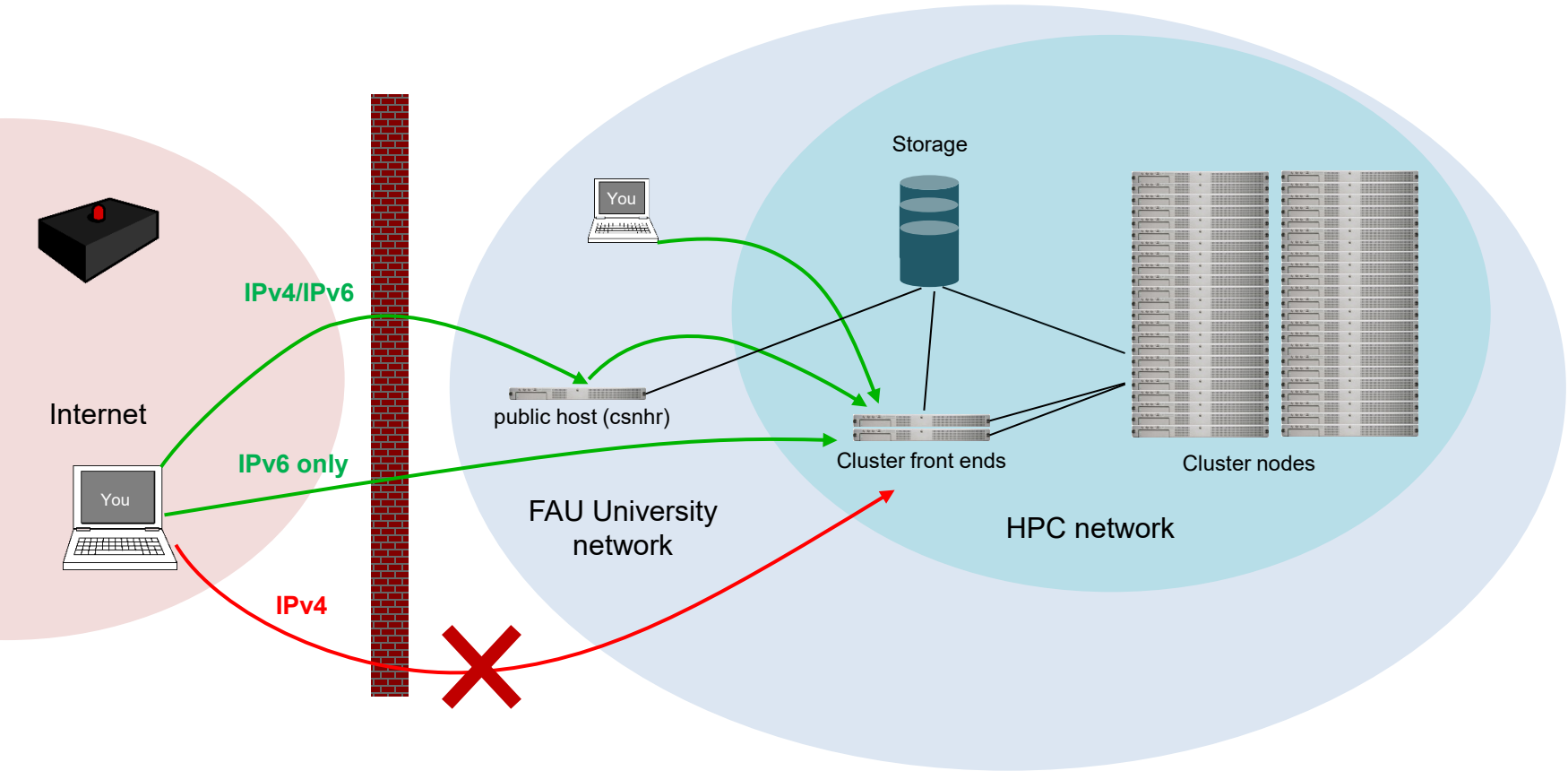
Which cluster(s) are you planning to use?



# Accessing HPC systems

<https://doc.nhr.fau.de/access/overview/>

# Cluster access



# Cluster access

---

- Cluster frontends: only directly available from within FAU network or via IPv6
  - `meggie.rrze.fau.de`
  - `tinyx.nhr.fau.de` (for TinyGPU/TinyFat)
  - `woody.nhr.fau.de`
  - `alex.nhr.fau.de`
  - `fritz.nhr.fau.de`
- Access from outside FAU network via ProxyJump over dialog server
  - `csnhr.nhr.fau.de`



# SSH – Secure Shell

- No GUI is available on the HPC systems. Basic knowledge of file handling, scripting, editing, etc. under Linux is required.

- SSH usage:

```
ssh hpcaccount@csnhr.nhr.fau.de
```

- Authentication only with SSH keys uploaded to HPC Portal, no passwords!
- General documentation on SSH, how to generate an SSH key and example ssh-config including ProxyJump: <https://doc.nhr.fau.de/access/ssh-command-line>

# Secure Shell client programs

---

- Linux: OpenSSH available
- Mac: OpenSSH available
- Windows
  - MobaXterm (<https://doc.nhr.fau.de/access/ssh-mobaxterm/>)
  - OpenSSH via Command/PowerShell (since Windows 10 v1803)
  - Linux Subsystem for Windows

# SSH – Troubleshooting

---

- Troubleshooting guide: <https://doc.nhr.fau.de/access/ssh-command-line/#troubleshooting>
- FAQs for most frequent SSH problems: <https://doc.nhr.fau.de/faq/#ssh>
- In case of problems with login, send output of the following command to [hpc-support@fau.de](mailto:hpc-support@fau.de): `ssh -vv hpcaccount@csnhr.nhr.fau.de`

# Working with data

<https://doc.nhr.fau.de/data/filesystems/>

# File systems overview

Available file systems differ in size, redundancy and how they should be used

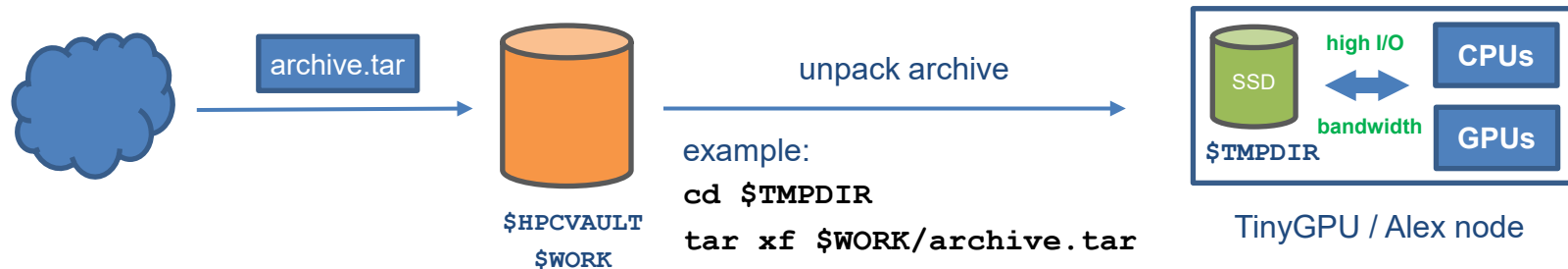
Mount point	Access	Purpose	Technology	Backup	Snapshots	Data lifetime	Quota
/home/hpc	\$HOME	Source, input, important results	NFS	YES	YES	Account lifetime	50 GB
/home/vault	\$HPCVAULT	Mid-/long-term storage	NFS	YES	YES	Account lifetime	500 GB
/home/{woody, saturn, titan, janus, atuin}	\$WORK	General-purpose, log files	NFS	NO	NO	Account lifetime	500 GB NHR project
/lxf /lustre	\$FASTTMP (fritz)	High performance parallel I/O	Lustre via InfiniBand	NO	NO	High watermark	Only inodes
/???	\$TMPDIR	Node-local job-specific dir	SSD/ ramdisk	NO	NO	Job runtime	NO

# Working with large datasets containing small files

**Best case:** use a container file format (HDF5, Parquet, ...)

Alternative: pack small files into archive. Do not unpack archive to  
`$HOME/$HPCVAULT/$WORK`

Unpack files to node-local SSDs only and use them from there



More details: <https://doc.nhr.fau.de/data/staging/>

# File system quotas

- File system may impose quotas on data volume and/or number of files
- Quotas may be set per user or per group (or both)
- Soft quota
  - Can be exceeded temporarily (7 days)
  - Turns into hard quota at end of grace period
- Hard quota:
  - absolute upper limit, cannot be exceeded

```
$ quota -s                # generic command

$ shownicerquota.pl      # only on RRZE systems
  Path                Used      SoftQ      HardQ      Gracetime  Filec      FileQ      FiHaQ      FileGrace
/home/hpc              5.7G      52.5G      104.9G      N/A         72K        500K       1,000K      N/A
/home/woody            112G      333.0G     499.5G      N/A         188K
```

<https://doc.nhr.fau.de/data/filesystems/#quotas>

# Data transfer

- `$HOME`, `$HPCVAULT`, `$WORK` are mounted on all HPC systems
- `scp` / `rsync` is used to transfer files from and to the outside  
<https://doc.nhr.fau.de/data/copying/>

```
scp [options] source destination
source/destination: local-path | username@host:remote-path
```

```
# local file to remote
# -r recurse into directories
scp -r code unrz55@csnhr.nhr.fau.de:/home/woody/unrz/unrz55
# remote file to local
scp unrz55@csnhr.nhr.fau.de:results/output.dat .
```

- Additionally on Windows: WinSCP, MobaXTerm
- Mount filesystems locally via `sshfs` or your Linux desktop  
<https://doc.nhr.fau.de/data/mounting/>



# Software

Environment modules: <https://doc.nhr.fau.de/environment/modules/>

Development and Tools: <https://doc.nhr.fau.de/sdt/overview/>

Applications: <https://doc.nhr.fau.de/apps/overview/>

What type of software are you using?



# One thing up front...

---

- The clusters at the computing center are not like your own PC
  - **You are not root**, even if the “How-To” in the github repo you found assumes that
  - **Do not blindly copy&paste** commands you do not understand
  - **Your home directory is not a local hard disk** but a shared volume that is mounted over the network
  
- Things that do not work:
  - `sudo apt install python-3.9`
  - `make && sudo make install`
  - `sudo <anything>`
  - `pip install pandas`

# Environment modules

---

- Linux standard distro packages are available on frontends and to some extent on compute nodes, but they might be outdated.
- Additional software is provided via environment modules
  - Compilers, libraries, commercial and open software
  - Installed on central server and available on all cluster nodes
- Environment modules are managed through the **module** command
- All **module** commands affect the current shell only!

# The module command

Show all available modules: `module avail`

```
$ module avail
----- /apps/modules/data/applications -----
amber/20p12-at21p11-impi-gnu                gromacs/2021.5-gcc11.2.0-impi-mkl
amber/20p12-at21p11-impi-intel             gromacs/2022.1-gcc11.2.0-ompi-mkl
amber/20p12-at21p11-openmpi-gnu-cuda11.5   gromacs/2022.1-gcc11.2.0-mkl-cuda
----- /apps/modules/data/compiler -----
gcc/10.3.0  gcc/11.2.0  gcc/12.1.0  intel/2021.4.0  intel/2022.1.0  nvhpc/22.1  nvhpc/22.2
----- /apps/modules/data/development -----
cuda/11.3.1                intelmpi/2021.4.0                openmpi/4.1.2-gcc11.2.0-cuda
cuda/11.4.2                intelmpi/2021.6.0                openmpi/4.1.2-intel2021.4.0-cuda
cuda/11.5.0                openmpi/4.1.2-gcc10.3.0-cuda     openmpi/4.1.2-oneapi2021.4.0-cuda
```

# The module command

Load a module: `module load <modulename>`

```
$ module load intel/2021.4.0
$ icc -V
Intel(R) C Intel(R) 64 Compiler Classic for applications running on Intel(R) 64, Version 2021.4.0 Build
20210910_000000
Copyright (C) 1985-2021 Intel Corporation. All rights reserved.
```

Display loaded modules: `module list`

```
$ module load openmpi/4.1.2-intel2021.4.0
$ module list
Currently Loaded Modulefiles:
1) intel/2021.4.0 <aL>  2) openmpi/4.1.2-intel2021.4.0
```

# Module command summary

<b>Command</b>	<b>What it does</b>
module avail	List available modules
module whatis	Shows verbose listing of all modules
module list	Shows which modules are currently loaded
module load <pkg>/<version>	Loads specific version of module package, i.e. adjusts environment
module unload <pkg>	Undoes what the load command did
module help <pkg>	Shows a detailed description of package
module show <pkg>	Shows which environment variables are modified and how

<https://doc.nhr.fau.de/environment/modules/>

# Using Python

- Use anaconda modules instead of system installation

```
$ module avail python
----- /apps/modules/modulefiles/tools -----
python/3.6-anaconda  python/3.7-anaconda(default)  python/3.8-anaconda
```

- Install packages via conda/pip with `--user` option
- Change default package installation path from `$HOME` to `$WORK`
- It might be necessary to configure a proxy to access external repositories
- Build packages in an interactive job on the target cluster (especially for GPUs)
- More details:
  - <https://doc.nhr.fau.de/sdt/python/>
  - <https://doc.nhr.fau.de/environment/python-env/>



# Running jobs

[https://doc.nhr.fau.de/batch-processing/batch\\_system\\_slurm/](https://doc.nhr.fau.de/batch-processing/batch_system_slurm/)

# Interactive work on the front-ends

---

- The cluster frontends are for interactive work
  - Editing, compiling, preparing input,...
  - Front-ends are shared among all users, so be considerate!
  - Amount of compute time per binary is limited by system limits
    - E.g., after 1 hour of CPU time your process will be killed
  - MPI jobs are not allowed on front ends
- Submit computational intensive work to the batch system to be run on the compute nodes!
- Use interactive batch jobs for debugging and testing.

# Batch System

- Users can interact with the resources of the cluster via the “Batch system”
- “Batch jobs” encapsulate:
  - Resource requirements (number of nodes, number of GPUs, ...)
  - Job runtime (usually max. 24 hours)
  - Setup of runtime environment
  - Commands for application run
- Batch system will handle queuing of jobs, resource distribution and allocation
- Job will run when resources become available



# Example: Batch script for Fritz

```
#!/bin/bash -l
```

```
#SBATCH --nodes=4
```

Resource requirements

```
#SBATCH --ntasks-per-node=72
```

Max. runtime

```
#SBATCH --time=06:00:00
```

```
#SBATCH --job-name=testjob_cpu
```

Other job options (name, notifications,...)

```
#SBATCH --export=NONE
```

Prevent export of environment to job

```
unset SLURM_EXPORT_ENV
```

Set up job environment

```
module load openmpi
```

```
srun ${HOME}/bin/a.out -i inputfile -o outputfile
```

Actual run of your binary

# Example: Batch script for Alex

```
#!/bin/bash -l
#SBATCH --gres=gpu:a40:1
#SBATCH --time=06:00:00
#SBATCH --job-name=testjob_gpu
#SBATCH --export=NONE

unset SLURM_EXPORT_ENV

module load python
conda activate test-environment

python train.py
```

Resource requirements

Max. runtime

Other job options (name, notifications,...)

Prevent export of environment to job

Set up job environment

Actual run of your binary

# Example: Batch script for Alex

```
#!/bin/bash -l
#SBATCH --gres=gpu:a40:1
#SBATCH --time=06:00:00
#SBATCH --job-name=testjob_gpu
#SBATCH --export=NONE

unset SLURM_EXPORT_ENV

module load python
conda activate test-environment

cd $TMPDIR
tar xzf $WORK/large-archive-with-small-files.tar.gz

python train.py
```

Resource requirements

Max. runtime

Other job options (name, notifications,...)

Prevent export of environment to job

Set up job environment

Actual run of your binary

# Slurm batch job submission

```
hpcuser@fritz1$ sbatch script.sh
```

```
Submitted batch job 675329
```

```
hpcuser@fritz1:~ $ squeue -l
```

JOBID	PARTITION	NAME	USER	STATE	TIME	TIME_LIMI	NODES	NODELIST (REASON)
675329	multinode	testjob_cpu	hpcuser	RUNNING	0:06	06:00:00	4	f[0116,0120,0159,0264]

## Specific for TinyFat/TinyGPU:

- All jobs are submitted from the frontend `tinyx.nhr.fau.de`
- Wrapper scripts have to be used for all Slurm commands (e.g. `sbatch.tinygpu`, `sbatch.tinyfat`, `squeue.tinygpu`, ...)

# GPU Jobs on TinyGPU / Alex

---

- Nodes are shared, GPUs are always exclusive
- Granularity is one GPU with a corresponding portion of CPU and main memory
- Request GPUs with **sbatch** option e.g.
  - `--gres=gpu:rtx3080:1` (to request a specific type)
  - `--gres=gpu:a100:1 --partition=a100` (necessary for V100 and A100 GPUs on TinyGPU)
- More details and examples:  
<https://doc.nhr.fau.de/clusters/alex/>  
<https://doc.nhr.fau.de/clusters/tinygpu/>



# Interactive batch job with Slurm

- TinyGPU / Alex

```
iww042@tinysx$ salloc.tinygpu --gres=gpu:1 --time=01:00:00
```

```
iww042@alex1$ salloc --gres=gpu:a40:1 --time=01:00:00
```

- Meggie / Fritz:

```
iww042@meggie1$ salloc --nodes=1 --time=01:00:00
```

- Woody / TinyFat

```
iww042@woody$ salloc --ntasks=1 --time=01:00:00
```

```
iww042@tinysx$ salloc.tinyfat --cpus-per-task=10 --time=01:00:00
```

# Slurm documentation

- NHR@FAU
  - General: [https://doc.nhr.fau.de/batch-processing/batch\\_system\\_slurm/](https://doc.nhr.fau.de/batch-processing/batch_system_slurm/)
  - Cluster-specific: <https://doc.nhr.fau.de/clusters/overview/>
  - HPC Café on “Slurm - basics, best practices and advanced usage”:  
<https://hpc.fau.de/files/2022/04/2022-04-12-hpc-cafe-slurm.pdf>,  
<https://www.fau.tv/clip/id/41306>
- Official Slurm documentation
  - Separate documentation for every command and the available options:  
[https://slurm.schedmd.com/man\\_index.html](https://slurm.schedmd.com/man_index.html)
  - Slurm commands and their counterparts in different batch systems:  
<https://slurm.schedmd.com/rosetta.pdf>
  - Slurm tutorials: <https://slurm.schedmd.com/tutorials.html>

# Some Dos and don'ts

# Good practices

---

- Be considerate. Clusters are valuable shared resources that have been paid by the taxpayer.
- Use the appropriate amount of parallelism
  - Most workloads are not highly scalable
  - Best to run scaling experiments to figure out “sweet spot”
- Use the appropriate file system(s)
  - #1 mistake: Overload file servers by doing tiny-size, high-frequency I/O to FS
  - Delete obsolete data

# Good practices

---

- Check your jobs regularly
  - Are the results OK?
  - Does the job actually use the allocated nodes in the intended way? Does it run with the expected performance?
  - Check if your job makes use of the GPUs
    - **Attach to a running job** ([https://doc.nhr.fau.de/batch-processing/batch\\_system\\_slurm/#attach-to-a-running-job](https://doc.nhr.fau.de/batch-processing/batch_system_slurm/#attach-to-a-running-job))
    - Use e.g. nvidia-smi to check GPU utilization
- Job Monitoring
  - How to use it and what to look out for: <https://doc.nhr.fau.de/job-monitoring-with-clustercockpit/>

# Good practices

---

- Talk to co-workers who are more experienced cluster users; let them educate you
- Do not re-use other people's job scripts if you don't understand them completely
- Look at tips and tricks for various applications (e.g. example batch scripts): <https://doc.nhr.fau.de/apps/overview/>
- Have a look at HPC Café talks from past events: <https://hpc.fau.de/systems-services/support/hpc-cafe/>

# Good practices

---

## When reporting a problem to NHR@FAU:

- Mail to [hpc-support@fau.de](mailto:hpc-support@fau.de) – this will immediately open a helpdesk ticket
- Provide as much detail as possible so we know where to look
  - “My jobs always crash” will not do
  - Cluster, JobID, file system, time of event, ...
  - Batch script, output files, ...
  - SSH problems: `ssh -vv`



THANK YOU.

NHR@FAU

<https://doc.nhr.fau.de>

[hpc-support@fau.de](mailto:hpc-support@fau.de)