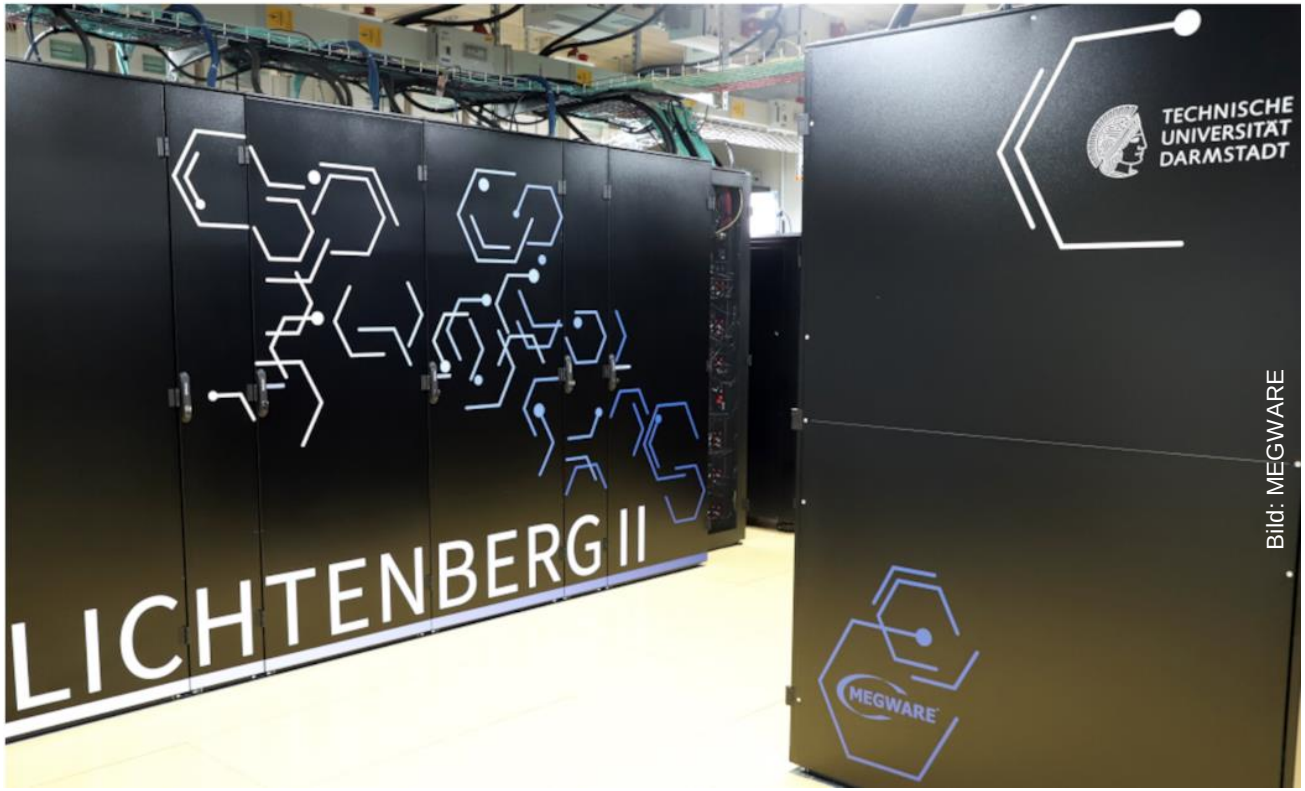
Introduction to the *Lichtenberg* High Performance Computer

TECHNISCHE UNIVERSITÄT DARMSTADT

November 14, 2023, 10:00–13:00 | *online*

Documents: www.hrz.tu-darmstadt.de/hlr → Training & Support → Events → Introduction

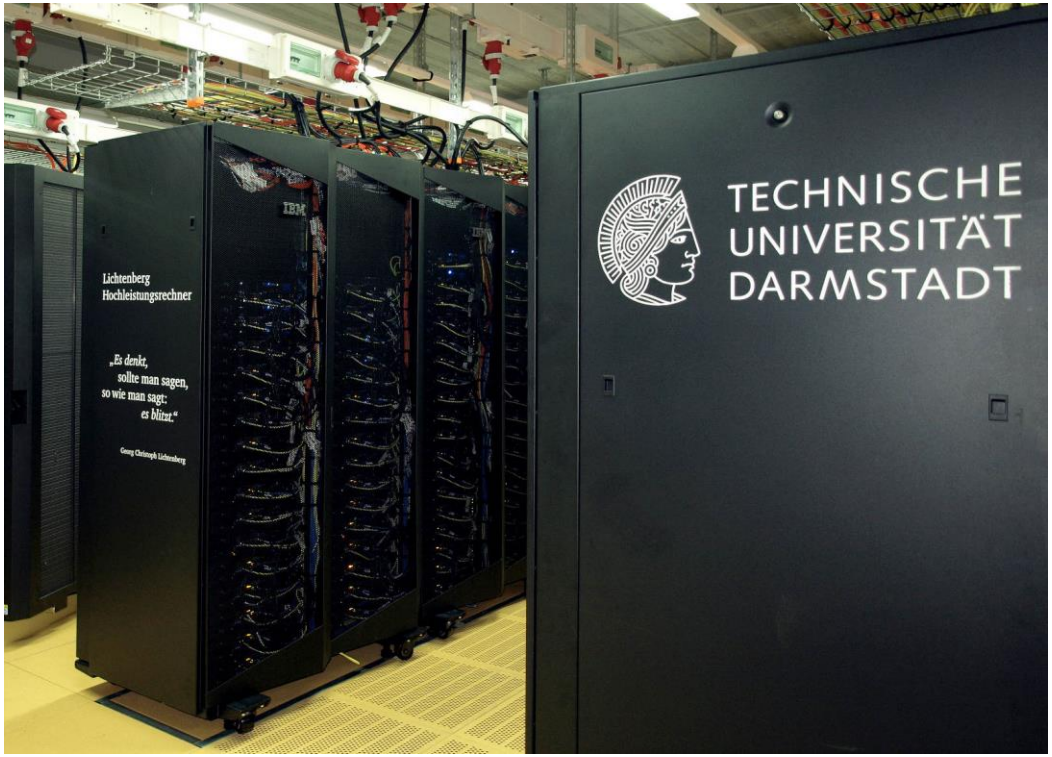


HRZ Hochschulrechenzentrum

Introduction to the *Lichtenberg* High Performance Computer

TECHNISCHE UNIVERSITÄT DARMSTADT

Infrastructure / How to get Access / Basic Usage



HRZ Hochschulrechenzentrum

Georg-Christoph LICHTENBERG

1st July 1742 – 24th February 1799

TECHNISCHE UNIVERSITÄT DARMSTADT

- born in Ober-Ramstadt near Darmstadt
- 1763 studies at the University of Göttingen
- 1769 extraordinary professor of experimental physics,
  1775 ordinary professor of experimental physics
- first to introduce life experiments in lectures
- 1793 member of the Royal Society in London
  (≙ british Academy of Sciences)

Famous quotes:

- “It thinks, one should say, the way one says it flashes.”
- "I thank the Lord a thousand times for having made me become an atheist."

HRZ Hochschulrechenzentrum

Starting point

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Researchers who plan to use HPC resources when devising their overarching research project
2. Researchers who discover during their research project that their local computing infrastructure is not sufficiently powerful to pursue all their goals
3. Lecturers who want to provide a HPC training/course/workshop

Do not worry about € $ ¥ £ – using the cluster is free of charge!

No hidden costs, no strings attached, neither for you nor your organization.

Can I use the Lichtenberg HPC for my research?

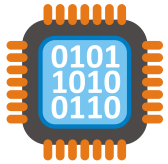
How to proceed

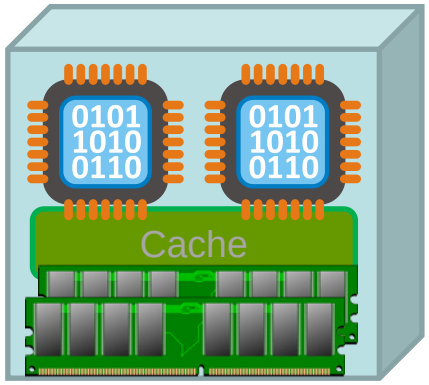
TECHNISCHE UNIVERSITÄT DARMSTADT

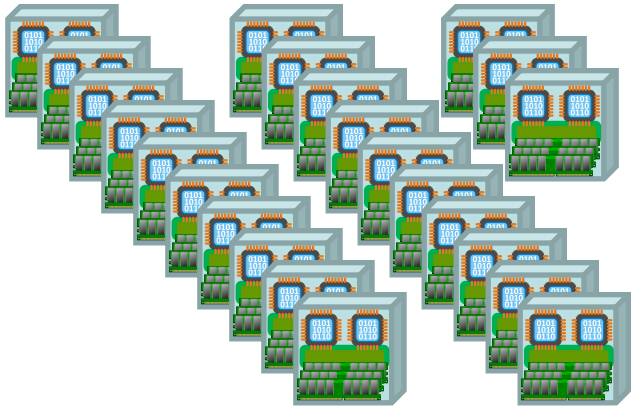
1. Are my computational requirements compatible with the Lichtenberg HPC resources?
   - Do I have a *parallelized* linux program to solve my scientific problem?
   - Do I have a serial linux program, but *sufficient distinct tasks* to solve my scientific problem?
2. Submit a **project proposal** for the Lichtenberg HPC
3. Get a **user account** for Lichtenberg HPC
   (Nutzungsantrag – only available in German)
4. Do the computations
   (more details in the 2nd part of the introductionary course.)

HRZ Hochschulrechenzentrum

Lichtenberg – A Parallel, Yet *Hierarchical* System







Processor = CPU (known from your desktop PC)

- Multiple cores per CPU (in our nodes typically: 48)
- Shared memory (all cores access the very *same* memory chips)

Compute node (similar to a better workstation or server)

- Server with multiple processors (typical node: 2)
- Very fast connection between processors and memory
- Shared memory between processors

Cluster

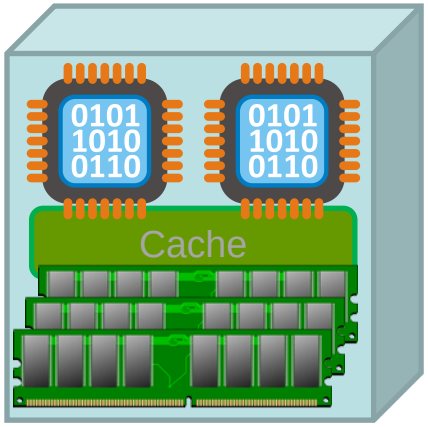
- All compute nodes as a battalion, commanded by a sophisticated scheduler
- Very fast interconnect between the nodes for high-throughput and low latency communication
- *No* shared memory between nodes

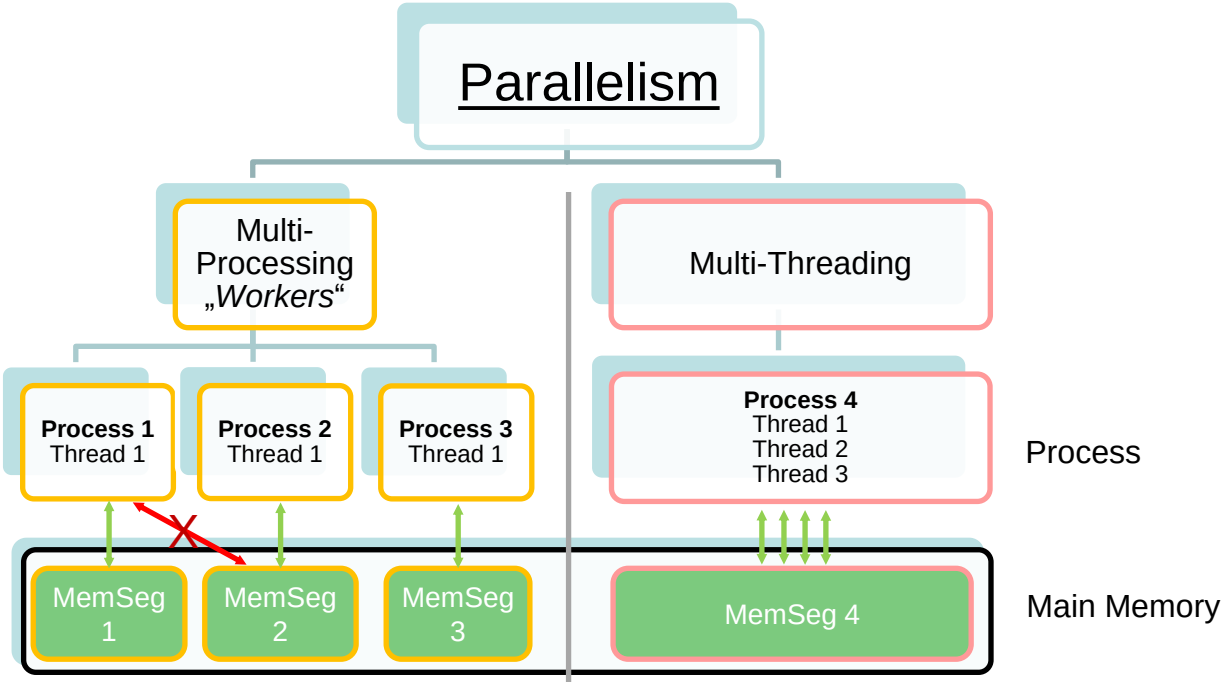
Parallelisation (1)

Serial vs. Parallel, Process vs. Thread

TECHNISCHE UNIVERSITÄT DARMSTADT

inside a Compute Node





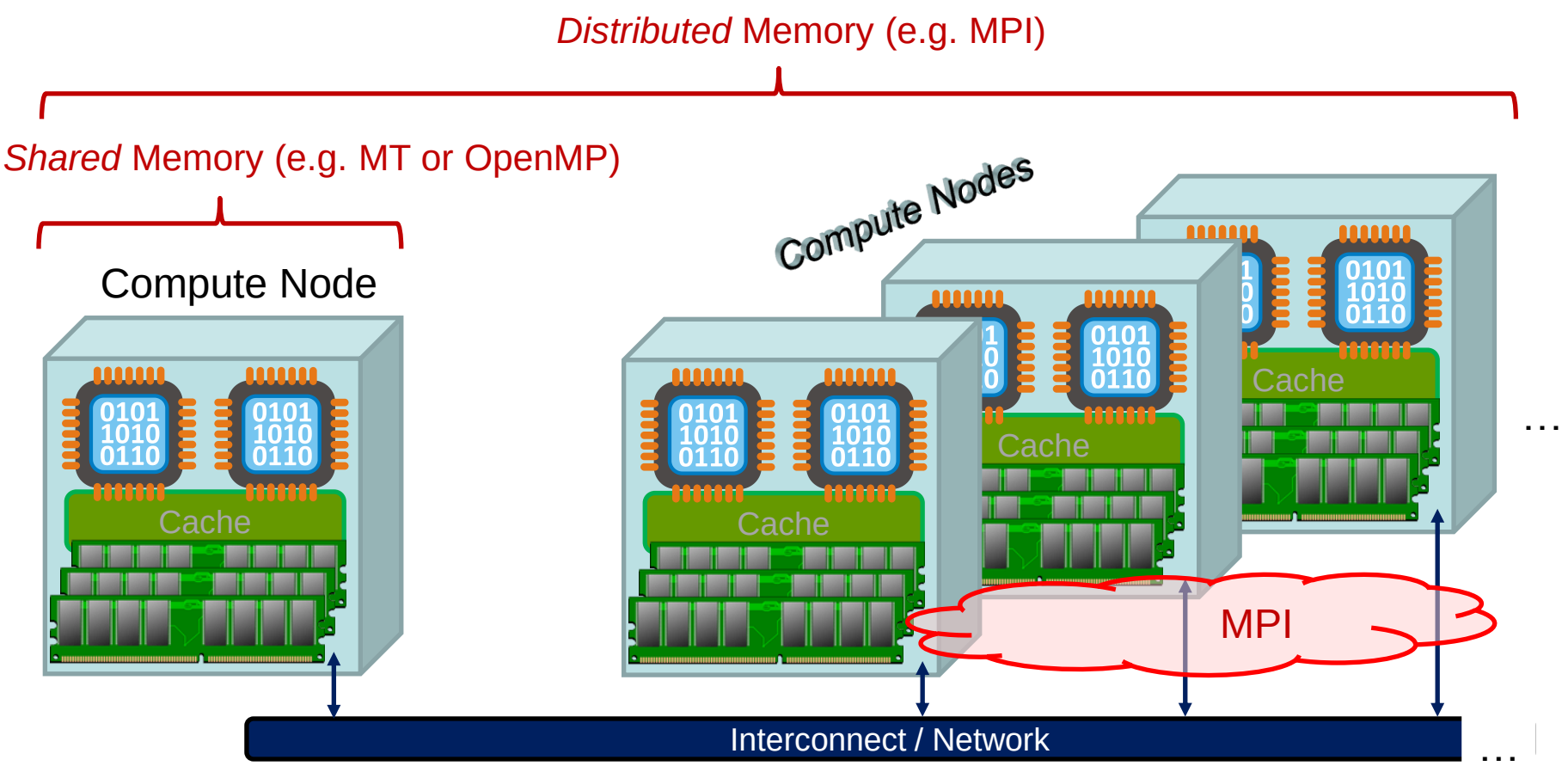
A serial program can be run *concurrently*, but only in *distinct* instances (“workers”). Even if running on the *same* computer, the distinct workers *cannot* access the memory segments (data) of their “siblings”.

A multithreaded program can use *more* CPU cores on a given compute node in parallel. All its threads can access the very *same* memory segment, i.e. can work in parallel on the *same* data.

HRZ Hochschulrechenzentrum

Parallelisation (2):
Shared and distributed memory

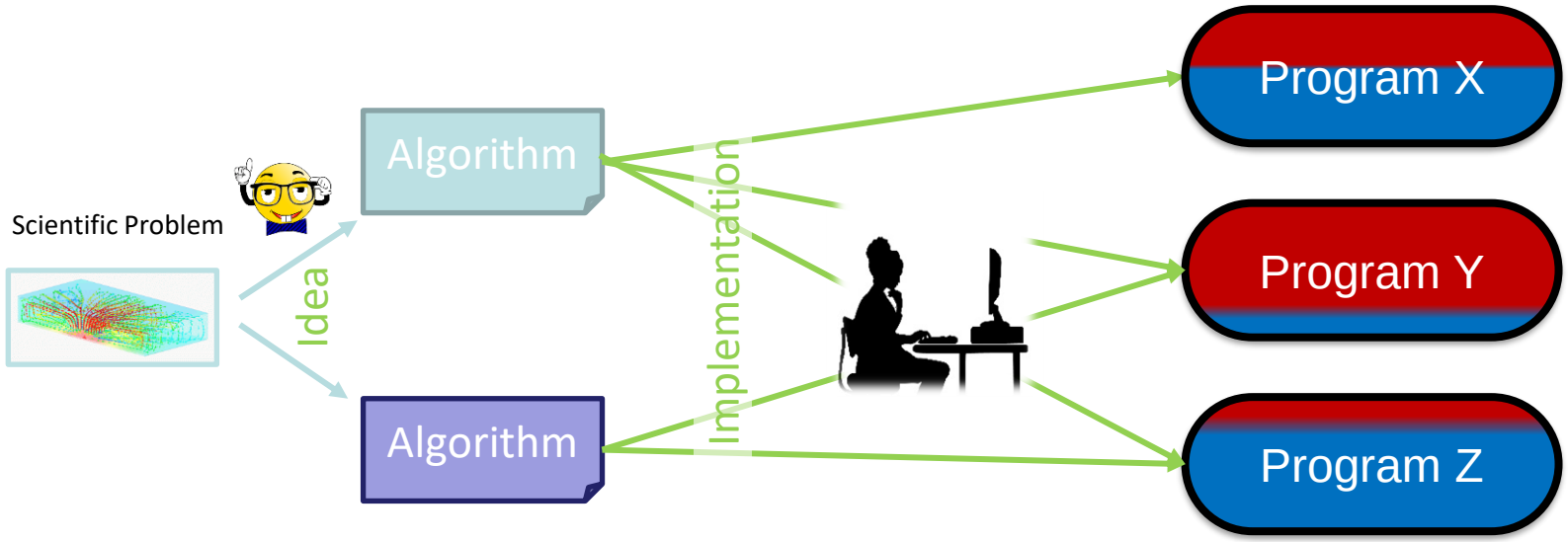
TECHNISCHE UNIVERSITÄT DARMSTADT

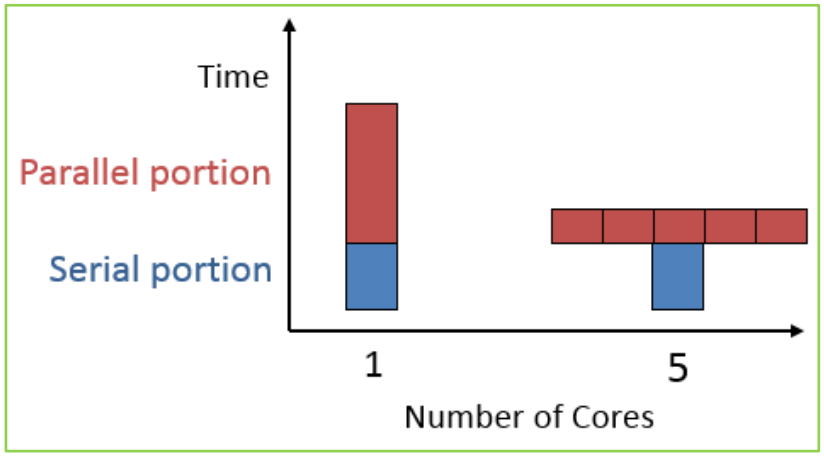


MT = *Multi-Threading (only „inside“ a node)*
OpenMP = *Open Multi Processing (only „inside“ a node)*
MPI = *Message Passing Interface („inside“ and also „across“ nodes)*

HRZ Hochschulrechenzentrum

Parallelisation (3):
Scalability





AMDAHL's Law: Only the *parallel* portion of a program benefits from more CPU cores.

Your mileage may vary.

Parallelisation (4):

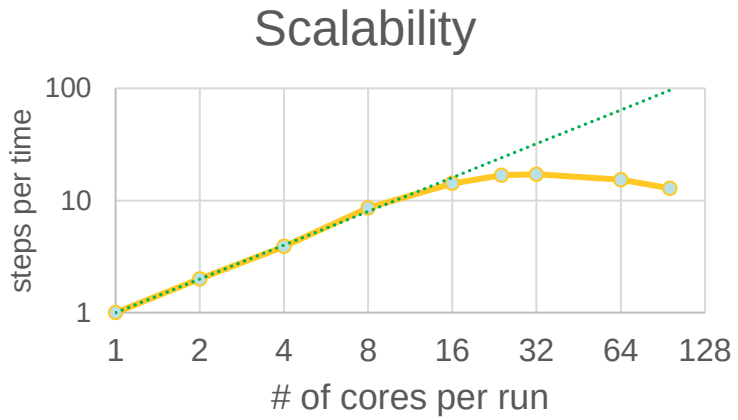
Scalability – what to gain with more CPU cores?

TECHNISCHE UNIVERSITÄT DARMSTADT

Strong scaling: speed-up when using more CPU cores by *keeping* the *same* problem size (CPU-bound algorithms).

Weak scaling: speed-up when scaling up both number of CPU cores *and* problem size (Memory-bound algorithms).

- choose an *exemplary representation* of your simulation or calculation sized so that 4 CPU cores can complete it within a reasonable time frame
- run this set with 4, 8, 16, 24, 32, 64, 96, … cores and measure your speed-up (eg. in “number of atoms analyzed” or “simulation steps completed per single run” or any other meaningful measure)

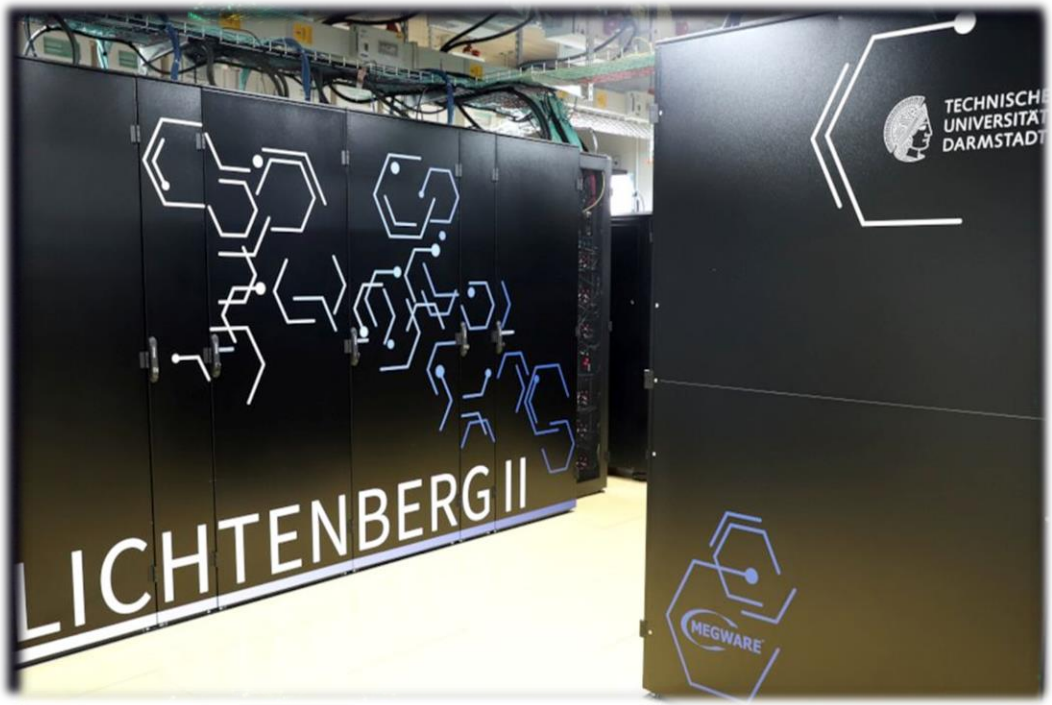


This example suggests to run the analyses with **16** or **24** cores per single run.

HRZ Hochschulrechenzentrum

Lichtenberg HPC – a tightly coupled cluster of compute nodes

TECHNISCHE UNIVERSITÄT DARMSTADT



1 hour runtime of the cluster at full load:
→ ~ 620 kWh
→ ~ 78 liters of gazoline
→ ~ 1,000 km journey by car

- lots of compute nodes (*independent* servers) running **Linux**
- very fast (and expensive) interconnect (**Infiniband**) with high bandwidth and low latency
- unified management
- computing *non-interactively* on *distinct* tasks (batch jobs)
- "job scheduler" distributes your tasks to suitable nodes, according to your job specifications

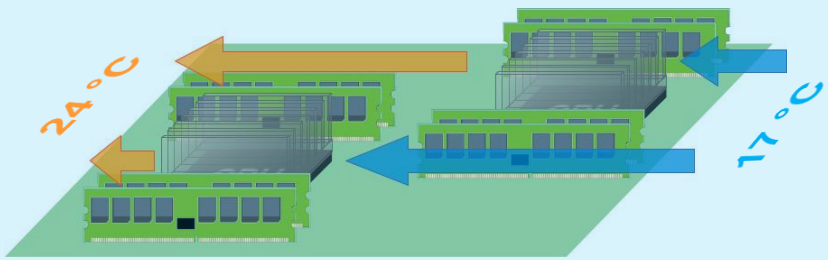
HRZ Hochschulrechenzentrum

Energy Efficiency
Hot Water Cooling

TECHNISCHE UNIVERSITÄT DARMSTADT

Formerly

legacy cooling by air

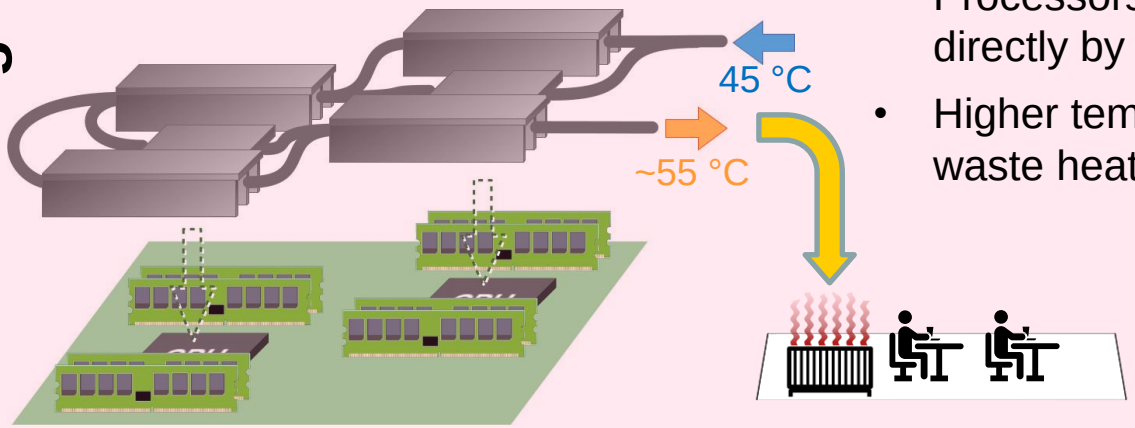


- Waste heat of processors and memory carried off by cool air
- Heated air is chilled down by air conditions wasting extra energy and heat going in the environment

Temperatures	
inlet	outlet
17 °C	24 °C

Lichtenberg

hot water cooling



- Processors and memory modules cooled directly by cooling fluid
- Higher temperatures allow for reuse of waste heat

Temperatures	
inlet	outlet
45 °C	52 - 55 °C

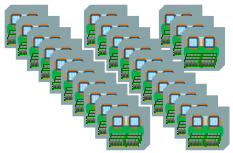
HRZ Hochschulrechenzentrum

Compute Nodes

Details on https://www.hrz.tu-darmstadt.de/hlr → Operations → Hardware

TECHNISCHE UNIVERSITÄT DARMSTADT

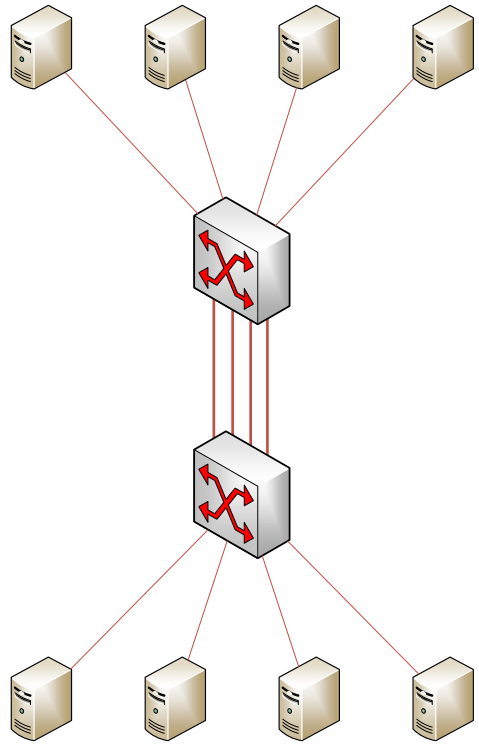
MPI section



630 nodes à

- 96 cores (2x Intel Cascade-Lake AP @2.3 GHz)
  - Intel® Virtualization Technology (VT-x)
  - Intel® TSX-NI
  - 2x Intel® AVX-512
  - VNNI (for DL/ML Inference)
  - ≥ 4 NUMA domains
- 384 GBytes RAM (2.933 GHz memory clock)

Lichtenberg 2



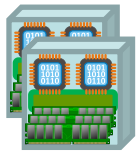
HRZ Hochschulrechenzentrum

Big-Mem Nodes

Details on https://www.hrz.tu-darmstadt.de/hlr → Operations → Hardware

TECHNISCHE UNIVERSITÄT DARMSTADT

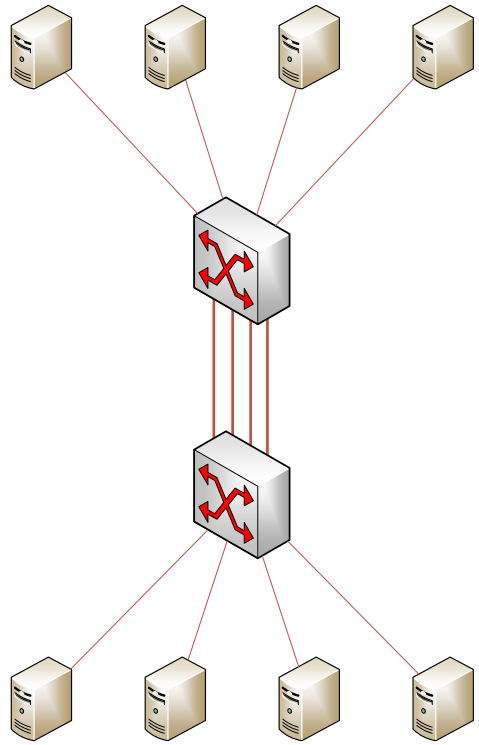
MEM section



2 nodes à

- 96 cores (2x Intel Cascade-Lake AP @2.3 GHz)
  - Intel® Virtualization Technology (VT-x)
  - Intel® TSX-NI
  - 2x Intel® AVX-512
  - VNNI (for DL/ML Inference)
  - ≥ 4 NUMA domains
- **1536 GBytes RAM** (2.933 GHz memory clock)

Lichtenberg 2



HRZ Hochschulrechenzentrum

Accelerator Nodes

Details on https://www.hrz.tu-darmstadt.de/hlr → Operations → Hardware

TECHNISCHE UNIVERSITÄT DARMSTADT

ACC section

4 nodes à

- 96 CPU cores (2x Intel Cascade-Lake AP @2.3 GHz)
- 384 GBytes RAM (2.933 GHz memory clock)
- **4x** GPU accelerator **Nvidia Volta 100**
  - 5 120 CUDA cores
  - 640 Tensor cores

4 nodes à

- 96 CPU cores (2x Intel Cascade-Lake AP @2.3 GHz)
- 384 GBytes RAM (2.933 GHz memory clock)
- **4x** GPU accelerator **Nvidia Ampere 100**
  - 8 192 CUDA cores
  - 432 TensorFlow32 cores

Accelerator Nodes

Details on https://www.hrz.tu-darmstadt.de/hlr → Operations → Hardware

TECHNISCHE UNIVERSITÄT DARMSTADT

ACC section

3 nodes (Nvidia DGX) à

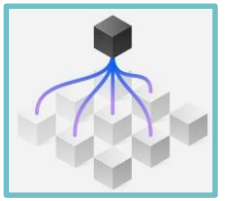
- 128 CPU cores (2x AMD EPYC 7742 @3.4 GHz)
- 1024 GBytes RAM (3200 MT/s)
- **8x** GPU accelerator **Nvidia Ampere 100**
  - 8 192 CUDA cores
  - 432 TensorFlow32 cores

Though primarily used for GPU codes, mind the *difference to all other compute nodes*:

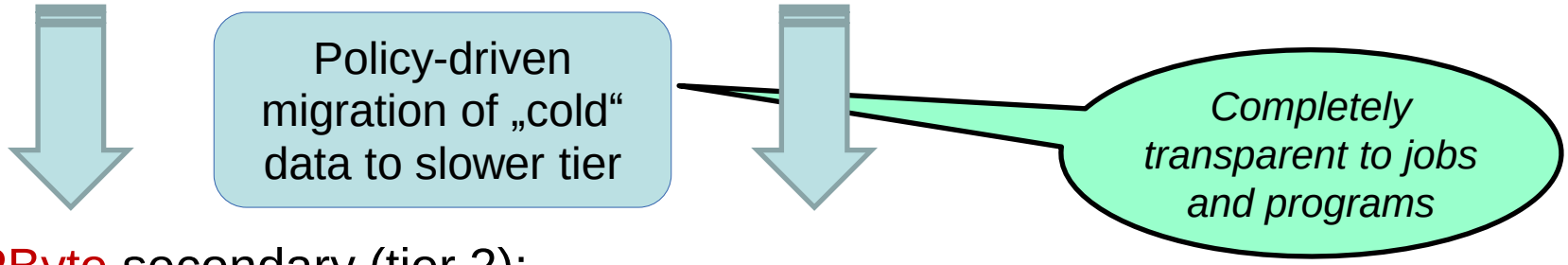
Codes / programs relying on AVX512 will ***not run here***, only those compiled with / expecting AVX2!

Storage Technology

TECHNISCHE UNIVERSITÄT DARMSTADT



- Massively parallel file system "*Storage Scale*" by IBM (formerly known as **G**eneral **P**arallel **F**ile **S**ystem)
- 2 PByte primary (tier 1):
  IBM/Lenovo ***E****lastic* ***S****torage* ***S****ystem* ("all-flash" with NVMe), directly connected to IB fabric (to all nodes)



- 3 PByte secondary (tier 2):
  IBM/Lenovo ***D****istributed* ***S****torage* ***S****olution* (legacy magnetic hard disks, SSDs only for metadata)

HRZ Hochschulrechenzentrum

File Systems

*Do not use home, groups or projects for **running** jobs!*

TECHNISCHE UNIVERSITÄT DARMSTADT

Mountpoint	**/work/home/**	**/work/groups/** **/work/projects/**	**/work/scratch/**
Symlinks	`/home`	`/home/groups/` -	`/work/scratch/`
Env. Variable	`$HOME`	-	`$HPC_SCRATCH`
Size	∑ 5 PByte		
Access time	Fast (GPFS via IB)		
Accessibility	**Global (cluster-wide)**		
Persistence	permanent		**8 weeks**
Quota*	40 GB** 5 Mio. files**	Individual	10 TB** 20 Mio. files**
Backup	→ tape, and snapshots		none
Usage Pattern	**low-volume I/O** (*static* input data, results of *finished* jobs)		**high-volume I/O** (*running* jobs' input/output, intermediary files)

More details on https://www.hrz.tu-darmstadt.de/hlr → Usage → File systems

* Use the command **`cquota`** to find out your current usage and quota.
** Can be increased on request.

HRZ Hochschulrechenzentrum

File System Best Practices

TECHNISCHE UNIVERSITÄT DARMSTADT

low-volume I/O

`/home/` `(=/work/home)`

- Compiling and saving of your own programs
- Saving of important data (results of computations)
- Job scripts and logs

`/work/{projects,groups}/`

- Compiling and saving of programs *shared* between project or group members
- Sharing of project- or group-relevant input and output files

high-volume I/O

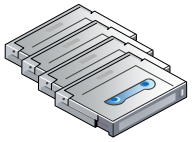
`/work/scratch/`

- Saving of large (intermediary) data
- Parallel input/output (ie. from *many* nodes concurrently) with high performance

Remember the deletion policy on files not read/written for >8 weeks!

HRZ Hochschulrechenzentrum

Backup



TECHNISCHE UNIVERSITÄT DARMSTADT

- Lichtenberg file systems are ***not*** intended as permanent storage for research data → ULB
- The **backup to tape** mechanism is ***not*** intended to recover your deleted files, only for recovery in case of a disaster
- To recover your deleted files in the low-volume I/O file systems, use the **snapshot** mechanism:
  - Go to the ***invisible*** folder "`.snapshots`" in your home or projects or group folder (also in each subdirectory)
  - There you will find several older snapshots of your **home** or **project** folder and directories below it
- **Beware**: there are absolutely **no** **backups/snapshots** for the **`/work/scratch`** file system!

HRZ Hochschulrechenzentrum

Recap

TECHNISCHE UNIVERSITÄT DARMSTADT

- lots of cores in lots of nodes available, but for starting out and assessing scalability, it is recommended to stay *within 4 to 8 nodes*, and thus to limit your scaling tests to a maximum of **768** cores in total
- ~3 PFLOP/s total computing power
  (LB2P1: #100 on Nov 2020's and #150 on Nov 2021's Top500 list of the most powerful computer systems)
- maximum # of cores per single node: 96 (128 on DGX, only avx2)
- maximum main memory per single node: 1.5 TByte
- maximum # of GPUs per Intel / avx512 node: 4
- maximum # of GPUs per AMD / avx2 node: 8
- 1+3 PByte shared file system

Lichtenberg HPC suits the computational needs of the project

HRZ Hochschulrechenzentrum

… what if not?

TECHNISCHE UNIVERSITÄT DARMSTADT

? Use an Institute Cluster (if available)

? Wait for Lichtenberg 2 Phase II, the next stage of expansion

? Use another HPC of the GAUSS-Allianz
(Tier 1 systems: Jülich, München, Stuttgart)

? Use a Commercial Cloud – Drawbacks:

- costs money
- usually no fast interconnect available
(only High *Throughput* Computing possible)

How to proceed

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Are my computational requirements compatible with the Lichtenberg HPC infrastructure?

   Yes. Then how to get access?

2. Submit a **project proposal** for the Lichtenberg HPC
3. Get a **user account** for the Lichtenberg HPC

   (Nutzungsantrag – only available in German)

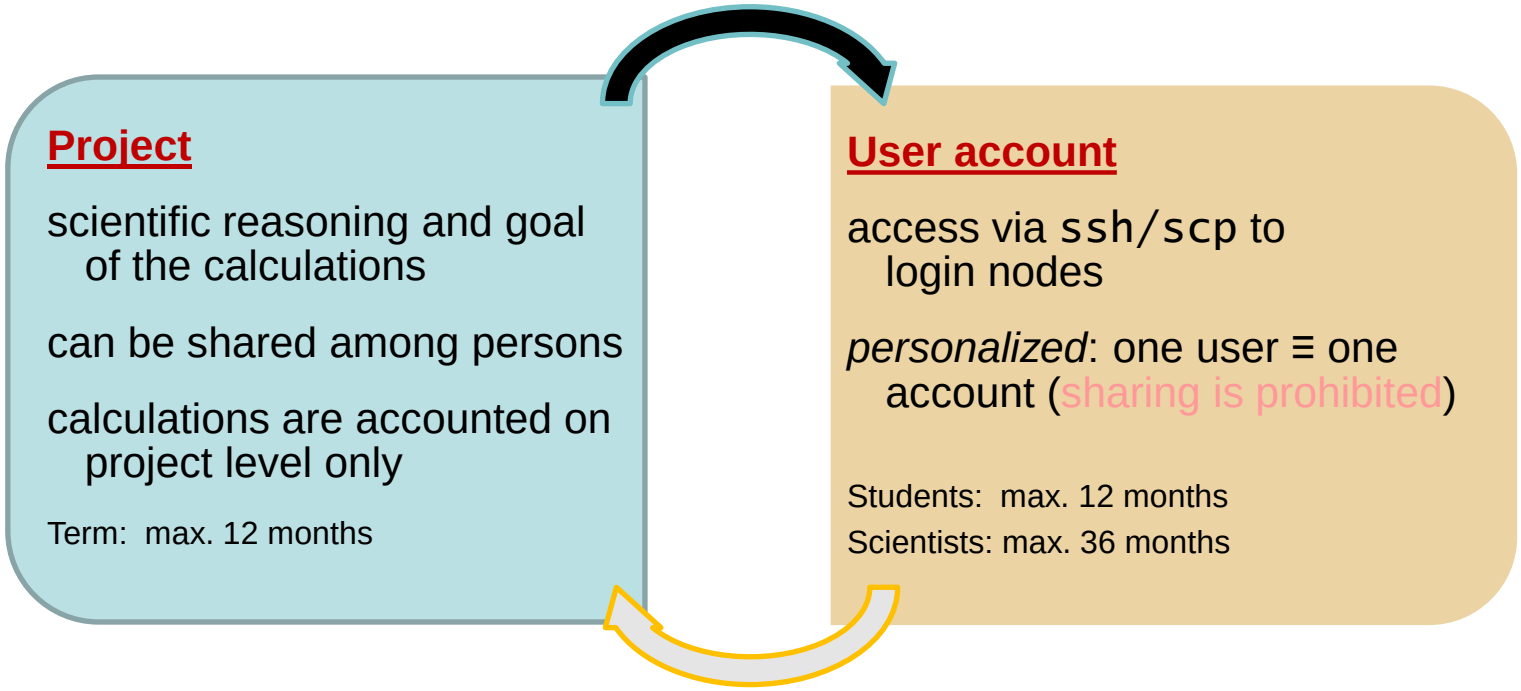
4. Do the computations

   (more details in the 2nd part of the introductionary course.)

Project vs. User Account

TECHNISCHE UNIVERSITÄT DARMSTADT

- **No fees** for users or their faculties
- No plain *commercial* usage, though collaboration projects in applied sciences are welcome, as long as the leading institution is an academic one



At least you need to apply for your personal **User Account**. For *doing* computations, you need to become member of an existing project or you apply for your own project.

HRZ Hochschulrechenzentrum

Project proposal – classes

TECHNISCHE UNIVERSITÄT DARMSTADT

SMALL ("local" projects)

Maximum of **360,000** core hours, also used for estimating resources and to test for larger proposals.

≙ about 1 Lichtenberg compute node for a year

TUDa-Portal

Nationales **H**ochleistungs**r**echnen

NHR **N**ormal

> 1 Mio to **< 12 Mio** core hours.

≙ about 10 Lichtenberg compute nodes for a year

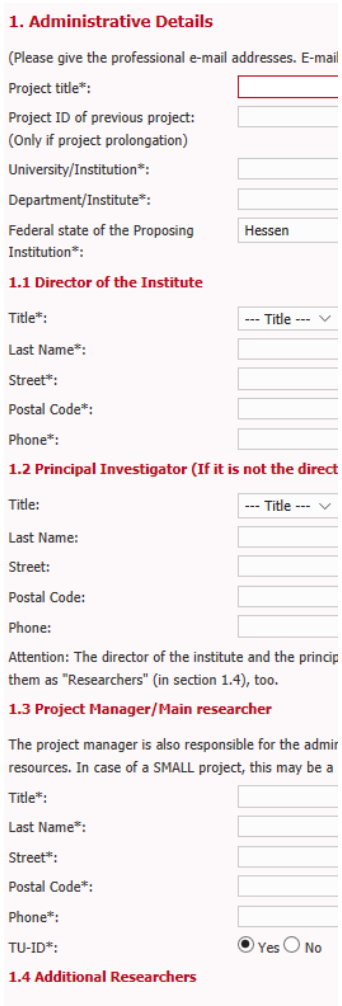
NHR-Portal

NHR **L**arge

> 12 Mio to **< 50 Mio** core hours.

≙ about 42 Lichtenberg compute nodes for a year.

Small Project – 1. Administrative details


1. Administrative Details

(Please give the professional e-mail addresses. E-mail

Project title*:

Project ID of previous project:
(Only if project prolongation)

University/Institution*:

Department/Institute*:

Federal state of the Proposing Institution*: Hessen

1.1 Director of the Institute

Title*: --- Title ---

Last Name*:

Street*:

Postal Code*:

Phone*:

1.2 Principal Investigator (If it is not the direct

Title: --- Title ---

Last Name:

Street:

Postal Code:

Phone:

Attention: The director of the institute and the princip
them as "Researchers" (in section 1.4), too.

1.3 Project Manager/Main researcher

The project manager is also responsible for the admir
resources. In case of a SMALL project, this may be a

Title*:

Last Name*:

Street*:

Postal Code*:

Phone*:

TU-ID*: Yes No

1.4 Additional Researchers

Forms and templates are on the HRZ web page

https://hlr-hpc1.hrz.tu-darmstadt.de/pp

Most fields should be self explanatory

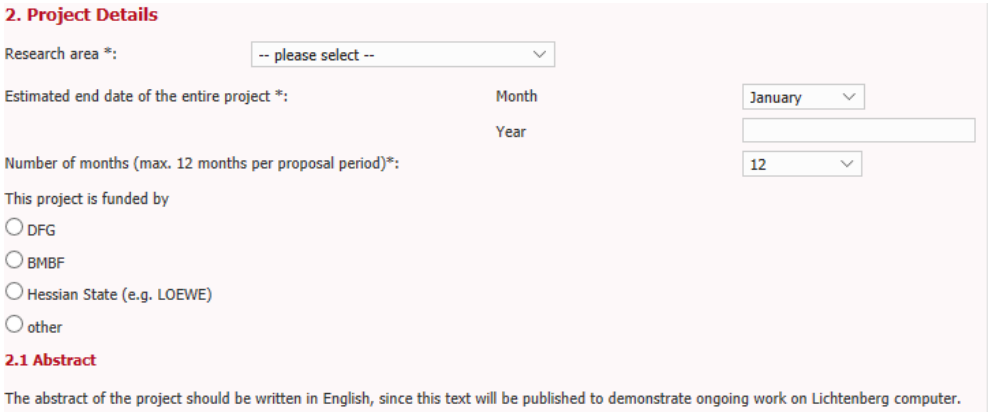
- for follow-up projects: please supply previous project ID
- **P**erson of **C**ontact (formerly, **P**roject **M**anager): usually the person we contact and who is responsible for the handling of the project.
- **P**rincipal **I**nvestigator: 1st researcher working on the project
- **D**irector of the **I**nstitute – needs to sign the project
- **project members** can always be added later by the PoC, while the project is running

Reminder: use only professional/institutional eMail addresses!

Small Project – 2. Project details

TECHNISCHE UNIVERSITÄT DARMSTADT

- The HPC project refers to the overarching *research* project.
- A project on the Lichtenberg HPC can last max. 12 months. If your research takes longer, follow-up proposals have to be submitted.



2.1 *Scientific Abstract*: is made public on the HRZ/HKHLR web page and in HPC reports, so please write something generally intelligible that represents you and your research appropriately (< 300 characters).

HRZ Hochschulrechenzentrum

Small Project – 3. Technical description

TECHNISCHE UNIVERSITÄT DARMSTADT

3.1 Core hours (accelerator hours will count 24x) → project class

3. Technical Description

Please have a look at the hardware overview of the HRZ first, and verify that you really need a machine of this size for your project and that the required resources are available. The CPU time requirement for MIDDLE and LARGE project class must be justified in a detailed project description.

In the following, always count the number of individual cores that your program will need (estimated).

3.1 Project class

CPU time (without accelerators, in core-hours)*: 500000

(This is the total compute time for your project, e.g.: wall clock time for parallel application*number of cores*number of runs + testing + etc.)

Accelerator Type (in card-hours):
The following accelerator card hours are additional compute hours. For the compute hours we need to account 8 cores per card in average. In case you only want to use accelerator cards (and corresponding host CPUs), please let the above field empty (zero).

NVIDIA: 50000 Xeon Phi:

Total CPU time (in core-hours): 900000 (Project class: MIDDLE)

3.2 Detailed resource requirements of the project

Base architecture of the estimated core hours above*: AVX 2

Required CPU architecture for your jobs*: AVX 2

(Total for home directories. Change this default value only if it is really necessary! This is an expensive resource. Files are backed up regularly.)

Home (in GB, default 10 GB)*: 50

Independently from the proposal each user has access to the /work/scratch area (with a default quota of 100 TB). In case your project needs more, please contact us independently via email request.

3.2 Special requirements (eg. avx512 or more space in `/home`)

Small Project – 3. Technical description

TECHNISCHE UNIVERSITÄT DARMSTADT

3.3 What does your *typical* **single batch job** look like in terms of:

- number of CPU cores
- memory – rough estimate in GBytes per CPU core (justification required if more than 5 GB / core are needed)
- duration (“wall clock time”) – ideally **≤ 24h** (max 168h = 7d)

3.3 Resource requirements of a typical single batch run

For a typical single run (as well as for a typical interactive run), fill in the expected average values in the production phase of your project (in contrast to developing and debugging phases). Of course, the numbers can only be estimated. Use section 3.6 (Special Requirements) for extraordinary maximum resource requirements.

Number of cores*:	512
Main memory **per core** (in GB)*:	1

If you need more than 5 GB main memory per core, you have to provide a special statement (concerning e.g. limited scalability of the code or model with respect to memory, communication, or algorithm bottlenecks; number of available software licences; benchmarking):

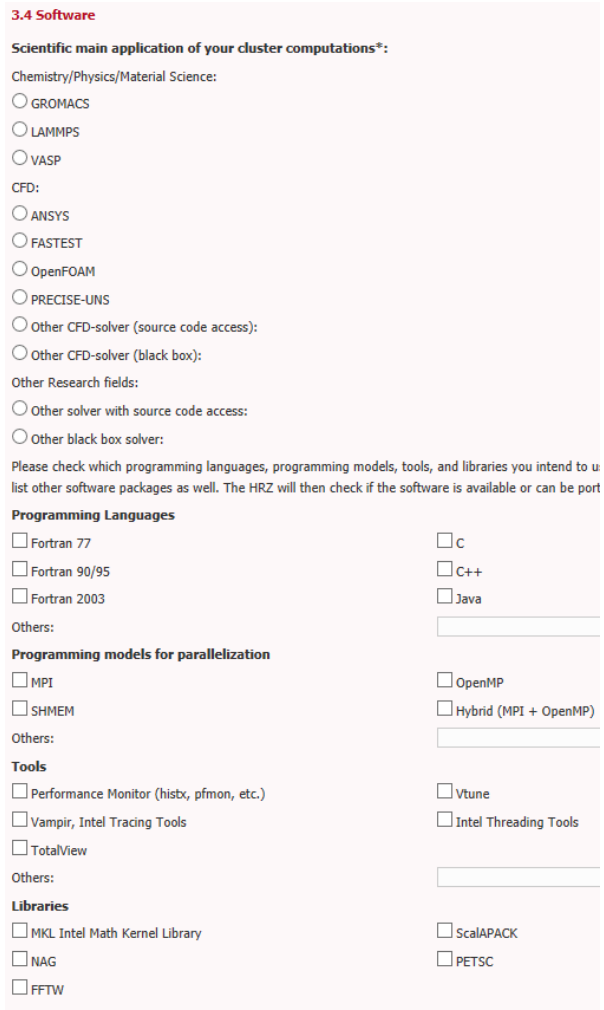
Run time (wall-clock time, in hours)*:	24

If you really need more than 24 hours, provide the reason and give a statement about checkpointing and restart capabilities of your application:

this is just a prospect–the „truth“ is what your later job submission scripts set as requirements!

HRZ Hochschulrechenzentrum

Small Project – 3. Technical description



3.4 Software

Scientific main application of your cluster computations*:

Chemistry/Physics/Material Science:

- ○ GROMACS
- ○ LAMMPS
- ○ VASP

CFD:

- ○ ANSYS
- ○ FASTEST
- ○ OpenFOAM
- ○ PRECISE-UNS
- ○ Other CFD-solver (source code access):
- ○ Other CFD-solver (black box):

Other Research fields:

- ○ Other solver with source code access:
- ○ Other black box solver:

Please check which programming languages, programming models, tools, and libraries you intend to u
list other software packages as well. The HRZ will then check if the software is available or can be port

Programming Languages

- ☐ Fortran 77
- ☐ Fortran 90/95
- ☐ Fortran 2003
- ☐ C
- ☐ C++
- ☐ Java

Others:

Programming models for parallelization

- ☐ MPI
- ☐ SHMEM
- ☐ OpenMP
- ☐ Hybrid (MPI + OpenMP)

Others:

Tools

- ☐ Performance Monitor (histx, pfmon, etc.)
- ☐ Vampir, Intel Tracing Tools
- ☐ TotalView
- ☐ Vtune
- ☐ Intel Threading Tools

Others:

Libraries

- ☐ MKL Intel Math Kernel Library
- ☐ NAG
- ☐ FFTW
- ☐ ScalAPACK
- ☐ PETSC

- Software
- Programming languages
- Parallelization models
- Tools and libraries

  (mostly for our analytics)

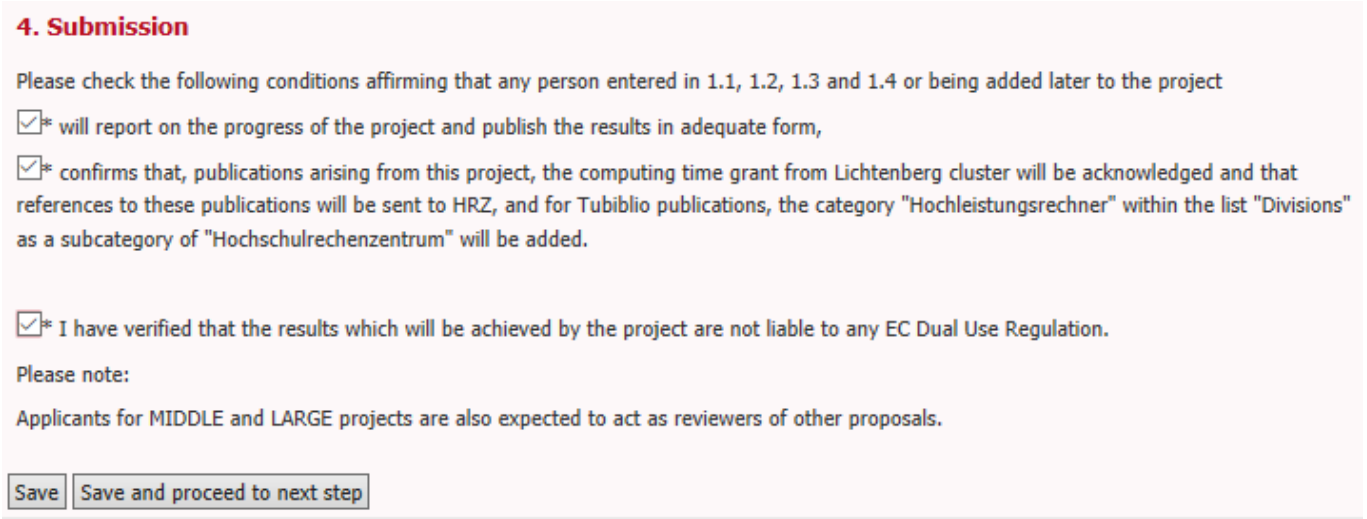
- Any other special requirements

Small Project – 4. Submission

TECHNISCHE UNIVERSITÄT DARMSTADT

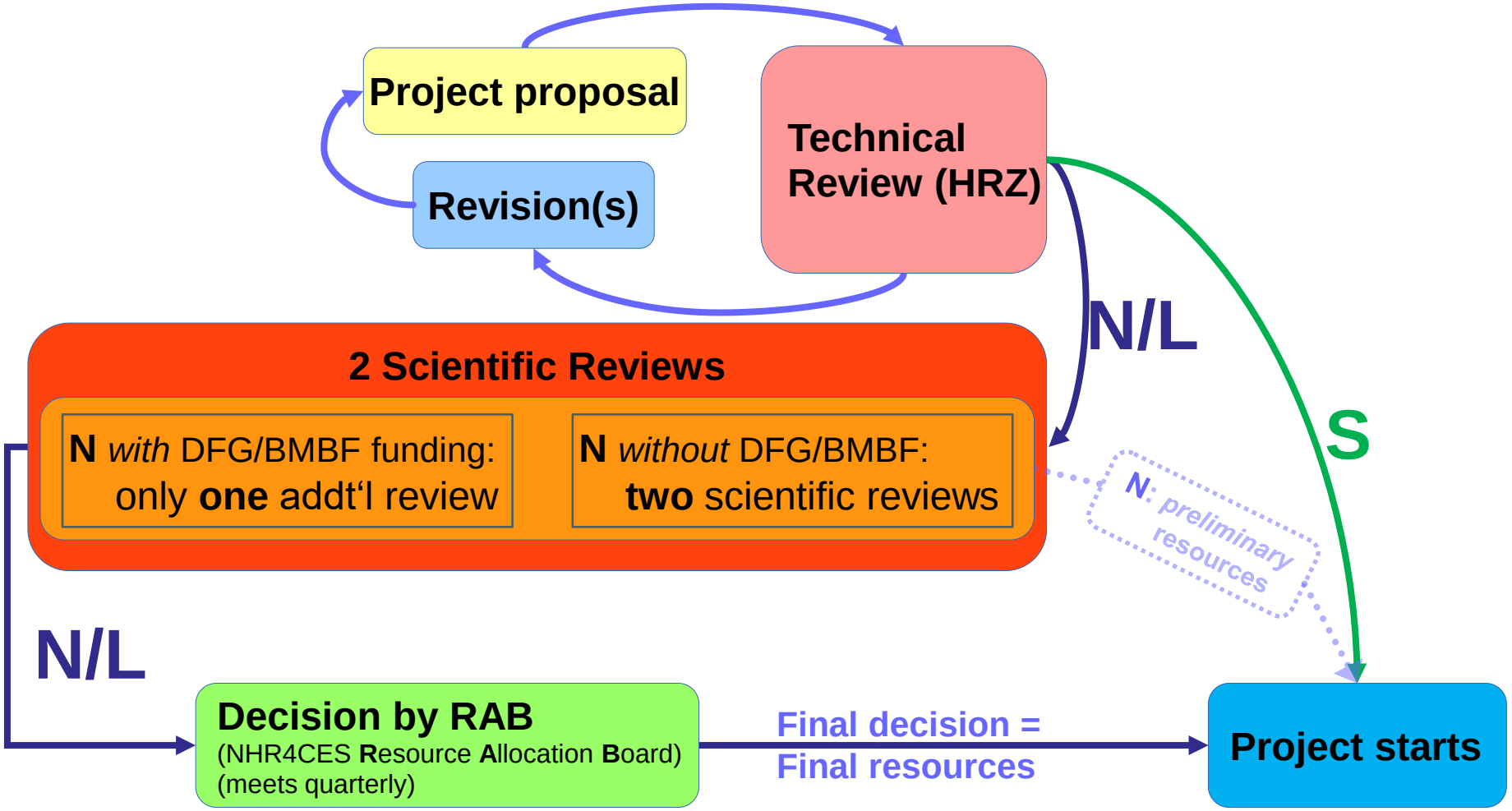
4 Confirm your obligations to:

- write and submit a **final report** after the HPC project is finished (ie. each year)
- acknowledge the Lichtenberg cluster in your publications, notify the HRZ about publications by including them in your final report (TU-Da faculty members: add the proper category “Hochleistungsrechner” for TUBiblio publications)
- comply with the European Commission’s **Dual-Use Regulations**



HRZ Hochschulrechenzentrum

Project Review Process



Final steps

TECHNISCHE UNIVERSITÄT DARMSTADT

- HRZ may give feedback on your proposal and can re-open the web form for the requestor to upload a revised proposal.

- Once all modifications are done and submitted, HRZ sends back a checksum'ed PDF of the proposal.

- this PDF then needs to be ***printed*** and ***signed*** (usually by PI or DI), and sent back via office post to the HRZ.
  Due to the COVID-19 pandemic, we accept scans via eMail/ticket to open the project preliminary, but for legal reasons, we need the paper original nonetheless).

Approval, and your active projects

TECHNISCHE UNIVERSITÄT DARMSTADT

- After **T**echnical **R**eview and signatures of a **S**mall project are completed, you get a mail "*HLR Project ID ... Decision about computing time request*" with a recap of the project's details
- The *next* day, you can submit jobs to your new project, even though its official starting date might be next month

Notification Project starts

Your **currently active projects** and their runtime are updated nightly into the `$HOME/.project` file. To see them, use

```
cat ~/.project
```

or

```
member
```

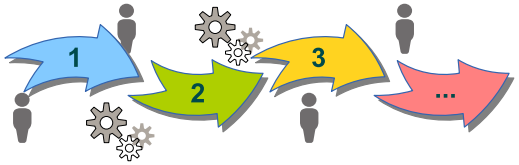

HRZ Hochschulrechenzentrum

Project memberships

TECHNISCHE UNIVERSITÄT DARMSTADT

Since Nov 2022, you can use a new *self-service* for requesting and controlling memberships in (your) HPC projects:

The **member** command



% **member**

lists all your **active projects**

% **member -p <***project01234***> -m**

lists this project's manager / coordinator – ie. whom to ask for becoming member

% **member add -p <***project01234***> -u TU-ID** (only allowed by **PoC/PM**)

makes **TU-ID** a new member of this project

HRZ Hochschulrechenzentrum

How to proceed

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Are my computational requirements compatible with the Lichtenberg HPC infrastructure?
2. Submit a **project proposal** for the Lichtenberg HPC
3. Get a **user account** for the Lichtenberg HPC
   (Nutzungsantrag – only available in German)
4. Do the computations
   (more details in the 2nd part of the introductionary course.)

HRZ Hochschulrechenzentrum

How to get access – user regulation

TECHNISCHE UNIVERSITÄT DARMSTADT

- do *not* share your account! One account ≡ one person!
- no plain commercial projects (though academic-industry cooperations eg. in applied sciences are okay)
- „dual use“ projects (military/weaponry research) not allowed
- embargo regulations for several countries
- LB2: not only Hessia, but nation-wide use, due to funding from the federal ministry of education and research

How to get access – User application / Nutzungsantrag

TECHNISCHE UNIVERSITÄT DARMSTADT

- available on the HRZ web page
- only in German
- ***print*** and fill out *the last **2** pages 7+8*
- have it signed by the director of the institute
- send the signed form to us (HRZ) via office post (due to the current COVID-19 pandemic, we accept scans via eMail/ticket, but need the signed original nonetheless)

If the new user is *not* from the TU Darmstadt:

- get a “guest TU-ID” first
- send the signed form to your *local contact person* (list of contact persons is available on the HRZ web page)

Recap of the “paper work”

TECHNISCHE UNIVERSITÄT DARMSTADT

1.) **Project**

- scientific reasoning of computations
- scientific abstract to be published
- **final report required**!
- **acknowledgment** of LB computing time grant in all publications required!

Small:

Max **360 000** core hours - about ½ Lichtenberg Compute Node / year

NHR **N**ormal:

1 – 12 Mio core hours - about **1-15** Lichtenberg Compute Nodes / year

NHR **L**arge:

12 – 50 Mio core hours - about **15-60** Lichtenberg Compute Nodes / year

use `cat $HOME/.project` or `member` to see your actives

distinct forms & processes & validity terms

2.) **User account**

- independent of project!
- personalized (TU-ID)
- necessary for *each* researcher doing computations
- expiry time independent of project’s runtime

- do **not share** your account with others!
- accounts do expire (you get a warning in due time):
  students: max. 12 months
  scientific staff: max. 36 months
  use `/shared/bin/account_expire` to see yours
- any user is member of *one or several* projects, and projects can have *several users* – that’s why there are different terms and no unison between them.

HRZ Hochschulrechenzentrum

Questions?

TECHNISCHE UNIVERSITÄT DARMSTADT

Coffee break and quiz after this slide…

Good to know: even after approval and start of a project:

- any number of *additional* researchers/students/co-workers can be added to a running project (see the '`member`' command)
- Assigned core hours can be adjusted during a project's runtime if the need arises (and is well reasoned for)
- your proposal's "main memory per core" and "number of tasks" is *not* a fixed ceiling. In your productive batch job scripts, you can freely request more `--mem-per-cpu` or more `--ntasks` (though the latter is of course accounted to your project)
- Even if not initially requested in your project application, you can use GPU accelerators at will

HRZ Hochschulrechenzentrum

Coffee break

TECHNISCHE UNIVERSITÄT DARMSTADT

Introduction to the *Lichtenberg* High Performance Computer

Part 2: basics of resource allocation and job submission

We will continue in ~5-6 minutes!

Online Quiz (anonymous, non-tracking) for the break:

try “part 1” only

https://university.quizacademy.io/course/CIWBTC

HRZ Hochschulrechenzentrum

How to proceed

TECHNISCHE UNIVERSITÄT DARMSTADT

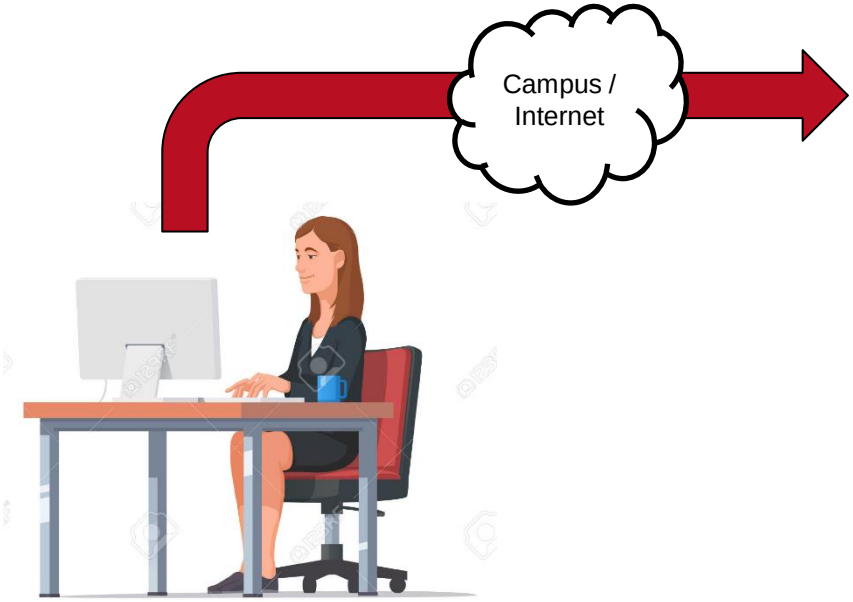
1. Are my computational requirements compatible with the Lichtenberg HPC infrastructure?
2. Submit a **project proposal** for the Lichtenberg HPC
3. Get a **user account** for the Lichtenberg HPC (Nutzungsantrag – only available in German)
4. Do the computations

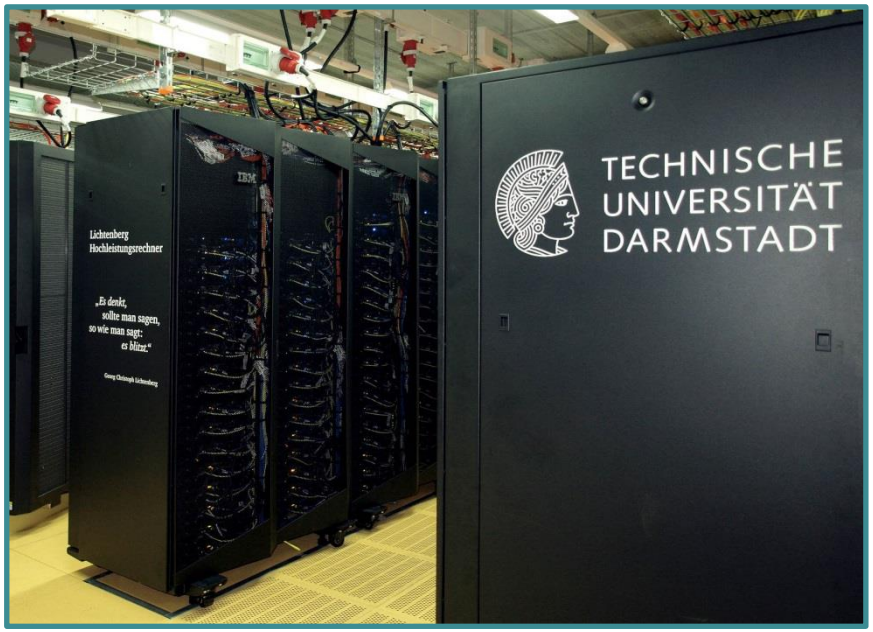
Remote access

TECHNISCHE UNIVERSITÄT DARMSTADT

You cannot (and would not want to) work directly in the HPC building on the Lichtenberg HPC.

→ Access the cluster **remotely**.



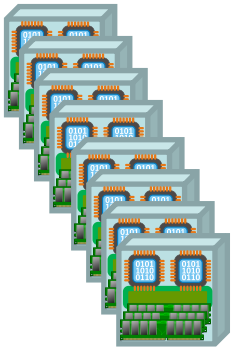


HRZ Hochschulrechenzentrum

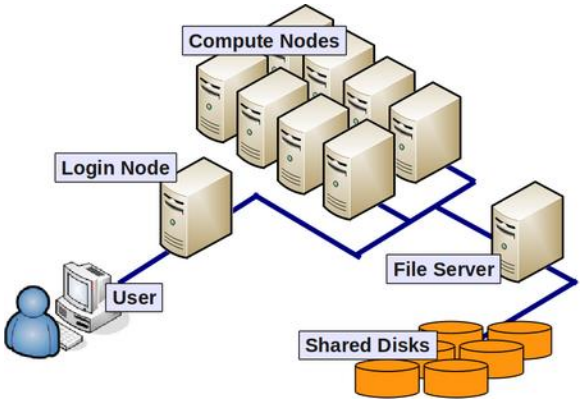
Login Nodes Lichtenberg 2

Details on https://www.hrz.tu-darmstadt.de/hlr → Operations → Hardware

TECHNISCHE UNIVERSITÄT DARMSTADT



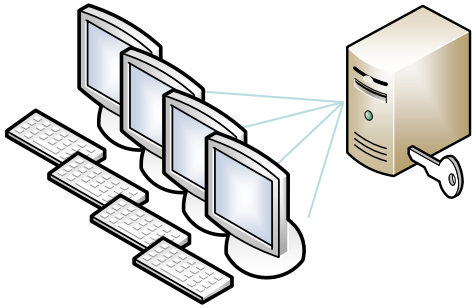
8 login nodes



- Processors:
  - 2 Intel Xeon Platinum 9242 processors
    Cascade Lake, AVX512
    ≙ 2 · 48 = **96 CPU cores**
  - **2.3 GHz**
    (up to 3.8 GHz in turbo mode)
- Main Memory:
  - **768 GB RAM**
- Network:
  - 2 · 10 Gigabit Ethernet
  - HDR-100 InfiniBand
- Accelerators:
  - **1x Nvidia Tesla T4 GPU**
    (every “odd”-numbered login node)

HRZ Hochschulrechenzentrum

Purpose of the Login Nodes



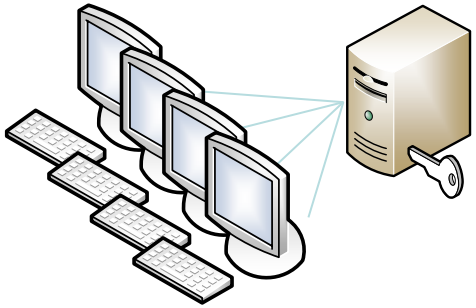
- Transferring files to and from the Lichtenberg HPC
  (there are no special "copy" or "I/O" nodes)
- Editing files on the cluster
- Compiling software
- Testing software **(time limit: 30 minutes** and **nice** or **<96 cores)**
- Debugging small test cases

- Submitting computations to the batch system (explained later)
- Checking status of compute jobs

Connecting to the Lichtenberg HPC

TECHNISCHE UNIVERSITÄT DARMSTADT

Open an **SSH connection** to one of the login nodes:



```
lcluster13
lcluster14
lcluster15
lcluster16.hrz.tu-darmstadt.de
lcluster17
lcluster18
lcluster19
```

“Cascade Lake”, AVX512, 96 Cores, 768 GB RAM

lcluster13/15/17/19: NVidia Tesla T4 GPU

- All login nodes are equal (except for the four with GPUs)
- There is no load balancer – just choose whichever of the above
- Switch to another login node if your current one is too much loaded
- initial login **password** is your main TU-ID password

Connecting to the Lichtenberg HPC

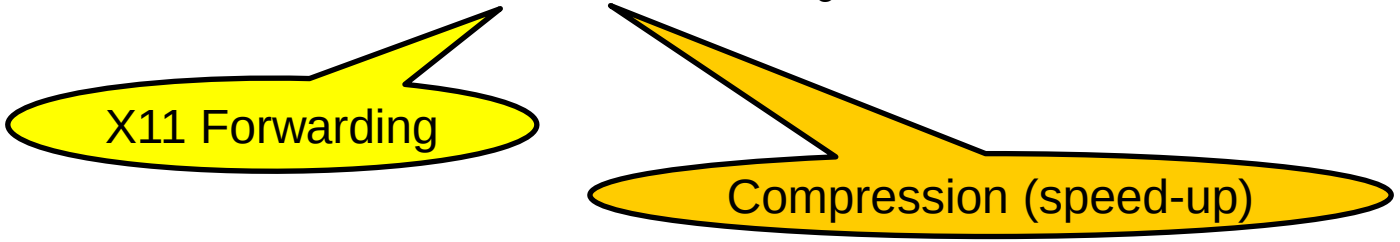
TECHNISCHE UNIVERSITÄT DARMSTADT

SSH – the **S**ecure **Sh**ell (encrypted + secured network channel)

- is the *only* remote connection protocol available for Lichtenberg (no `rsh` nor `telnet`)
- provides for terminal (text-based) access ***and*** for secure file transfer (`scp`) into and from the cluster

From a Windows PC: use the SSH client built into Windows 10 or PuTTY or Bitvise (cf. next slide) or similar software.

From Linux/Mac: `ssh -X -C <tu-id@>lcluster13.hrz.tu-darmstadt.de`



HRZ Hochschulrechenzentrum

Connecting to the Lichtenberg HPC

TECHNISCHE UNIVERSITÄT DARMSTADT

In case of login problems:

- Read the (ssh) error message in its entirety. Sometimes it even explains how to fix the actual problem.
- raise verbosity: `ssh -v[vv]` will show more (debug) messages
- Try another login node (the one you tried may be down)
- Try to log in explicitly with IPv4 or IPv6:

```
ssh -XC -4 <tu-id@>lcluster15.hrz.tu-darmstadt.de
ssh -XC -6 <tu-id@>lcluster17.hrz.tu-darmstadt.de
```

- Change your TU-ID password (https://www.idm.tu-darmstadt.de) and wait a few minutes before retrying
- If nothing of the above works out: open a ticket.

Some Software for Windows Users

TECHNISCHE UNIVERSITÄT DARMSTADT

- **SSH** (plain command prompt):
  - **`ssh`** built in (since Win10 Update “Apr2018”): `cmd` → `ssh TU-ID@lcluster…`
  - **PuTTY** (or one of its forks like KiTTY) (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)
- **File transfer** between your computer and the cluster:
  - **WinSCP** (http://winscp.net), **FileZilla** (http://filezilla-project.org)
- **Both** (SSH *and* file transfer)
  - **Bitvise** (https://www.bitvise.com/ssh-client)

➢ **Hint**: To obtain some speedup, **enable compression and always use the most recent version**.

The HRZ cannot support the (above mentioned) software, nor can it give any guarantee against security holes. Thus, it is *your responsibility* to keep your software up to date!

HRZ Hochschulrechenzentrum

Some Software for Windows Users

TECHNISCHE UNIVERSITÄT DARMSTADT

- For **graphical** applications:
  - Win10 `ssh` client or PuTTY or Bitvise (enable “X11 forwarding”!) with **VcXsrv** (http://sourceforge.net/projects/vcxsrv).
    - For fast connections, e.g. network of TU Darmstadt
  - **VNC** (a little difficult to configure here, still possible)
    - Also works well on slower connections.
  - **MobaXterm** (https://mobaxterm.mobatek.net/)
    - Multi-terminal ssh/scp client with built-in Xserver ("swiss army knife")
  - ***Do not use ~~Xming~~!*** The free version has not been updated since 2007 and has several bugs. VcXsrv is by far superior.

The HRZ cannot support the (above mentioned) software, nor can it give any guarantee against security holes. Thus, it is *your responsibility* to keep your software up to date!

Work environment

Operating System

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Linux** only
- Red Hat Enterprise Linux [RHEL] 8
- `x86_64` architecture (64bit)
- Shell (command interpreter): **`/bin/bash`** (BOURNE-again shell)

- Some **general-purpose programs** are included in the operating system's base installation, ready to use:
  - You can just run these programs, e.g. text editors like **`vi`**, **`nano`**, **`mc`**, **`gedit`** (for editing files directly on the login nodes)
  - Only a *limited number* of packages is provided this way, because the OS resides completely in the main memory (RAM) of each node (diskless)

The Module System*

TECHNISCHE UNIVERSITÄT DARMSTADT

* not to be confused with `perl` or `python` modules!

- unified Lua script kit to set up and tear down the *environment* (`$PATH`, `$LD_LIBRARY_PATH`, `$MANPATH` …) for different scientific programs, libraries and packages
- resides on the cluster-wide shared file system → available to all login & compute nodes

- A lot of software is **provided by the module system**:
  - **Compilers** (GCC 8.5 – 13.1, Intel 2020-2022, LLVM 8-16, …)
  - **Libraries** (MPI, MKL, ACML, CUDA, NAG, Boost, FFTW, GSL, …)
  - **Tools** (TotalView, SVN, Git, GDB, Valgrind, Java, Python 2.7 – 3.10, CMake, …)
  - **Application software** (MATLAB, ANSYS, Abaqus, Gurobi, LAMMPS, PETSc, QuantumEspresso, R, …)
    Make sure you hold (or are entitled to use) the necessary **licenses** for commercial packages!

- Use **`module avail`** to obtain a list of all modules *currently* loadable and **`module whatis`** for their description.

- Use **`module help <modulename>`** to get some further details about a specific module

HRZ Hochschulrechenzentrum

The Module System*

* not to be confused with `perl` or `python` modules!

TECHNISCHE UNIVERSITÄT DARMSTADT

- the module tree is *hierarchical* in terms of
  - the loaded compiler
    - the loaded MPI implementation

That is: compiler- or MPI-*dependent* modules will *not* be shown nor listed until you have loaded a compiler or an MPI, respectively!

\+ avoids unfavorable as well as mutually exclusive module combinations (ie. two different compilers at the same time)

\+ makes sure you get modules compiled with *exactly* your loaded compiler/MPI variety and version

- use **`module --show_hidden avail`** to really see *all* modules available

HRZ Hochschulrechenzentrum

The Module System*

* not to be confused with `perl` or `python` modules!

TECHNISCHE UNIVERSITÄT DARMSTADT

- To load a module, use the command **`module load <modulename>`**.
  - Read the *hints* that might appear. Sometimes you have to load additional modules.
  - Typically, the module name has to contain a version number, e.g. **ansys/21**
  - For some modules where it makes sense, we set a *default* version, e.g. you can use **`module load gcc`** (instead of **`gcc/8.5`**).
- To show all currently loaded modules, use **`module list`**.
- Shortcut: **`ml`** (= "**m**odule **l**ist") / **`ml myapp`** (= "**m**odule **l**oad myapp")
- To remove a module, use **`module unload <modulename>`**.
- To remove *all* modules, use **`module purge`**, eg. to get a clean environment

HRZ Hochschulrechenzentrum

Other Software

TECHNISCHE UNIVERSITÄT DARMSTADT

How to use application software (not provided by the OS) on the cluster:

1. **Program is provided by our *module system*.**
   - You have to load the corresponding module(s)
   - We can create new modules if the software is useful for *several* people
2. **Install** the software **to your `$HOME` folder (`/home/<TU-ID>`).**
   - If software is not available via the module system.
   - Installation procedure dependent on the program you want to install (refer to manuals, website, README/INSTALL files, etc).
3. **Install** the software **to a common group folder.**
   - group/project area (`/work/projects/p00XXX`) can be made available on request.
   - (Same) installation procedure as for a home folder.

HRZ Hochschulrechenzentrum

Python: `virtualenv` instead of `conda`

TECHNISCHE UNIVERSITÄT DARMSTADT

Python *conda* environments could interfer with and do not harmonize well with python modules from our module system.

Thus we strongly recommend using **python virtual environments**, if you need specific python packages not available in the module system.

```
ml gcc/8 python/3.10
mkdir test; cd test
python -m venv myenv
source myenv/bin/activate
pip install --upgrade pip
pip install spython
deactivate
```

Load a suitable compiler & python version

Create a vEnv named „*myenv*“…

and activate it.

Now you can use `pip` without „`--user`“…

to install eg. the „spython“ module.

These „*create & install* steps“ are only required **once**, *outside* of a batch job!

HRZ Hochschulrechenzentrum

Python: `virtualenv` in a batch job

TECHNISCHE UNIVERSITÄT DARMSTADT

In each batch job, use only the lines

```
ml gcc/8 python/3.10
cd test

source myenv/bin/activate
python myScript

deactivate
```

switch to the same directory

Activate the vEnv…

and run your „payload“, ie. your script requiring „spython“

and deactivate it (optional)

to run your scripts using your special python modules from this vEnv.

HRZ Hochschulrechenzentrum

Use `docker` containers with `singularity`

TECHNISCHE UNIVERSITÄT DARMSTADT

There is no `docker` available on the Lichtenberg cluster. `Podman` offers docker functionality, but has severe limitations, and is difficult to set up. We recommend `singularity`, respectively `apptainer` (next year).

From docker registry:

```
singularity build lolcow.sif docker://godlovedc/lolcow
singularity run lolcow.sif
```

From docker-archive (`podman pull` only works if podman is correctly configured):

```
podman pull docker://godlovedc/lolcow
singularity build lolcow_from_archive.sif docker-
     archive://$(pwd)/lolcow_docker_archive.tar
singularity run lolcow_from_archive.sif
```

Use `docker` containers with `singularity`

TECHNISCHE UNIVERSITÄT DARMSTADT

While you cannot build a singularity container directly from a dockerfile, you can *convert* a dockerfile to a singularity definition, and build from that (requires „fakeroot“):

```
cd test
git clone https://github.com/GodloveD/lolcow
source myenv/bin/activate
spython recipe lolcow/Dockerfile ./sing.lolcow
singularity build --fakeroot lolcow_from_dockerfile.sif sing.lolcow
singularity run lolcow_from_dockerfile.sif
deactivate
```

These „*convert & install* steps“ are only required **once**, *outside* of a batch job!

To run the container inside a batch job, use only

```
cd test
singularity run lolcow_from_dockerfile.sif
```

- Questions up to here?

- (Live demo)

Batch System

Batch (Queuing) System

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Do not confuse the login nodes with the cluster!**

  There are only a few login nodes, but >**630** compute nodes.

  Do not simply start your program on the login node hoping it will use the cluster!

  → Login nodes are (only) your *main access point* to the system.

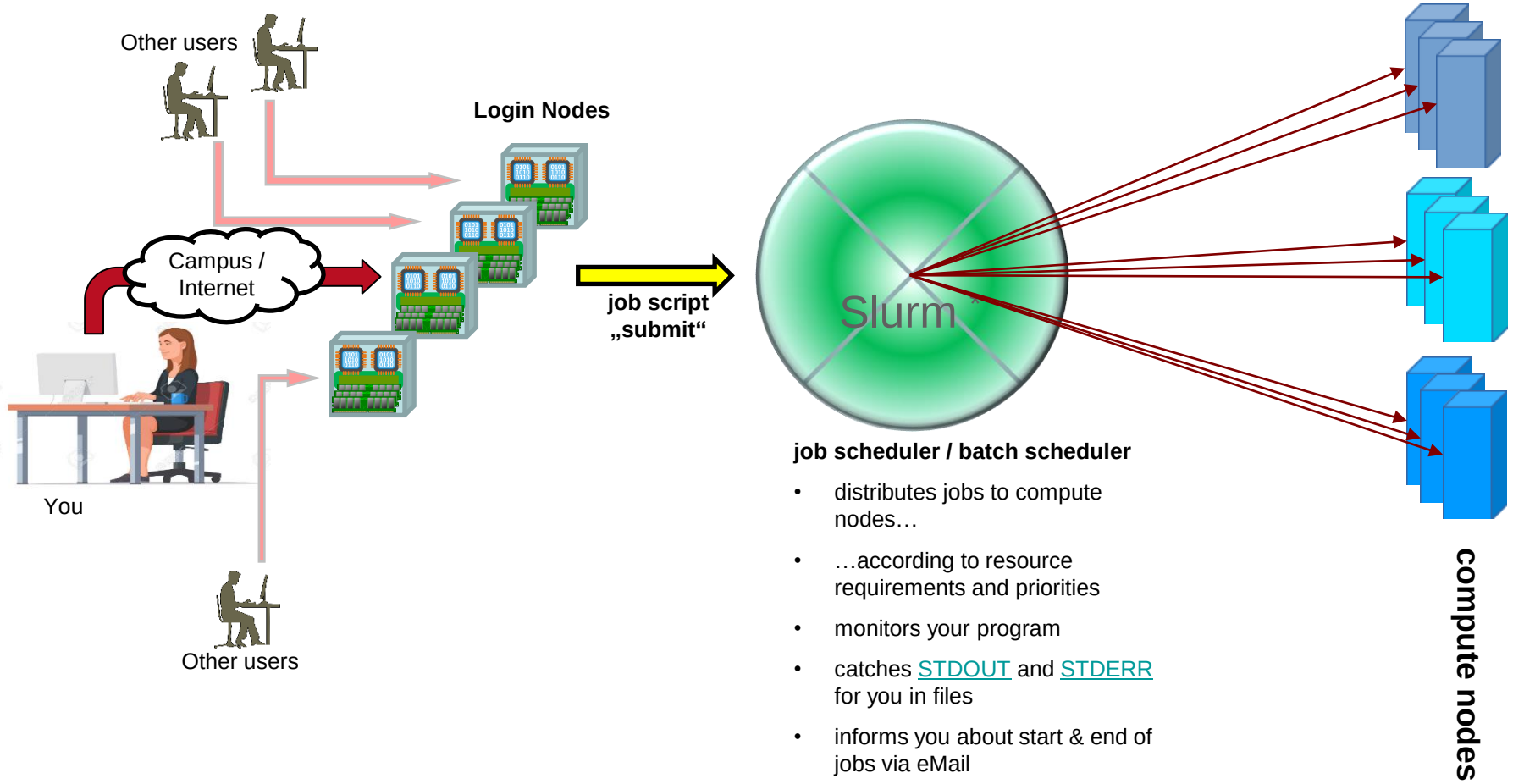
- The cluster runs 24/7 (including the weekend) and many users might want to do far more calculations than there are CPU cores available

  → **Batch queueing system** arbitrates between jobs by priorities, based on allotted and consumed computing time.

- You *cannot* connect or login directly to the compute nodes.

  → from the *login nodes*, use the batch system to **submit jobs**

Computations via batch jobs only



job scheduler / batch scheduler

- distributes jobs to compute nodes…
- …according to resource requirements and priorities
- monitors your program
- catches STDOUT and STDERR for you in files
- informs you about start & end of jobs via eMail

* formerly: **S**imple **L**inux **U**tility for **R**esource **M**anagement

Shell Script / Job Script

TECHNISCHE UNIVERSITÄT DARMSTADT

A **shell script** is simply a plain text file (with UNIX line ends), listing all commands to execute one after another, line by line (comparable to a "batch script" in Windows).

```
#!/bin/bash
# a comment line ignored by bash
# another comment, also ignored
echo "Command list starting..."
myProg1 --inf=input1.dat  --outf=output1.txt --param1=value1 ...
myProg2 --inf=output1.txt --outf=output2.txt --param2=value2 ...
echo "Command list finished."
```

/home/<*TU-ID*>/bin/myScript

"Shebang line" - tells what command to use for execution

List of commands to execute, ie. the "*payload*" so to say

HRZ Hochschulrechenzentrum

Job Script

… above all, is such a simple shell script (plain text file with UNIX line ends)

TECHNISCHE UNIVERSITÄT DARMSTADT

```
#!/bin/bash

echo "This is not yet a job."

module purge
module load  gcc  openmpi
srun <myMPIprogram> …
```

HRZ Hochschulrechenzentrum

Important Submission Parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

You should **always** set the following parameters:

- **`#SBATCH -n <numtasks>`**: Number of tasks (separate [MPI] processes)
  (This often corresponds to the number of CPU cores. *Exception*: OpenMP/Hybrid)
- **`#SBATCH --mem-per-cpu*=< memory | 3800 >`**: Maximum memory per core in MBytes
- **`#SBATCH -t <time>`**: Time limit for the job (“wall clock time”)
  (‘mm’, ‘mm:ss’, ‘hh:mm:ss’, ‘dd-hh’, ‘dd-hh:mm’, ‘dd-hh:mm:ss’)

Recommended (but optional) parameters:

- **`#SBATCH -A <project_name>`**: project to account on. Important if you have access to several projects. (project_name = “`project`” + 5 digits, e.g. “`-A project00123`”)
- **`#SBATCH -J <jobname>`**: Name of the job (your choice, does not have to be unique)
- **`#SBATCH -o <filename>`**: Write the job’s standard output (‘`stdout`’) to a file
- **`#SBATCH -e <filename>`**: Write the job’s standard error (‘`stderr`’) to a file (optional)

** means „memory per **core**“!*

HRZ Hochschulrechenzentrum

Further useful parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

- **#SBATCH -c <number>**: # of CPU cores *per task/process*, in case of OpenMP / Multi-Threading
- **#SBATCH --mail-type=<NONE | BEGIN | END | ALL | …>**
  Send an email (or not) when jobs start and/or end
- **#SBATCH -D <path>**: "working directory" for all commands below
  (is like "`cd <path>`" before running your program)

- **#SBATCH --exclusive**: Exclusive node (no other jobs) – caution!
- **#SBATCH -C <feature>**: Select special resources (discussed later)
- **#SBATCH -a <index range>**: Submit a job array (discussed later)

See **man sbatch** for even more additional parameters if required.

HRZ Hochschulrechenzentrum

Job Script

… is a simple shell script with SLURM pragmas (camouflaged as shell comments), using slurm placeholders as **%…** and in the payload, shell variables as $...

TECHNISCHE UNIVERSITÄT DARMSTADT

```
#!/bin/bash

#SBATCH -J my_job
#SBATCH --mail-type=END
#SBATCH -e /work/scratch/<tu-id>/somewhere/%x.err.%j
#SBATCH -o /work/scratch/<tu-id>/somewhere/%x.out.%j
#SBATCH --mem-per-cpu=250
#SBATCH -t 00:05:00
#SBATCH -n 4

echo "This is Job $SLURM_JOB_ID"

module purge
module load  gcc  openmpi
srun /usr/bin/hostname
```

%x will be replaced by the JobName.

%j will be replaced by the actual job id.

…while **$** variables are *shell script* placeholders

your program (**hostname** here being just an example for a simple command)

HRZ Hochschulrechenzentrum

Submitting a Job and other SLURM Commands

TECHNISCHE UNIVERSITÄT DARMSTADT

- `sbatch <JobScriptFile>` : submit the job and returns the JobID.
- `squeue`: list all your jobs
- `sjobs <JobID>`: print detailed information about a job
- `scancel <JobID>`: Kill a particular job
- `scancel -u <TU-ID>`: Kill all your own jobs (*use with care!*)
- `csreport`: shows information about used computing time per project
- `csum` or `seff <JobID>` or `tuda-seff <JobID>`: computing time and resource usage (efficiency)

see also the SLURM “cheat sheet”: https://slurm.schedmd.com/pdfs/summary.pdf

HRZ Hochschulrechenzentrum

Problems in Submitting Jobs

TECHNISCHE UNIVERSITÄT DARMSTADT

`sbatch <JobScriptFile>` : submit the job and returns the JobID.

In case of errors:

- Read the error message
- Check your project (`#SBATCH -A ...` vs. `cat ~/.project`)
  `sacctmgr show assoc account=<your project>`
- Check your job script for *windows line ends:*
  `file myJobScript` (if it says "ASCII text, with CRLF", re-edit the job script to have UNIX line endings, eg. with the `dos2unix` command)
- check for *conflicting* requirements (ie. >376 GByte RAM *and* a GPU)
- Avoid using "`--nodes/-N`" and "`--ntasks-per-node`"

see also the SLURM “cheat sheet”: https://slurm.schedmd.com/pdfs/summary.pdf

Special Features / Constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

- By default, jobs are executed on arbitrary nodes

- when jobs have special requirements, such as
  - particular CPU architectures (avx512 or avx2)
  - more main memory than is available on the average compute node

  → **select** so-called "**features**"

  - GP GPUs (accelerators based on graphics cards)

  → **select** so-called "**GRes**" (**G**eneric **Res**ources)

Available Features / GRes

TECHNISCHE UNIVERSITÄT DARMSTADT

CPU	Acc	Memory	Misc

`-C avx512`
`-C avx2`
	`--gres=gpu`
	any Nvidia GPU
	`--gres=gpu:a100`
	a100 (NVidia AMPERE 100)
	`--gres=gpu:v100`
	v100 (NVidia VOLTA 100)
		`-C mem`
		`-C mem1536g`
			`-C mpi`

Details on those features not being part of this talk can be found here:

https://www.hrz.tu-darmstadt.de/hlr/nutzung_hlr/rechenjobs_und_batchsystem_hlr/index.en.jsp

Selecting Features

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use submission parameter **-C** to select features (aka “**C**onstraints”)
- Examples:
  - **`#SBATCH -C avx512`**

    Job is dispatched to nodes with AVX512 CPUs (default)
  - **`#SBATCH -C "avx512&mem1536g"`**

    Job is dispatched to nodes with AVX512 CPUs ***and*** 1.5 TByte of RAM
  - **`#SBATCH -C "A|B"`**

    Job is dispatched to nodes either with feature A ***or*** nodes with feature B

See **`man sbatch`** for more details.

HRZ Hochschulrechenzentrum

Selecting GRes

Unlike constraints/features, **g**eneric **res**ources allow for more precise specifications of what a job needs

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use submission parameter **`--gres=Class:Type:#`** to select specific resources (if not specified, the defaults are Type=any and #=1)
  - **`#SBATCH --gres=gpu`**

    Job is dispatched to nodes with *any* GPU card(s) and can use 1 of them
  - **`#SBATCH --gres=gpu:4`**

    Job is dispatched to a node with *any* GPU card(s) and can use 4 of them
  - **`#SBATCH --gres=gpu:v100`**

    Job is dispatched to a node with NVidia GPUs of type "Volta 100" and will have access to 1 of these cards
  - **`#SBATCH --gres=gpu:a100:3`**

    Job is dispatched to nodes with NVidia GPUs of type "Ampere 100" and will have access to 3 of these cards

Do not specify >4 with "`-C avx512`" as there are only Intel avx512 nodes with 4 GPUs.
To request up to **8** GPUs/node, add "`-C avx2`" (Nvidia **DGX** nodes with AMD CPUs).

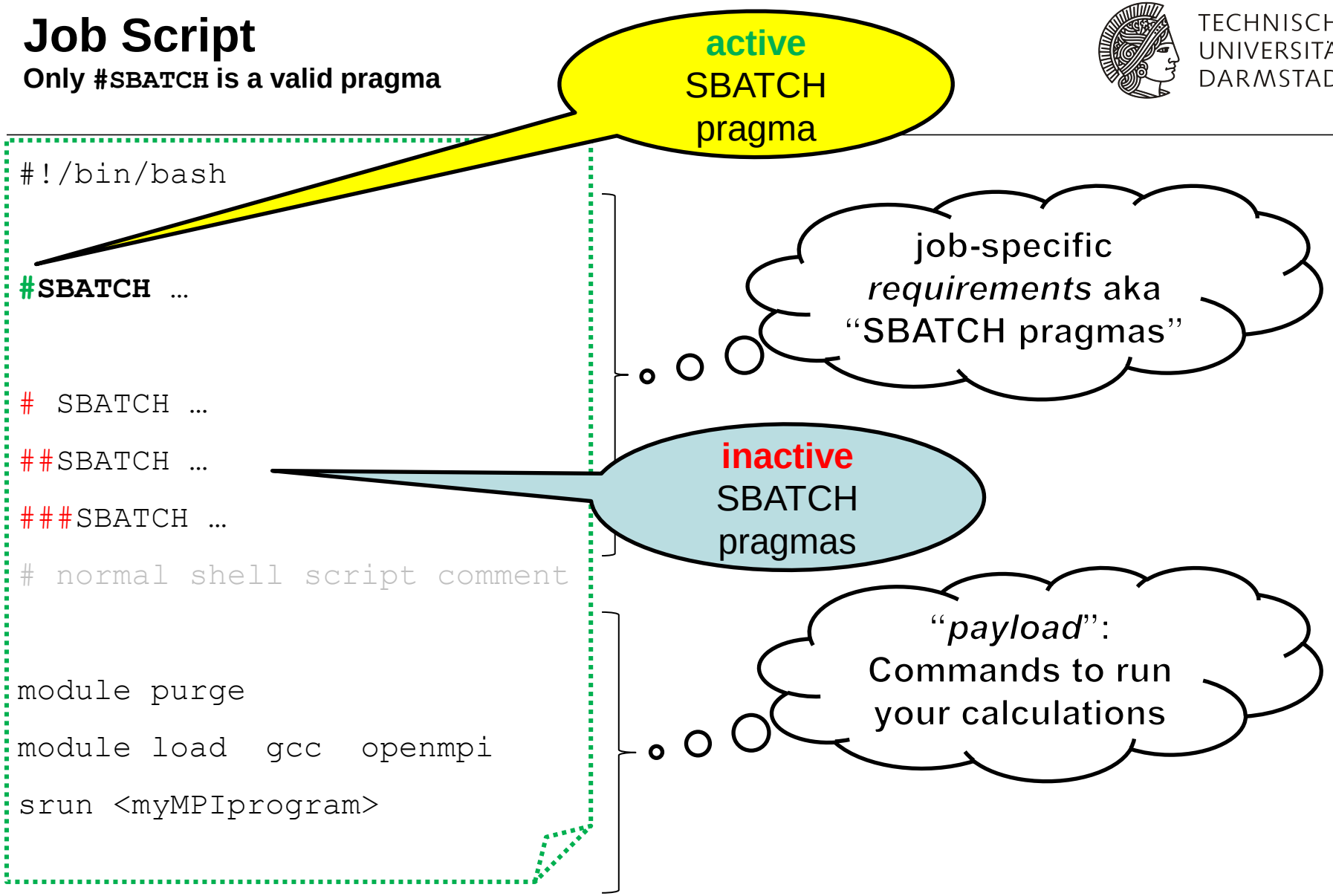
- Questions up to here?

- (Live demo)

Tips & Tricks

Job Script

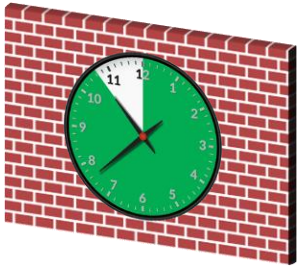
Only `#SBATCH` is a valid pragma



Job run time requirements

TECHNISCHE UNIVERSITÄT DARMSTADT

… are always taken as *wall clock time* (not accumulated CPU time) = total real time needed to complete all steps and components of your job.



- ≤ 30 min (“short” jobs)
  - Jobs can run on almost all nodes
  - Additionally, a few nodes are *reserved* just for such jobs
- ≤ 24 h (“default” jobs)
  - Jobs can run on almost all nodes
- > 24 h (“long” jobs)
  - Only a limited number of nodes processes these jobs
  - Currently limited to one week

`-t 1-` and `-t 24:00:00` are inclusive – no need to specify `23:59:59`

HRZ Hochschulrechenzentrum

How to make my jobs efficiently placeable? How do I “size” them?

TECHNISCHE UNIVERSITÄT DARMSTADT

- use the most *generic* form possible, but be as *specific* as necessary
  Why specifying „mem1536“ unless your program *really* requires it? Your job would have to wait for the scarce „big-mem“ nodes, even though lots of other nodes were idling around.
- to assess the 2 key requirements necessary for your program, run it with the UNIX `time` command:
  ```
  /bin/time --format='MaxMem: %Mkb, WCT: %E' myProgram
  ```
  then: `--mem-per-cpu` = (MaxMem (MB) / # of cores used) + safety margin
  and: `--time` = Wall Clock Time + safety margin
- use `--ntasks=` values in even multiples of 96 cores (or even divisors like 24 and not setting `--exclusive`), to best exploit all cores on your allocated nodes
- in case of several *similar* jobs, make use of **job arrays**
  (discussed in a minute)

The crux of -n / --ntasks= vs. -c / --cpus-per-task=

TECHNISCHE UNIVERSITÄT DARMSTADT

-n number of *distinct processes* to run (MPI)

Processes could always be dispatched to arbitrary and distinct nodes = requires MPI!

-c number of CPU cores to use per single process (Multi-Threading/OpenMP)

*Threads will **never** be placed on distinct nodes (only on the same node)!*

If you run a *non-MPI* program via **-n 4**, it is simply started 4 times. These 4 *distinct instances* will most likely overwrite eachother's log and output files.

→ Use **-n** *only* to run MPI programs!

Multithreaded program on one LB2 node (à 96 cores):

```
#SBATCH -n 1
#SBATCH -c 96
#SBATCH --exclusive
OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export OMP_NUM_THREADS

myMTprogram
```

MPI program on four LB2 nodes (à 96 cores)**:**

```
#SBATCH -n 384
#SBATCH --exclusive

srun myMPIprogram
```

Hybrid program on four LB2 nodes (à 96 cores)**:**

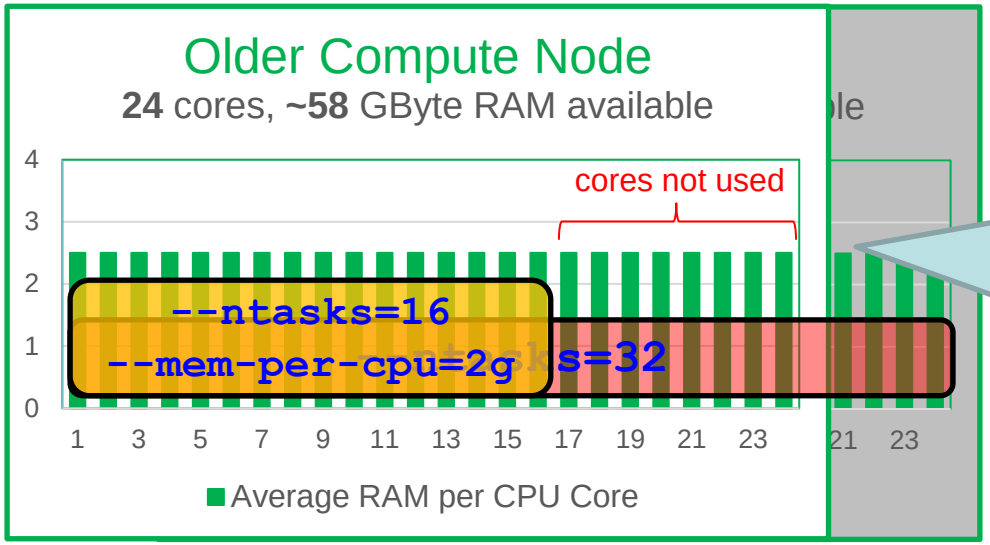
```
#SBATCH -n 4
#SBATCH -c 96
#SBATCH -exclusive
OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export OMP_NUM_THREADS

srun myMPI+MTprogram
```


HRZ Hochschulrechenzentrum

How to make my jobs efficiently placeable? How do I “size” them?

TECHNISCHE UNIVERSITÄT DARMSTADT



Older node type with 24 cores taken *just as an example* – 96 cores would „detonate“ this figure to be unreadable.
Transform the principle to 96 cores and ~4096 MByte/core

Non-MPI jobs (confined to run *inside* a node)

- should be run with
  ```
  #SBATCH -n 1
  #SBATCH -c 96
  ```
  to use all cores on a node
- in case your program scales best at 24 cores, use
  ```
  #SBATCH -n 1
  #SBATCH -c 24
  ```
  to place **4** of your jobs concurrently on the same node

MPI jobs (running *across* several nodes)

- asking for any number of cores *not* being an even multiple of 96
- asking for more main memory-per-core than 3,800 GiB

will waste resources by not using all cores of all nodes involved.

Remember: even these „wasted“ cores not doing anything for your job will be accounted on your project!

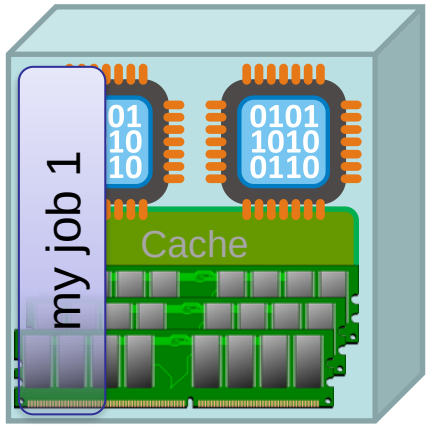
HRZ Hochschulrechenzentrum

Exclusive or Shared

- Default node config on the Lichtenberg: ***user-exclusive***, meaning only *your* jobs can share a given node (and none of other users)
- `#SBATCH --exclusive` assigns node(s) exclusively to *this* single job

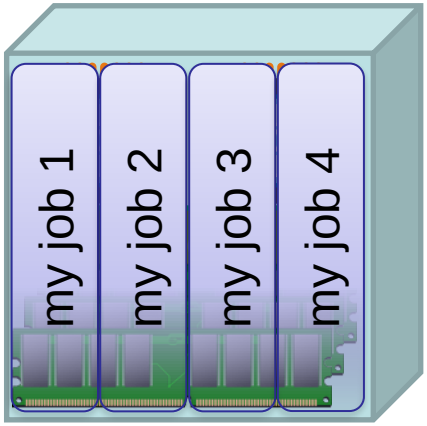
Assuming your scientific program scales best at using **24 cores** per run: why not running 4 jobs concurrently on the same node?

`#SBATCH --exclusive`



☹

`###SBATCH --exclusive`

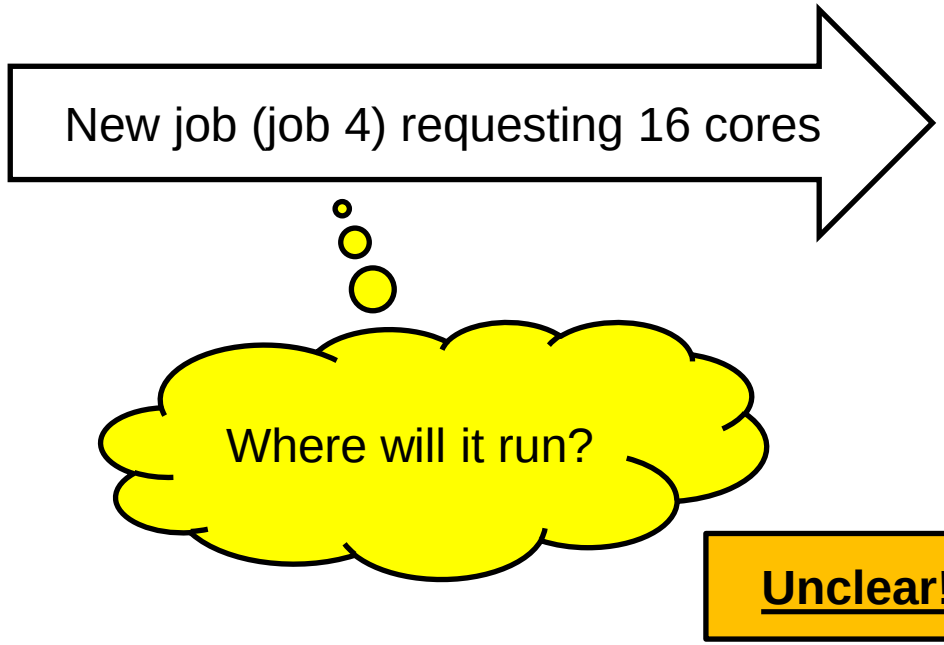


☺

Dispatching Jobs
(assuming node with 16 cores)

TECHNISCHE UNIVERSITÄT DARMSTADT

	Node 1	Node 2	Node 3
Core 1	Job 1		Job 2
Core 2	Job 1		Job 2
Core 3	Job1		Job 2
Core 4	Job 1		Job 2
Core 5	Job 1		Job 3
Core 6	Job 1		Job 3
Core 7	Job 1		Job 3
Core 8	Job 1		Job 3
Core 9			
Core 10			
Core 11			
Core 12			
Core 13			
Core 14			
Core 15			
Core 16			



HRZ Hochschulrechenzentrum

Exclusive vs. Non-exclusive Jobs
(assuming node with 16 cores)

TECHNISCHE UNIVERSITÄT DARMSTADT

	Node 1	Node 2	Node 3
Core 1	Job 1		Job 2
Core 2	Job 1		Job 2
Core 3	Job1		Job 2
Core 4	Job 1		Job 2
Core 5	Job 1		Job 3
Core 6	Job 1		Job 3
Core 7	Job 1		Job 3
Core 8	Job 1		Job 3
Core 9	Job 4		Job 4
Core 10	Job 4		Job 4
Core 11	Job 4		Job 4
Core 12	Job 4		Job 4
Core 13	Job 4		Job 4
Core 14	Job 4		Job 4
Core 15	Job 4		Job 4
Core 16	Job 4		Job 4

This might or might not be the desired behavior!

	Node 1	Node 2	Node 3
Core 1	Job 1	Job 4	Job 2
Core 2	Job 1	Job 4	Job 2
Core 3	Job1	Job 4	Job 2
Core 4	Job 1	Job 4	Job 2
Core 5	Job 1	Job 4	Job 3
Core 6	Job 1	Job 4	Job 3
Core 7	Job 1	Job 4	Job 3
Core 8	Job 1	Job 4	Job 3
Core 9		Job 4	
Core 10		Job 4	
Core 11		Job 4	
Core 12		Job 4	
Core 13		Job 4	
Core 14		Job 4	
Core 15		Job 4	
Core 16		Job 4	

This is what users often are expecting! Ensure with `'--exclusive'`.

HRZ Hochschulrechenzentrum

Exclusive Jobs: Not Always Useful

TECHNISCHE UNIVERSITÄT DARMSTADT

	Node 1	Node 2	Node 3
Core 1	Job 1		Job 2
Core 2	Job 1		Job 2
Core 3	Job1		Job 2
Core 4	Job 1		Job 2
Core 5	Job 1		Job 3
Core 6	Job 1		Job 3
Core 7	Job 1		Job 3
Core 8	Job 1		Job 3
Core 9			
Core 10			
Core 11			
Core 12			
Core 13			
Core 14			
Core 15			
Core 16			

16 new jobs requesting one core each (exclusive mode)

HRZ Hochschulrechenzentrum

Exclusive vs. Non-exclusive Jobs

	Node 1	Node 2	Node 3
Core 1	Job 1	Job 4	Job 2
Core 2	Job 1	-----	Job 2
Core 3	Job1	-----	Job 2
Core 4	Job 1	-----	Job 2
Core 5	Job 1	-----	Job 3
Core 6	Job 1	-----	Job 3
Core 7	Job 1	-----	Job 3
Core 8	Job 1	-----	Job 3
Core 9		-----	
Core 10		-----	
Core 11		-----	
Core 12		-----	
Core 13		-----	
Core 14		-----	
Core 15		-----	
Core 16		-----	

This might or might not be the desired behavior!

	Node 1	Node 2	Node 3
Core 1	Job 1	Job 4	Job 2
Core 2	Job 1	Job 5	Job 2
Core 3	Job1	Job 6	Job 2
Core 4	Job 1	Job 7	Job 2
Core 5	Job 1	Job 8	Job 3
Core 6	Job 1		Job 3
Core 7	Job 1		Job 3
Core 8	Job 1		Job 3
Core 9	Job 9		Job 14
Core 10	Job 10		Job 15
Core 11	Job 11		Job 16
Core 12	Job 12		Job 17
Core 13	Job 13		Job 18
Core 14			Job 19
Core 15			
Core 16			

This is what users often are expecting! Ensure by *omitting* '--exclusive'.

Resource requests

TECHNISCHE UNIVERSITÄT DARMSTADT

Possible effects of **too little** resources requested:

- Runs into time limit (job will be `SIGTERM`'inated, then `SIGKILL`'ed)
- Job crashes (e.g. `SEGMENTATION FAULT` due to "insufficient memory" or similar problems)
- Jobs (and thus your scientific project) takes „forever"

Possible effects of **too much** resources requested:

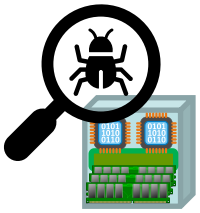
- Blocked cores remaining unused
- Longer queue (pending) times!
- Resources are accounted on your core*h budget, yet not fully used

Looking over a job's shoulder

TECHNISCHE UNIVERSITÄT DARMSTADT

How to see how a job runs *en detail*, eg. to debug it?

While a batch job of yours runs, you are entitled to log in to its *executing* compute node.



- run `squeue -t RUNNING` to determine your own running jobs and list their executing compute nodes (last column „NODELIST“)
- from a login node, run `ssh <name_of_executing_CN>` to „hop“ on the compute node
  or
  `sattach <jobid.stepid>` (no nodename required)
- then, use your favorite linux commands like „`top`“ / „`htop`“ (or „`nvidia-smi`“ on GPU nodes) to see what resources your job uses and in what way
  Even syscall tracing: `strace -p <your job's PID>` is possible.

Pending Jobs

TECHNISCHE UNIVERSITÄT DARMSTADT

My jobs are pending – when will they start?

Depending on

- priority (used resources up to now, see "`csreport`")
- required run time (especially when targeting the "long" queue!)
- available compute nodes (w.r.t "features" like GPUs)

your jobs might remain in "pending" state for quite a long time.

You can ask the scheduler as to when it deems your jobs startable:

```
squeue --start
```

Since the scheduler doesn't touch *all* pending jobs in *every* scheduling cycle, even this estimation might take some time to return something other than "N/A".

Exit Code

… is a program's only way of telling its parent process about success or failure – how to preserve it for Slurm

TECHNISCHE UNIVERSITÄT DARMSTADT

```
#!/bin/bash
#SBATCH …
echo "This is Job $SLURM_JOB_ID"
module purge
module load  gcc  openmpi
srun /my/scientific/program …
EXITSTATUS=$?
mv resultFile /path/to/my/resultFiles/
rm *.tmp
echo "Job $SLURM_JOB_ID ended at $(date)."
exit $EXITSTATUS
```

SLURM has *no means* to identify THIS line as your job script's most important command!

… if you do not save it like here with eg. `EXITSTATUS=$?`

Not doing so would yield jobs being *falsely* reported as COMPLETED, masking any failures of your real "payload" program.

HRZ Hochschulrechenzentrum

Interactive Jobs

TECHNISCHE UNIVERSITÄT DARMSTADT

… are requested via

```
srun -n1 -t10 --mem-per-cpu=200 --pty /bin/bash
```

start an interactive job

- one task/process
- for at most 10 minutes
- with at most 200 MBytes of RAM per core
- allocate a pseudo terminal
- command to run on the compute node

… but:

- has to wait until resources are free (can take seconds/minutes/hours…)
- are killed inevitably as soon as time is up or RAM is exceeded
- should be used only for testing / debugging of special cases, eg.
  - accelerator jobs (NVidia GPUs: add `--gres=gpu:1`)
  - when the login nodes do not evoke the problem, eg. due to being too different from the compute nodes

Job Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

… are the *most efficient way* of submitting many *similar* tasks, and do *not* choke the scheduler as much as individual jobs would do!

Whenever you're tempted to submit more than ~20 distinct but similar jobs, always consider using a job array instead!

Additional Benefit: you avoid the TU being blamed as **mail spammer** and our mail servers from being blacklisted! While each *job* creates separate mails, a job array's *subtasks* do not (at least not by default).

Use **C**ases:

1. The same scientific program should run
   a. with the same input, but X times with a different parameter set ("parameter study")
   b. with the same parameters, but on X different input files

2. I have to run a lot of independent subtasks, not communicating with or dependent on each other

3. My simulation takes way longer than 24h, but waiting for a 7-day slot ("long" queue) takes too much time

Job Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

How to use the **Job Array** feature of SLURM:

```
#!/bin/bash

#SBATCH -J testArr
#SBATCH -a 1-8%2
#SBATCH -e /home/<tu-id>/some/path/%x.err.%A_%a
#SBATCH -o /home/<tu-id>/some/path/%x.out.%A_%a
#SBATCH -n 1
#SBATCH --mem-per-cpu=1000
#SBATCH -t 00:02:00

echo "This is Job $SLURM_ARRAY_JOB_ID, Index: $SLURM_ARRAY_TASK_ID"
myProgram --parameters=$SLURM_ARRAY_TASK_ID.params --input=same.in      = UC 1a

                                  or

myProgram --parameters=same.params --input=img$SLURM_ARRAY_TASK_ID.jpg  = UC 1b
```

Start index

End index

Number of elements that may run in parallel (optional)

Job ID (same for all array elements)

Index of the current array element

HRZ Hochschulrechenzentrum

Job Arrays - Details

TECHNISCHE UNIVERSITÄT DARMSTADT

UC 2

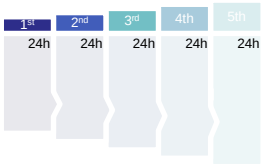
High Throughput Computing: *independent* subtasks

☹ for eg. 25 subtasks not specified as a job array, SLURM would wait for 25 nodes to be free at the same time.

☺ Specifying the 25 subtasks as an "`#SBATCH -a 1-25`" array (without a limiting `%#`), SLURM would run as much subtasks in parallel as there are nodes free: your subtasks can "slip" onto free nodes as they become available, without waiting for a set of 25 free nodes!

UC 3

Chain of jobs: exceeding the maximum runtime limit of 7 days



If your simulation needs eg. ~19 days total runtime and your program is capable of CPR (**c**heck**p**oint & **r**estart), you would specify an array with

`#SBATCH -a 1-19%1` (telling SLURM to run **19** array tasks *sequentially*, one after another)

`#SBATCH -t 24:00:00` (with 24h runtime each)

That way, you can avoid waiting for the "long" / 7d queue and submit this array into the "24h" class.

This yields a *chain of 19 one-day tasks*.
While your job's program will be terminated every 24h, the next array task will pick up the CPR file and continue to run the simulation where the former array task left off a few minutes before.

HRZ Hochschulrechenzentrum

- Questions up to here?

- (Live demo)

Other Common Issues

TECHNISCHE UNIVERSITÄT DARMSTADT

```
#!/bin/bash

#SBATCH -J my_job
#SBATCH --mail-type=END
#SBATCH -e /work/scratch/<tu-id>/someDir/%x.err.%j
#SBATCH -o /work/scratch/<tu-id>/someDir/%x.out.%j
#SBATCH --mem-per-cpu=250
#SBATCH -t 00:05:00
#SBATCH -n 4

echo "This is Job $SLURM_JOB_ID"

module purge
module load  gcc  openmpi
srun /usr/bin/hostname
```

Make sure these directories *exist beforehand*, as else your job fails silently, without any clue why.

HRZ Hochschulrechenzentrum

Other Common Issues (2)

Automatic loading of modules at login

TECHNISCHE UNIVERSITÄT DARMSTADT

When tempted to automate all your "`module load …`" in a shell startup file, only do it after the following check line:

```
# my own $HOME/.bashrc
# for automation
…
# place any output-generating
# commands below next check
# if you don't want to run into
# issues with "scp" or "rsync":
[ -z "$PS1" ] && return
module load gcc openmpi
echo "Ready for commands."
```

verbose & chatty (output-generating) commands

Due to SSH protocol issues, shell startup files generating STDOUT or STDERR are known to cause issues with "`scp`" and similar tools.

With this check line, their "login" ends at "`return`", so these do not see any output (only shown during interactive logins).

HRZ Hochschulrechenzentrum

Other Common Issues

TECHNISCHE UNIVERSITÄT DARMSTADT

- Only software that is *designed* to run in parallel *will* run in parallel.
  - Multithreading / OpenMP: Only *inside* a single node: **-c** <24 | 48 | 96>
  - MPI: inside a node **and/or** *across* several nodes: **-n** <24 | 48 | 96 | ... | … >
- **Licenses**:
  - For commercial software (e.g. Ansys), your institute has to hold licenses.
  - For other software (eg. MATLAB), TU Da has acquired a campus license. Otherwise, you *must* ***not*** *use* the software.
- **Account**:
  - **Do not pass on your account to other persons** (e.g. by sharing the password or `ssh`-keys). Otherwise, your account will be **locked** due to violating the user regulations!

User Support

- **Video Consultation**

  every Wednesday 10am – 11am as a Zoom meeting:

  Meeting-ID: 665 7163 2291

  Kenncode: 113329

- **Office Hours**

  in presence – every first Wednesday (monthly) between 9:30am and 3pm:

  Alexanderstraße 2, 3rd Floor, Room 205, 213 und 214

- Contact us via hhlr@hrz.tu-darmstadt.de

HKHLR: the Experts in Parallel Programming and Optimizing of HPC

TECHNISCHE UNIVERSITÄT DARMSTADT

Local support:

University of Kassel
kassel@hpc-hessen.de

Philipps University Marburg
marburg@hpc-hessen.de

Justus Liebig University Giessen
giessen@hpc-hessen.de

Goethe University Frankfurt
frankfurt@hpc-hessen.de

TU Darmstadt
darmstadt@hpc-hessen.de

High Performance Computing in Hessen

brain ware for science

educate

We offer periodic training courses on high performance computing at all Hessian locations

apply

We support users in accessing Hessian high performance computers

investigate

An overview of our research results is available on
www.hpc-hessen.de

Office:
Hessisches Kompetenzzentrum für Hochleistungsrechnen
Alexanderstraße 2
D-64283 Darmstadt

Email: office@hpc-hessen.de
Phone: +49 6151 16-76038
URL: www.hpc-hessen.de

support

Our team will advise users individually

funded by:
HESSEN Hessisches Ministerium für Wissenschaft und Kunst

Hessisches Kompetenzzentrum für Hochleistungsrechnen

How to Find Information Online

TECHNISCHE UNIVERSITÄT DARMSTADT

https://www.hrz.tu-darmstadt.de/hlr

HRZ Hochschulrechenzentrum

Coming to an end…

TECHNISCHE UNIVERSITÄT DARMSTADT

For “homework”, we recommend part 2 of our

Online Quiz:

https://university.quizacademy.io/course/CIWBTC

(Completely anonymous and non-tracking).

HRZ Hochschulrechenzentrum