lrz

Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

AI Training Series: Introduction to the LRZ Linux Cluster

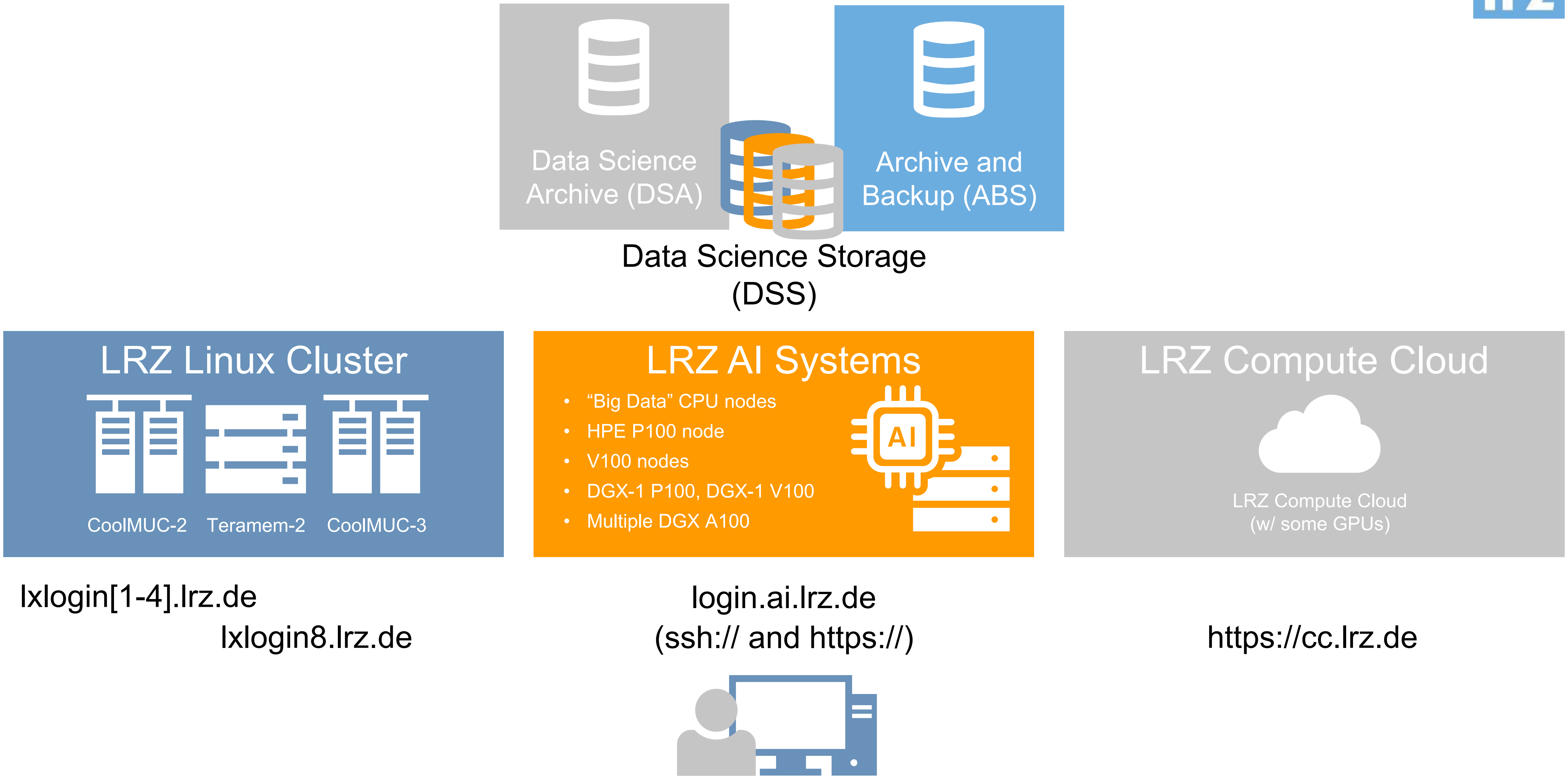
2023-11-16

Course Information

- The aim of this course is to provide an introduction to the characteristics of and the basic interaction with the LRZ Linux Cluster of the Leibniz Supercomputing Centre (LRZ; as part of the High-Performance Computing (HPC) infrastructure) for High Performance Data Analytics / High Performance AI (HPDA / AI)
- You will probably benefit the most if you’re not yet familiar with the LRZ Linux Cluster, but plan to work on this system in the future

-> by the end of this workshop, you should have the basic skills to successfully interact remotely with the LRZ Linux Cluster

HPC & AI Systems for Bavarian Universities



Data Science Archive (DSA)

Archive and Backup (ABS)

Data Science Storage (DSS)

LRZ Linux Cluster

CoolMUC-2 Teramem-2 CoolMUC-3

LRZ AI Systems

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

LRZ Compute Cloud

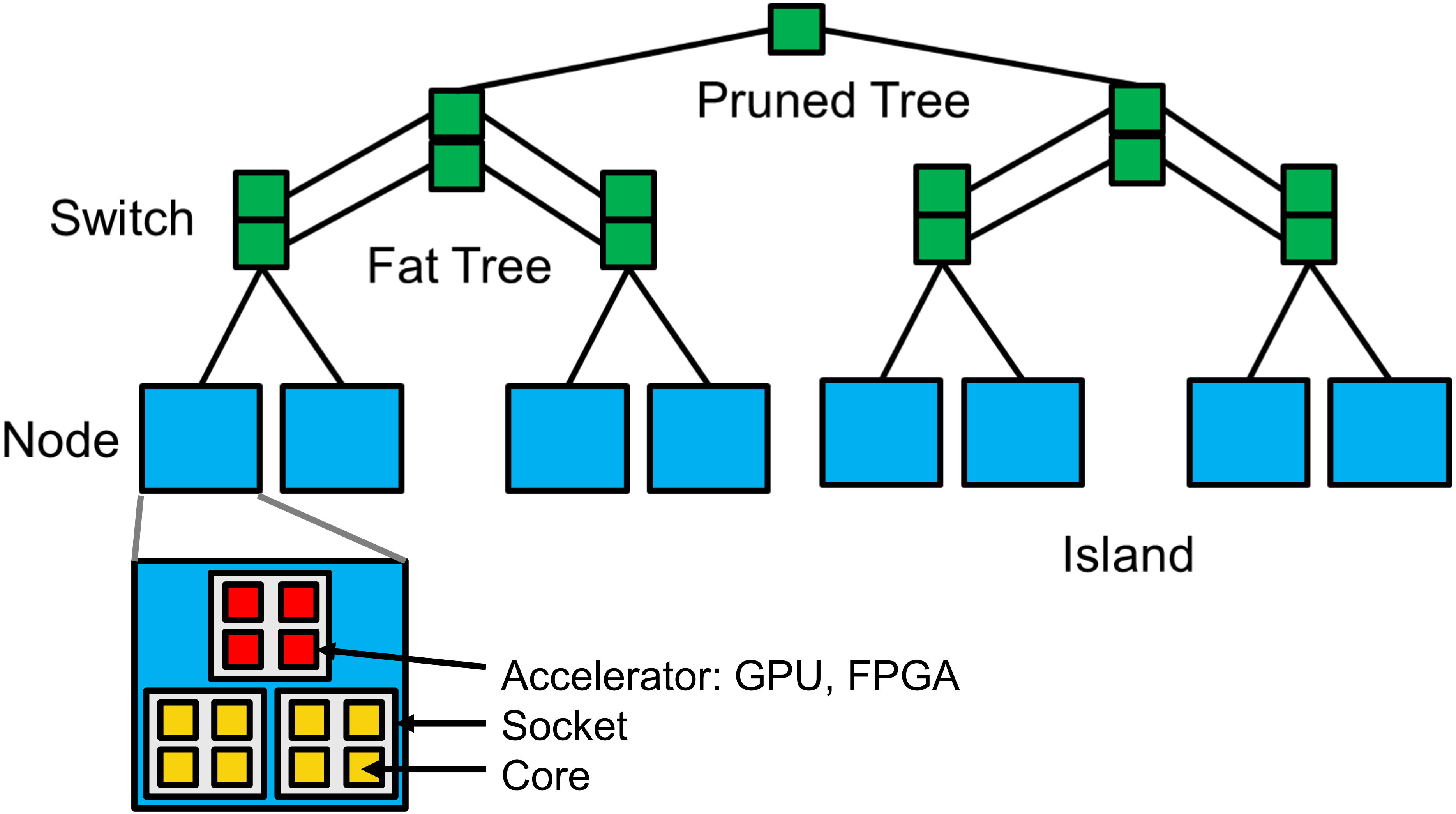
LRZ Compute Cloud
(w/ some GPUs)

lxlogin[1-4].lrz.de
lxlogin8.lrz.de

login.ai.lrz.de
(ssh:// and https://)

https://cc.lrz.de

HPC Cluster Systems



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
  - Process spawning/forking (e.g., Python module “multiprocessing”, R package “parallel”)
  - (POSIX p)threads (e.g., Python module “threading” or other modules supporting parallel execution, e.g., “PyTorch” and “TensorFlow”)

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 cores of as many compute nodes you can get access to.
- This is referred to as job or task farming.
- Slurm provides some built-in functionality to support this approach (“job steps”, “job arrays”).
- Alternatively and/or additionally, e.g. 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 a (single) 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 on 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 e.g. the distributed deep learning framework Horovod as well as PyTorch Distributed do (conceptually) build on it (other common backends are Gloo and NCCL).

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-2	Intel Xeon Platinum 8360HL	96	6144	1	96

… you may also be able to catch a glimpse of CoolMUC-4, the next generation of nodes.

https://doku.lrz.de/display/PUBLIC/Compact+Guide+to+first-time+Linux+Cluster+Access+Process

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 provide LRZ user IDs with access to the Linux Cluster (see previous slide)

https://doku.lrz.de/display/PUBLIC/Compact+Guide+to+first-time+Linux+Cluster+Access+Process

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 (in some cases) import your existing LRZ user ID or create a new personal LRZ user ID and enable Linux Cluster access rights 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

https://doku.lrz.de/display/PUBLIC/File+Systems+and+IO+on+Linux-Cluster

Data Storage: Linux Cluster

- DSS-backed home directory $HOME (managed by LRZ)
  - 100GB per user
  - Access: /dss/dsshome1/###/<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

https://doku.lrz.de/display/PUBLIC/File+Systems+and+IO+on+Linux-Cluster

Data Storage: Linux Cluster

- DSS project storage (Linux Cluster DSS)
  - 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)

https://doku.lrz.de/display/PUBLIC/File+Systems+and+IO+on+Linux-Cluster

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 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 availability is not guaranteed. Do not save any important data exclusively on these file systems! Seriously, don’t do it!

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
- Use port 22 for password-based authentication, port 2222 for public-key authentication:
  ```
  $ ssh -p 2222 <user>@lxlogin1.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:
```

- Continue by typing your password.
  As this may become a repetitive burden, you may choose to use SSH-keys instead.

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:
!!! 2FA !!! 2FA !!! 2FA !!! 2FA !!! 2FA !!! 2FA !!! 2FA !!! 2FA !!! 2FA !!! All LRZ cluster
login nodes require now a second factor for authentication! Please refer to
https://doku.lrz.de/x/EgDxBw Note for using push tokens: Without PIN directly press enter
upon the 2FA prompt to receive the push token!
Token_Response:
```

- Finally, a two-factor authentication token needs to be provided (different methods are supported).
  For the purpose of this course, **use your course account ID** as second factor!

Linux Cluster – CoolMUC-2

```
user@localhost:~$ ssh user@lxlogin1.lrz.de
Last login: Mon Nov 13 15:14:06 2023 from pb167.int.vpn.lrz.de
  Welcome to the CoolMUC-2 Infiniband Cluster, one of the linux cluster systems
  operated by Leibniz Supercomputing Centre (LRZ).
  Please submit SLURM batch scripts with a VALID email address for production
  jobs. Short-running programs and testing are most suited for being carried
  out within SlURM interactive shells.
  DO NOT run any extensive computational programs on login nodes!
  Misuse of the interactive resources will lead to violating accounts being
  blocked from access to the cluster.
--------------------------------------------------------------------------------
  Documentation:
         https://doku.lrz.de/display/PUBLIC/Linux+Cluster
  Messages/System Status:
         https://doku.lrz.de/display/PUBLIC/High+Performance+Computing
  Slurm Usage Instructions:
         https://doku.lrz.de/display/PUBLIC/Available+SLURM+clusters+and+features
--------------------------------------------------------------------------------
intel-mpi: using intel wrappers for mpicc, mpif77, etc
user@cm2login1:~$
```

Linux Cluster – CoolMUC-2

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

Linux Cluster – CoolMUC-2

- Get your bearings… where did you end up on the file system?
  Use `$ pwd` to print your current directory.

Linux Cluster – CoolMUC-2

- Use `dssusrinfo all` to query DSS-containers available to you (for additional options, see `dssusrinfo -h`)
- Look at and explore $HOME (this is a environment variable):

```
$ echo $HOME
```

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        3/10/0/13 hlegr1409n[01-13]
biohpc_gen_production     up 14-00:00:0        3/10/0/13 hlegr1409n[01-13]
biohpc_gen_normal         up 2-00:00:00        3/10/0/13 hlegr1409n[01-13]
biohpc_gen_inter*         up   12:00:00        3/10/0/13 hlegr1409n[01-13]

CLUSTER: c2pap
PARTITION      AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
c2pap_parallel    up 2-00:00:00      114/6/0/120 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      114/6/0/120 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_preempt     up 2-00:00:00      114/6/0/120 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]

CLUSTER: cm2
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
cm2_std*     up 3-00:00:00     386/11/7/404 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]
...
```

- Look for the cluster segments
  - inter (dedicated to interactive usage)
  - cm2 (the main CoolMUC-2 cluster)
  - cm2_tiny (CM2 for small jobs)
  - 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          0/3/0/3 mpp3r03c05s[01-03]
teramem_inter           up 10-00:00:0          1/0/0/1 teramem1
cm2_inter               up    2:00:00        12/0/0/12 i22r07c05s[01-12]
cm2_inter_large_mem     up 4-00:00:00          0/6/0/6 i22r07c01s[01-06]

user@cm2login1:~$ sinfo -M cm2 -s
CLUSTER: cm2
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
cm2_std*     up 3-00:00:00     377/20/7/404 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     377/20/7/404 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-InteractiveSLURMshellforparalleltesting 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

Teramem-2

- A single node with 6TB main memory
- Use any of the login nodes of CooLMUC-2
- To start an interactive shell, use

```
module load salloc_conf/teramem
salloc --cpus-per-task=<ncpus> --mem=<mem>

(srun ./my-program)
```

- For an example batch script, see documentation

https://doku.lrz.de/x/1waVAg

Job Processing

- For the purposes of this training, a Slurm reservation has been created, i.e. a certain number of CoolMUC-2 compute nodes have been dedicated exclusively to participants of this training.
- Add Slurm option `--reservation=hdta4w23` to any upcoming resource allocation on the cm2 cluster, partition cm2_tiny.
  (you may still be able to run jobs on other clusters/partitions or without using this reservation, but it does increase your chances of faster execution)

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>

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. “playground”) 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”.
- 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 (e.g. to today’s jobs), add the `-S` option to specify a user-defined start time:
  ```
  $ sacct … -S <YYYY-MM-DD>
  ```

AI Training Series

Upcoming Sessions

21.11.2023 The **LRZ Compute Cloud** for AI Support <- Registration Deadline 24.11.2023

13.12.2023 High Performance **Data Analytics Using R** at LRZ

… more to follow in 2024!

Further details and registration: https://app1.edoobox.com/en/LRZ/