ScaDS.AI
DRESDEN LEIPZIG
CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Introduction to High Performance Computing

Physics of Life - Bio Image Analysis Training School

Neringa Jurenaite, Apurv D. Kulkarni, Taras Lazariv

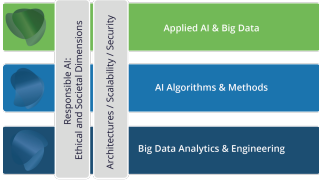
Dresden, 29 August 2023

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Instructors

ScaDS.AI
DRESDEN LEIPZIG

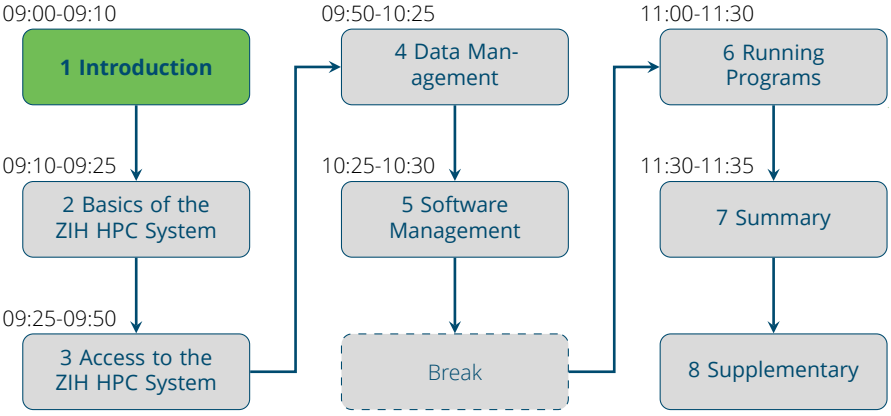


https://scads.ai/

Neringa Jurenaite,
Research Associate, ScaDS.AI
NLP and Omics Data, Living Lab

Apurv D. Kurkarni,
Research Associate, ScaDS.AI
Big Data Analytics and Distributed Computing, Living Lab

Taras Lazariv,
Senior Researcher, ScaDS.AI
Statistics, Machine Learning, Deep Learning


09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

Intention of this Training

- Making things understandable esp. for new users without a background in computer science
- Complete workflow examples are shown which are based on Python (can be used as a blueprint for other software/tools)
- From the user's perspective: What are the most important things to work with ML on an HPC cluster?

Note

- We do **not** want to show everything possible. We want to show what is needed to get started.
- We do **not** want to show the perfect HPC-ML workflow. We want you to comprehend ML principles with an HPC machine starting with concrete examples.

Hint

Please interrupt and ask immediately if something is not clear.

What is an HPC machine?

Terminology

- **Compute or login Node**: An individual computer, part of an HPC cluster
- **CPU, Core**: Central Processing Unit. A modern CPU is composed of numerous cores
- **Cluster**: A group of machines interconnected in a way that work together as a single system

Note

HPC cluster is a relatively tightly coupled collection of compute nodes, the interconnect typically allows for high bandwidth, low latency communication. Access to the cluster is provided through a login node. A resource manager and scheduler provide the logic to schedule jobs efficiently on the cluster.

Why do we use HPC?

Question

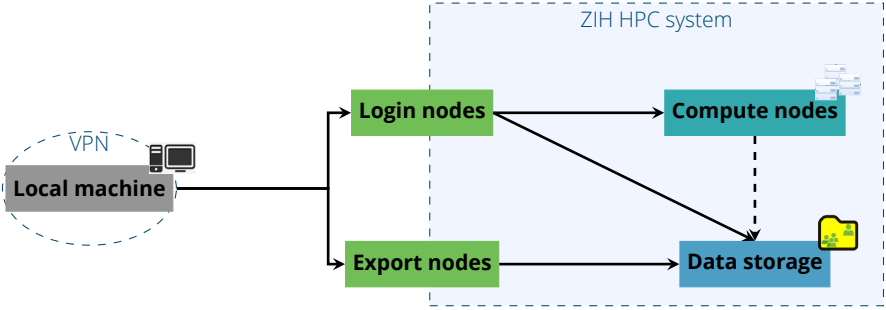
What are the things that we look for, while performing our task on a computer? What are the issues that we come across?
Computation? Memory? or both?

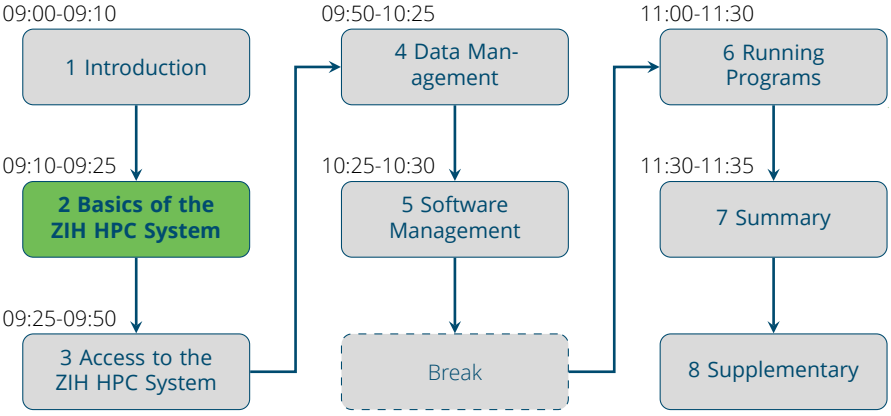
- **High speed**: Massive calculations in a relative short time
- **Reduced cost**: Faster answers mean less money; Affordable cloud-based HPC
- **Scalability**
- **Innovation**: Science, technology, business, and academia
- More information on HPC: https://hpc-wiki.info/hpc/HPC_Wiki

HPC system – Taurus

- **Name**: HPC-Cluster Taurus
- **Model**: The second high-performance computing/memory complex (HRSK-II)
- **Facility**: Center for Information Services and High Performance Computing (ZIH)
- **Details**:
  - More than 60,000 cores,
  - GPU partition with 128 dual GPU.All nodes have local SSD.
  - Flexible storage hierarchy with about 16 PB total capacity,
  - Linux, shared login nodes, filesystems, batchsystem Slurm,
  - Perfect platform for highly scalable, data-intensive and compute-intensive applications.

The ZIH System




09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

Typical Workflow

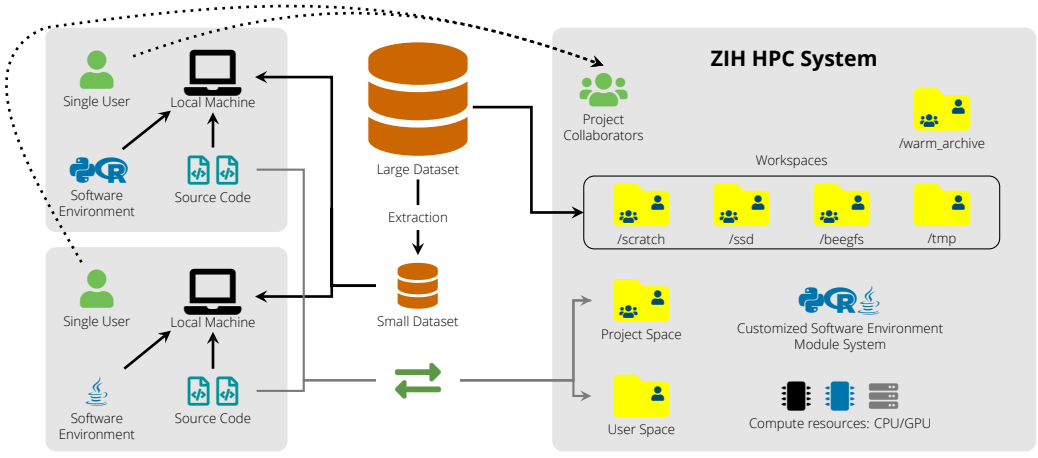
Recommended workflow to achieve a productive command line (CLI) based pipeline (some steps are optional)

1. Local machine: development of a pipeline as prototype (maybe with virtual environments like virtualenv or conda), usage of IDE or within jupyter notebook
2. Local machine: fully working CLI-based pipeline for small model/data
3. HPC machine: switch to a compute cluster with need for larger resources
4. HPC machine: use graphical front-end as jupyter notebook for testing purposes (software, hardware, algorithms)
5. HPC machine: fully working CLI-based pipeline with full data

Hint

See the official docs for the ZIH system: https://doc.zih.tu-dresden.de/.

Switching from Local Machines to HPC Systems

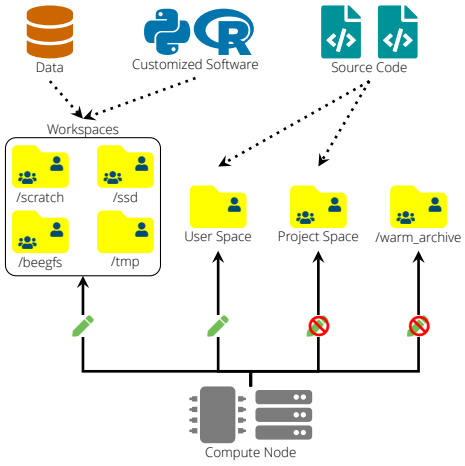


Collaborative Working

- We can achieve a collaborative working environment at two different levels based on the access/permission restrictions on the cluster:
  - Consideration 1: setting permissions to **different users** in your project (esp. important for data and customized software environments)
  - Consideration 2: Defined access possibilities based on **which node** you are on (login or compute) and on **which storage filesystem** you are trying to access (e.g. beegfs vs archive)
- It is important to distinguish usage of data, software environment, source code.

Hint

- **Data**: in workspaces with group access
- **Software environment**: module system or workspaces with group access
- **Source code**: user space or user specific directory in the project space



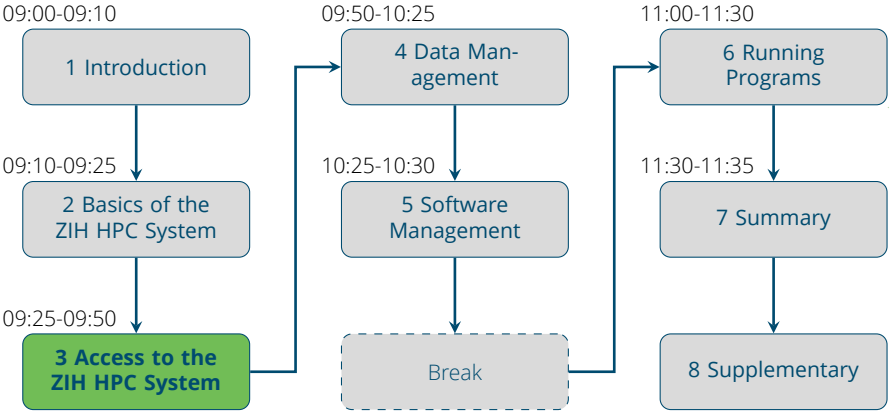
HPC project application

The ZIH system is structured by HPC projects. A HPC project on the ZIH system includes:

- project directory
- project group (linux group)
- project members (at least project leader and project administrator)
- resource quotas for compute time (CPU/GPU hours) and storage

There are different possibilities to work with the ZIH HPC:

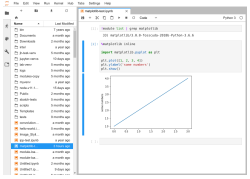
- create a new project
  - fill in the application form: https://hpcprojekte.zih.tu-dresden.de/
  - find additional information on the wiki:
- join an existing project: e.g. new researchers in an existing project, teaching purposes


09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

Access the ZIH HPC System

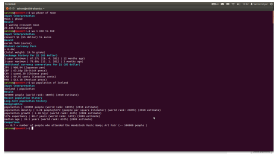
JupyterHub

- browser based approach
- easiest way for beginners



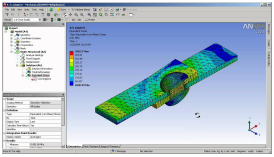
SSH connection (CLI)

- "classical" approach
- command line interface (CLI) knowledge is necessary



Desktop visualization

- esp. in the case of using GUI-based software
- e.g. Ansys, Vampir,...



Note

- When working from **outside of the university network**, the ZIH HPC system can be accessed **only via Virtual Private Network (VPN)**
- Please use **provided** login information for this training

Hint

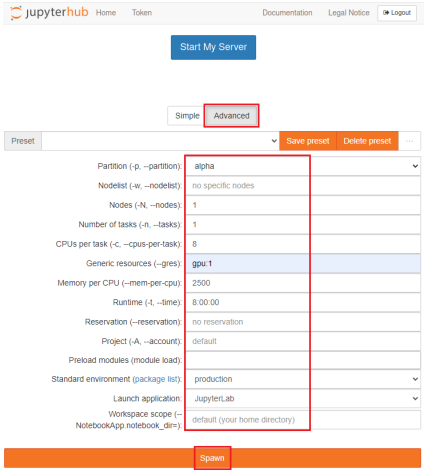
More info on VPN:

Access via JupyterHub

1. Connect to TU Dresden network via VPN, if you are accessing outside university network
2. Go to following JupyterHub link using any browser: https://taurus.hrsk.tu-dresden.de/jupyter
3. Login using ZIH-ID and password
4. Click on "Start My Server". This doesn't start any job.
5. Allocate resources via the spawner interface, and click on "Spawn" to start job

Hint

More information in the wiki:



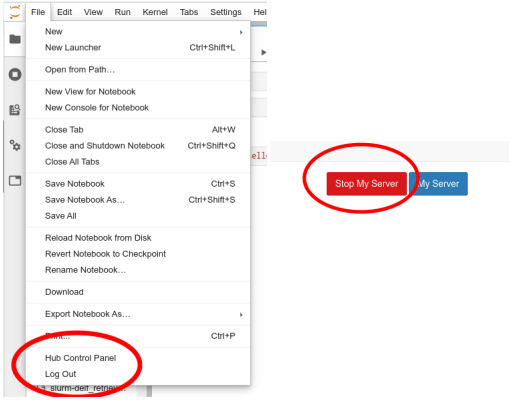
Access via JupyterHub

- When using JupyterHub do not forget to **logout** and then **stop your server**! Otherwise, the resources will not be available to others and will be included in your CPU quota!

Hint

In case of any issues while accessing/launching JupyterNotebook, it is recommended to:

- clear browser cookies and cache, and/or
- use incognito/private browser mode



Access via SSH connection (CLI)

ZIH systems can be accessed via CLI using SSH connection.

1. Connect to TU Dresden network via VPN, if you are accessing outside university network
2. Launch terminal
3. Connect to ZIH HPC login-nodes via `ssh` using ZIH-ID, login-node address (`taurus.hrsk.tu-dresden.de`) and ZIH account password
4. Allocate resources via `srun` or `sbatch`

Hint

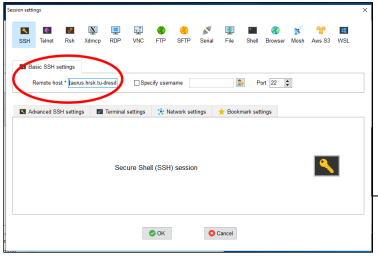
More info at:

Example

```
# Connect to the cluster via SSH
marie@local$ ssh marie@taurus.hrsk.tu-dresden.de

# Allocate resources via srun (for an interactive session)
marie@login$ srun --partition=alpha --gres=gpu:1 --cpus-per-task=8 --mem=8000 --time=6:00:00
      --account=p_scads_trainings --reservation=p_scads_trainings_1149 --pty bash

# Now you can use the resources
marie@compute$ [...]
```

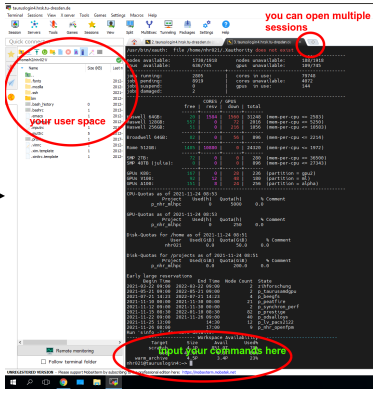
Access via SSH connection (CLI) (contd.)

- Accessing the ZIH system from a **Windows system** is done via SSH client
- We recommend using **MobaXterm** (download free version at https://mobaxterm.mobatek.net/download.html)



Remote host: `taurus.hrsk.tu-dresden.de`
username:`<ZIH-ID>`

successful login



Hint

More info at:

Try it Yourself

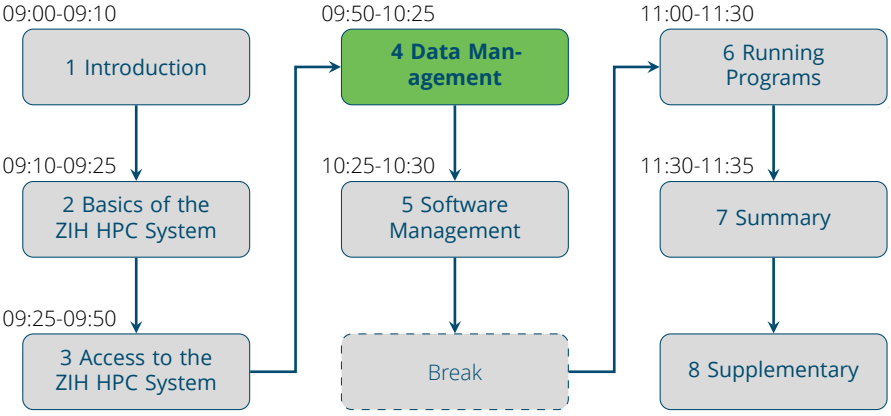
- Accessing HPC via MobaXterm, JupyterHub
- Use basic CLI Commands
- Use nano open, modify and save a text file

Once you are logged in try doing following commands in CLI:

- copy/insert from/to with CLI: `Ctrl` +`Shift` +`c` or `Ctrl` +`Shift` +`v`
- copy/insert from/to in MobaXterm: just select/Shift+insert or right-click-mouse
- use arrow up and down for command history
- use Tab-completion
- nano is command line text editor for Linux

Hint

More info about basic CLI commands can be found at 51


09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

Filesystems

- There are different areas for storing your data on the ZIH HPC system, called **filesystems**
- The filesystems have different properties (available space, time limit, size, permission rights). Therefore, choose the one that fits your project best.
- You need to create a workspace for your data on one of these filesystems.

Note

For this presentation: a filesystem refers to a "space/place" to store data and a workspace refers to the "access" you created to that filesystem

- Overview of filesystems

scope	mount point	speed	size	duration
local	/tmp	+++	---	---
Global temporary	/beegfs	++	--	--
Global temporary	/ssd	+	-	--
Global temporary	/scratch	-	+	+
Global permanent	/projects /home	--	++	++
archiving	/warm_archive	---	+++	+++

Note

There are also other archives available via datamover nodes.

Workspaces

- On the ZIH system, data has a **limited lifetime** depending on the filesystem
- User creates a workspace on filesystems with defined **expiration date**
- Data is **deleted** automatically after expiration.

Note

`/projects` and `/home` are permanent filesystems **without expiration** date

Some commands to manage workspaces on the ZIH system:

Command	Description
`ws_find --list`	Find available workspace filesystems
`ws_allocate -F <filesystem> <name_of_your_ws> <duration_of_your_ws>`	Allocate workspace
`ws_list`	List your workspaces and get information
`ws_extend -F <filesystem> <name_of_your_ws> <duration>`	Extend workspace

Hint

More info at:

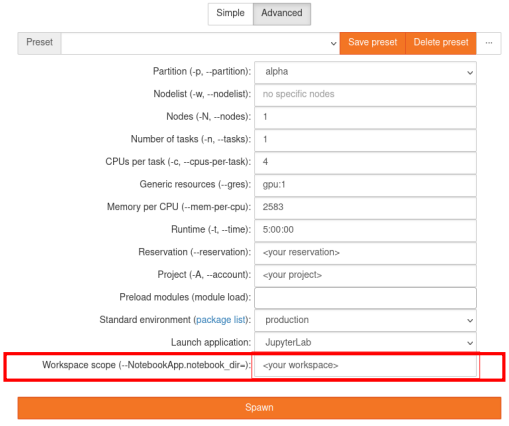
Workspaces

Example

```
#Check the available filesystems
marie@login$ ws_find --list

#Create a beegfs workspace and retrieve workspace path
marie@login$ ws_allocate -F beegfs myworkspace 10

#Check workspace information (e.g. duration, path,...)
marie@login$ ws_list

#Create a directory for the data in the workspace
marie@login$ mkdir /beegfs/ws/1/marie-myworkspace/src

#Create a directory for the model in the workspace
marie@login$ mkdir /beegfs/ws/1/marie-myworkspace/output
```

Workspace: JupyterHub

- On JupyterHub, the working directory can be set in the spawner options
- Set `Workspace scope` parameter to the full path of required workspace
- Specified workspace will be set as the default directory for the notebook execution



Access Permissions

Permission management is done by the 'chmod' command via CLI:

```
chmod [class][operator][mode] myobject
chmod [ugoa][-+=][rwx] myobject
```

where:

- class = `[ugoa]` = **u**ser/owner, **g**roup, **o**ther, **a**ll three classes
- operator = `[-+=]` = remove(-), add(+), set(=) the specified modes for the specified classes
- mode = `[rwx]` = **r**ead, **w**rite, e**x**ecute

Hint

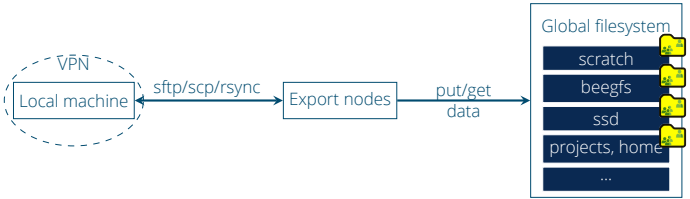
More info at: and or `man chmod`

Example

```
# check permissions of myobject i.e. the workspace
marie@login$ ls -l /beegfs/ws/1/marie-myworkspace/

# give "read, write and execute" access to the group that owns myobject
# -R means recursively i.e. on all files and directories at myobject
marie@login$ chmod g+rwx -R /beegfs/ws/1/marie-myworkspace/
```

Datatransfer: Local Machine and ZIH System

- Connection with local machine in VPN to export nodes via `scp`, `sftp`, `rsync`
- Transfer data to your target directory via export node



Hint

More info at:

Example

```
1 marie@local$ scp myscript.py marie@taurusexport.hrsk.tu-dresden.de:/home/marie/
2 marie@local$ rsync -avz myscript.sbatchr marie@taurusexport.hrsk.tu-dresden.de:/home/marie/
3 marie@local$ rsync -avz requirements.txt
      marie@taurusexport.hrsk.tu-dresden.de:/beegfs/ws/1/marie-myworkspace/
```

Datatransfer: External sources

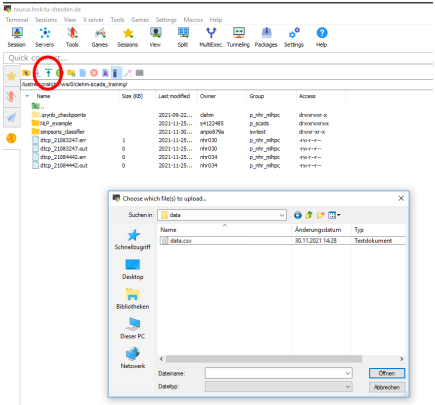
- Any file from external sources can also be downloaded on ZIH HPC system directly using commands like `wget` .

Example

```
1 #Use wget command to get dataset from the web
2 marie@compute$ wget --directory-prefix=/beegfs/ws/1/marie-myworkspace
      https://cloudstore.zih.tu-dresden.de/index.php/s/bm9HMTRGbN9ibrn/download/myfile.txt
3
4 #Check the content of the file with the nano editor
5 marie@compute$ nano /beegfs/ws/1/marie-myworkspace/myfile.txt
```

Datatransfer: Windows

With an SSH client (such as MobaXterm), data can be copied with a GUI



Datatransfer: Large vs Small Data

- Runtime of applications on login nodes is **limited** to 10min
- Data transfer that takes longer than that, will be **canceled**
- Therefore, it is recommended to use **export nodes** for large data transfer (might take longer)

Note

For very small data (up to 100MB), you can also use the login nodes (i.e. a single script file)

Datatransfer: Within the ZIH system

- Special data transfer nodes for moving large data between different filesystems
- Commands for data handling and transfer are prefixed with `dt`
  - `dtcp`, `dtls`, `dtmv`, `dtrm`, `dtrsync`, `dttar`
  - `dtqueue --me` to check the status of the data transfer
  - These commands create a **Slurm job** with dedicated resources to conduct the data handling/transfer.

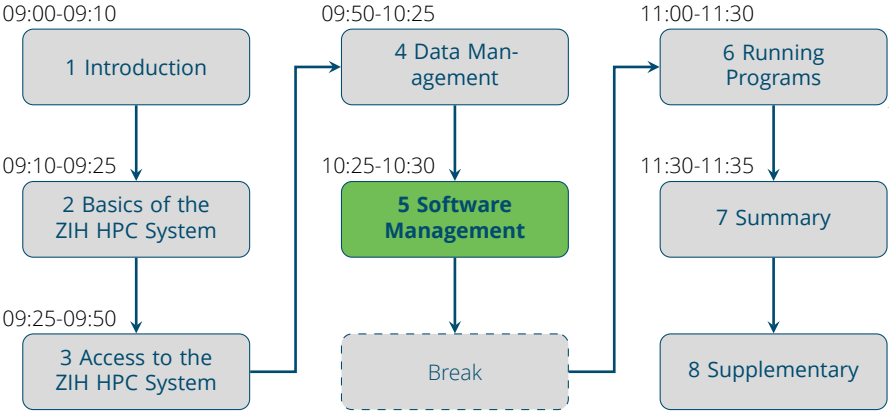
Hint
More info at:

Example

```
1 # Copy the training data into beegfs workspace directory
2 marie@login$ cp myscript.py /beegfs/ws/1/marie-myworkspace/src/
```

Example

```
1 # Copy the training data into beegfs workspace directory
2 marie@login$ dtcp myscript.sbatch /beegfs/ws/1/marie-myworkspace/src/
3
4 # Check copy job status
5 marie@login$ dtqueue --me
```


09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

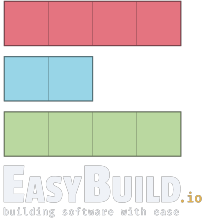
Module System

- On the ZIH system, software is organized in modules.
- A **module** is a user interface, that:
  - allows you to easily switch between different versions of software
  - dynamically sets up user's environment (PATH, etc.) and loads dependencies

- ZIH HPC system use **EasyBuild** framework to build and install scientific frameworks.

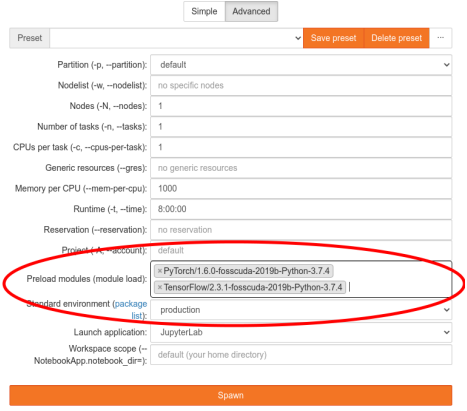
Hint

More info at: and

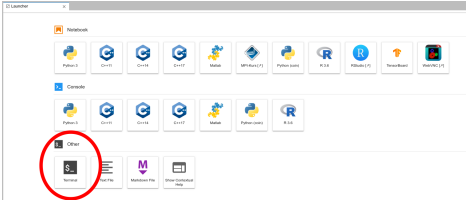


Module Management: JupyterHub

Preload modules before spawning a notebook



After spawning (wihtout preloading modules), loading modules in the JupyterHub terminal **doesn't** make them available inside Jupyter Notebook



Module Management: CLI

After allocating resources, setting up your software can be done using following module commands:

Command	Description
`module avail/spider <your software>`	Find required software
`module show <module name>`	Show additional module information
`module load <module name>`	Load module
`module list`	List all loaded modules
`module rm <module name>`	Unload module
`module purge`	Unload all module
`module save`	Save modules for next login

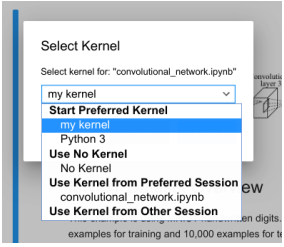
Example

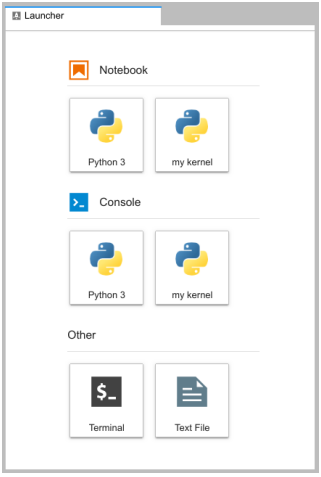
```
1 marie@compute$ module load modenv/hiera GCC/10.2.0 CUDA/11.1.1 OpenMPI/4.0.5 PyTorch/1.9.0
      tqdm/4.56.2 matplotlib/3.3.3
2
3 marie@compute$ module list
```

Custom Jupyter Kernel

Allows you to install your own preferred Python packages and use them in your notebooks.

- You can switch kernels of existing notebooks in the menu
- Your kernels are listed on the launcher page





Hint

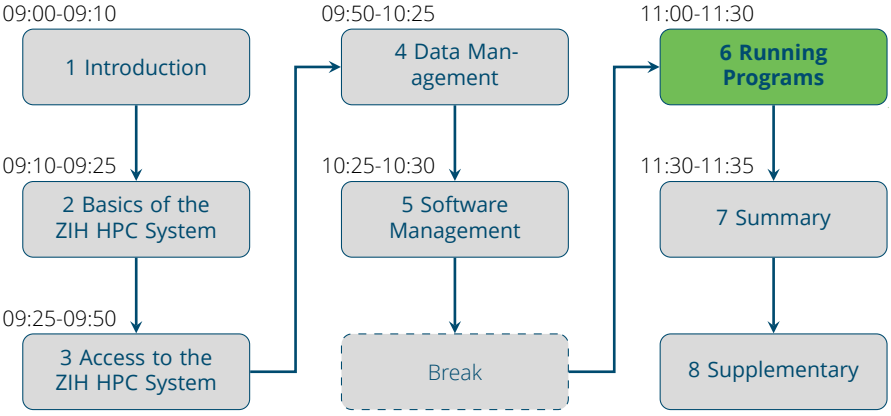
To create and use your own kernel, see here: and

Python Virtual Environment + Kernel

- Virtual environments are isolated run-time environments and allow users to install additional Python packages
- For managing virtual environments on the ZIH HPC system, `virtualenv` is preffered to be used, which is a part of the Python modules

Example

```
# Load default Python
marie@compute$ module load Python

# Install virtual environment
marie@compute$ virtualenv --system-site-packages /path/to/desired/location/my-env

# Activate the virtual environment
marie@compute$ source /path/to/desired/location/my-env/bin/activate

# User can now install additional packages using pip command
(env) marie@compute$ pip install -r requirements.txt

# register and name your kernel
(env) marie@compute$ pip install ipykernel
(env) marie@compute$ python -m ipykernel install --user --name my-env --display-name="my kernel"
```


09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

Workload Management System

- User-defined calculations, programs etc. are **not run directly** and **interactively** on an HPC system (as commonly on a personal workstation or laptop)
- Running some program on an HPC machine means running it within a **temporary resource allocation**
- The definition of a program and a resource allocation is a **job**

Note

Most crucial issue esp. for HPC beginners: You need to specify in advance compute, memory, and time resources according to your program's needs. We will refer to this later.

- On the ZIH system: job scheduling and workload management with **slurm**
  - **Manages** jobs and provides an **interface** for the users to submit their jobs
  - **Evaluates** resource requirements and **priorities**, **distributes** jobs to suitable compute nodes –> "job queue"

slurm workload manager

More details about job schedulers can be found at: `https://hpc-wiki.info/hpc/Scheduling_Basics`

Job Scheduler Basics: Slurm

- Some basic commands used to manage Slurm jobs

Type	Command	Description
Job submission	`srun <params> <command-to-run>`	Allocate resources and execute an application (interactive)
	`sbatch <params> <command-to-run>`	Run a command, script etc. with dedicated hardware resources (non-interactive)
Job management	`scancel <jobid>`	Cancel job
	`squeue --me`	List own jobs in queue and retrieve jobid
	`whypending <jobid>`	Get info why a job has not started yet
	`sinfo <params>`	View information about nodes and partitions

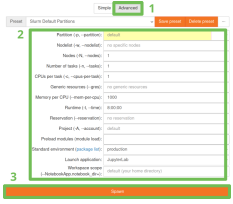
- Hardware limits can be found at:
- More slurm parameters can be found at:
- Additional commands in the docs can be found at:

Job Scheduler Basics: Running Jobs

JupyterHub

Login to JupyterHub (slide 16)

Set parameters and load the required modules (slide 16)



Running the application either using **Console** or **Jupyter Notebook**

Terminal Notebook

Interactive

Login to ZIH HPC System login-node using console (slide 18)

Using `srun` to alocate resources and attach shell using `--pty bash -l`

```
marie@login$ srun <params> --pty bash -l
```

Load the required modules (slide 35)

```
marie@compute$ module load <module>
```

Run the application

```
marie@compute$ python my_script.py
```

Batch

Login to ZIH HPC System login-node using console (slide 18)

Create a sbatch script

```
#!/bin/bash
#SBATCH --partition=alpha    # Use alpha partition
#SBATCH --nodes=1            # Use one node
#SBATCH --cpus-per-task=2    # Use 2 threads per task
#SBATCH --mem=8000           # Use 8gb of RAM
#SBATCH --gres=gpu:1         # Use 1 GPU per node
#SBATCH --time=00:05:00      # five minutes should be enough

# Loading required modules
module load modenv/hiera CUDA/11.1.1 OpenMPI/4.0.5 PyTorch/1.9.0

# Run the application
python my_script.py >> output
```

Run the application by submitting the sbatch script usng `sbatch` command

```
marie@login$ sbatch my_sbatch_script.sbatch
```

Note

JupyterHub job is inherently a batch job. Since it is a little different approach than batch job submission using CLI, it is shown here as a separate method.

Slurm Parameters: Example

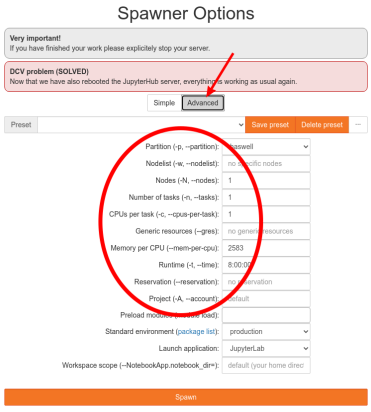
- Source code is stored in `<workspace-full-path>/myscript.py`
- alpha partition: `--partition=alpha`
- one node (no distributed application): `--nodes=1`
- two CPUs for preprocessing: `--cpus-per-task=2`
- 8GB memory for the data: `--mem=8000`
- one GPU : `--gres=gpu:1`
- run the job for 1 hr: `--time=01:00:00`
- account: `--account=p_scads_trainings` (only for this login)
- reservation: `--reservation=p_scads_trainings_1149` (only for today)

Hint

- Typically, the most crucial parameters are `--gres` , `--cpus-per-task` , `--mem` , `--time` .
- Choosing the right parameters needs application-specific knowledge and experience.
- For more insights on resources use the performance monitoring tool PIKA (see slide 53).

Slurm Parameters: JupyterHub

- Many slurm parameters are usable directly in the JupyterHub GUI (“Advanced” mode)
- Access overview is provided on slide 16.
- Note the choice of Standard environment and workspace option.
- Once the the JupyterHub is spawned, open `your-script.ipynb` notebook execute the code in it.



Slurm Parameters: Interactive Job

Example

```
# Allocate resources via slurm and run a bash interactively (--pty)
marie@login$ srun --nodes=1 --partition=compute --gres=gpu:1 --cpus-per-task=2 --mem=8000 --time=
      01:00:00 --account=p_scads_trainings  --reservation=p_scads_trainings_1149 --pty bash -l

# Now, within our temporary resource allocation, we run our interactive job
# [..do what you want and use the resources...]

marie@compute$ module load modenv/hiera GCC/10.2.0 CUDA/11.1.1 OpenMPI/4.0.5 PyTorch/1.10.0
      tqdm/4.56.2

marie@compute$ cd /beegfs/ws/1/marie-myworkspace/src

marie@compute$ python myscript.py
```

Slurm Parameters: Batch Job

Create the a batch script (`myscript.sbatch`) with the any editor:

Example

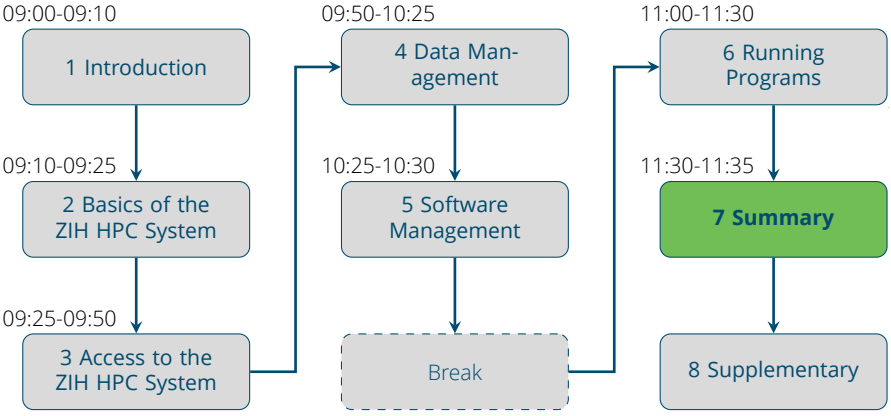
```
#!/bin/bash
#SBATCH --job-name=example
#SBATCH --nodes=1                # Number of nodes
#SBATCH --ntasks=1               # Run on a single CPU
#SBATCH --cpus-per-task=2        # use 2 threads per task
#SBATCH --gres=gpu:1             # 1 GPU per node
#SBATCH --time=0-00:05:00        # d-hh:mm:ss
#SBATCH --partition=alpha        # use alpha partition
#SBATCH --mem=2GB                # Memory per node
#SBATCH --output=%j.out          # Standard output and error log
#SBATCH --reservation=p_scads_trainings_1149
#SBATCH --account=p_scads_trainings

# From here code is executed line by line

module load modenv/hiera GCC/10.2.0 CUDA/11.1.1 OpenMPI/4.0.5 PyTorch/1.10.0 tqdm/4.56.2
cd /beegfs/ws/1/marie-myworkspace/src
python myscript.py
```

Slurm Parameters: Batch Job

Run batch job files using command: `sbatch <jobfile>`

Example

```
# run sbatch job file via sbatch command and retrieve the job id
marie@login$ sbatch myscript.sbatch
Submitted batch job 20815837
#show status of submitted job
marie@login$ squeue --me
```


09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

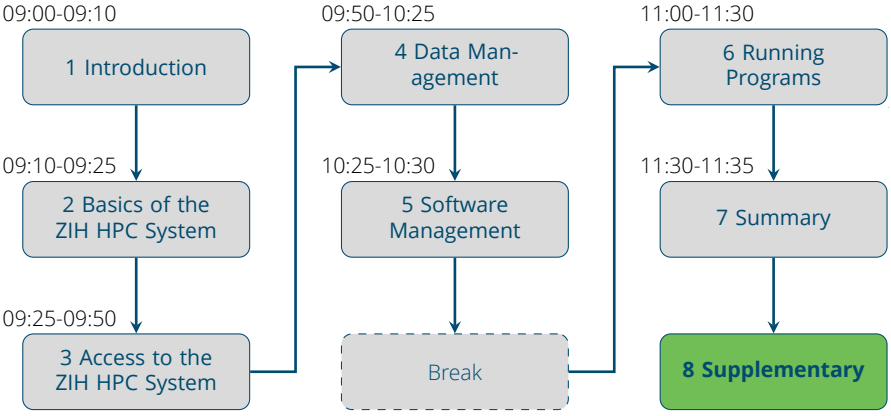
7 Summary

- There are three ways to work on HPC: JupyterHub, interactive, non-interactive.
- Do not use too much resources, make resources free after you are done.
- For more details read https://doc.zih.tu-dresden.de/
- Contact us with your own ideas, experiences and wishes!
- For getting support, esp. on ML, contact our ScaDS.AI service center https://scads.ai/.
- Logins (scads0xx) will work next 9 days, afterwards please apply for your own HPC project

ScaDS.AI DRESDEN LEIPZIG

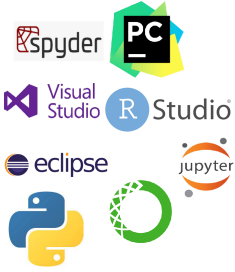
Enjoy HPC!

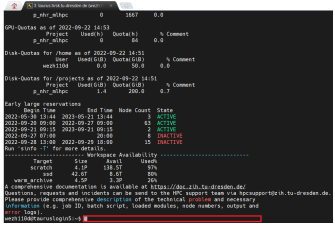
Contact us on:
neringa.jurenaite@tu-dresden.de
apurv.kulkarni@tu-dresden.de

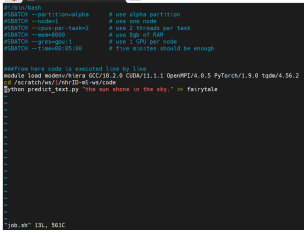

09:00-09:10
1 Introduction
09:10-09:25
2 Basics of the ZIH HPC System
09:25-09:50
3 Access to the ZIH HPC System
09:50-10:25
4 Data Management
10:25-10:30
5 Software Management
Break
11:00-11:30
6 Running Programs
11:30-11:35
7 Summary
8 Supplementary

Basics: General terms

- **Secure shell (SSH)**: is a cryptographic network protocol
- **Integrated development environment (IDE)**: software application with comprehensive facilities for software development
- **Command-line interface (CLI)**: processes commands to a computer in the form of lines of text
- **Terminal, Bash, shell**: often used as synonym for CLI (esp. on Linux systems)







Basic CLI Commands

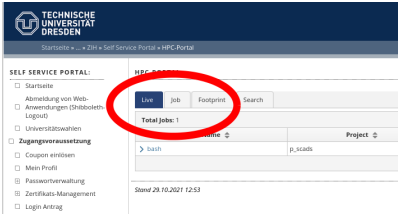
Command	Description
`ssh <hpc_name>`	connect to HPC
`ls`	list files and/or directories
`cd <path_destination>`	change directory
`mkdir <directory_name>`	create a new directory
`pwd`	Show full path of the current directory
`cp <path_source> <path_destination>`	copy a file
`cp -r <path_source> <path_destination>`	copy a directory
`mv <path_source> <path_destination>`	move a file or directory
`rm <path_source>`	delete a file or directory
`wget`	get datasets from the web (using HTTP, HTTPS, FTP and FTPS)

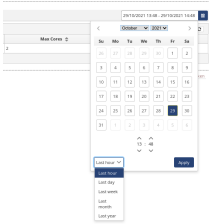
Performance Analysis – Overview

- Performance analysis: basis for optimizing parallel programs on HPC w.r.t. run-time and efficiency
- challenging task for complex applications
- analysis of an application: collecting relevant information at runtime
- different approaches to collect info:
  - easiest way: simple monitoring of CPU/GPU processes and e.g. RAM (simple tools on OS level).
  - advanced possibilities: using advanced tools level for tracing and profiling (e.g. analysis with Score-P)
- user-friendly starting point: simple monitoring using PIKA for answering questions as:
  - Are supposedly parallelized parts of the program actually executed in parallel?
  - Is the GPU actually used?

Performance Analysis – Job Monitoring with PIKA

- Simple approach for overview and to check used resources for a job: PIKA (hardware performance monitoring stack)
- monitoring during and after runtime: access PIKA web interface at https://selfservice.zih.tu-dresden.de/l/index.php/hpcportal/jobmonitoring/z../jobs_and_resources





Choose between “Live” (for running jobs) and “Jobs” (finished jobs with filtering by date/time if needed)

Hint

More info about PIKA at:

Available Resources on ZIH System

General partitions:

- `romeo`: 192 nodes, 128 cores, 512GB RAM, 200GB /tmp
- `haswell`: 1456 nodes, 24 cores, 64/128/256GB RAM, 128GB /tmp
- `julia`: 1 node, 896 cores, 47TB RAM

Machine Learning:

- `alpha`: 32 nodes, 48 cores, 1TB RAM, 3.5TB /tmp, 8x Nvidia A100-SXM4
- `ml`: 32 nodes, 44 cores IBM Power9, 256GB RAM, 6x Nvidia Volta V100

Hint

There are some more partitions available. More info at:

Workflow: Using git

Git is a version-control system, commonly used for managing source code and coordinating the work of multiple contributors on a software project.

Please consider the following basics while using git on the ZIH system:

- git is available on all cluster nodes
- can be used on HPC in the same way as in the shell of your local machine, e. g. `git clone`, `git add`, `git commit`, `git push` etc.
- every project collaborator is working on his own copy and changes are pushed to the remote repository
- no binary files are contained in the repo (this keeps the storage requirement low)
- cloned repo copies are located in user space

Note

Using the same repo copy for different users will cause permission and access issues - DON'T DO THAT.

Workflow: Using your favorite IDE

Instead of using the personal favorite IDE[1] (e.g. Eclipse, Code::Blocks, PyCharm, VisualStudio etc.) on an HPC system editors as vi or vim are available.

- Without establishing any SSH session, directories from the ZIH system can be mounted into the filesystem of the local machine with SSHFS:

Example

```
marie@local$ sshfs <loginname>@taurus.hrsk.tu-dresden.de:</path/of/your/workspace/on/zihsystem/>
      </path/on/your/localmachine/> -o nonempty
```

- after calling SSHFS files are accessible from local file browsers and can be opened with your locally installed IDE

Hint

For running source code on the ZIH system just open a CLI in an interactive session and execute the code there. Do not forget to save in your IDE and therewith writing changes to the ZIH filesystem.

[1] Integrated Development Environment