NHR FAU

Friedrich-Alexander-Universität Erlangen-Nürnberg

High Performance Computing in a Nutshell

HPC Services, RRZE / NHR@FAU

High Performance Computing

Survey

Which application area do you come from?

NHR FAU

Friedrich-Alexander-Universität
Erlangen-Nürnberg

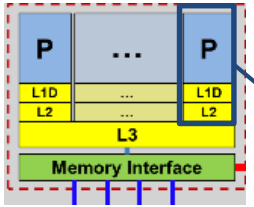
HPC systems at RRZE / NHR@FAU

https://hpc.fau.de/systems-services/documentation-instructions/

High Performance
Computing

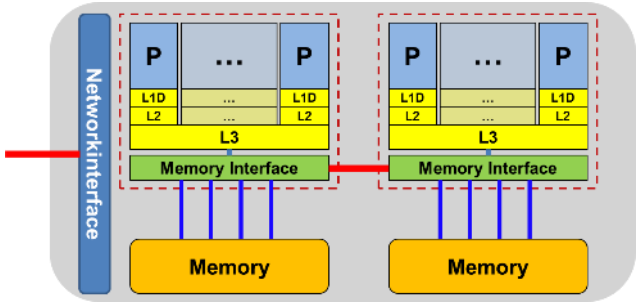
Parallel computing hardware terminology

=



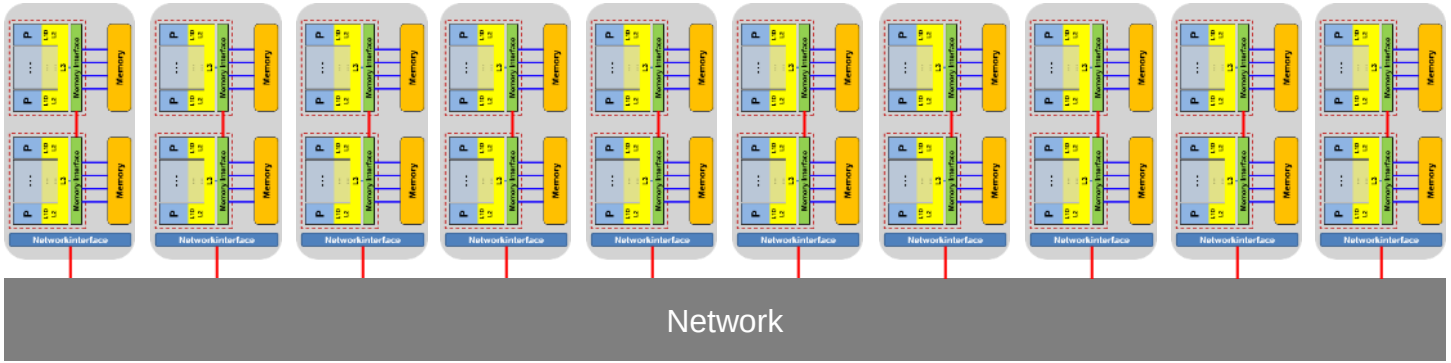
core

chip/socket
“CPU”

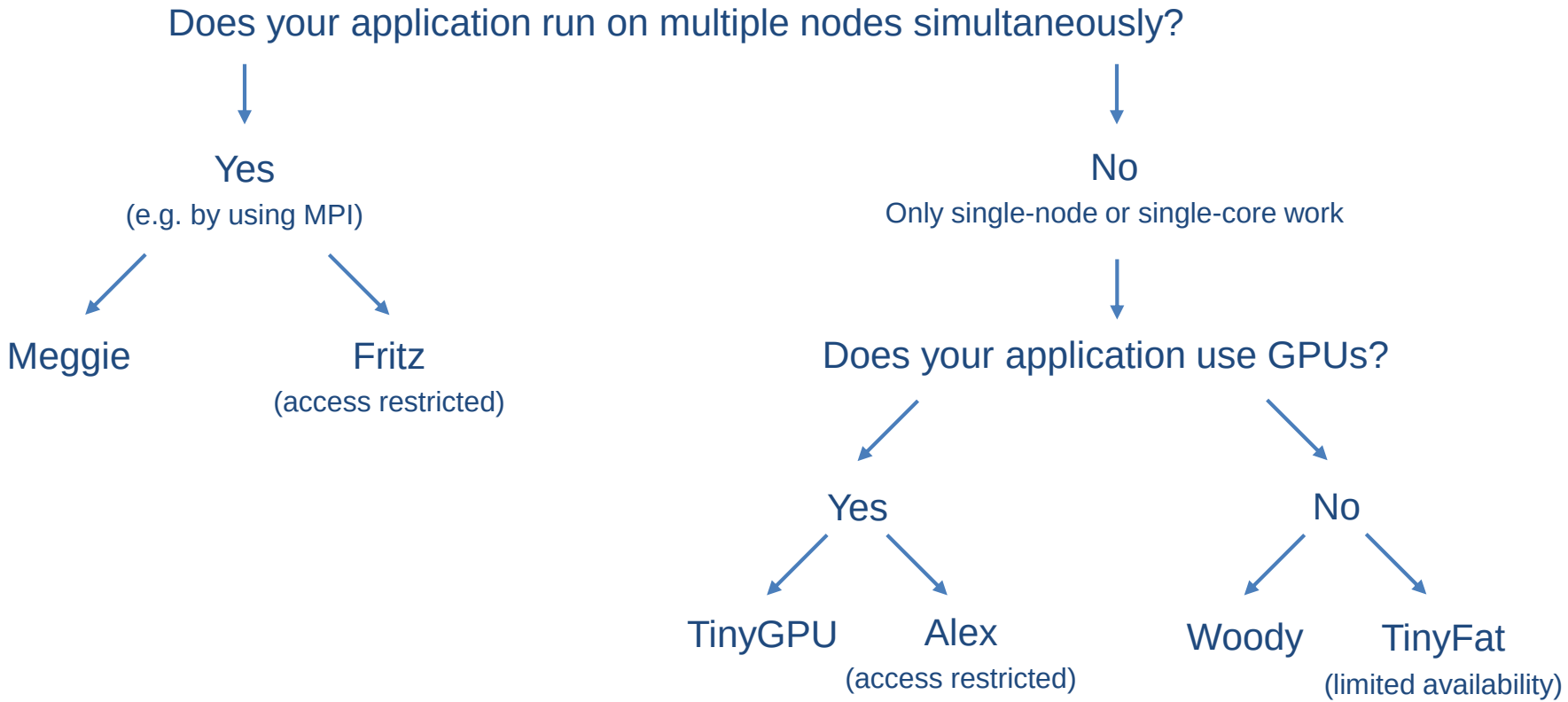


shared-memory compute node

distributed-memory cluster



Which cluster should I use?



“Meggie” cluster

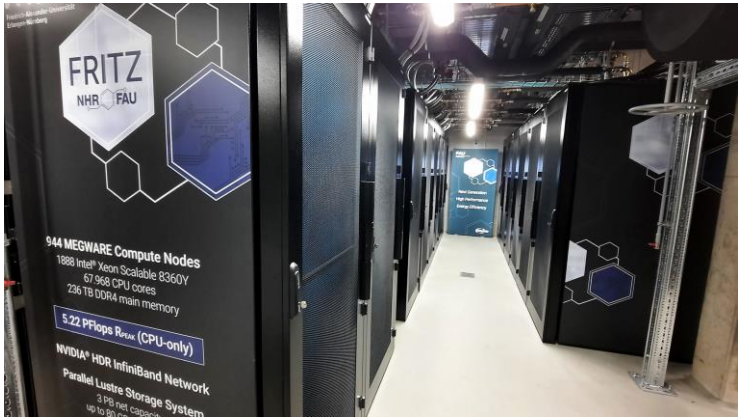
current main cluster for parallel jobs, intended for highly parallel jobs

- 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

“Fritz” cluster

NHR&Tier3 parallel cluster, access restricted

- 992 compute nodes (71.424 cores)
  - 2 Intel Xeon Platinum 8360Y “Ice Lake” 2.4 GHz (36 cores)
  - 72 cores/node (SMT disabled)
  - 256 GB main memory per node
- Blocking (1:4) HDR100 Infiniband network, up to 100 GBit/s
- Parallel file system with 3,5 PB capacity



“TinyGPU” cluster

for GPU workloads – not all nodes always generally available

- 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

“Alex” cluster

NHR&Tier3 GPGPU cluster, access restricted

- 35 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

“Woody” cluster

main workhorse for throughput and single-node jobs

- 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

Survey

Which cluster(s) are you planning to use?

NHR FAU

Friedrich-Alexander-Universität Erlangen-Nürnberg

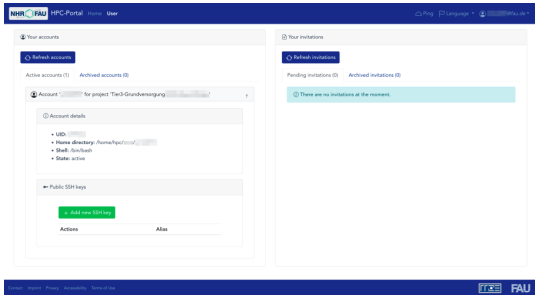
Accessing HPC systems

https://hpc.fau.de/systems-services/documentation-instructions/ssh-secure-shell-access-to-hpc-systems

High Performance Computing

HPC account – How to get one

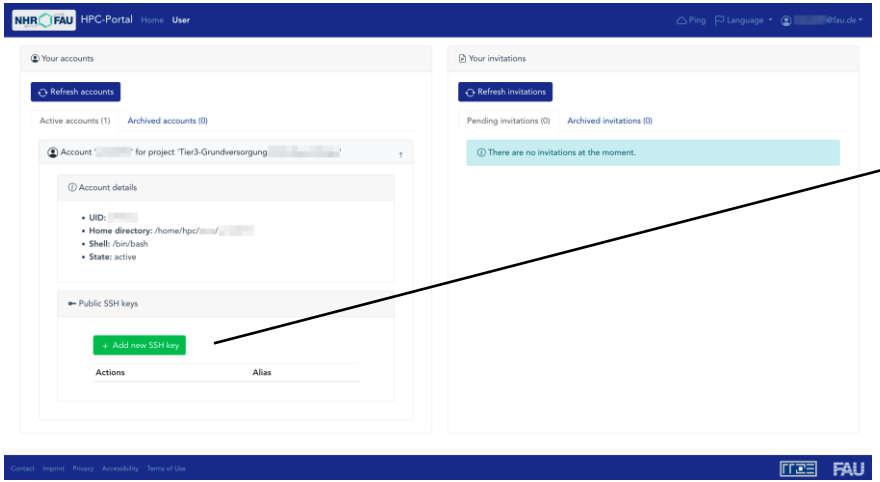
- You need a separate account (not your IdM account)
- Transition from paper application forms to HPC Portal currently in progress

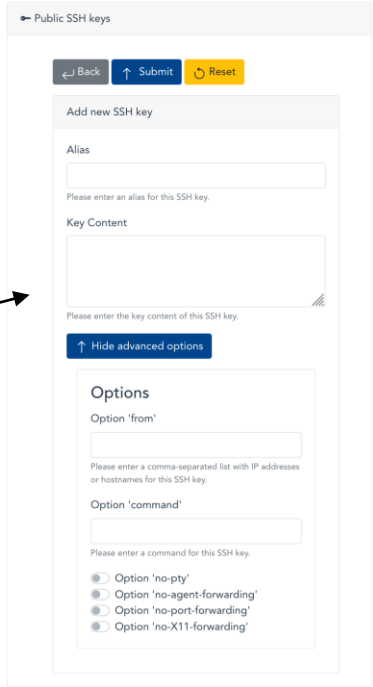


- If in doubt, ask your local RRZE contact person for help
- If you change your affiliation, you need a new HPC account. Data migration may be required

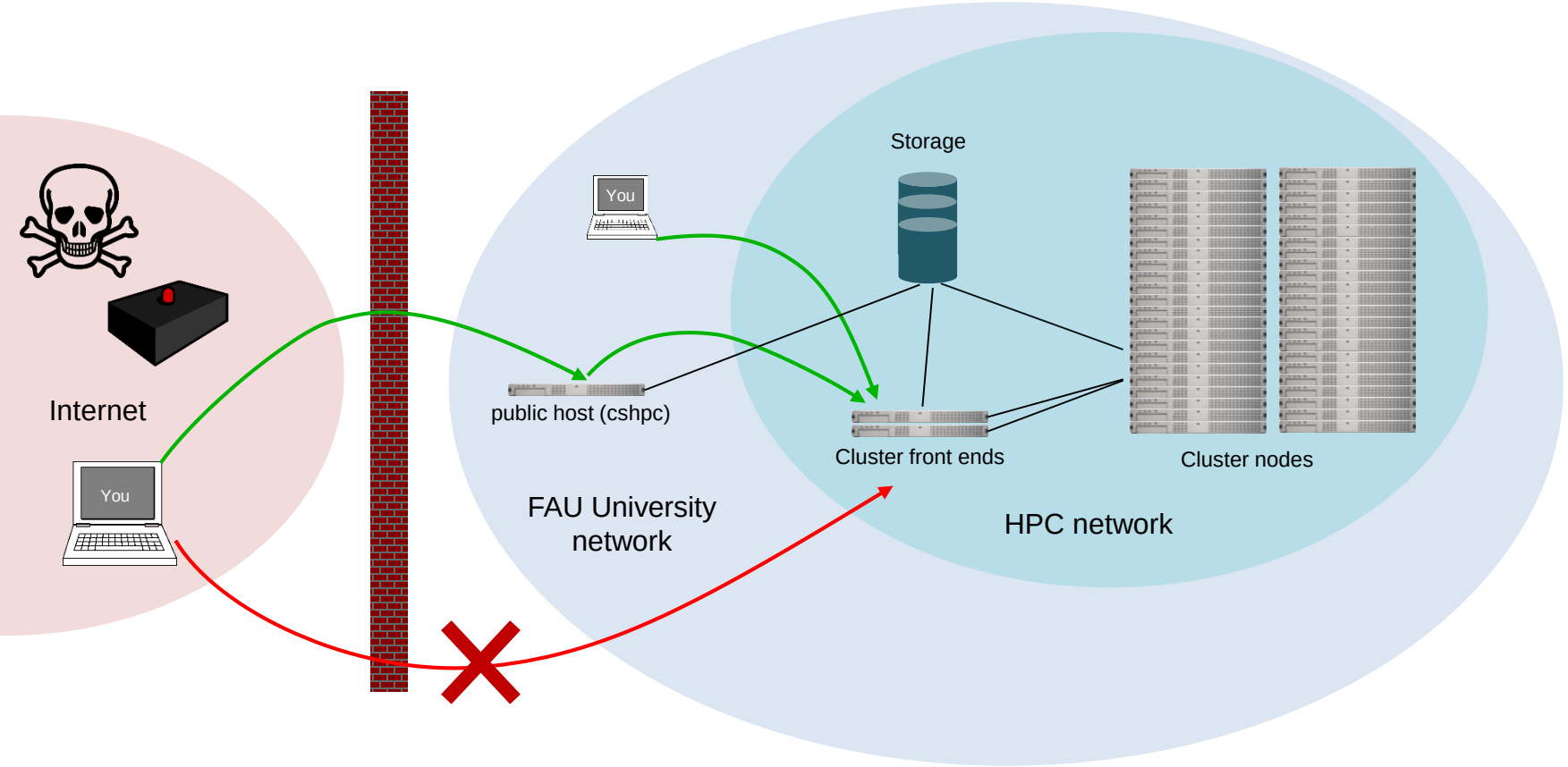
HPC account – HPC-portal

- Account via portal.hpc.fau.de
- Invitation has to be send by project PI/technical contact/contact person at chair
- Login to HPC portal with credentials of home institution via SSO
- No account passwords, cluster login only via SSH keys!
- SSH keys can be managed over HPC portal





Cluster access



Cluster access

- Primary point of contact: cluster frontends
  - **meggie.rrze.fau.de**
  - **tinyx.nhr.fau.de** (for TinyGPU/TinyFat)
  - **woody.nhr.fau.de**
  - **alex.nhr.fau.de**
  - **fritz.nhr.fau.de**
  - Only directly available from within FAU network or via IPv6
- Access from outside FAU network via dialog server
  - **cshpc.rrze.fau.de**
  - The only machine with a public IPv4 address

Secure Shell

- By default: text mode only

```
$ ssh ihpc02h@meggie.rrze.fau.de
```

- Basic knowledge of file handling, scripting, editing, etc. under Linux is required
- X11 forwarding with option **`-X`** or **`-Y`**
  - Requires local X server

- How to log into HPC systems:
  https://youtu.be/J8PqWUfkCrI
- More information on ssh:
  https://hpc.fau.de/systems-services/documentation-instructions/ssh-secure-shell-access-to-hpc-systems

Cluster access - HPC portal users

- Authentication only with SSH keys uploaded to HPC Portal, no passwords!

- Access to frontends from outside FAU:
  - directly via IPv6
  - by proxyjump over dialog server **`cshpc.rrze.fau.de`**

- Example for ssh-config:

  https://hpc.fau.de/systems-services/documentation-instructions/ssh-secure-shell-access-to-hpc-systems/#ssh_config_hpc_portal

Secure Shell client programs

- Linux: OpenSSH available in any distribution
- Mac: ditto
- Windows
  - MobaXterm (**https://mobaxterm.mobatek.net/**)
    - includes an embedded X server
  - OpenSSH via Command/PowerShell
  - Linux Subsystem for Windows
  - WinSCP (data transfer only) (**https://winscp.net**)

NHR FAU

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Working with data

https://hpc.fau.de/systems-services/documentation-instructions/hpc-storage/

High Performance
Computing

File systems

- File system == directory structure that can store files
- Several file systems can be “mounted” at a compute node
  - Similar to drive letters in Windows (C:, D:, …)
  - Mount points can be anywhere in the root file system

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

- HPC Café on “Using file systems properly“ (especially for data-intensive applications):
  https://hpc.fau.de/files/2022/01/2022-01-11-hpc-cafe-file-systems.pdf
  https://www.fau.tv/clip/id/40199

RRZE file systems overview

Mount point	Access	Purpose	Technology	Backup	Snap-shots	Data lifetime	Quota
/home/hpc	$HOME	Source, input, important results	NFS on central servers, small	YES	YES	Account lifetime	50 GB
/home/vault	$HPCVAULT	Mid-/long-term storage	Central servers	YES	YES	Account lifetime	500 GB
/home/{woody, saturn, titan, janus, atuin}	$WORK	General-purpose, small files	Central NFS server	NO	NO	Account lifetime	500 GB
/lxfs /lustre	$FASTTMP (only within fritz+alex)	High performance parallel I/O	Lustre parallel FS via InfiniBand	NO	NO	High watermark	Only inodes
/???	$TMPDIR	Node-local job-specific dir	SSD/ ramdisk	NO	NO	Job runtime	NO

File system quotas

- File system may impose quotas on
  - Stored data volume
  - Number of files and directories (inodes, actually)
- Quotas may be set per user or per group (or both)
- Hard quota
  - Absolute upper limit, cannot be exceeded
- Soft quota
  - May be exceeded temporarily (e.g., for 7 days – grace period)
  - Turns into hard quota at end of grace period

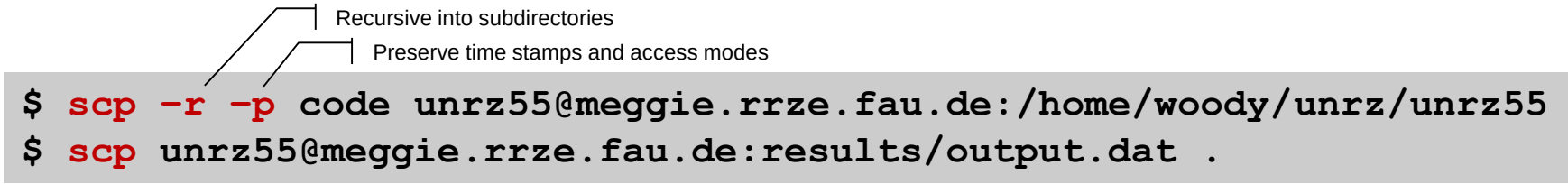
Displaying the quota limits

```
$ quota -s            # generic command
Disk quotas for user unrz55 (uid 12050):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
10.28.20.201:/hpcdatacloud/hpchome/shared
                 5544M  51200M    100G           72041    500k   1000k
wnfs1.rrze.uni-erlangen.de:/srv/home
                 112G   318G      477G            199k       0       0

$ 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                          N/A
```

Data transfer

- Most RRZE file systems are mounted at all HPC systems
  - Exception: parallel FS and node-local storage
- No NFS mounting from or to systems outside of RRZE

- → scp / rsync is the preferred file transfer tool from and to the outside



- Windows: **https://winscp.net/**

Software

https://hpc.fau.de/systems-services/documentation-instructions/environment/

Survey

What type of software are you using?

The modules system

- Linux standard distro packages available on frontends and to some extend on compute nodes, but might be outdated
- Software provided locally by RRZE via modules system
  - Compilers, libraries, commercial and open software
  - Installed on central server and available on all cluster nodes
- A package must be made available in the user's environment to become usable
  - Command: **`module`**
  - All module commands affect the current shell only!

The `module` command

Show 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

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

https://hpc.fau.de/systems-services/documentation-instructions/environment/#modules

Using Python

- Use anaconda modules instead of system installation

```
$ module avail python
------------ /apps/modules/modulefiles/tools ------------
python/2.7-anaconda   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`**
- Build packages in an interactive job on the target cluster (especially for GPUs)
- It might be necessary to configure a proxy to access external repositories

- More details: https://hpc.fau.de/systems-services/documentation-instructions/special-applications-and-tips-tricks/python-and-jupyter/

NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Running jobs

https://hpc.fau.de/systems-services/documentation-instructions/batch-processing/

HPC High Performance Computing

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: Simple Slurm batch script

- Most simple batch script (job1.sh):

```
#!/bin/bash -l
~/bin/a.out arg1 arg2 arg3
```

- Submission:

```
iww042@meggie1$ sbatch --nodes=1 --time=01:00:00 job1.sh
1051341.madm
```

Example: Complex Slurm batch script

```
#!/bin/bash -l
#SBATCH --nodes=4 --ntasks-per-node=20 --time=06:00:00
#SBATCH --job-name=Sparsejob_33

#SBATCH --export=NONE
unset SLURM_EXPORT_ENV                  # avoid login shell settings

# create a temporary job dir on $WORK
mkdir ${WORK}/$SLURM_JOB_ID
cd ${WORK}/$SLURM_JOB_ID

# copy input file from location where job was submitted, and run
cp ${SLURM_SUBMIT_DIR}/inputfile .

srun --mpi=pmi2 ${HOME}/bin/a.out -i inputfile -o outputfile
```

Job submission options:
Nodes, cores per node, time, name,…

Job option sentinel

$SLURM_* variables contain job-relevant data

Actual run of your binary

Slurm batch job submission

```
iww042@meggie1$ sbatch job3.sh
Submitted batch job 357074

iww042@meggie1:~ $ squeue -l
Mon Jan 28 17:38:52 2022
             JOBID   PARTITION    NAME     USER    STATE      TIME  TIME_LIMI  NODES NODELIST(REASON)
             357074  work         Testjob  iww042  RUNNING    0:35  1:00:00    4     m[0101-0104]
```

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 proportion of CPU and main memory
- Request GPUs with **`sbatch`** option e.g.
  - **`--gres=gpu:1`** (if you don‘t care which type you get)
  - **`--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://hpc.fau.de/systems-services/systems-documentation-instructions/clusters/tinygpu-cluster
  https://hpc.fau.de/systems-services/systems-documentation-instructions/clusters/alex-cluster/

Interactive batch job with Slurm

- TinyGPU / Alex

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

Slurm documentation

- NHR@FAU
  - General: https://hpc.fau.de/systems-services/systems-documentation-instructions/batch-processing/
  - Cluster-specific: https://hpc.fau.de/systems-services/systems-documentation-instructions/clusters/
  - 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
  - Slurm commands and their counterparts in different batch systems: https://slurm.schedmd.com/rosetta.pdf
  - Slurm tutorials: https://slurm.schedmd.com/tutorials.html

NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Some Dos and don’ts

HPC High Performance Computing

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 metadata servers by doing tiny-size, high-frequency I/O to parallel FS
  - Delete obsolete data
- Try to come up with a reasonable job runtime estimate
  - 24 hours is only the maximum
  - Job scheduling is easier if jobs are shorter

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
    - Use ssh to log into a node where you have a job running
    - Use e.g. nvidia-smi to check GPU utilization
  - For pytorch/tensorflow, check if GPUs are detected

    https://hpc.fau.de/systems-services/systems-documentation-instructions/special-applications-and-tips-tricks/tensorflow-pytorch/
  - Job Monitoring: https://monitoring.nhr.fau.de/
  - How to use it and what to look out for: https://hpc.fau.de/files/2023/01/2023-01-10-HPC-Cafe-ClusterCockpit.pdf

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://hpc.fau.de/systems-services/systems-documentation-instructions/special-applications-and-tips-tricks/
- 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 RRZE:

- Use the official contact 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, …

NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

THANK YOU.

HPC@RRZE / NHR@FAU

https://hpc.fau.de

HPC High Performance Computing