Introduction to the LUIS Computing Cluster

Scientific Computing Group
Leibniz Universität IT Services

Outline

- What is a Compute Cluster? Why use it?
- Accessing the LUIS computing cluster & data transfer
- Available hardware, how to properly use the storage
- How to use/install software
- Submitting jobs to the SLURM workload manager
- How to get help

Why use a compute cluster?

- **Case A:** Imagine a computation that would take 2 years (730 days) to complete on a normal workstation.

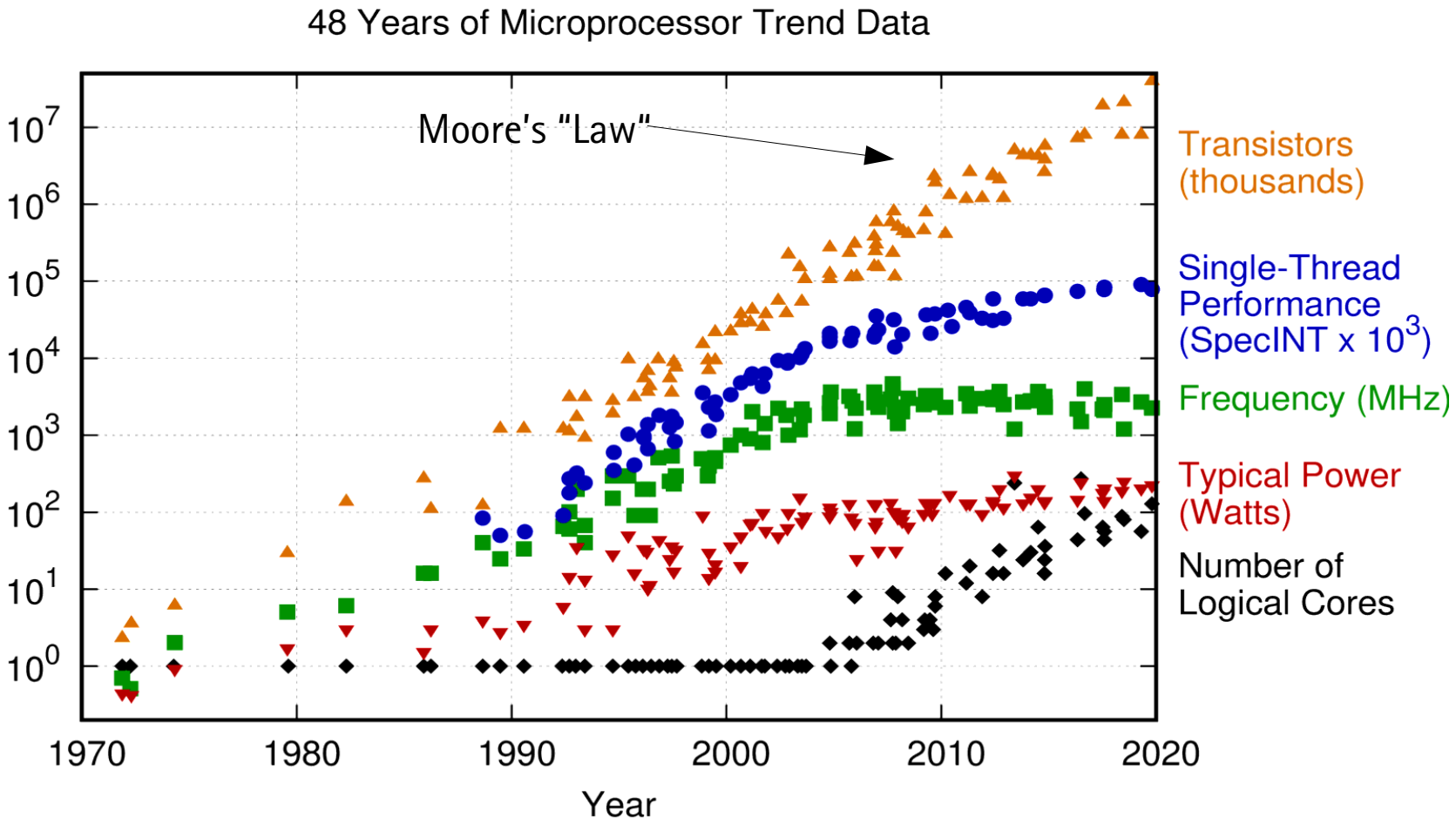
  If you could somehow compute 100 times faster (e.g. by using more CPUs), you would get your results next week.

- **Case B:** A parameter study with four parameters, each with 10 different settings and each computation needing 10 minutes would take 100.000 minutes (~ 70 days).

  If you could bundle your jobs into 100 packages of 100 computations each and run 50 of these jobs in parallel, you would be done in less than 2 days.

- Drawback: you may need to do some work to make this possible.

Computers get faster all the time. Why not buy faster CPUs?



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

Moore's "Law" & Parallel Processing

- Number of transistors ("switches") in a microprocessor doubles about every 18 months → Moore's "Law", self-fulfilling prophecy
- Result: structure size shrinks, from (1971) 10 µm to 28 nm (2010). Structure sizes claimed thereafter (2024, "2 nm") are mostly marketing.
  Size of a Si-atom: ca. 111 pm.
- CPU Performance (FLOPs) ~ Cores x Frequency x Instructions per Cycle
- Higher frequencies **in**crease power consumption and **de**crease signal integrity. 2.5 GHz has been proven to be a good balance between performance and stability.
- A typical 2.5 GHz CPU consumes 135 Watts.
  The heat dissipation limit for air cooling about 150 W/cm²
- Multicore CPUs and thus parallelization necessary to increase performance.

How does a "typical" compute/HPC cluster look like today?

- *Compute* cluster (suitable for "easy" parallelization): a collection of servers ("**compute nodes**"), usually equipped with CPUs using x64 architecture (Intel, AMD).

- *HPC* cluster (suitable for parallel computation of complex problems): connected by at least one **high speed** and **low latency** internal network (usually InfiniBand) for both inter-process communication and access to common file systems (Lustre, NFS).

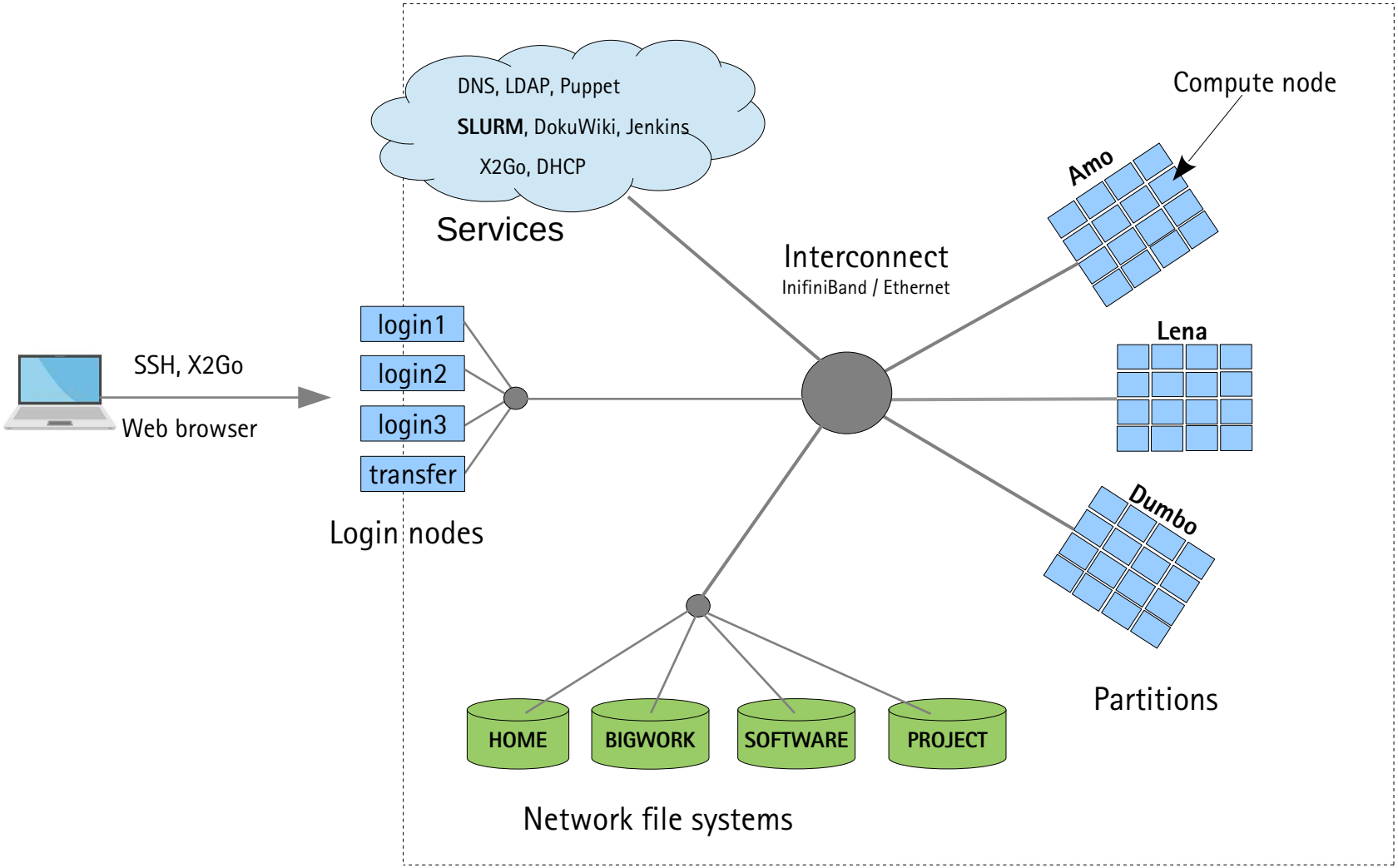
- **Single node** (intranode) parallelization, also called shared memory or SMP, is frequently based on the **OpenMP** (Open Multi-Processing) API. **Multinode** (also: massively parallel, MPP, distributed memory) parallelization is typically realized using **MPI** (Message Passing Interface) libraries.
  Hybrid parallelization is also possible.

- A **single** core on a cluster CPU usually does **not** run faster than your desktop computer/notebook. Typical compute server CPUs carry much more CPU cores than workstation CPUs and run at a lower frequency to remain within the power budget. RAM access is also **not** always faster than on your desktop computer, but nodes often have significantly larger memory.
  Compute Cluster : Workstation is similar to **Container Ship : Porsche**.

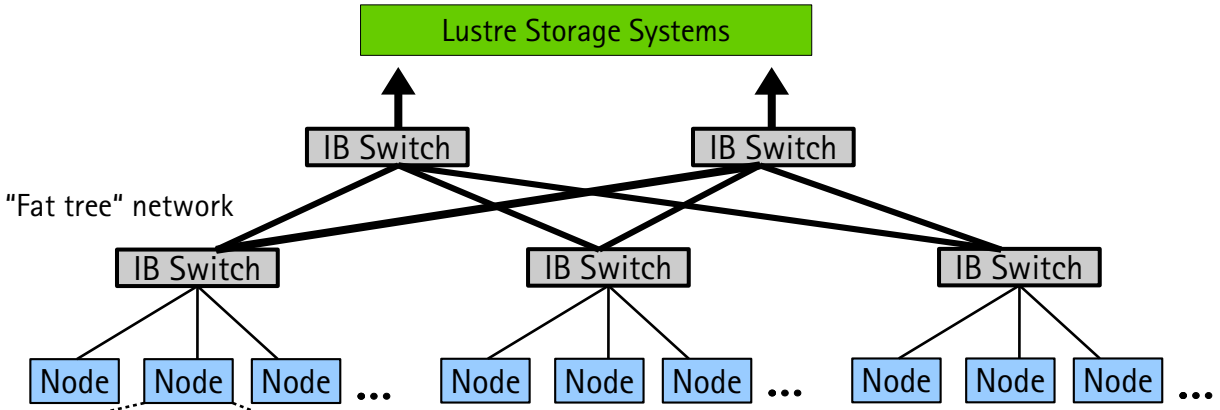
- Each node usually runs one Linux instance (CentOS, Rocky Linux, Ubuntu etc.).
  Compute clusters usually do **not** run Windows applications.

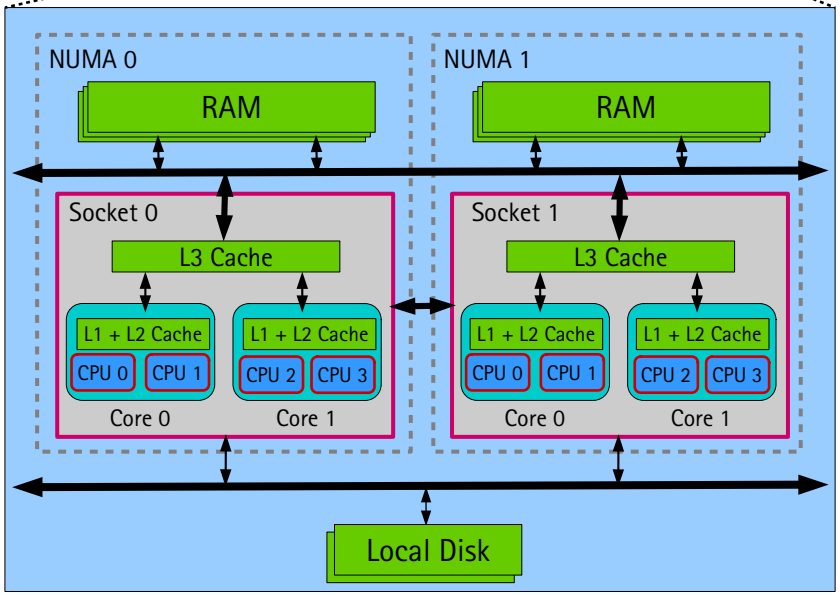
- Compute nodes are usually not accessible directly. They get allocated by a workload manager (**SLURM** in our case) that runs jobs that are submitted from the login nodes.

LUIS Cluster Systems



Compute Cluster





Latencies (approximate, in ns)	
CPU cycle	0,4
L1 Cache	1
L2 Cache	3
L3 Cache	10
RAM	100
SSD	50.000
HDD	5.000.000

Accessing the LUIS cluster

- The LUIS compute cluster is available to LUH employees (free of charge)
- The basis for access is a "project":
  - Project administration can use bias.luis.uni-hannover.de to manage (create, delete, ...) user accounts.
  - User accounts allow access to the cluster
  - Please use **only** LUH email address for accounts and contact to support
- Frequently, a project will already exist at your institute.
  Just contact the project management and ask for an account.
- Do not share accounts – one person per account, one account per person
- Students can not apply for a project, but may get an account for their work at an institute.
- User accounting via BIAS is ***not*** part of the service *Scientific Computing*
- LUIS compute cluster hardware and software licenses may ***only*** be used for academic purposes

Accessing the LUIS cluster, data transfer

- From *within* (!) the university network, connect to the login machines (login.cluster.uni-hannover.de) or the data transfer node (transfer.…), depending on what you want to do
- We support the following ways to connect to the LUIS compute systems
  - Open OnDemand web portal at https://login.cluster.uni-hannover.de
    - No need to install anything. Mobile devices can also be used
    - Works with Chrome, Firefox, Microsoft Edge and Safari (partially)
  - SSH secure shell
    - *Linux and Mac*: clients available by default
    - *Windows*: MobaXTerm (recommended, inc. X11, sftp), PuTTY, WSL
  - X2Go Remote Desktop Session
    - Client available for all major operating systems
    - See Cluster documentation wiki for help on installation
- Data transfer
  - *Linux and Mac*: command line: *scp*, *rsync*, GUI tools*: FileZilla*, *Web Portal*
  - *Windows: e.g. FileZilla*, *MobaXTerm*, *WinSCP*, *Web Portal*

Exercise

- How to access the LUIS compute systems
  - Web browser
  - SSH client
  - X2Go client

Login & Transfer Nodes

- Connecting to login.cluster.uni-hannover.de routes to one of the login nodes
- On the login machines, you can:
  - Prepare, submit and monitor batch jobs
  - Run small tests and do GUI analysis
- Do **NOT** run resource-intense (CPU, RAM) applications on the login nodes, as they will be killed automatically after 30 minutes of elapsed CPU-time. Parallel applications are killed faster: use 10 cores and get killed after 3 minutes.
- Data transfer (e.g. via scp) may qualify as resource-intense, depending on size, due to encryption. Therefore, we recommend to use transfer.cluster.uni-hannover.de for any large (> 10 GB) data transfers between your desktop computer and the cluster file systems (or the LUIS archive). Transferring data via the Open OnDemand Portal will also fail for files larger than 10 GB.
- Process execution time on the transfer node is unlimited, but no jobs are accepted from there.

Compute resources

4 compute clusters for **MPI**-Jobs that require a lot of CPU

Cluster	Nodes	CPU	Cores/ Node	Cores Total	Memory / Node (GB)	Memory Total (GB)	GFLOPs/ Core
Amo	80	2x Cascade Lake 2.30 GHz	40	3200	192	15360	75
Lena	80	2x Haswell 2.40 GHz	16	1280	64	5120	38
Taurus	24	2x Skylake 2.10 GHz	32	768	128	3072	67
Haku	20	2x Broadwell 2.10 GHz	16	320	64	1280	34

GPU compute nodes

Cluster	Nodes	CPU/GPU	Cores / Nodes	Memory / Node (MB)	GPU Double-Precision-Performance / Node
GPU	4	CPU: 2x Cascade Lake GPU: 2x Tesla V100	CPU: 2x 20 CUDA: 2x 10340	CPU: 125.000 GPU: 2x 16.384	2x 7 TFLOPs
GPU	3	CPU: 2x Skylake GPU: 2x Tesla A100	CPU: 2x 24 CUDA: 2x 6912	CPU: 1025.000 GPU: 2x 80.384	2x 9.7 TFLOPs

Compute resources

SMP (Symmetric MultiProcessing) nodes for jobs that require a lot of memory

Cluster	Nodes	CPU	Cores/ Node	Cores Total	Memory/ Node (GB)	Memory Total (GB)	GFLOPs / Core
Dumbo	18	4x IvyBridge 2.4 GHz	40	720	**512**	9216	19
SMP	4	4x Broadwell 2.5 GHz	32	128	**256**	1024	40
SMP	9	4x Westmere 2.1 GHz	32	288	**256**	2304	8.4
SMP	9	4x Backton 2.0 GHz	24	96	**256**	3072	8.0
SMP	3	4x Westmere 2.1 GHz	32	96	**1024**	1024	8.5

Forschungscluster-Housing (FCH)

Cluster	Nodes	CPU	Cores / Node	Cores Total	Memory/ Node (GB)	Memory Total (GB)
Various	~150	Intel, AMD	12 - 128	~6000	various	5000

Total	**~400**			**~15k**		**~90000**

Research Cluster Housing ("FCH")

- Institutes can arrange for their own computing hardware to be integrated into the cluster via "ForschungsCluster-Housing"(FCH), provided the service's conditions are met.
- We provide housing, electrical power, cooling, administration, software stack, network file systems, batch system, consulting, backup, access ... just as we do for the public cluster
- FCH nodes may be reserved exclusively for institute projects during office hours. For the remainder of the time, the nodes help on the common workload. The reservation provides a huge priority boost (indirectly even outside exclusive hours because of wall times).
- Advantages:
  - You invest only in compute (and optionally storage) hardware
  - Immediate access to the full cluster software stack, storage systems and cluster resource management, easy project administration and user account creation.
  - Much better utilization of resources and financial funds (good for grant applications).
  - In case you have to buy your resources now, but can not yet use them fully in a project yourself, others may benefit for a time. You will see the same situation from the opposite side in future years.

Compute Cluster **Amo**

- In production since July 2021
- 80x Dell PowerEdge C6420 dual processor compute nodes
- Interconnect: InfiniBand HDR100 (100 Gbs)
- 3200 CPU cores, 15 TB RAM
- ~235 TFLOPs theoretical performance
  - Linpack-value ~195 TFLOPs

Storage

Storage	Quota Soft / Hard	Speed	Backup	Available on
$HOME	10 GB / 12 GB	slow	Yes (daily)	All nodes
$BIGWORK	100 GB* / 1 TB*	high	No	All nodes
$SOFTWARE	50 GB / 100 GB	slow	No	All nodes
$PROJECT	10 TB / 12 TB	moderate	No	login & transfer

To display your HOME, BIGWORK and PROJECT quota, execute the *checkquota* command on a login node
* We will try to accommodate reasonable time-limited requests for more quota on BIGWORK

Cluster storage systems

$HOME

- Globally (i.e. cluster-wide) available network filesystem-based ("NFS") home directory → default storage after login
- Intended for your configuration and small scripts
- General rule: "Never WORK at HOME" → do NOT use as input/output location for your running applications. You will only overstep your quota and get into trouble
- Intentionally low quota will not be increased
- Slow

$BIGWORK

- Globally available Lustre-based parallel file system
- Intended for temporary storage of large datasets (high bandwidth)
- Poor performance when using many small files
- May fail and get destroyed – regard it as scratch and make backups of important data
- Can be accessed using the environment variable $BIGWORK
- Default quota ("storage limits") can be raised upon reasonable request

Cluster storage systems

$TMPDIR

- Shared local storage available to jobs
- Automatically removed after job completion (no backup/copy)
- Intended as a working directory for software that reads/writes many small files
- Can be accessed from within a job using the variable $TMPDIR
- Contents are strictly local to each node

$SOFTWARE

- NFS-based file system available on all nodes
- Intended for installation of software
- Can be accessed using environment variable $SOFTWARE
- Default quota may be increased upon request

Cluster storage systems

$PROJECT

- Lustre-based parallel file system available only on login and transfer nodes
- Intended for long term retention of your datasets
- Can be accessed using environment variable $PROJECT
- Regarded as being owned by the project/group
- Default per project quota may be increased upon reasonable request

Exercise

- Data transfer *from/to/within* the compute systems
  - Cluster storage ⟺ Desktop: **`scp, rsync`**
  - Within cluster: you may additionally use the commands provided by the module **`mpifileutils`**
  - **`checkquota`**

How to use/install software applications

- Most software packages and applications available in the cluster are not installed locally on the nodes' operating systems.
- Instead, they are installed in a central directory and activated loading a module (Lmod system).
  - Lmod provides tools for a dynamic modification of your software environment
  - Different versions of a software package may be available
- Software installation utilizes a hierarchical software module naming scheme
- Loading a compiler or MPI implementation module will make available all the software built with those applications ("toolchain concept")
- Use module spider to find out which software versions are available and how to load them

module spider [<name>]	show all available modules
module avail [<name>]	show modules available for immediate load
module [un]load <name>/version	[un]load a module of <name> and version <version>
module list	list currently loaded modules
module purge	unload all modules
module show <name>	show environment variables set by module <name>

How to use/install software applications

You can also manage software packages yourself (as always, check https://docs.cluster.uni-hannover.de):

- Build from source: use EasyBuild, or install software in a Singularity container
- Additional Python packages can be installed using pip or conda
  For "large" installations, you may want to consider changing the default installation directory

```
module spider Python/3
module load GCCcore/.12.2.0 Python/3.10.8
pip config list
mkdir -p ${SOFTWARE}/${USER}/pip
pip config set global.target "${SOFTWARE}/${USER}/pip"
pip install --user matplotlib
```

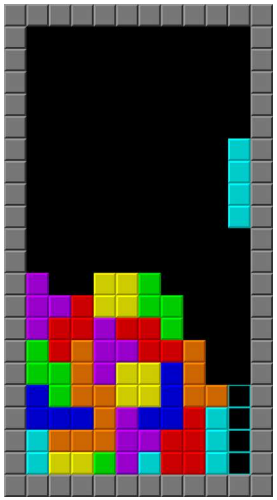
```
module load Miniconda3
conda config -show envs_dirs
[… conda config -prepend …]
conda create -n myconda python
conda search matplotlib
conda install matplotlib==2.4
```

```
module load Miniconda3
conda info --envs
conda activate myenv
```

Module usage commands

```
# find available octave modules
module spider octave

# determine how to load the selected Octave/4.0.0 module
module spider Octave/4.0.0

# load prerequisites and module itself
# as described by previous output
module load GCC/4.9.3-2.25 OpenMPI/1.10.2 Octave/4.0.0

# start the application
octave

# list modules loaded in the current shell
module list

# clean your environment - remove all modules
module purge
```

Submitting jobs — the SLURM workload manager



- Cluster compute nodes can not be accessed directly.
  Instead, a resource request must be submitted to the SLURM workload manager
  - Allows for efficient use of available compute resources
  - Users can execute multiple jobs in parallel
  - Job priority is not assigned on First In - First Out (FIFO) basis,
    but based on several factors that promote fairness.
- Use login nodes to submit SLURM jobs

salloc	allocate compute resources **interactively**
sbatch <script>	submit a **batch** script
srun	allocate compute resources and launch job-steps
squeue <--me>	check status of <own> jobs
scancel <jobid>	delete job with jobid
scontrol show job <job>	show details of a specific job
scontrol show nodes <nodename>	information about all <specific> compute nodes
sinfo	view information about nodes and partitions
sacctmgr -s show user	list limits relevant for you
sacct <jobid>	accounting information about jobs
slurmtop	text-based view of nodes' free/allocated resources and job status

Batch system limits

- Maximum walltime: 200 hours
- Maximum number of simultaneously running jobs per user: 64
- Maximum number of CPU cores used by running jobs per user: 768
- Maximum number of jobs in the queue per user: 500
- Maximum number of jobs in one array job container: 300

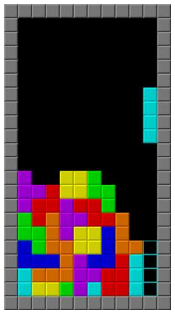
To list the job limits relevant for you, use the *sacctmgr* command

```
sacctmgr -s show user

sacctmgr -s show user format=user,account,maxjobs,maxsubmit,maxwall,qos
```

Serial batch job script (consult `man sbatch` for details)

```
#!/bin/bash -l
#SBATCH --job-name=my_serial
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2G
#SBATCH --time=0:42:00
#SBATCH --constraint=[skylake|haswell]
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --output my_serial-job-%j.out
#SBATCH --error my_serial-job-%j.err

# Change to my work dir
cd $SLURM_SUBMIT_DIR

# Load modules
module load my_modules


# Start my serial program
srun ./my_serial_app
```

Use shortest reasonable time needed for job + some reserve to get your job to execute as soon as possible while allowing for variation of common system load



Only the email address assigned to the username can be used

By default, standard *output* & *error* are merged into the same file

By default, the files are put in the job's submit directory

Applications can be executed without *srun*, but then no accounting record per app run is available in SLURM DB.

An OpenMP parallel job script

```
#!/bin/bash -l
#SBATCH --job-name=my_openmp
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=2G
#SBATCH --time=11:30:00
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --output my_openmp-job-%j.out


# Load modules
module load my_modules


# Set the number of threads to the number of
# CPUs per task you requested for the job
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK


# Start my OpenMP program
echo "start of job: `date`"
srun ./my_openmp_app
```

If you request less than 12 hours, you have a high chance to run on an FCH partition during night hours.

Load your set of modules or use some other way of application activation, e.g. *conda*

Do NOT use srun for commands that are single-threaded by definition (e.g. mkdir, rm, ...)

An MPI parallel multinode job script

```
#!/bin/bash -l
#SBATCH --job-name=my_mpi
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=40
#SBATCH --mem=0
#SBATCH --time=2:30:00
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --output logs/my_mpi-job-%j.out
#SBATCH --error logs/my_mpi-job-%j.err

# Change to my work dir
cd $SLURM_SUBMIT_DIR

# Load your modules including MPI stack
module load GCC/11.2.0 MPI/4.1.1

# Start my MPI program
srun ./my_mpi_app
```

First fill complete nodes.
Only then consider expanding to multiple nodes (***intra***-node communication is usually faster than ***inter***-node communication)

If you are using a complete node, you may as well request all of its configured memory

Provide complete path to the *logs* directory, if files are not to be put in the job's submit directory

mpirun can also be used to run parallel apps instead of *srun*, but then no accounting record per app. run is created in SLURM DB. **Do NOT use srun or mpirun** for commands that are single-threaded by definition (e.g. mkdir, rm, ...)

Hybrid MPI-OpenMP job script

```
#!/bin/bash -l

#SBATCH --job-name=test_hybrid

#SBATCH --nodes=4

#SBATCH --ntasks-per-node=4

#SBATCH --cpus-per-task=10

#SBATCH --mem-per-cpu=4G

#SBATCH --time=2-00:00:00

#SBATCH --constraint=skylake

#SBATCH --mail-type=BEGIN,END,FAIL

module load GCC/11.2.0 MPI/4.1.1

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

mpicc -fopenmp test_hybrid.c -o test_hybrid
srun ./test_hybrid
```

4 *MPI ranks* per node

10 cpu cores per process, reasonable for Amo architecture

4GB per core

2 days run time. Default: 24h

Select nodes with *skylake* cpu type

Get notified about job begin, end and failure

Set the number of *threads* to the number of CPUs

Job steps: tasks which must be executed

A Job Array script

```
#!/bin/bash -l
#SBATCH --job-name=my_job_array
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2G
#SBATCH --array=1-20%5
#SBATCH --time=1-00:00:00
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --output my_array-job-%A_%a.out
#SBATCH --error my_array-job-%A_%a.err

# Change to my work dir
cd $SLURM_SUBMIT_DIR

# Load your modules
module load my_module

# Start my program
srun ./my_app $SLURM_ARRAY_TASK_ID
```

Submit 20 independent jobs, each requesting 4 CPU cores and 2GB memory per CPU for 24 hours. Only 5 jobs will run at a time.

%A refers to the master job ID
%a refers to the job IDs in the array

Environment variable SLURM_ARRAY_TASK_ID uniquely identifies each job in the job array

Some SLURM environment variables

SLURM sets many shell variables in the **running** job's environment that might be useful in job scripts. Find out **in a job** using **`set | grep SLURM`**

$SLURM_JOB_ID	Job id
$SLURM_JOB_NUM_NODE	Number of nodes assigned to the job
$SLURM_JOB_NODELIST	List of nodes assigned to the job
$SLURM_NTASKS	Number of tasks in the job
$SLURM_NTASKS_PER_CORE	Number of tasks per allocated CPU
$SLURM_NTASKS_PER_NODE	Number of tasks per assigned node
$SLURM_CPUS_PER_TASK	Number of CPUs per task
$SLURM_CPUS_ON_NODE	Number of CPUs per assigned node
$SLURM_SUBMIT_DIR	Directory the job was submitted from
$SLURM_ARRAY_JOB_ID	Job id for the array
$SLURM_ARRAY_TASK_ID	Job array index value
$SLURM_ARRAY_TASK_COUNT	Number of jobs in a job array
$SLURM_GPUS	Number of GPUs requested

Frequently used *sbatch/srun/salloc* options

Options	Default Value	Description
–nodes=<N> or -N <N>	1	Number of compute nodes
–ntasks=<N> or -n <N>	1	Number of tasks to run
–cpus-per-task=<N> or -c <N>	1	Number of CPU cores per task
–ntasks-per-node=<N>	1	Number of tasks per node
–ntasks-per-core=<N>	1	Number of tasks per CPU core
–mem-per-cpu=<mem>	partition dependent	memory per CPU core in MB
–mem=<mem>	partition dependent	memory per node in MB
–gres=gpu:<type>:<N>	-	Request nodes with GPUs
–time=<time> or -t <time>	partition dependent	Walltime limit for the job
–partition=<name> or -p <name>	none	Partition to run the job
–constraint=<list> or -C <list>	none	Node-features to request
–job-name=<name> or -J <name>	job script's name	Name of the job
–output=<path> or -o <path>	slurm-%j.out	Standard output file
–error=<path> or -e <path>	slurm-%j.err	Standard error file
–mail-user=<mail>	your account mail	User's email address
–mail-type=<mode>	-	Event types for notifications
–exclusive	nodes are shared	Exclusive acccess to node

SLURM Exercises

- Submitting a job
  - Interactive
  - batch

How to get help

If it does not run

- Don't panic 😇 A cluster is a complex system. Others have failed before.
- Analyze error messages, check logs, step through your program and batch script, try to narrow down what may have failed. Do your own homework.
- Did we mention to read the cluster documentation? https://docs.cluster.uni-hannover.de
- Check the FAQ page. Check your disk quota.
- Use the "Getting help" page in the docs to create a case that helps us help you
- We are in no way experts on your specific usage of specific software packages (Ansys, Abaqus, Comsol, Matematica, Matlab, …). Sometimes we can guess, but only if you really tell us what you do.
- You can also call us by phone (number listed on the docs page), but we often need to figure out for ourselves first what could have failed.
- To get, change or delete accounts, ask the project management at your own institute
- If you find out something that you think could help others – tell us, so we can improve the documentation
- Wir sprechen natürlich auch Deutsch. Your choice whether you want to contact us in English or German