




Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

The background of the slide is a photograph of a modern, multi-story building with a glass facade and a prominent vertical tower section. The image is overlaid with a semi-transparent blue filter. The building is situated in an urban environment with trees and other structures visible in the background.

Introduction to Multiuser Cluster Systems at LRZ

2022-04-19 | J. Albert-von der Gönna, F. Dufour

Course Information

- The aim of this course is to provide an introduction to multiuser cluster systems in general and to those operated at the Leibniz Supercomputing Centre (LRZ), specifically
- You will probably benefit the most if you're not yet familiar with the LRZ HPC/HPDA/HPAI infrastructure, but plan to work with these systems in the future
- A majority of systems will be covered in more detail in dedicated sessions later this week

→ by the end of today's workshop, you should have a general understanding of multiuser HPC/HPDA/HPAI cluster systems and the basic skills to successfully interact remotely with such systems at LRZ





IT Service Backbone for the Advancement of Science and Research



> 250
employees



Computer Centre
for all Munich Universities

Regional Computer Centre
for all Bavarian Universities

National Supercomputing Centre
(GCS)

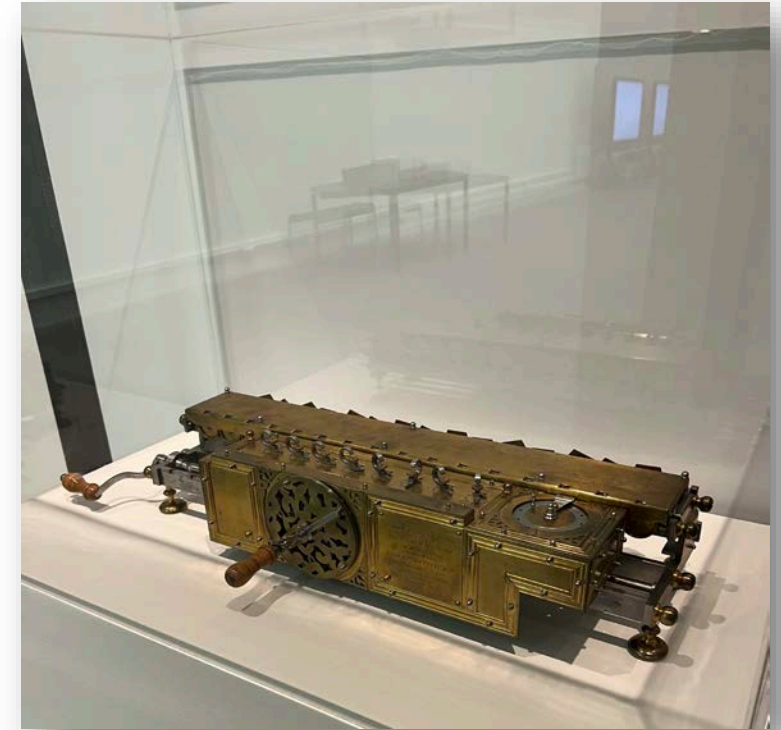
European Supercomputing Centre
(PRACE)



60
years of
IT support

A (Very, Very Short) History of Computing

- **1672/1673:** Early calculating tools & mechanical calculators with Gottfried Wilhelm Leibniz' stepped reckoner
- **1804:** The advent of punched-card data processing with the Jacquard loom
- **1837:** Charles Babbage's Analytical Engine as first general-purpose computing device with Ada Lovelace as the first programmer
- **1936:** Alan Turing describing the principles of the modern computer („On Computable Numbers“)
- Several generations of digital computers: vacuum tubes, transistors on printed circuit boards, integrated circuit chips and, finally, microprocessors



Replica of the Leibniz stepped reckoner (machina arithmetica) from the Deutsches Museum in Munich

A (Very, Very Short) History of Computing

- Computing in different shapes and sizes...
 - Third-generation mainframes and minicomputers
 - Fourth-generation supercomputers and microcomputers/personal computers (home computers vs. business use, „unified“ by the IBM PC in 1981)
- ... has seen various interaction models...
 - batch processing
 - time sharing
 - individual workstations and devices
- ... all of which are still relevant.



Promotional photo of the IBM mainframe 360 from 1964

The Macintosh 128K in 1984

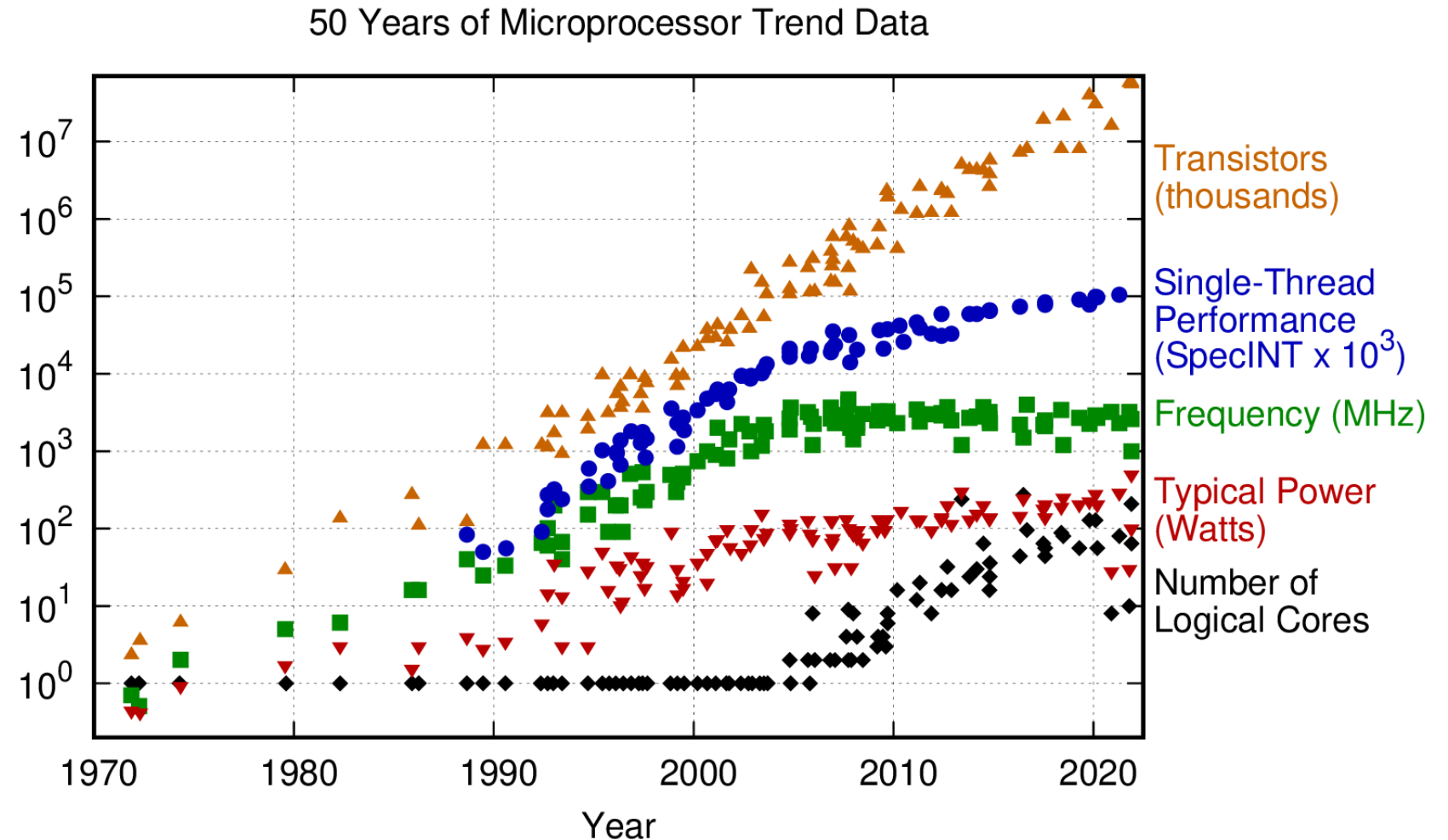


(Massively) Parallel Processing

Mid 2000s:
“heat death”

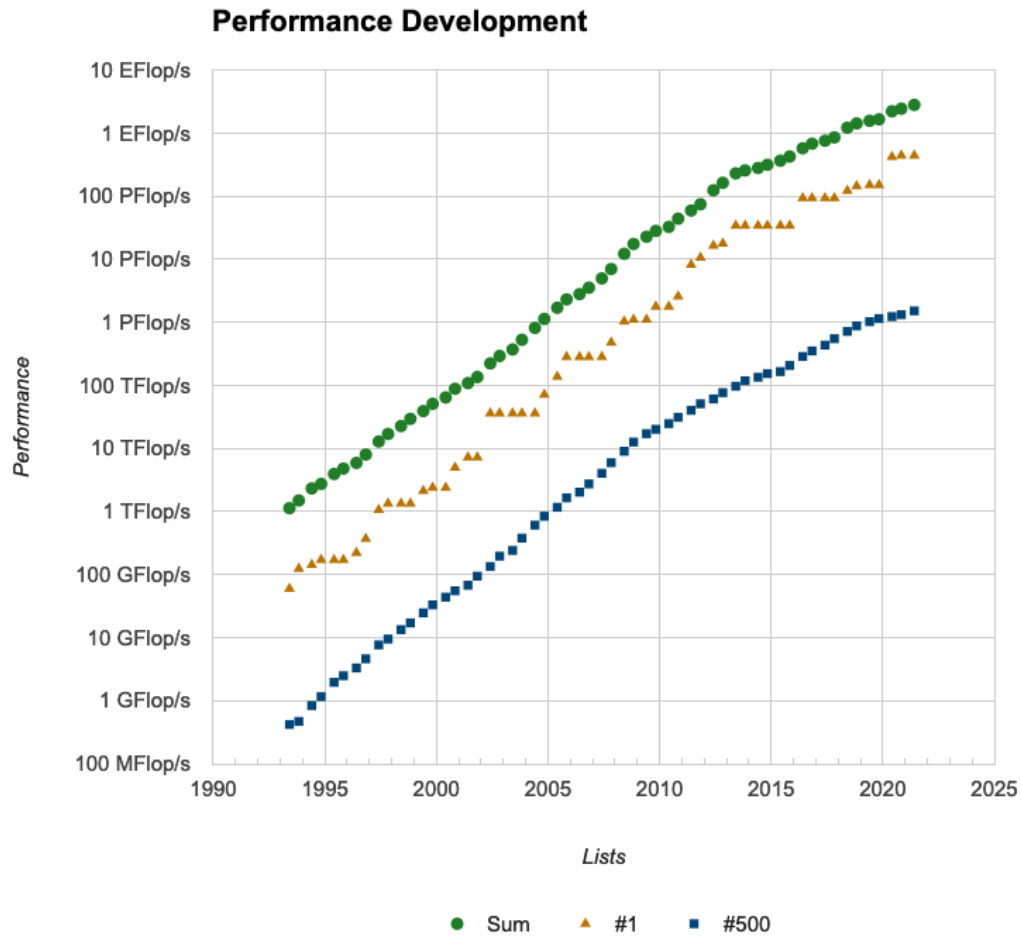
No more faster
processors, only
more of them.

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 2010-2021 by K. Rupp

Performance Development

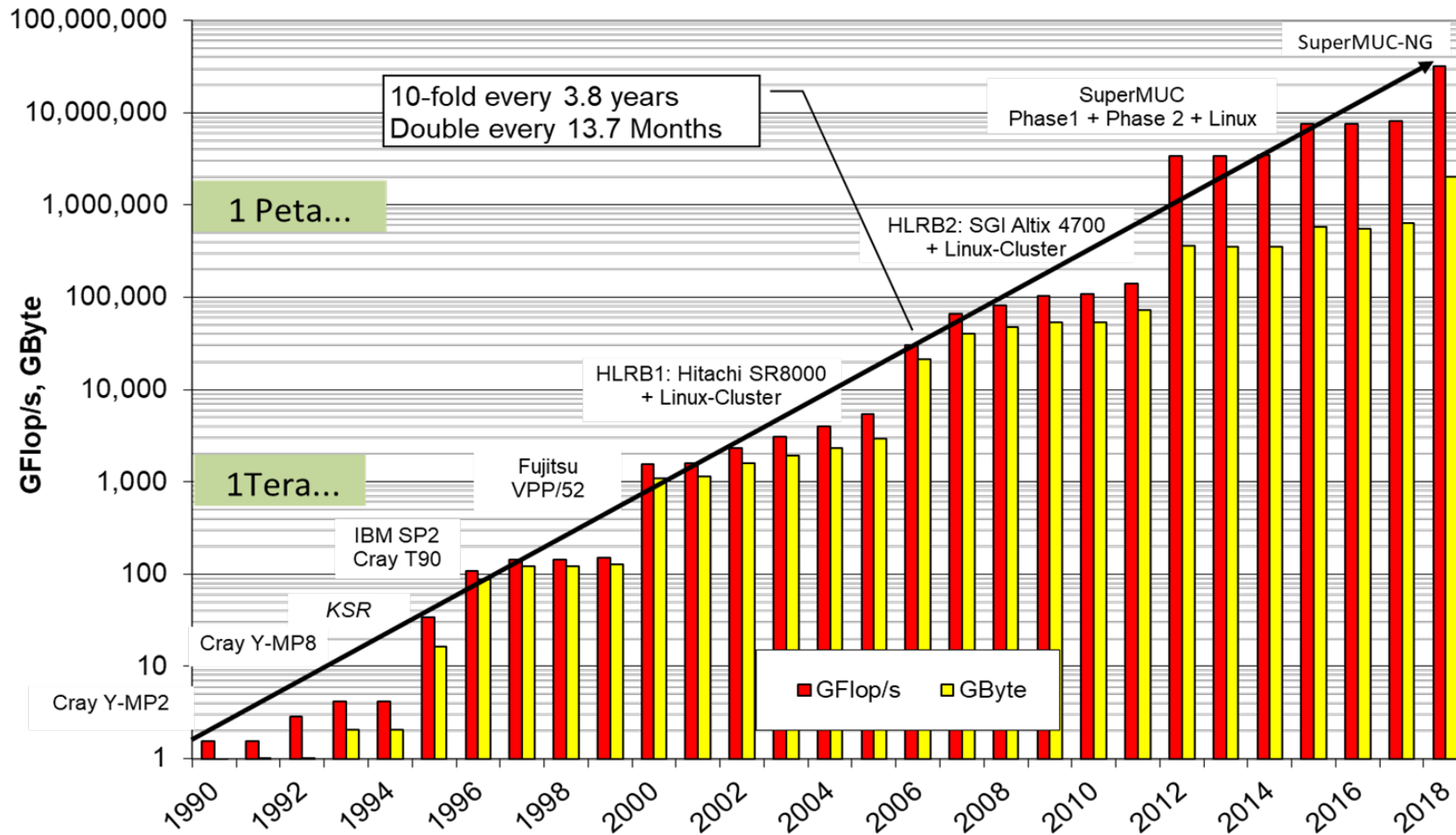


From #1 to #500:
6-8 years

From #500 to Notebook:
8-10 years



Evolution of Peak Performance and Memory



What is a Supercomputer or High-Performance Cluster... (Not)?

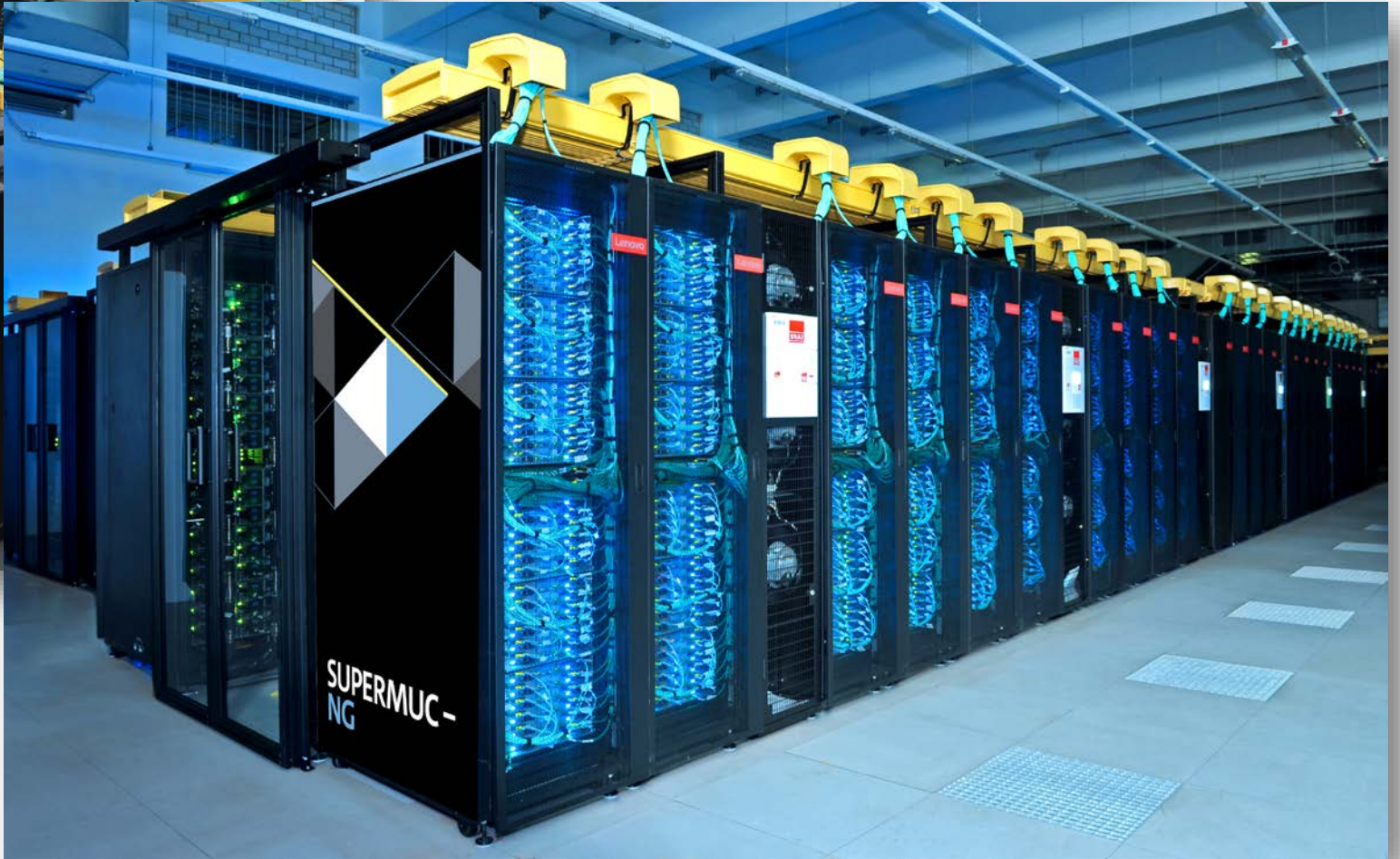


- It has overclocked high-speed processors? **Nope**
- The CPU runs faster than a desktop PC? **Nope**
- It has a large internal memory (RAM)? **Not necessarily (there are exceptions)**
- It runs Microsoft Windows? **Nope**
- It will run my Excel spreadsheet? **Nope**
- It will run my old tried and tested 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? **Some yes, some no**

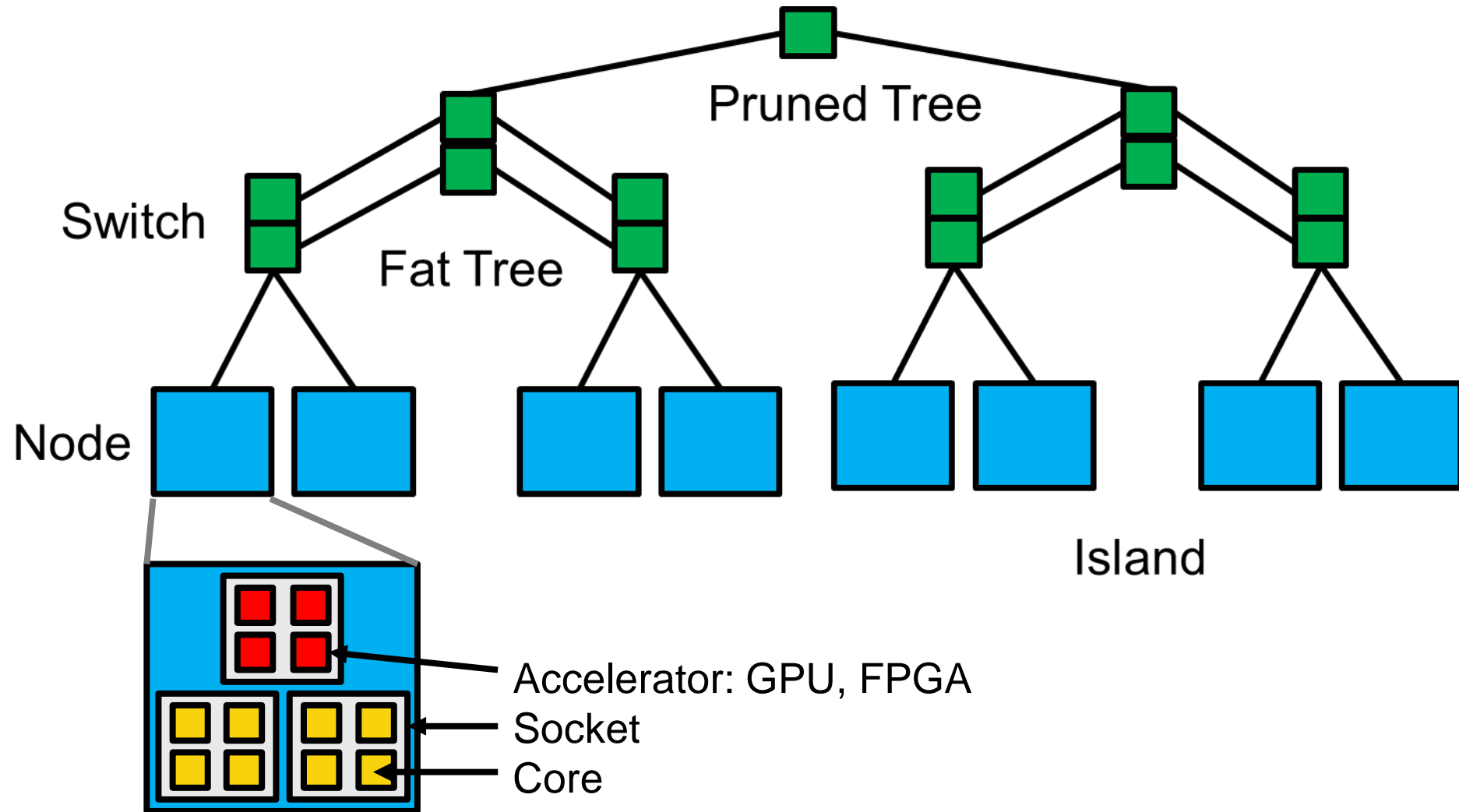
So... what is a Supercomputer or High-Performance Cluster?



- It consists of **many off-the-shelf (server) CPUs** with vector instructions (e.g., AVX2, AVX512) and, in many cases, **accelerators like GPUs** in multiple 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 – this is essentially the component making a supercomputer)
- 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, and sometimes via the web)
- 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/HPDA/HPAI Cluster Systems



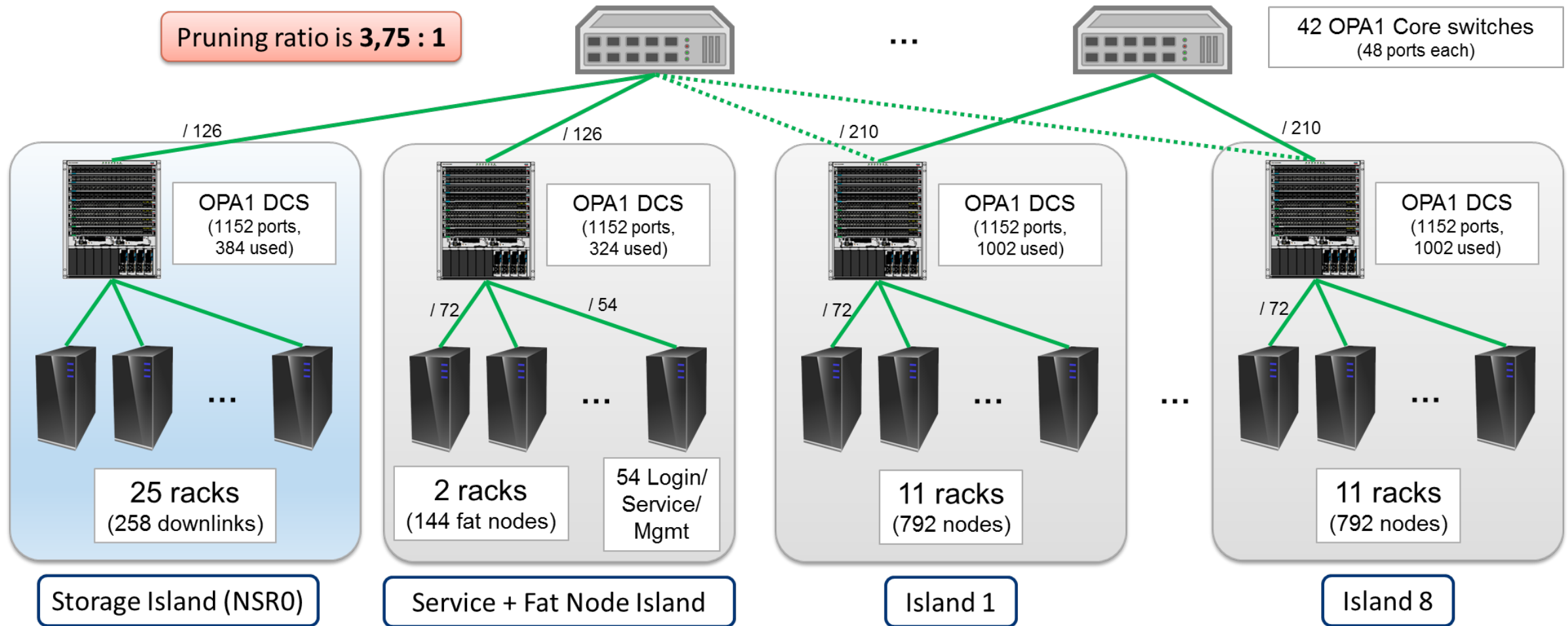
Levels of Parallelism

- **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 (total)	Cores (total)
SuperMUC-NG Thin Nodes	Intel Xeon ("Skylake")	48	96	6336	304,128
SuperMUC-NG Fat Nodes	Intel Xeon ("Skylake")	48	768	144	8,912

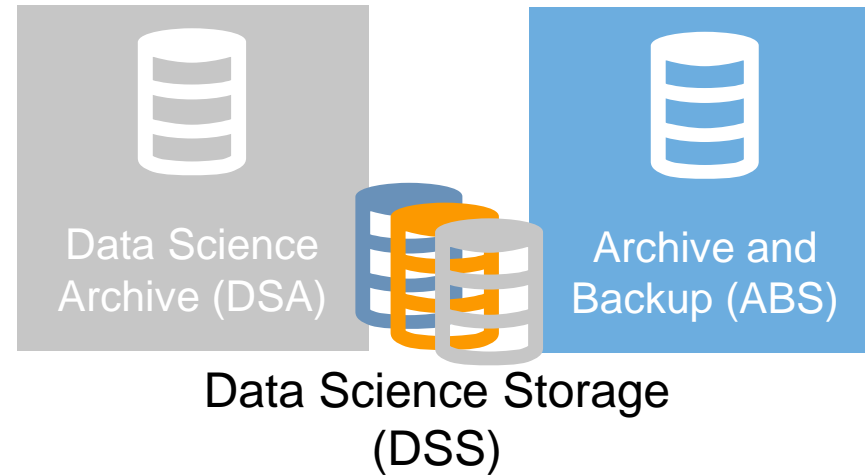
There are three (well, four) ways to apply for using SuperMUC-NG:

1. GCS test project: rolling call, fast review (short abstract), < 300.000 core-h
2. GCS regular project: rolling call, technical & scientific review, < 45m core-h
3. GCS large scale project: biannual, technical & scientific review, > 45m core-h
4. (biannual PRACE calls for academic users from any European country)

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

- In order to use LRZ services provided to Bavarian universities, a “**LRZ Kennung**” (user ID 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 access to other (high performance) systems (see <https://doku.lrz.de/x/64N6Aw> for an overview).
- 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 IDs and permissions within these LRZ projects.

HPC & BDAI Systems for Bavarian Universities



LRZ Linux Cluster

CoolMUC-2 Teramem CoolMUC-3

[lxlogin\[1-4\].lrz.de](https://lxlogin[1-4].lrz.de)

lxlogin8.lrz.de

LRZ AI Systems

- “Big Data” CPU nodes
- HPE P100 node
- V100 nodes
- DGX-1 P100, DGX-1 V100
- (Multiple DGX A100)

datalab2.srv.lrz.de

<https://datalab3.srv.lrz.de>



LRZ Compute Cloud

LRZ Compute Cloud
(w/ some GPUs)

<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 Case LRZ Project Exists

- The master user has to check if the LRZ project is already eligible for Linux Cluster usage.
 - If not, the master user must contact the LRZ contact person for the project (advisor). The LRZ advisor will then explain the next steps to the master user.
 - If the LRZ project is eligible for Linux Cluster usage, the master user can create a new personal LRZ user ID with Linux Cluster access rights for you through the LRZ Identity Management (IDM) Portal.

The master user will need your nationality. Please provide this information. It is a necessary requirement for the export control regulations affecting all HPC/HPDA/HPAI services at LRZ.
- After you get the new user ID from your master user, please use the password reset function of the LRZ IDM Portal using your new ID and your contact e-mail address:
<https://idmportal.lrz.de/pwreset>



Linux Cluster: Access in Case No LRZ Project Exists

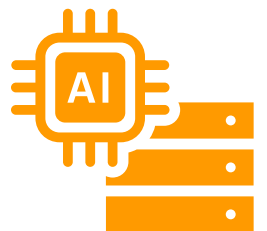
- Your chair/research group has to apply for a new LRZ project.
- Use PDF application form "Antrag auf ein LRZ-Projekt" to be found here: <https://doku.lrz.de/x/CgCiAQ> (only available in German, unfortunately)
 - Pay attention to "Gewünschte LRZ-Serviceklassen" in the application form. For Linux Cluster access you need to
 - select "High Performance Computing" and
 - fill in the phrase "Linux Cluster" at "andere Dienste:"
 - Send the filled in and signed application form to the responsible LRZ contact person (advisor). The original document is needed (you may send a scanned copy to speed up the process, but this does not replace sending the physical letter via snail mail).
- The master user of the newly requested LRZ project will get instructions from the LRZ advisor:
 - Fill out the Service Request Template for "Linux Cluster Project Activation" and submit it to LRZ Servicedesk (<https://servicedesk.lrz.de/en/ql/createsr/12>)
 - Afterwards, the Linux Cluster access for the new LRZ project is typically approved
 - Now, your new master user can create new LRZ user IDs with access to the Linux Cluster (see previous slide)



(BD)AI Systems: Hardware Overview

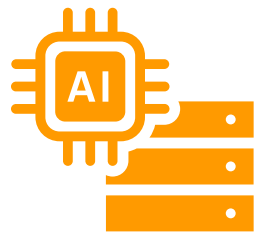


Type	Nodes	CPUs (Node)	Memory (Node)	GPUs (Node)	Memory (GPU)
CPU Nodes	7	up to 20	up to 850GB	-	-
HPE P100 Node	1	64	256 GB	4x P100	16 GB
V100 Nodes	4	40	368 GB	2x V100	16 GB
DGX-1 P100	1	80	512 GB	8x P100	16 GB
DGX-1 V100	1	80	512 GB	8x V100	16 GB
DGX A100/40	1	256	1 TB	8x A100	40 GB
DGX A100/80	4	256	2 TB	8x A100	80 GB



(BD)AI Systems: Access

- Account management of the LRZ AI Systems is associated with the LRZ Linux Cluster:
 - If you don't have an account for the Linux Cluster, you need to set that up first (see previous slides)
 - If you already have an account for the Linux Cluster, you will additionally have to request access to the LRZ AI Systems for this account
- Submit a dedicated service request to LRZ Servicedesk:
<https://servicedesk.lrz.de/en/ql/create/23>
Select "Service Request" from the drop-down list and subsequently "LRZ AI Systems - Request for Access".



Compute Cloud: Hardware Overview



Type	Nodes	CPUs (Node)	Memory (Node)	GPUs (Node)	Memory (GPU)	Access
Compute Nodes	82	40	384 GB	-	-	all users: up to 10 CPUs
GPU Nodes	22	40	768 GB	2x V100	16 GB	restricted
Huge Node	1	192	6 TB	-	-	restricted

Access to more than 10 CPUs and/or other restricted resources can be requested by contacting the cloud support team: <https://servicedesk.lrz.de/ql/create/105>



Compute Cloud: Hardware Overview



Compute Cloud: Access

- You need to have an **ID in a cloud-enabled LRZ project**.
 - If your user ID belongs to a project that is maintained by LRZ but not cloud-enabled yet, you might ask the project's master user(s) to contact LRZ and ask to activate this project for the compute cloud.
- The project's **master user can enable your account** for the LRZ Compute Cloud.
- It might take **some minutes to synchronize** your new permissions with the cloud system. You can not log in until your permissions have been synchronized with the cloud system. After your account's permissions have been synchronized you will receive emails providing further information on how to use the Cloud.



Data Storage: Overview

- The LRZ HPC/HPDA/HPAI 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)
- Additionally, we also provide a new type of Data Archive, based on the DSS Solution stack, called Data Science Archive (DSA) (this basically relates to DSS like AWS Glacier relates to AWS S3).
- 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., project storage).



Data Storage: Linux Cluster

- **\$HOME** (DSS-backed home directory, managed by LRZ)
 - 100GB per user
 - Access: `/dss/dsshome1/lxc###/<user>`
 - Automatic tape backup and file system snapshots (see `"/dss/dsshome1/.snapshots/"` directory)
 - All your important files/anything you invested a lot of work into should be here



Data Storage: Linux Cluster

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



Data Storage: Linux Cluster

- Legacy `$SCRATCH` (scratch file system, “temporary file system”)
 - 1.4 PB, shared among all users
 - Access: `/gpfs/scratch/<group>/<user>`
- New `$SCRATCH_DSS` (not yet available on CoolMUC-2 compute nodes)
 - 3.1 PB, shared among all users
 - Access: `/dss/lxclscratch/##/<user>`
- 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
- Data integrity is not guaranteed. Do not save any important data exclusively on these file systems! Seriously, don't do it!



Data Storage: AI Systems

- Currently, the LRZ AI Systems provide a separate `$HOME` directory which is available on all login and compute nodes of the system. This directory is located in a dedicated DSS container, managed by LRZ.
 - 150 GB per user
 - Access: `/home/<user>` (mapping to `/dss/dssfs03/pn69za/pn69za-dss-0001/<user>`)
 - Users will get an invitation to this DSS container when they are granted access to the LRZ AI Systems. In order to use the system, this invite has to be accepted.
- Going forward, work is being done to rely on the same `$HOME` directory as on the Linux Cluster and to provide access to the same DSS file systems, i.e. all DSS containers that are currently also available on the Linux Cluster.
- Additionally, dedicated low latency/high bandwidth storage is currently brought up and will be made available (potentially as transparent cache layer) soon(ish).



Data Storage: Compute Cloud

- The storage backend of the Compute Cloud is used to host the virtual disks belonging to the VMs in the cloud. It is not meant to store large data sets. No backups are created.
- DSS containers can be made available for VMs running in the LRZ Compute Cloud without the need to copy data into the VM.
 - The data curator of the data project, to which the relevant container belongs, needs to export the container to the IP address used by your VM via NFS.
 - You should only export DSS containers to IPs that are statically assigned to and trusted by you. NFS exports follow a "host based trust" semantic, which means the DSS NFS server will trust any IP/system to which a DSS container is exported. There is no additional user authentication between NFS server and client enforced.



User Perspective: System Access



```
di36pez — di36pez@cm2login1: ~ — ssh lxlogin1.lrz.de — 80x25
[BADWLRZ-CMS9145:~ di36pez$ ssh lxlogin1.lrz.de
Last login: Tue Apr 12 09:57:48 2022 from pb087.int.vpn.lrz.de
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
-----
Mar 18, 2022: Please note the announcement for a scheduled maintenance
of all cluster systems at https://www.lrz.de/aktuell/ali00936.html
-----
spack/release/22.2.1 22.2.1 release linux-sles15
intel-mpi: using intel wrappers for mpicc, mpif77, etc

di36pez@cm2login1:~$
```

Leibniz-Rechenzentrum
der Bayerischen Akademie der Wissenschaften

Shibboleth Web-Anmeldung

Anmeldung bei **Webbasierter Zug LRZ AI-Ressourcen**

Benutzername
Passwort

Anmeldung nicht speichern
 Lösche die frühere Einwilligung zur W meiner Informationen an diesen Dienst.

Anmelden

© Leibniz-Rechenzentrum

Log in

Domain
ADS

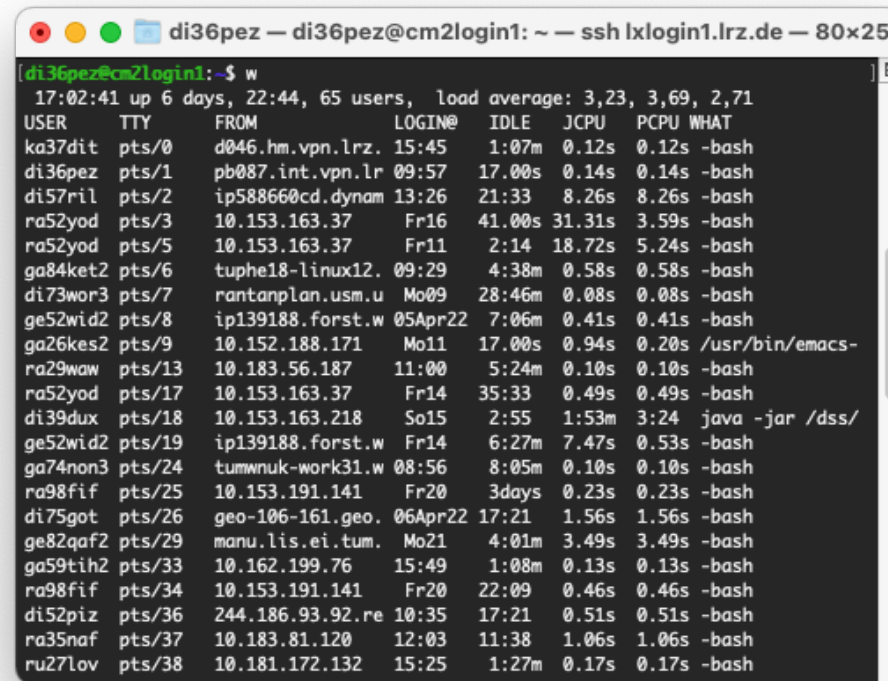
User Name
Password

Connect

Help and Support | Data Privacy Policy / Datenschutzerklärung | Legal Notice / Impressum

User Perspective: Environment & Workspace

- These are systems shared by many users, i.e. other people will be working on the same (login) node at the same time.
- Be aware of your surroundings and considerate of your fellow colleagues!



```
di36pez@cm2login1: ~ — ssh lxlogin1.lrz.de — 80x25
di36pez@cm2login1:~$ w
17:02:41 up 6 days, 22:44, 65 users,  load average: 3,23, 3,69, 2,71
USER  TTY  FROM          LOGIN@  IDLE   JCPU   PCPU WHAT
ka37dit pts/0 d046.hm.vpn.lrz. 15:45   1:07m  0.12s  0.12s -bash
di36pez pts/1 pb087.int.vpn.lr 09:57   17.00s 0.14s  0.14s -bash
di57ril pts/2 ip588660cd.dynam 13:26   21:33  8.26s  8.26s -bash
ra52yod pts/3 10.153.163.37   Fr16   41.00s 31.31s 3.59s -bash
ra52yod pts/5 10.153.163.37   Fr11    2:14  18.72s 5.24s -bash
ga84ket2 pts/6 tuphe18-linux12. 09:29    4:38m  0.58s  0.58s -bash
di73wor3 pts/7 rantanplan.usm.u Mo09   28:46m 0.08s  0.08s -bash
ge52wid2 pts/8 ip139188.forst.w 05Apr22 7:06m  0.41s  0.41s -bash
ga26kes2 pts/9 10.152.188.171  Mo11   17.00s  0.94s  0.20s /usr/bin/emacs-
ra29waw pts/13 10.183.56.187   11:00    5:24m  0.10s  0.10s -bash
ra52yod pts/17 10.153.163.37   Fr14   35:33  0.49s  0.49s -bash
di39dux pts/18 10.153.163.218  So15    2:55  1:53m  3:24 java -jar /dss/
ge52wid2 pts/19 ip139188.forst.w Fr14    6:27m  7.47s  0.53s -bash
ga74non3 pts/24 tumwnuk-work31.w 08:56    8:05m  0.10s  0.10s -bash
ra98fif pts/25 10.153.191.141  Fr20    3days  0.23s  0.23s -bash
di75got pts/26 geo-106-161.geo. 06Apr22 17:21  1.56s  1.56s -bash
ge82qaf2 pts/29 manu.lis.ei.tum. Mo21    4:01m  3.49s  3.49s -bash
ga59tih2 pts/33 10.162.199.76   15:49    1:08m  0.13s  0.13s -bash
ra98fif pts/34 10.153.191.141  Fr20   22:09  0.46s  0.46s -bash
di52piz pts/36 244.186.93.92.re 10:35   17:21  0.51s  0.51s -bash
ra35naf pts/37 10.183.81.120   12:03   11:38  1.06s  1.06s -bash
ru27lov pts/38 10.181.172.132  15:25    1:27m  0.17s  0.17s -bash
```

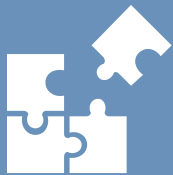
User Perspective: Environment & Workspace

- You don't have administrative rights on these systems, i.e. no root access.
- This may be in contrast to your local machine and certain usage patterns and/or expectations may therefore not apply, e.g.,
 - you will not be able to use the `sudo` command
 - you're prohibited from making system-wide modifications
 - disk access is restricted to your home directory (and possibly other storage areas accessible to your account, e.g., your DSS containers)
- That said, your home (directory) is your castle – there, anything goes!



- If available on the system, 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`
- Search for available modules:
`$ module available <module>` or
`$ module av <module>`
- Get more information about a specific module:
`$ module show <module>`
- Use `$ module load <module>` to apply the changes of a module to the environment

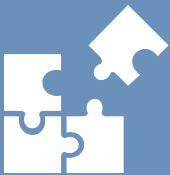
- Short of manually compiling all the binaries you need (of course, this is an option, too), you can resort to using package managers to set up the software/applications you need
- This will have to take place in your home directory, obviously
- **Spack** (<https://spack.io/>) is “a package manager for supercomputers, Linux, and macOS. It makes installing scientific software easy”. It is “supporting multiple versions, configurations, platforms, and compilers”.
It is used to set up (most of) the software available on the Linux Cluster and SuperMUC-NG. This setup can be extended by users.



- **Conda** (<https://conda.io>) is “a package, dependency and environment management for any language – Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN, and more”.
- **pip** (<https://pip.pypa.io>) is “the package installer for Python. You can use it to install packages from the Python Package Index and other indexes”.

Make sure to install packages to the home directory instead of the system-wide default location:

```
$ pip install --user <package>
```



User Perspective: OS-level Virtualization, Containers

- Isolated **user space** instances, called containers, allow programs running inside to only see the container's contents and devices assigned to the container.
- Thus, the environment inside a container can essentially be modified freely, typically **providing (encapsulated) root privileges**
- The most prominent container runtime, Docker, is typically not available on multiuser systems, but you will encounter alternatives
 - Charliecloud (<https://hpc.github.io/charliecloud/>)
 - Enroot (<https://github.com/NVIDIA/enroot>)
- Containers imposes no noticeable overhead, i.e. there should be no performance impact and parallelization, GPU access, etc. should – if set up correctly – work as expected
- Containers are UDSS: User Defined Software Stacks: you're basically independent from the environment created by system administrators, but you will only receive limited support for the environment created instead (inside the container).



Workload Management: Slurm

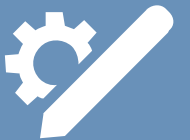
- Slurm Workload Manager is a job scheduler:
 - Allocates access to resources (time, memory, nodes/cores)
 - Provides framework for starting, executing, and monitoring work
 - Manages queue of pending jobs (enforcing “fair share” policy)
- Use the `sinfo` command to get information about the available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

```
$ sinfo --clusters=all or, shortened:
```

```
$ sinfo -M all
```

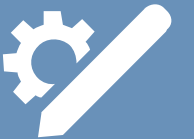
or, for information about a specific cluster:

```
$ sinfo -M <cluster>
```



Workload Management: Slurm

- Slurm allows for the (interactive) allocation of resources
 - `salloc` – obtain a Slurm job allocation (a set of nodes), execute a command, and then release the allocation when the command is finished.
 - `srun` – run parallel jobs, e.g., the most basic interactive job could look like this: `$ srun --pty bash`
- For production jobs, you want to prepare and submit batch scripts by using `sbatch` – they tell Slurm about the resources you need and the scripts/programs you want to run.



```
#!/bin/bash
#SBATCH --cluster=<cluster>
#SBATCH --nodes=1

./<executable>
```

- This is a very minimal example of a batch/job script to be saved in a text file.
- It is a shell script, eventually to be interpreted by /bin/bash. It may contain options preceded with "#SBATCH" before any executable commands in the script.
- These options are instructing Slurm to allocate
 - a single, exclusive compute node
 - of <cluster> for the joband when the job allocation is finally granted, Slurm runs a single copy of the batch script on the first node in the set of allocated nodes.
- Submit this job script to the queue:
`$ sbatch <script.sh>`

- Use the `squeue` command to query information about your jobs in the Slurm scheduling queue:
`$ squeue -M <cluster> -u <user>`
- If you're interested in the approx. start time of all your pending jobs:
`$ squeue -u <user> --start`
- Display accounting data of (finished) jobs by use of the `sacct` command, e.g.,
`$ sacct -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>`

Hands-on !

```
# Let's SSH into the machine...
```

```
~$ ssh <account>@<ip>
```

```
# ...give the password...
```

```
~$ sudo -i # ...and become root.
```

```
# Let's start a toy slurm cluster...
```

```
~# cd /opt/slurm/slurm-docker-cluster
```

```
~# docker compose up -d
```

```
~# slurm
```

```
# We're now in a toy login node!
```

```
[root@slurmctld /]#
```

Workload 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 smaller (more or less) independent tasks

...and you have access to a (massively parallel) supercomputer/multiuser cluster!



Parallelization Scenario: Shared Memory

- Imagine a situation, in which you have a task coordinated by a single script/program/task list that can utilize a few processes (~10-100) working closely together and results can typically be kept in memory.
- This situation could best be addressed by parallelization on a single, shared memory node, be it either your local multi-core desktop computer or the compute node of a multiuser cluster.
- The most widely used approaches are
 - OpenMP (<https://openmp.org/>), which supports multi-platform shared-memory multiprocessing programming in C, C++, and Fortran
 - POSIX Threads/pthreads (e.g., Python module “threading” or other modules supporting parallel execution, e.g., “TensorFlow” and “PyTorch”)
 - Process spawning/forking (e.g., Python module “multiprocessing”, R package “parallel”)



Parallelization Scenario: Embarrassingly/Pleasingly Parallel



- Imagine you have many fully independent processes (~10-100.000) with individual tasks, private memory requirements for each process and no communication between them, while results can be stored separately on a (large) storage medium (accessible to each process).
- In this scenario, you probably want to spawn as many of these processes on as many compute nodes as you can get access to.
- This is referred to as job or task farming.
- Slurm provides built-in functionality to support this approach (--exclusive "job steps" and "job arrays").
- Alternatively and/or additionally, OpenMP and/or the Message Passing Interface (MPI) can be used (see the following slides).

Parallelization Scenario: Worker Queue and Message Passing



- Imagine a situation similar to the previous one, but you want an additional central task scheduler (i.e. main process, database) to monitor your independent processes (e.g., to re-schedule failed tasks if needed) and to collect results.
- Or imagine a scenario in which many independent processes with private memory requirements and a common purpose should (in principle) be able to communicate with each other, also between nodes.

Message Passing Interface (MPI)



- Message Passing Interface (MPI; <https://www.mpi-forum.org/>) is a standard that defines syntax and semantics of library routines for parallel programs in C, C++, and Fortran.
- There are multiple implementations: MPICH, Open MPI, Intel MPI and others
- MPI manages the communication over the network between the running processes of an application. It can be used flexibly to address different needs:
 - Generally speaking, it can be used to distribute an application's workload among any number of cores located in any number of nodes.
 - In combination with OpenMP (or other methods) used for parallelization within a (multi-core) node, it can be used to manage parallelism between nodes (“hybrid mode”).
 - Applications supporting MPI can also be used in single-node setups if needed.
- Bindings are available for Python (mpi4py) and R (Rmpi, doMPI) and the distributed deep learning framework Horovod does (conceptually) build on it.