Leibniz Supercomputing Centre

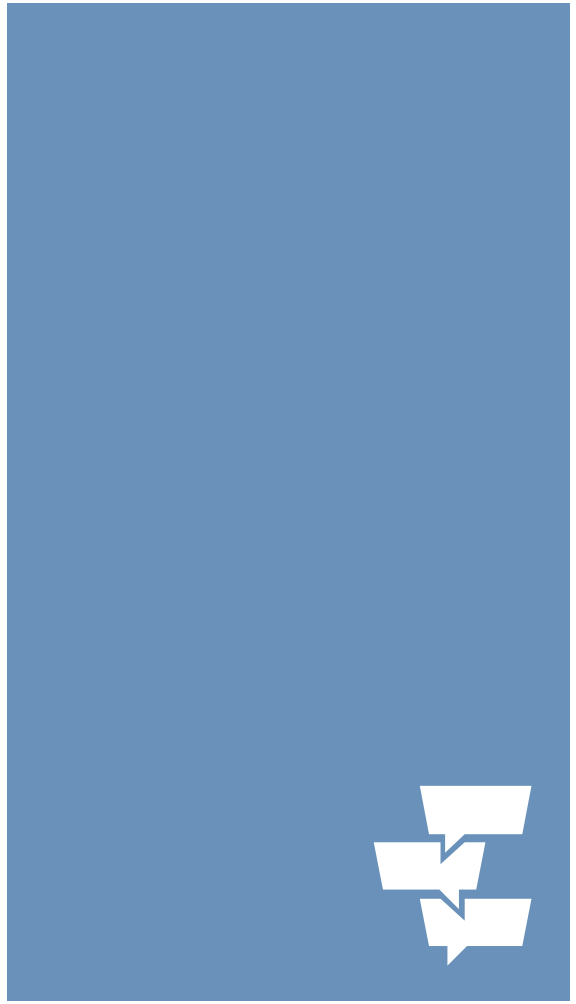
of the Bavarian Academy of Sciences and Humanities

Introduction to the LRZ HPC Infrastructure

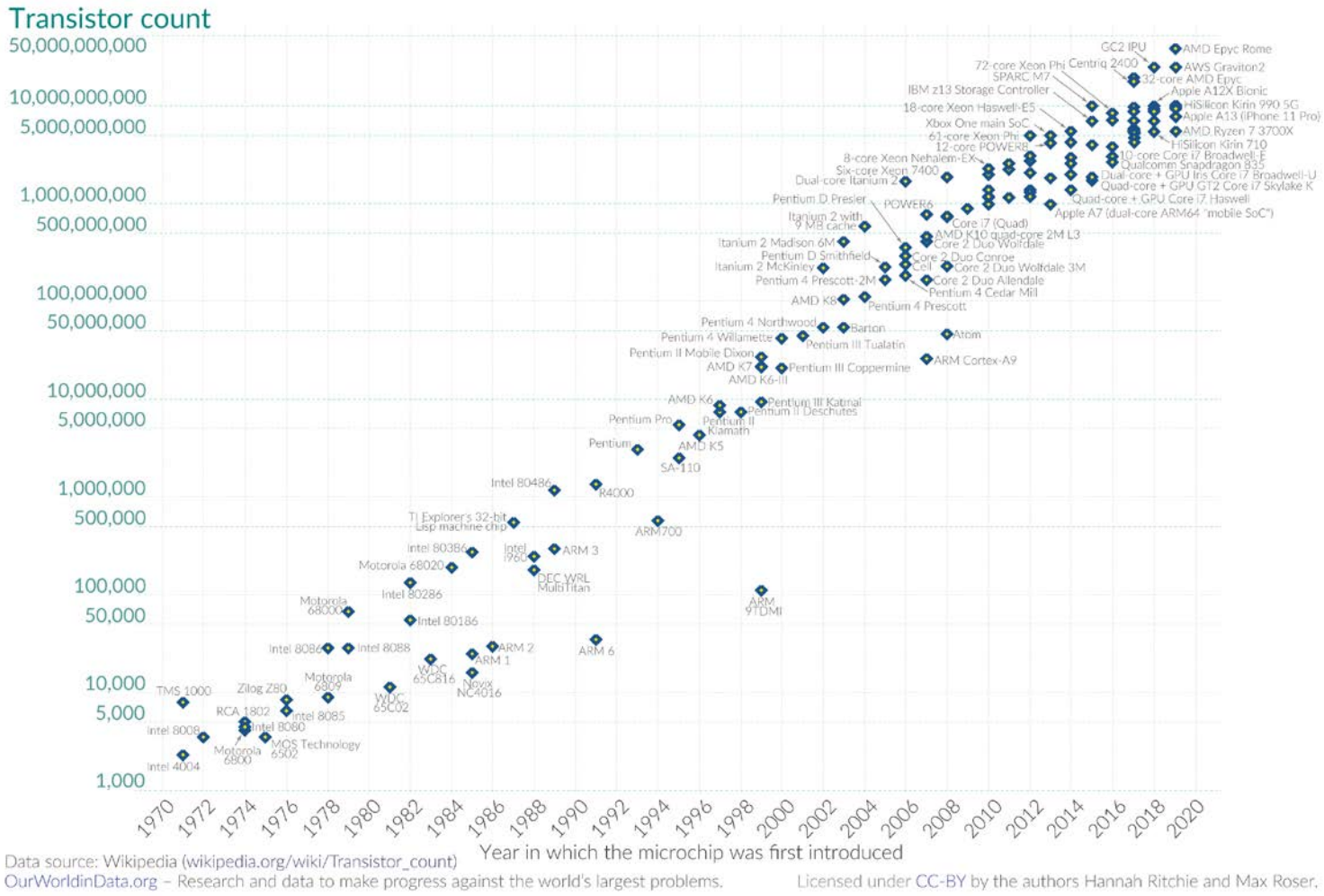
2021-10

Course Information

- The aim of this course is to provide an introduction to the High Performance Computing (HPC) infrastructure of the Leibniz Supercomputing Centre (LRZ)
- You will probably benefit the most if you’re not yet familiar with the LRZ HPC infrastructure, but plan to work on these systems in the future
  -> by the end of this workshop, you should have the basic skills to successfully interact remotely with LRZ HPC systems



Moore’s Law



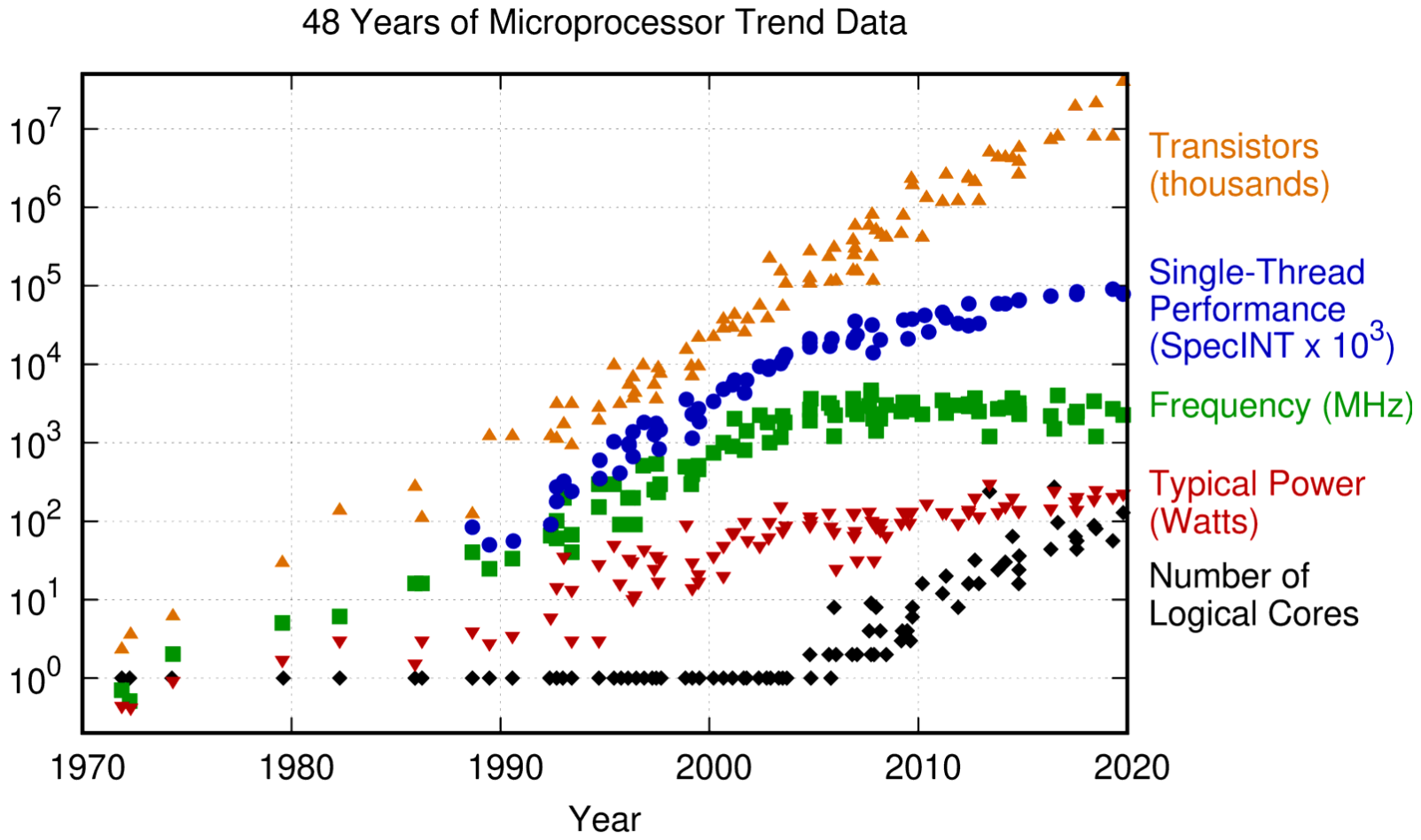
The number of transistors on integrated circuits is doubling about every two years.

(Massively) Parallel Processing

Mid 2000s:
“heat death”

No more faster processors, only more of them.

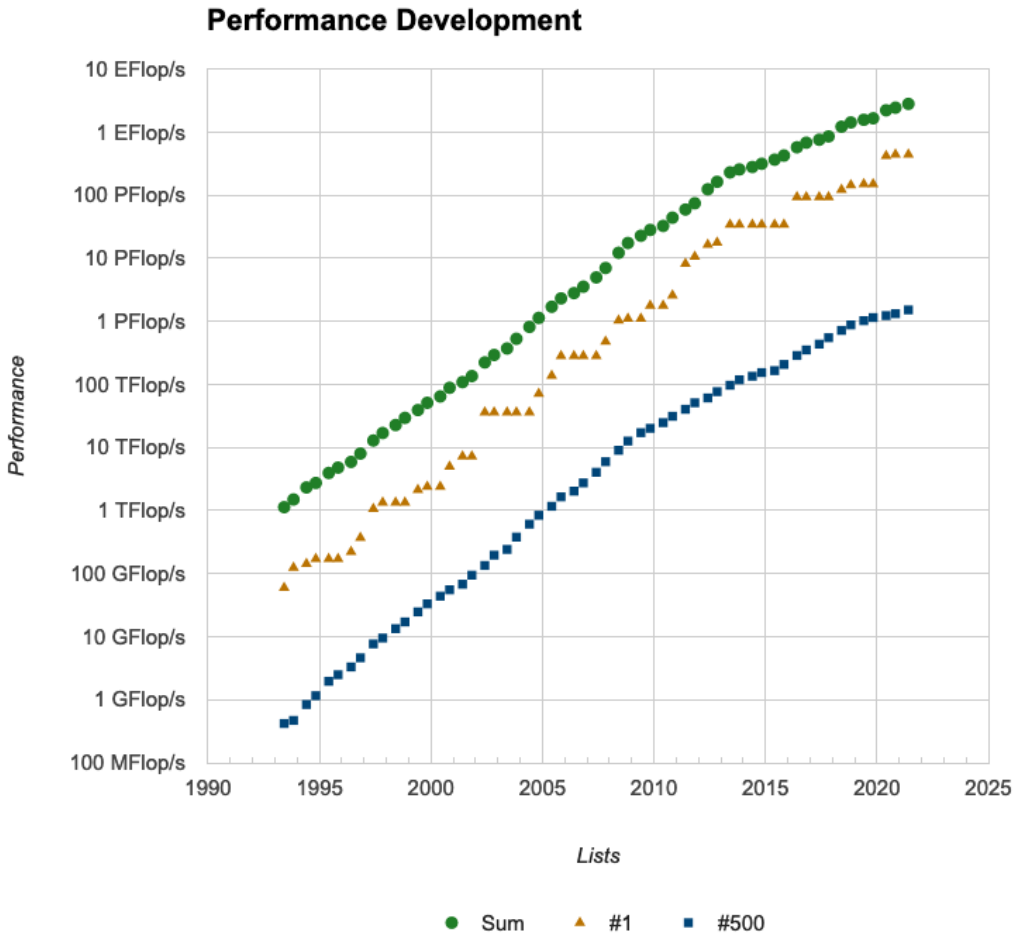
But:
2x3 GHz != 6 GHz



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

https://github.com/karlrupp/microprocessor-trend-data

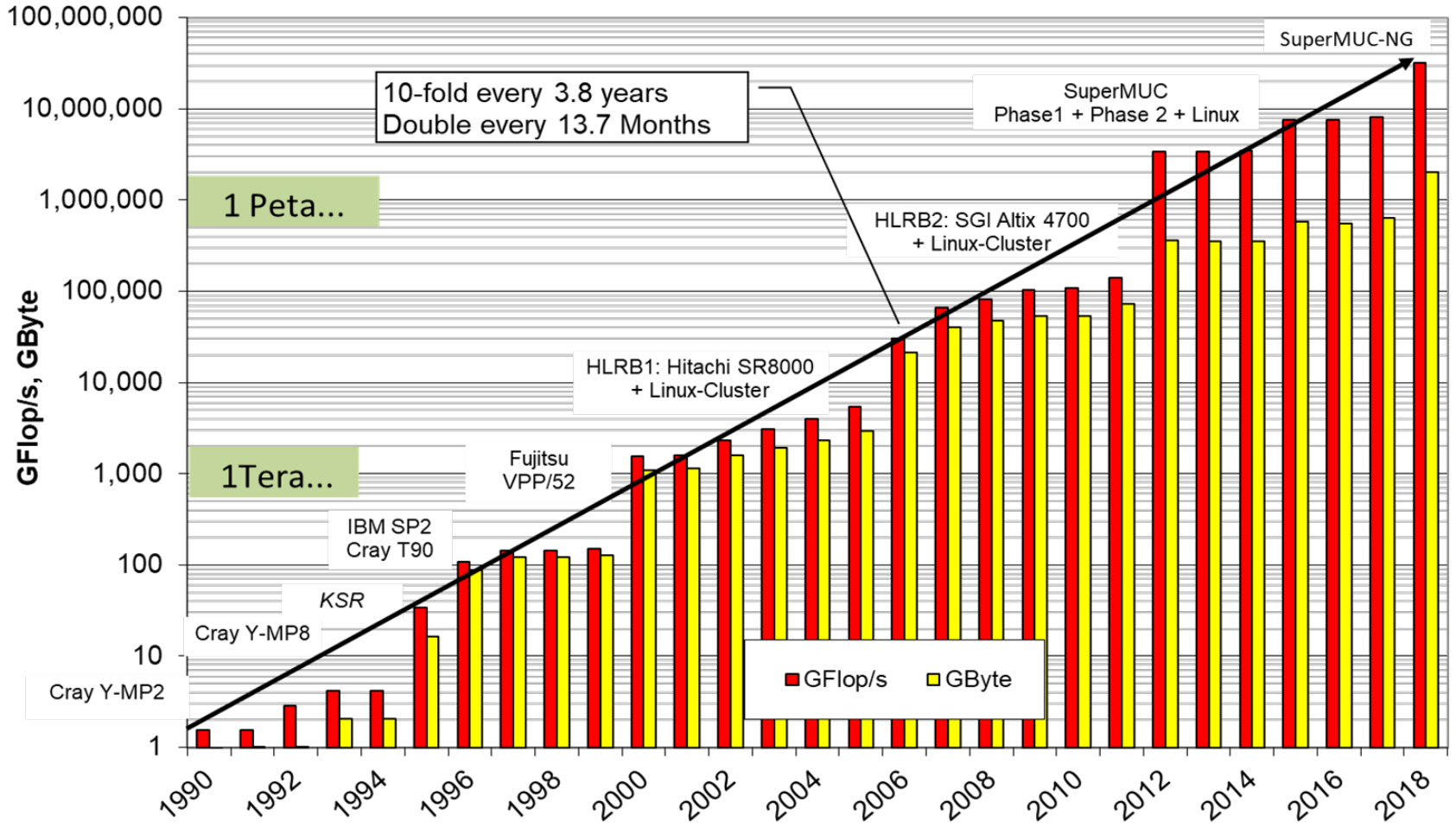
Performance Development



From #1 to #500:
6-8 years

From #500 to Notebook:
8-10 years

Evolution of Peak Performance and Memory



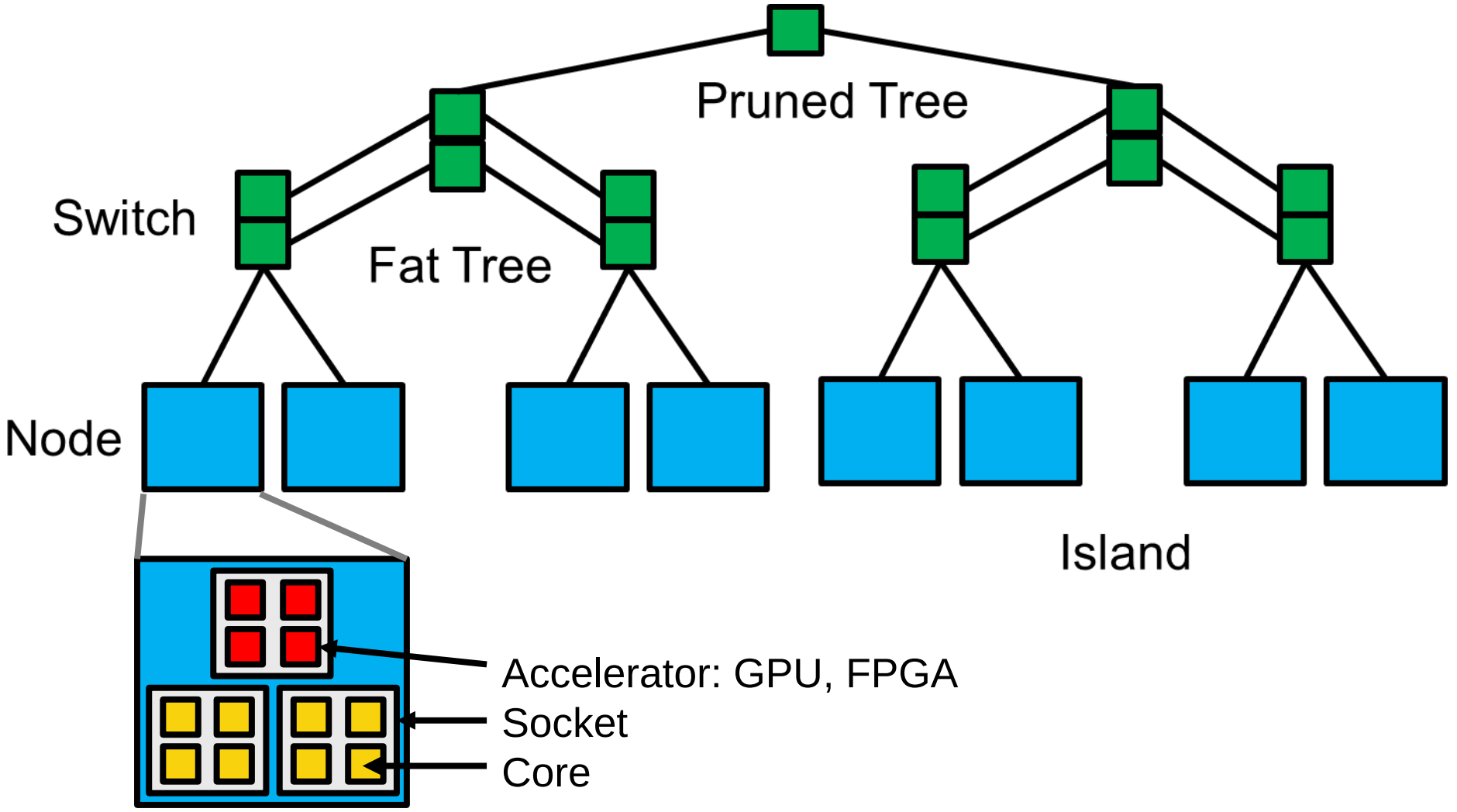
What is a Supercomputer… (not)?

- It has overclocked high-speed processors? **Nope**
- The CPU runs faster than a desktop PC? **Nope**
- It has a large internal memory (RAM)? **Nope (with some exceptions)**
- It runs Microsoft Windows? **Nope**
- It will run my Excel spreadsheet? **Nope**
- It will run my old tried and tested executable? **Probably not**
- It will run my software without changes? **Probably not**
- It will run my program with millions of threads? **Nope**
- It can be used interactively? **Mostly not**

So… what is a Supercomputer?

- It consists of many off-the-shelf (server) CPUs (and, possibly, accelerators like GPUs) with vector instructions (e.g. AVX2, AVX512) in login and compute nodes (as well as service/management nodes)
- All these nodes are connected by a high-speed internal network (interconnect, e.g. InfiniBand, OmniPath)
- They are typically diskless but have access to a parallel file system (e.g. Lustre, GPFS)
- The compute nodes can generally not be accessed directly, but programs have to be submitted to a batch scheduler application (e.g. Slurm) from the login nodes (which are usually accessible by SSH)
- Communication and parallelization is typically relying on the message passing interface standard (MPI) between nodes and the Open Multi-Processing API (OpenMP) on individual nodes
- The operating system is Unix-like, i.e. GNU/Linux

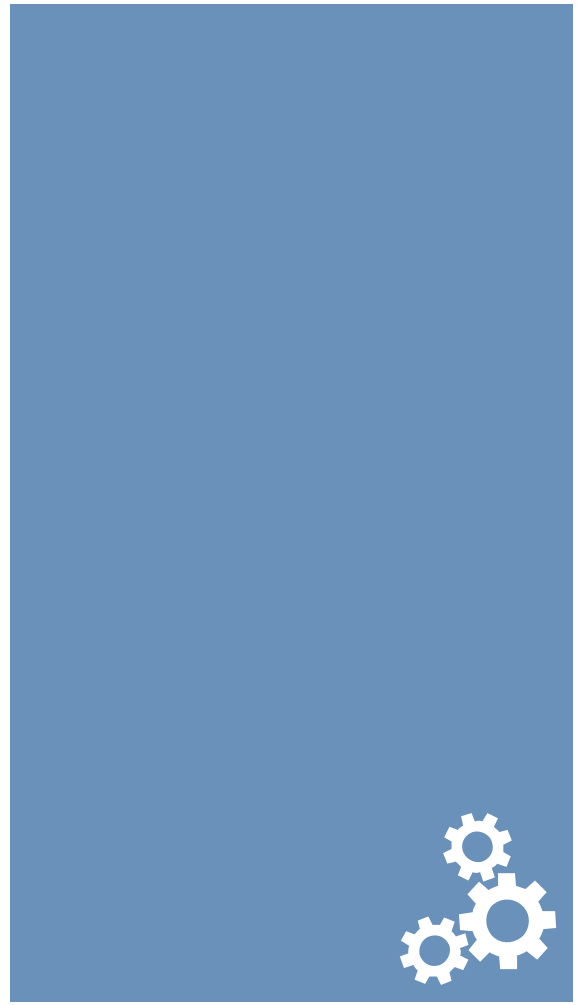
HPC Cluster Systems



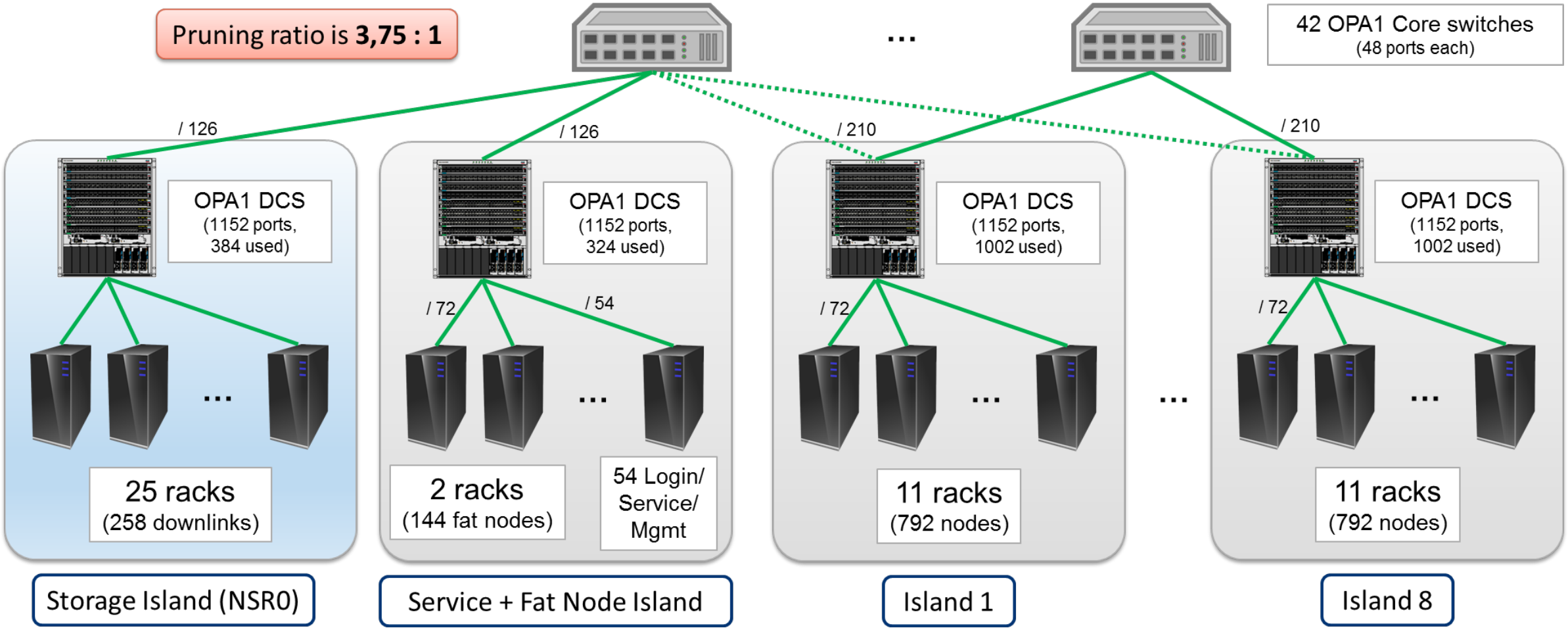
Levels of Parallelism

- Node Level (e.g. SuperMUC-NG has 6480 nodes)
- *Accelerator Level (e.g. a Nvidia DGX A100 has 8 GPUs)*
- Socket Level (e.g. Linux Cluster Teramem has 4 sockets [with 24 cores each])
- Core Level (e.g. Linux Cluster CoolMUC-3 nodes have 64 cores [on a single socket])
- *Thread Level*
  (e.g. Linux Cluster CoolMUC-2 nodes allow 2 threads per core)
- Vector Level (e.g. AVX-512 has 32 512-bit vector registers)

- SuperMUC-NG theoretical peak performance: **26,87 PFlop/s** = **6480** Nodes x **2** Sockets x **24** Cores x **32** Vectors x **2,7** GHz
  (= 26 873 856 000 000 000 Flop/s)



SuperMUC-NG: High Level System Architecture



SuperMUC-NG: Hardware Overview

Name	CPU	Cores/Node	RAM/Node (GB)	Nodes	Cores
SuperMUC-NG Thin Nodes	Intel Xeon ("Skylake")	48	96	6336	304128
SuperMUC-NG Fat Nodes	Intel Xeon ("Skylake")	48	768	144	8912

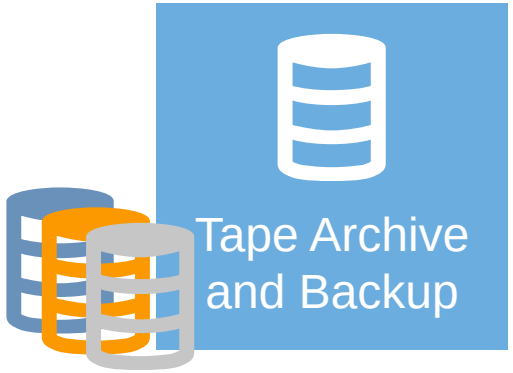
SuperMUC-NG: Access

There are three (well, four) ways to apply for using SuperMUC-NG:

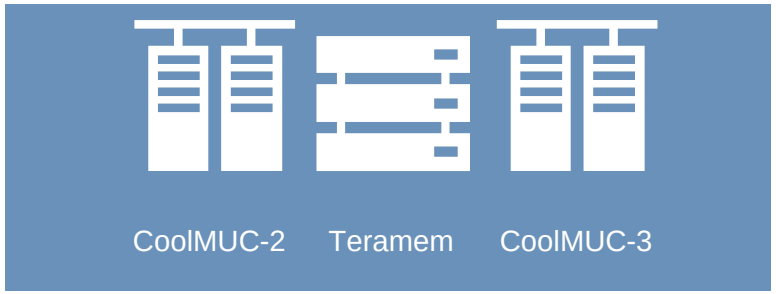
- GCS test project: rolling call, fast review (short abstract), < 300.000 core-h
- GCS regular project: rolling call, technical & scientific review, < 45m core-h
- GCS large scale project: twice per year, technical & scientific review, > 45m core-h
- (PRACE projects for academic users from other European countries)

For further details, see https://doku.lrz.de/x/XAAbAQ

HPC & BDAI Systems for Bavarian Universities



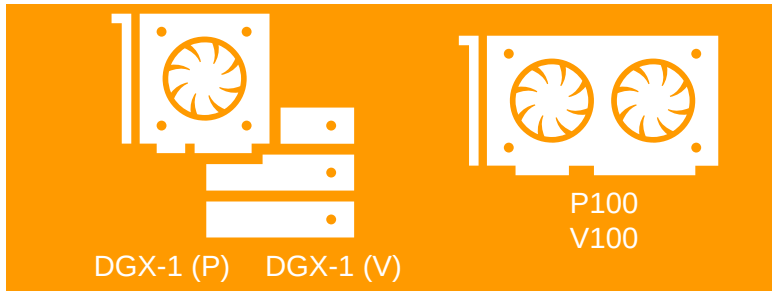
Data Science Storage
(DSS)



lxlogin8.lrz.de

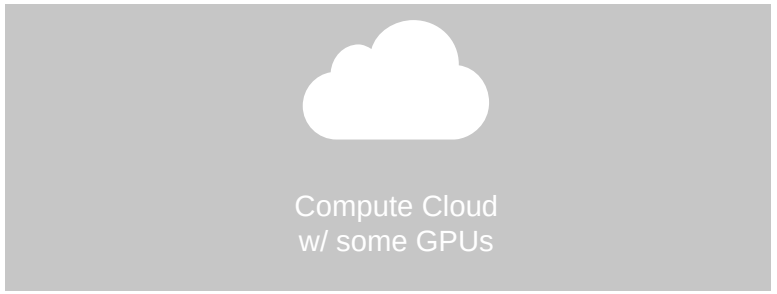
lxlogin[1-4].lrz.de

(https://www.rstudio.lrz.de)



datalab2.srv.lrz.de

https://datalab3.srv.lrz.de



https://cc.lrz.de

Linux Cluster: Hardware Overview

Name	CPU	Cores/Node	RAM/Node (GB)	Nodes (total)	Cores (total)
CoolMUC-2	Intel Xeon E5-2690 v3 ("Haswell")	28	64	812	22736
CoolMUC-3	Intel Xeon Phi (“Knights Landing”)	64	96	148	9472
Teramem	Intel Xeon E7-8890 v4 ("Broadwell")	96	6144	1	96

Linux Cluster: Access

- In order to use LRZ services provided to Bavarian universities, an LRZ account (belonging to an LRZ project) with appropriate permissions is needed.
  While student/staff accounts from LMU and TUM are (to some extent) managed by LRZ, they are restricted to certain services (e.g. E-Mail, Cloud Storage, LRZ Sync+Share) and can not be used to obtain Linux Cluster access.
- Department/institute heads and/or professors/PIs can request new LRZ projects and appoint a master user (or more) for the project. The master user(s) can manage accounts and permissions within these LRZ projects.
- If such an LRZ project already exists for your department (or institute, or research group), contact the master user and ask for an account with Linux Cluster permissions.
  If not, see https://doku.lrz.de/x/BQC7B

Data Storage

- The LRZ HPC Infrastructure is backed by the Data Science Storage (DSS)
  - Long-term storage solution for potentially vast amounts of data
  - Directly connected to the LRZ computing ecosystem
  - Flexible data sharing among LRZ users
  - Web interface for world-wide access and transfer
  - Data sharing with external users (invite per e-mail, access per web interface)
- Disk space and access is managed (as DSS projects and containers) by data curators. This can be LRZ personnel (e.g. Linux Cluster $HOME directories) or PIs/master users/dedicated data curators (e.g. data projects).

Linux Cluster: Data Storage

- $HOME (DSS-backed home directory, managed by LRZ)
  - 100GB per user
  - Automatic backup and snapshots
    (see “/dss/dsshome1/.snapshots/” directory)
  - All your important files/anything you invested a lot of work into should be here

Linux Cluster: Data Storage

- DSS-based project storage
  - Up to 10TB per data project **upon request**, shared among project members
  - Configuration (e.g. exports, backup, quota) to be managed by data curator
  - Use this for e.g. large raw data (and consider backup options)

Linux Cluster: Data Storage

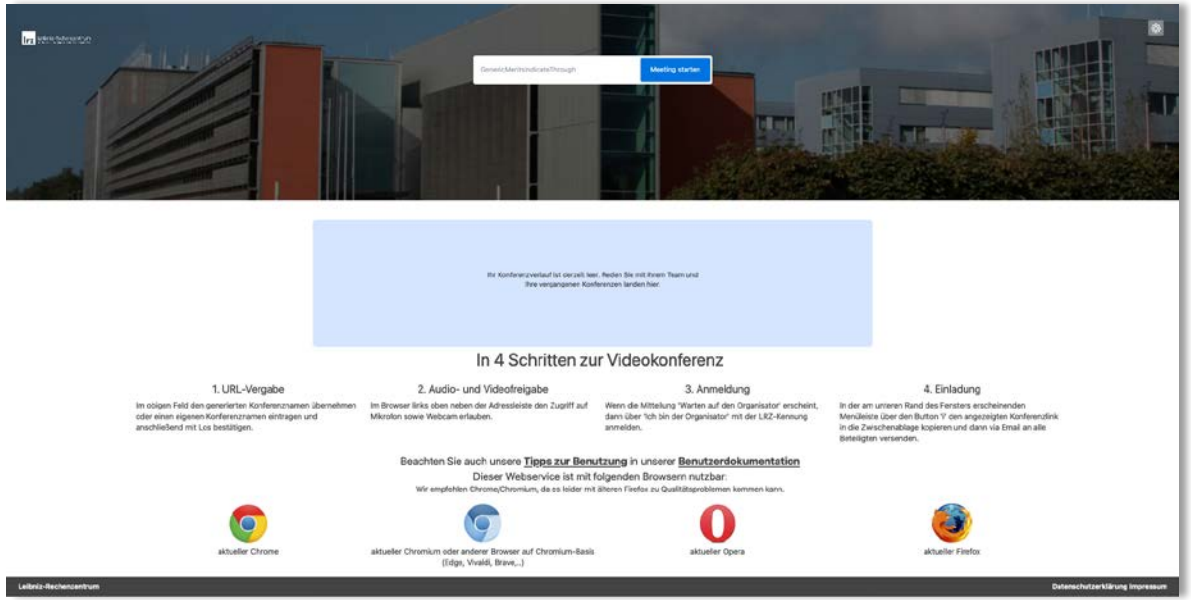
- $SCRATCH (scratch file system, “temporary file system”)
  - 1.4PB, shared among all users
  - No backup (!) and sliding window file deletion, i.e. old files will eventually be deleted (!!) – a data retention time of approx. 30 days may be assumed, but is not guaranteed
  - This is the place for e.g. very large, temporary files or intermediate results, directly feeding into additional analyses
  - Do not save any important data exclusively on this file system!
    Seriously, don’t do it!

Side Note: Collaboration Tools

- Videoconferencing
  https://meet.lrz.de
- Git-repository management, issue tracker, wiki…
  https://gitlab.lrz.de

https://meet.lrz.de

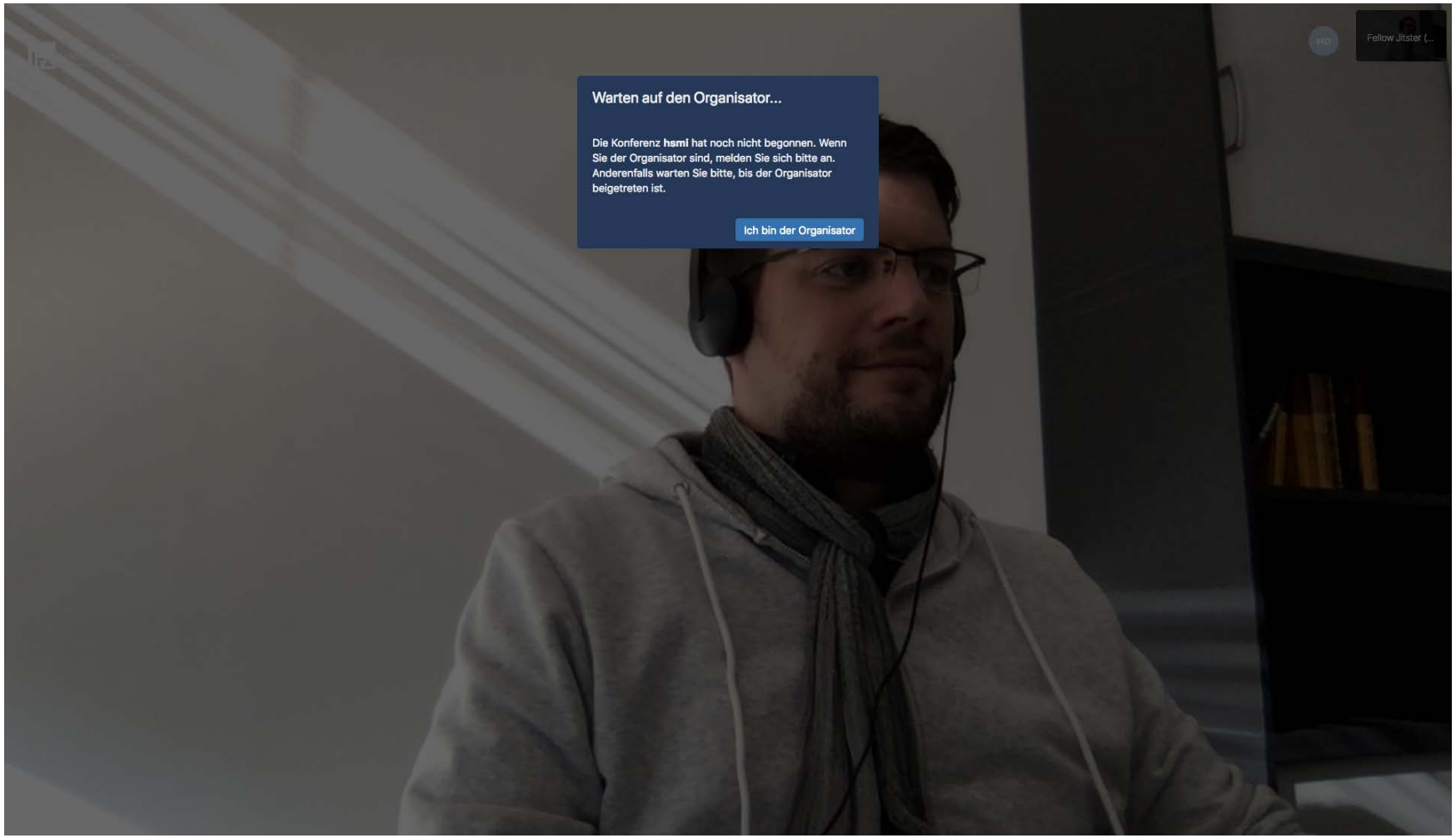
Video Conferencing



- Use a modern browser
  (Safari will likely not work properly)

https://meet.lrz.de

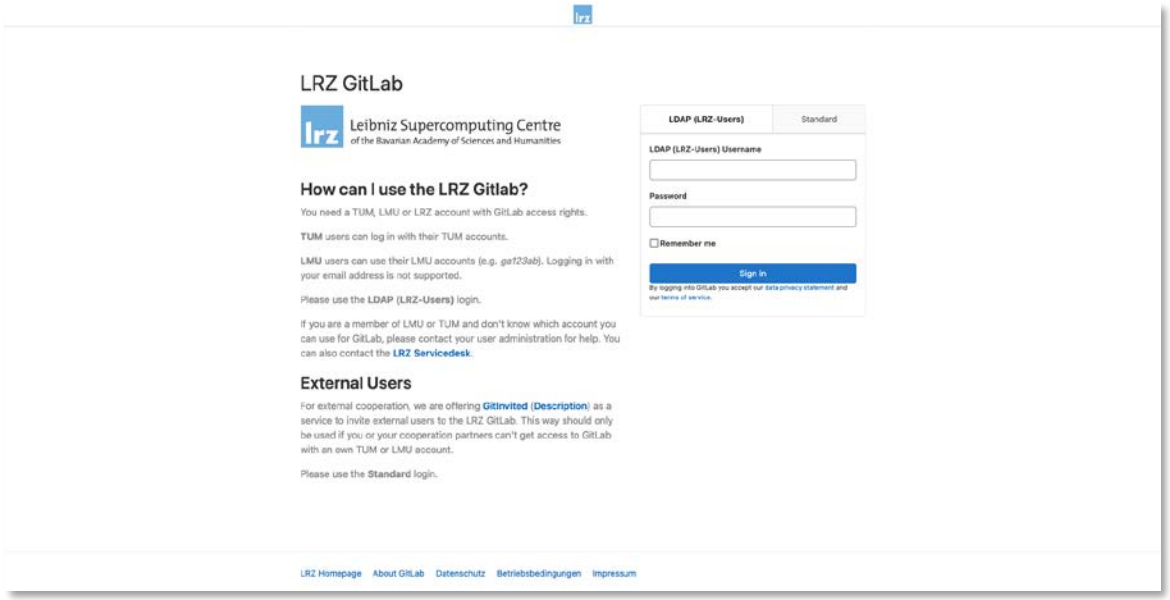
Video Conferencing



First person to join has to sign in using a valid LRZ account

https://gitlab.lrz.de

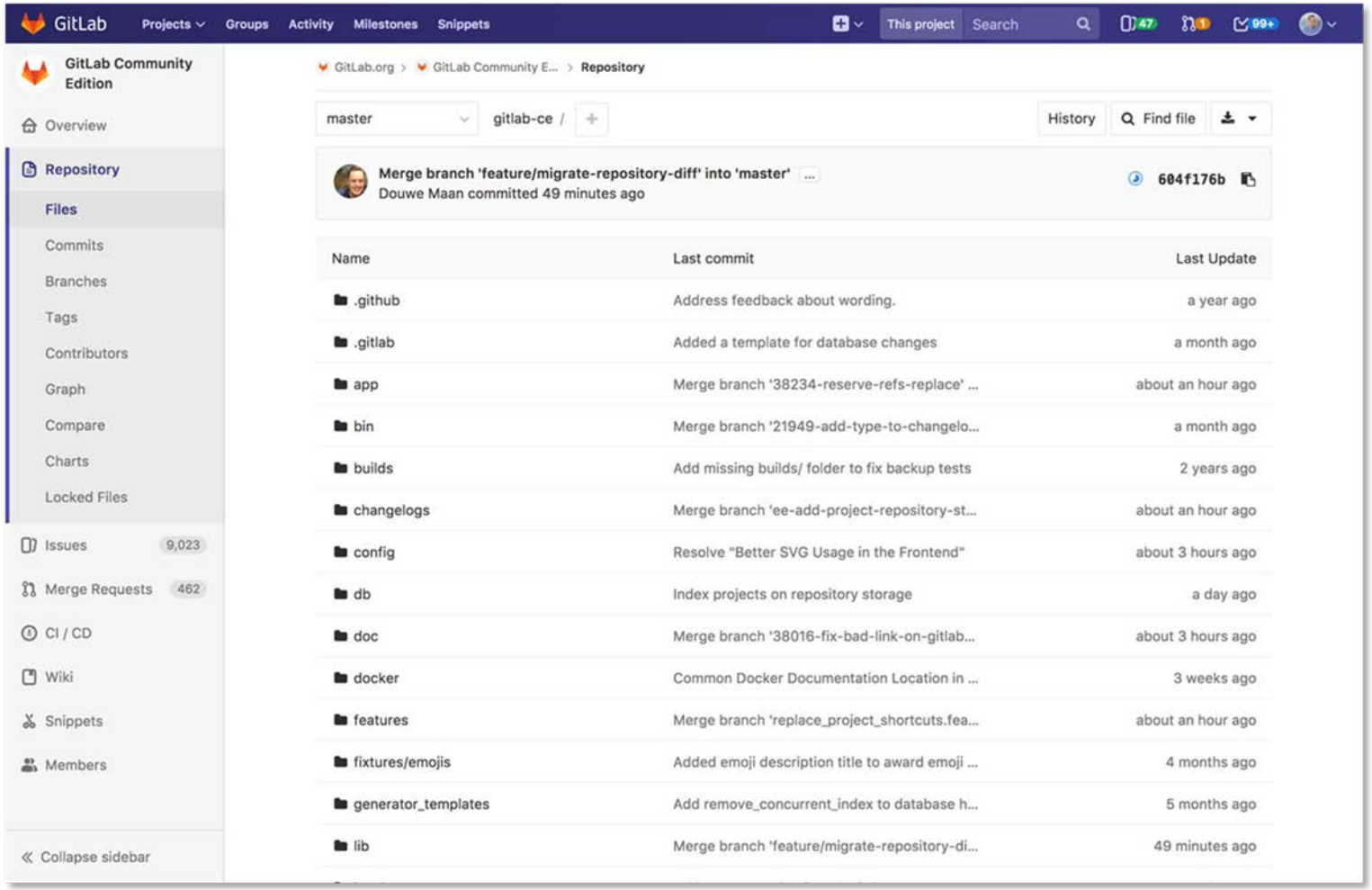
Git Repository Management



- Git repository manager providing wiki, issue-tracking and continuous integration and deployment pipeline features
- Every LRZ project member can invite up to 20 external users to the service (GitInvited), this is great for collaborations!

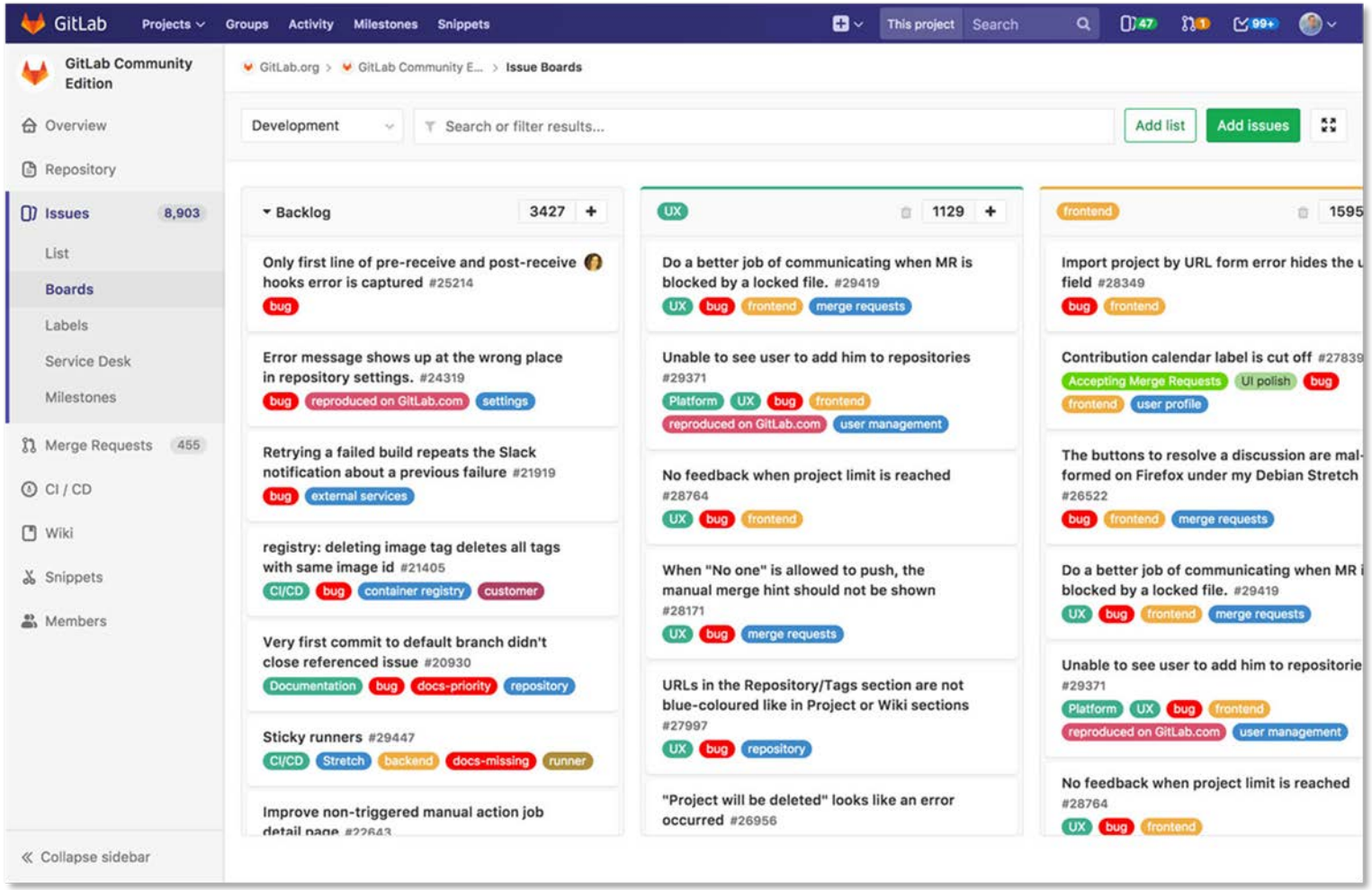
https://gitlab.lrz.de

Git Repository Management



https://gitlab.lrz.de

Git Repository Management



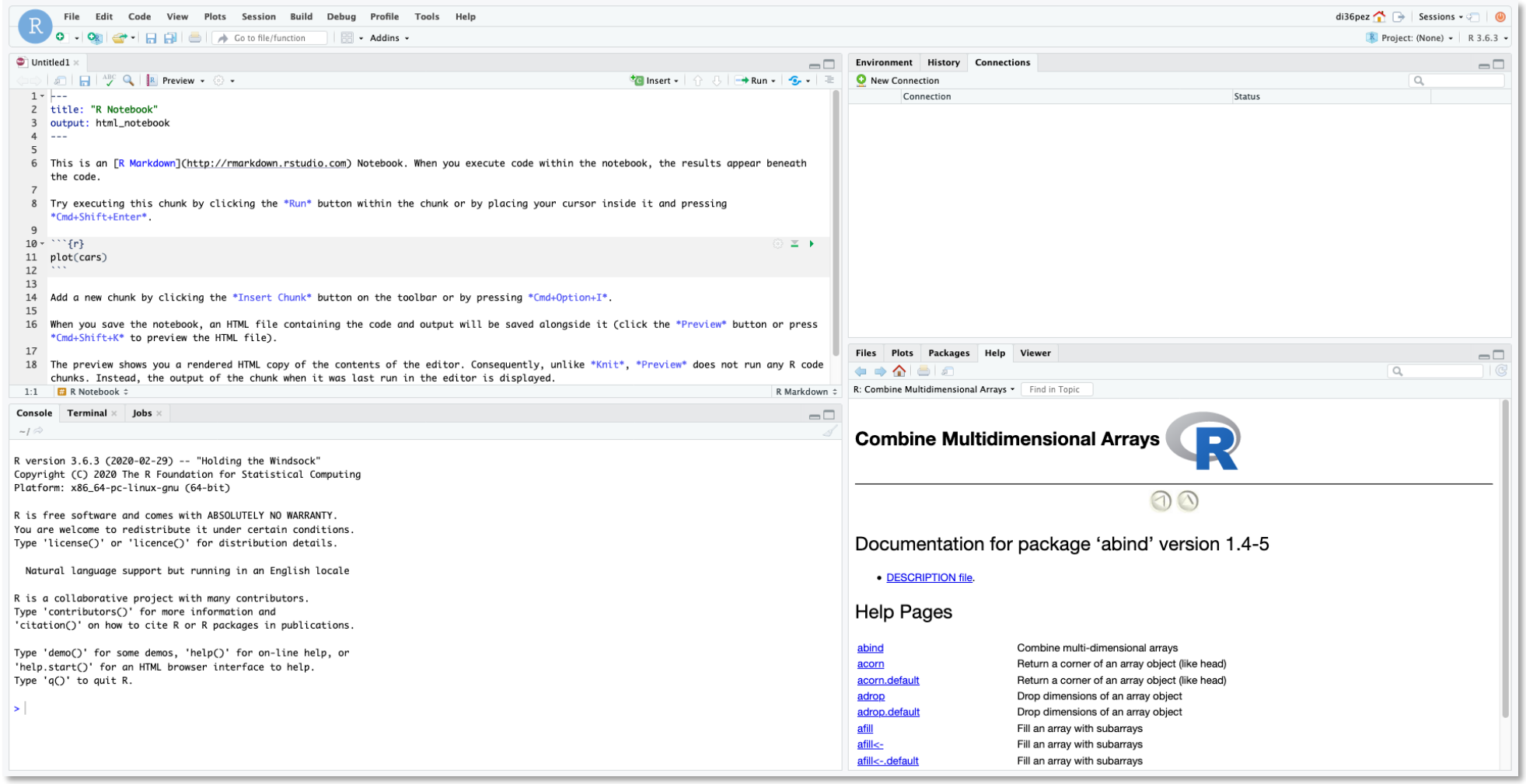
https://www.rstudio.lrz.de

RStudio Server

- Web-based RStudio frontend
- Cluster of multiple nodes, each with
  - 40 cores (please do not use more than 20 individually!) and
  - 360 GB RAM
- Integrates with the Linux Cluster:
  - Directly access the data in your DSS-backed Linux Cluster home directory ($HOME)
  - Allows to access any DSS-based storage container
    (NFS-export has to be set up by data curator)
  - Use the built-in Terminal to submit jobs to the Linux Cluster's batch queues via the Slurm Workload Manager
- For further details, see https://doku.lrz.de/x/zQWVAg
- **EOL: 30. November 2021 – use LRZ AI Systems instead!**
  Contact us if you encounter any (migration) issues!

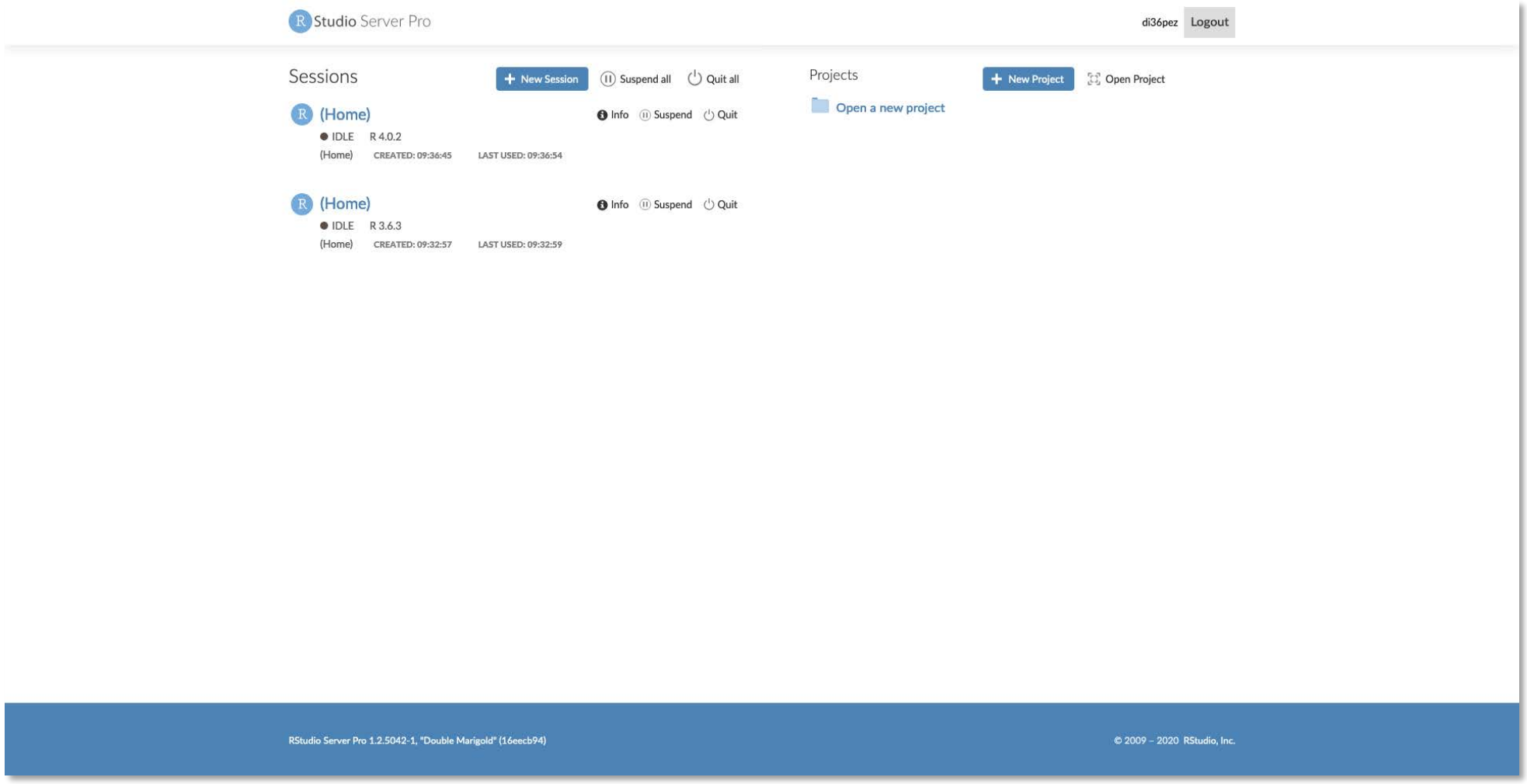
https://www.rstudio.lrz.de

RStudio Server



https://www.rstudio.lrz.de

RStudio Server



https://www.rstudio.lrz.de

RStudio Server

- Let’s get started:
  Log in to the RStudio Server and get familiar with the interface
- Create a new folder in your home directory
  (bonus points for using the built-in terminal)

https://www.rstudio.lrz.de

RStudio Server

- Let’s get started:
  Log in to the RStudio Server and get familiar with the interface
- Create a new folder in your home directory
  (bonus points for using the built-in terminal)
- Create a new text file, write down something nice and save it to the newly created folder
- Make sure you can locate the folder/file in the file system
  (using the Files pane and/or the terminal)
- Extra credit: what is the full path to your files on the file system?

Linux Cluster

- Let’s get started:
  Connect to the CoolMUC-2 segment of the Linux Cluster

- From a terminal application:
  `$ ssh <user>@lxlogin1.lrz.de`

Linux Cluster – CoolMUC-2

```
user@localhost:~$ ssh user@lxlogin1.lrz.de
```

- For CoolMUC-2 you can use the login nodes
  lxlogin1.lrz.de or
  lxlogin2.lrz.de or
  lxlogin3.lrz.de or
  lxlogin4.lrz.de
- (see https://doku.lrz.de/x/AAaVAg for all cluster segments/systems)

Linux Cluster – CoolMUC-2

```
user@localhost:~$ ssh user@lxlogin1.lrz.de
The authenticity of host 'lxlogin1.lrz.de (129.187.20.101)' can't be established.
ECDSA key fingerprint is SHA256:Q2NG5ofc7v/eW1kZYXcEUU69T3ESoIUkY9bITwNKJ5g.
Are you sure you want to continue connecting (yes/no)?
```

- The first time you connect to a (new) system, this message is expected, as the public-key of the remote system is not yet known to your local system, see https://superuser.com/questions/421997/what-is-a-ssh-key-fingerprint-and-how-is-it-generated for details.
- Information about the public keys of the LRZ Linux Cluster can be found here: https://doku.lrz.de/x/AAaVAg
- Type “yes” to import the public key locally and to continue.

Linux Cluster – CoolMUC-2

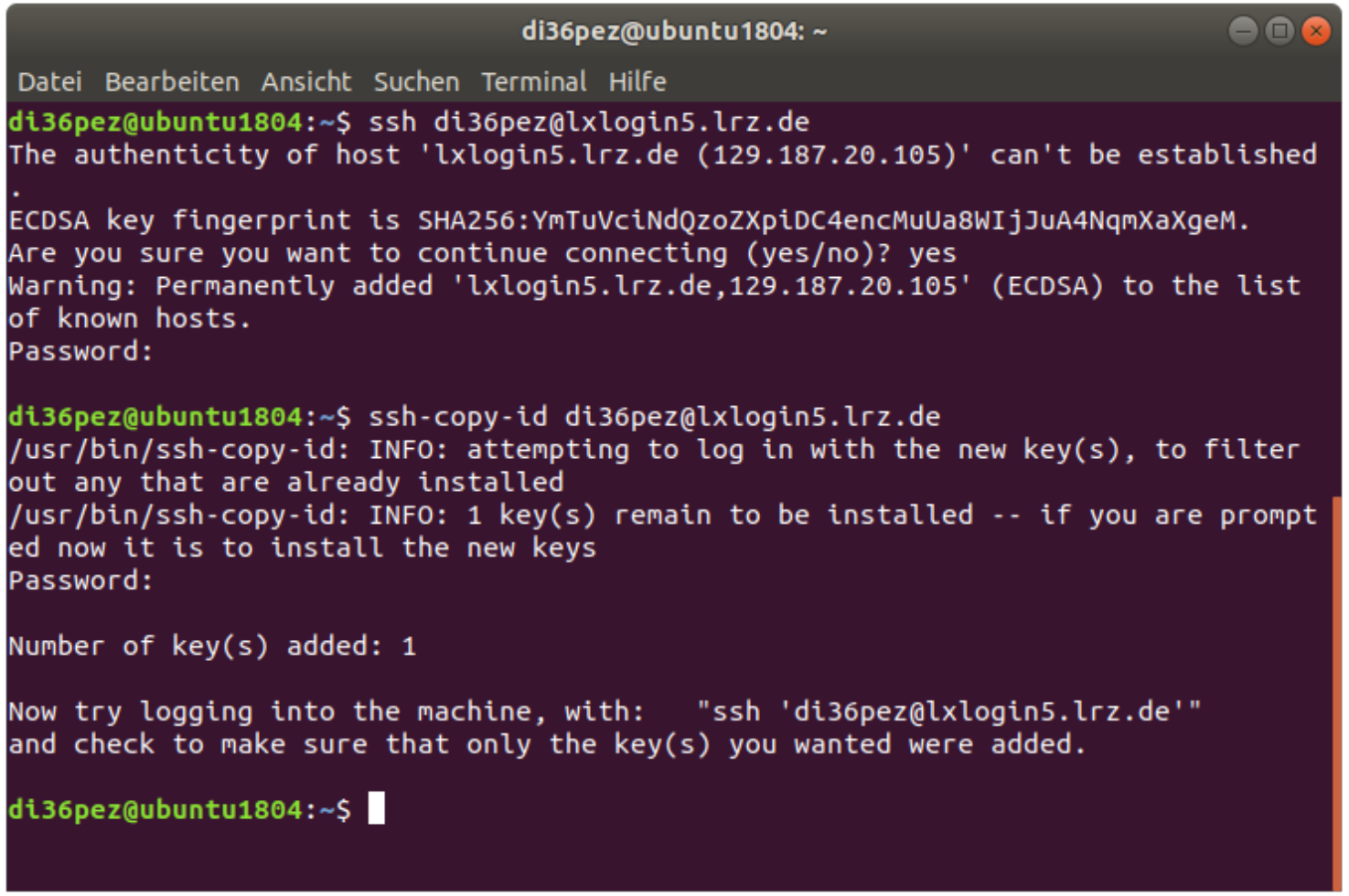
```
user@localhost:~$ ssh user@lxlogin1.lrz.de
The authenticity of host 'lxlogin1.lrz.de (129.187.20.101)' can't be established.
ECDSA key fingerprint is SHA256:Q2NG5ofc7v/eW1kZYXcEUU69T3ESoIUkY9bITwNKJ5g.
Are you sure you want to continue connecting (yes/no)?
Warning: Permanently added 'lxlogin1.lrz.de' (ECDSA) to the list of known hosts.
Password:
```

- You can now continue by typing your password.
  As this may become a repetitive burden, you may choose to do the following instead…
- (Interrupt the password prompt by pressing `Ctrl+C`)

Linux Cluster – CoolMUC-2

- Using a local terminal, add your SSH public key (not the private one!) to the authorized keys on the Linux Cluster!
- On Linux, this is the content of ~/.ssh/<your_key>.pub on your local machine…
- … which should go into ~/.ssh/authorized_keys on the login node
- Use the command `ssh-copy-id <user>@lxlogin1.lrz.de` on Linux,
  you may have to do it differently/manually on macOS and Windows (but only once)

Linux Cluster – CoolMUC-2



Demo content –

use the current login nodes instead!

Linux Cluster – CoolMUC-2

- Connect to the cluster again, it should now work without password! Crikey!
- You can use the `logout` or `exit` commands to close the connection

Linux Cluster – CoolMUC-2

```
user@localhost:~$ ssh user@lxlogin1.lrz.de
Last login: Wed Sep 22 13:08:36 2021 from SOME.IP.ADDRESS
  Welcome to the CoolMUC2 Infiniband cluster, one of the Linux cluster systems
  operated by Leibniz Supercomputing Centre (LRZ).

  Please do not run any extensive computational programs on login nodes.
  Instead, please submit SLURM batch scripts for production jobs, and SLURM
  interactive shells for testing and short-running programs.
  Misuse of the interactive resources will lead to violating accounts being
  blocked from access to the cluster.
  !!! Please note in particular this pertains to specifying invalid !!!
  !!! mail addresses in SLURM scripts, please read                  !!!
  https://doku.lrz.de/display/PUBLIC/Available+SLURM+clusters+and+features
------------------------------------------------------------------------------
  Documentation:  https://doku.lrz.de/display/PUBLIC/Linux+Cluster
  Messages/System Status:
                  https://doku.lrz.de/display/PUBLIC/High+Performance+Computing
------------------------------------------------------------------------------
spack/release/21.1.1 21.1.1 release linux-sles15
intel-mpi: using intel wrappers for mpicc, mpif77, etc

user@cm2login1:~$
```

Linux Cluster – CoolMUC-2

```
user@localhost:~$ ssh user@lxlogin1.lrz.de
Last login: Wed Sep 22 13:08:36 2021 from SOME.IP.ADDRESS
  Welcome to the CoolMUC2 Infiniband cluster, one of the Linux cluster
  systems operated by Leibniz Supercomputing Centre (LRZ).


  Please do not run any extensive computational programs on login nodes.
  Instead, please submit SLURM batch scripts for production jobs, and
  SLURM interactive shells for testing and short-running programs.
  Misuse of the interactive resources will lead to violating accounts
  being blocked from access to the cluster.
  !!! Please note in particular this pertains to specifying invalid !!!
  !!! mail addresses in SLURM scripts, please read                   !!!

https://doku.lrz.de/display/PUBLIC/Available+SLURM+clusters+and+features
------------------------------------------------------------------------
  Documentation:  https://doku.lrz.de/display/PUBLIC/Linux+Cluster
  Messages/System Status:
  https://doku.lrz.de/display/PUBLIC/High+Performance+Computing
------------------------------------------------------------------------
spack/release/21.1.1 21.1.1 release linux-sles15
intel-mpi: using intel wrappers for mpicc, mpif77, etc


user@cm2login1:~$
```

- This is the message of the day provided by the system administrators
- Take note of it, it may contain important information about the system status, scheduled maintenances, etc. …

Linux Cluster – CoolMUC-2

- Get your bearings… where did you end up on the file system?
  (Hint: `$ pwd`)
- Can you locate the folder and file created earlier (with RStudio Server)?

Linux Cluster – CoolMUC-2

- Take a look at the contents of your file
  `$ cat my_file`
- Use `dssusrinfo all` to query DSS-containers available to you
- Look at and explore $HOME (this is a environment variable):
  `$ echo $HOME`

Environment Modules

- Modules allow for the dynamic modification of environment variables,
  e.g. they provide a flexible way to access various applications and libraries available on the system
- List the currently active modules (loaded by default):
  ```
  $ module list
  ```

Environment Modules

```
user@cm2login1:~$ module list
Currently Loaded Modulefiles:
 1) admin/1.0   2) tempdir/1.0   3) lrz/1.0   4) spack/21.1.1   5) intel/19.0.5
 6) intel-mkl/2019   7) intel-mpi/2019-intel
user@cm2login1:~$
```

Get more info about e.g. the lrz module:

```
$ module show lrz
```

Environment Modules

```
user@cm2login1:~$ module show lrz
-------------------------------------------------------------------
/lrz/sys/share/modules/files_sles15/environment/lrz/1.0:

module-whatis   {Environment: Default setup for all LRZ users}
setenv          LRZ_ARCH x86_64_intel
append-path     PATH /lrz/sys/tools/slurm_utils/bin
setenv          SALLOC_PARTITION cm2_inter
setenv          LRZ_OS {SUSE Linux Enterprise Server 15 SP1}
setenv          LRZ_OS_VER 15
setenv          LRZ_OS_SUBVER 1
setenv          LRZ_NOCHECK yes
setenv          INTEL_LICENSE_FILE /lrz/sys/intel/licenses
setenv          LRZ_LICENSE {open source - no access restrictions}
-------------------------------------------------------------------
```

Environment Modules

- Suppose you need to use MATLAB
- It is not generally available (try `$ which matlab`)
- … or is it?
  Search for available modules:
  `$ module available matlab` or
  `$ module av matlab`

Environment Modules

```
user@cm2login1:~$ which matlab
which: no matlab in (/lrz/sys/tools/intel-mpi-wrappers/bin/:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel-mpi/2019.8.254-gcc-vyzek4m/compilers_and_libraries_2020.2.254/linux/mpi/intel64/libfabric/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel-mpi/2019.8.254-gcc-vyzek4m/compilers_and_libraries_2020.2.254/linux/mpi/intel64/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel-mpi/2019.8.254-gcc-vyzek4m/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel-mkl/2019.5.281-gcc-2gmjfxz/mkl/tools:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel-mkl/2019.5.281-gcc-2gmjfxz/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel/19.0.5-gcc-uglchea/compilers_and_libraries_2019.5.281/linux/bin/intel64:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel/19.0.5-gcc-uglchea/compilers_and_libraries_2019.5.281/linux/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel/19.0.5-gcc-uglchea/compilers_and_libraries_2019.5.281/linux/mpi/intel64/libfabric/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel/19.0.5-gcc-uglchea/compilers_and_libraries_2019.5.281/linux/mpi/intel64/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel/19.0.5-gcc-uglchea/debugger_2019/gdb/intel64/bin:/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/intel/19.0.5-gcc-uglchea/bin:/lrz/sys/tools/modules/4.6.1/bin:/lrz/sys/bin:/dss/dsshome1/06/di36pez/bin:/usr/local/bin:/usr/bin:/bin:/lrz/sys/tools/slurm_utils/bin)
user@cm2login1:~$ module av matlab
------- /lrz/sys/spack/release/21.1.1/modules/x86_64/linux-sles15-x86_64 -------
matlab-mcr/R2020a_Update5-generic  matlab/R2020a_Update5-generic
matlab-mcr/R2020b_Update3-generic  matlab/R2020b_Update5-generic
matlab-mcr/R2021a-generic          matlab/R2021a-generic

--------------- /lrz/sys/share/modules/files_sles15/applications ---------------
matlab-inter/coolmuc-2  matlab-inter/coolmuc-3
```

Environment Modules

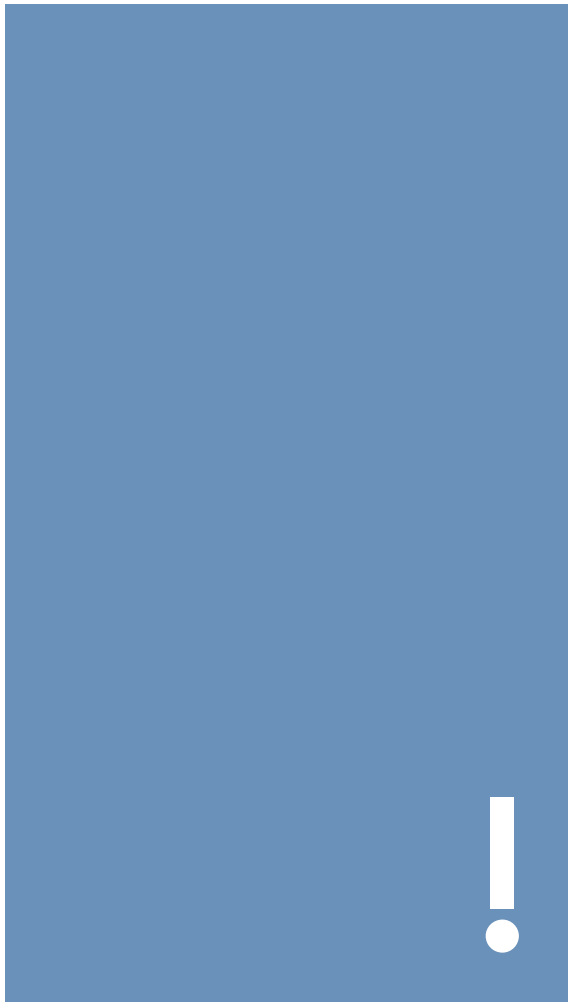
- Look at all these options…!
- Most modules are maintained using the Spack package manager, i.e. always prioritize modules in the “/lrz/sys/spack/…” path!
- Load any module you like, e.g. the latest MATLAB version:
  ```
  $ module load matlab/R2021a-generic
  ```

Environment Modules

```
user@cm2login1:~$ module load matlab/R2021a-generic

  WARNING: Please note that the dynamic loader is overloaded by this
           MATLAB environment module!

           Please note further that the setting of the KMP_AFFINITY
           environment variable is also modified by MATLAB environment
           module! This may have negative impact on the performance and
           functionality of other OpenMP based programs.

           Use a different shell to start programs other than MATLAB,
           otherwise those programs may not function properly.

user@cm2login1:~$ which matlab
/dss/dsshome1/lrz/sys/spack/release/21.1.1/opt/x86_64/matlab/R2021a-gcc-cbij4ux/bin/matlab
```

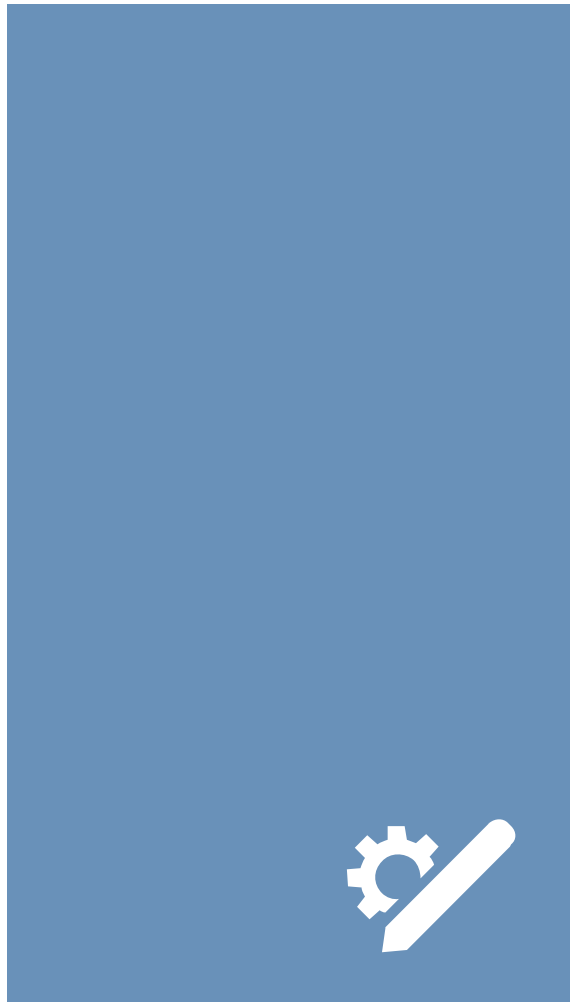
Environment Modules

- Modules can/should be unloaded when you don‘t need them anymore (e.g. before trying another version):
  `$ module unload <module/version>`
- Loading modules is not persistent across sessions,
  i.e. once you log out and back in again, only the default modules will be loaded!
- For further documentation, see
  https://modules.readthedocs.io/en/latest/module.html



Slurm Workload Manager

- Slurm is a job scheduler:
  - Allocates access to resources (time, memory, nodes/cores)
  - Provides framework for starting, executing, and monitoring work
  - Manages queue of pending jobs (enforcing “fair share” policy)
- Use the `sinfo` command to get information about the available clusters
  `$ sinfo --clusters=all` or, shortened:
  `$ sinfo –M all`



Slurm Workload Manager

```
user@cm2login1:~$ sinfo -M all -s
CLUSTER: biohpc_gen
PARTITION             AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
biohpc_gen_highmem       up 21-00:00:0         6/7/0/13  hlegr5vc[01-
02],hlegr1409n[02-12]
biohpc_gen_production    up 14-00:00:0         6/7/0/13  hlegr5vc[01-
02],hlegr1409n[02-12]
biohpc_gen_normal        up 2-00:00:00         6/7/0/13  hlegr5vc[01-
02],hlegr1409n[02-12]
biohpc_gen_inter*        up   12:00:00         6/7/0/13  hlegr5vc[01-
02],hlegr1409n[02-12]

CLUSTER: bsbslurm
PARTITION    AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
bsb_konvert*    up   infinite          6/0/0/6  i22r07c01s[01-06]

CLUSTER: c2pap
PARTITION      AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
c2pap_parallel    up 2-00:00:00      25/94/1/120  i23r07c01s[01-
12],i23r07c02s[01-12],i23r07c03s[01-12],i23r07c04s[01-12],i23r07c05s[01-
12],i23r08c01s[01-12],i23r08c02s[01-12],i23r08c03s[01-12],i23r08c04s[01-
12],i23r08c05s[01-12]
c2pap_serial*     up 2-00:00:00      25/94/1/120  i23r07c01s[01-
12],i23r07c02s[01-12],i23r07c03s[01-12],i23r07c04s[01-12],i23r07c05s[01-
12],i23r08c01s[01-12],i23r08c02s[01-12],i23r08c03s[01-12],i23r08c04s[01-
12],i23r08c05s[01-12]
```

- Look for the cluster segments
  - inter (allows for interactive usage)
  - cm2 (the main CoolMUC-2 cluster)
  - serial (shared nodes for serial jobs)
- What is their current status?
- Get information about a specific cluster segment, e.g.
  `$ sinfo -M inter` or
  `$ sinfo -M cm2`

CoolMUC-2 Overview

Slurm cluster	Slurm partition	Node range per Job	Slurm job settings

cm2	cm2_large	25-64	`--clusters=cm2`
			`--partition=cm2_large`
			`--qos=cm2_large`

	cm2_std	3-24	`--clusters=cm2`
			`--partition=cm2_std`
			`--qos=cm2_std`

cm2_tiny	cm2_tiny	1-4	`--clusters=cm2_tiny`

serial	serial_std	1	`--clusters=serial`
			`--partition=serial_std`
			`--mem=<memory_per_node>MB`

	serial_long	1	`--clusters=serial`
			`--partition=serial_long`
			`--mem=<memory_per_node>MB`

inter	cm2_inter	1-4	`--clusters=inter` `--partition=cm2_inter`
	teramem_inter	1	`--clusters=inter` `--partition=teramem_inter`

For additional details see https://doku.lrz.de/display/PUBLIC/Job+Processing+on+the+Linux-Cluster

Interactive Allocation

```
usercm2login1:~$ sinfo -M inter -s
CLUSTER: inter
PARTITION     AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
mpp3_inter*      up    2:00:00          1/2/0/3  mpp3r03c05s[01-03]
teramem_inter    up 10-00:00:0          0/0/1/1  teramem1
cm2_inter        up    2:00:00         5/6/1/12  i22r07c05s[01-12]

user@cm2login1:~$ sinfo -M cm2 -s
CLUSTER: cm2
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
cm2_std      up 3-00:00:00    275/120/9/404  i22r01c01s[01-12],i22r01c02s[01-
12],i22r01c03s[01-12],i22r01c04s[01-12],i22r01c05s[01-12],i22r01c06s[01-12],i22r02c01s[01-
12],i22r02c02s[01-12],i22r02c03s[01-12],i22r02c04s[01-12],i22r02c05s[01-12],i22r02c06s[01-
12],i22r03c01s[01-12],i22r03c02s[01-12],i22r03c03s[01-12],i22r03c04s[01-12],i22r03c05s[01-
12],i22r03c06s[01-12],i22r04c01s[01-12],i22r04c02s[01-12],i22r04c03s[01-12],i22r04c04s[01-
12],i22r04c05s[01-12],i22r05c01s[01-12],i22r05c02s[01-12],i22r05c03s[01-12],i22r05c04s[01-
12],i22r05c05s[01-12],i22r06c01s[01-12],i22r06c02s[01-12],i22r06c03s[01-12],i22r06c04s[01-
12],i22r06c05s[01-12],i22r07c02s[11-12],i22r07c04s[07-12]
cm2_large    up 2-00:00:00    275/120/9/404  i22r01c01s[01-12],i22r01c02s[01-
12],i22r01c03s[01-12],i22r01c04s[01-12],i22r01c05s[01-12],i22r01c06s[01-12],i22r02c01s[01-
12],i22r02c02s[01-12],i22r02c03s[01-12],i22r02c04s[01-12],i22r02c05s[01-12],i22r02c06s[01-
12],i22r03c01s[01-12],i22r03c02s[01-12],i22r03c03s[01-12],i22r03c04s[01-12],i22r03c05s[01-
12],i22r03c06s[01-12],i22r04c01s[01-12],i22r04c02s[01-12],i22r04c03s[01-12],i22r04c04s[01-
12],i22r04c05s[01-12],i22r05c01s[01-12],i22r05c02s[01-12],i22r05c03s[01-12],i22r05c04s[01-
12],i22r05c05s[01-12],i22r06c01s[01-12],i22r06c02s[01-12],i22r06c03s[01-12],i22r06c04s[01-
12],i22r06c05s[01-12],i22r07c02s[11-12],i22r07c04s[07-12]

user@cm2login1:~$
```

- The inter cluster can be used for interactive resource allocation:
  `$ salloc -p cm2_inter -N 1`

Interactive Allocation

```
user@cm2login1:~$ salloc -p cm2_inter -N 1
salloc: Granted job allocation 141265
user@i22r07c05s06:~$ hostname
i22r07c05s06
user@i22r07c05s06:~$ exit
exit
salloc: Relinquishing job allocation 141265
user@cm2login1:~$
```

- Notice the change of the hostname, you're now logged in on a compute node!
- See https://doku.lrz.de/display/PUBLIC/Running+parallel+jobs+on+the+Linux-Cluster#RunningparalleljobsontheLinuxCluster-InteractiveSLURMshellforparallel testing and https://doku.lrz.de/x/MgKoAg for further details
- For production jobs, you want to prepare and submit batch scripts – they tell Slurm about the resources you need and the scripts/programs you want to run

Job Processing

```
#!/bin/bash
#SBATCH --clusters=cm2_tiny
#SBATCH --nodes=1

module load slurm_setup

./<executable>
```

- A very minimal example of a job script (not recommended, but working in principle), requesting
  - a single, exclusive node (with 28 cores)
  - of the cm2_tiny partition/cluster, part of
  - the CoolMUC-2 system
- Submit this job script to the queue:
  $ sbatch <script.sh>

Parallelization

Motivation:

- You have a lot of (more or less) independent tasks or

- You want to accelerate a single complex task -> it might be possible to turn the single complex task into many (more or less) independent tasks

…and you have access to a (massively parallel) supercomputer!

Parallelization Scenario: Embarrassingly/Pleasingly Parallel

- many independent processes (10 - 100.000)
- individual task (list) for each process
- private memory for each process
- no communication between processes
- results are stored separately on a (large) storage medium

Parallelization Scenario: Worker Queue

- many independent processes (10 - 100.000)
- central task scheduler (database)
- private memory for each process
- results are sent back to task scheduler
- re-scheduling of failed tasks possible

Parallelization Scenario: Shared Memory

- a few processes working closely together (10-100)
- single script/program/task list
- shared memory
  (cache coherent non-uniform memory architecture aka ccNUMA)
- results are kept in shared memory

Parallelization Scenario: Message Passing

- many independent processes (10 - 100.000)
- one task list (script/program) for all processes
- each process can (in principle) talk to every other process
- private memory
- needs communication strategy in order to scale (area of optimization, e.g. nearest neighbor communication)
- beware of deadlocks!

Job Processing – Give it a try…

```
#!/bin/bash
#SBATCH --clusters=serial
#SBATCH --partition=serial_std

module load slurm_setup

hostname
```

- Create a new folder in your home directory (e.g. “tmp”) and change into it.
- Create this very, very minimal example of a job script (again, this is generally not recommended!) and save it as “script.sh”. What will it do?
- Submit this job script to the queue:
  `$ sbatch script.sh`
- Keep your eyes open for output in the current folder. What can you find?

Job Processing

```
#!/bin/bash
#SBATCH -J <job_name>
#SBATCH -o ./%x.%j.%N.out
#SBATCH -D ./
#SBATCH --get-user-env
#SBATCH --clusters=cm2
#SBATCH --partition=cm2_std
#SBATCH --nodes=3
#SBATCH --ntasks-per-node=28
#SBATCH --mail-type=end
#SBATCH --mail-user=<email_address>@<domain>
#SBATCH --export=NONE
#SBATCH --time=08:00:00

module load slurm_setup

mpiexec –n $SLURM_NTASKS ./<executable>
```

- A more practical example…
  - assigning a job name
  - defining custom output file(s)
  - setting a working directory
  - configuring mail notifications
  - managing the environment
  - limiting walltime explicitly
- See documentation for more details: https://doku.lrz.de/x/AgaVAg

Job Queue and Accounting

- Use the `squeue` command to query information about your jobs in the Slurm scheduling queue, e.g. of the cm2 cluster:
  `$ squeue -M cm2 -u <user>`
- If you‘re interested in the approx. start time of your pending jobs (in the the cm2 queue):
  `$ squeue -M cm2 -u <user> --start`

- Display accounting data of (finished) jobs by use of the `sacct` command, e.g.
  `$ sacct -M cm2 -u <user>`
- Per default, this is limited to today’s jobs, add the `-S` option to specify a user-defined date:
  `$ sacct … -S <YYYY-MM-DD>`