High Performance Computing - Advanced

June 23, 2026

WAGENINGEN
UNIVERSITY & RESEARCH

Instructors

- **Leonardo Honfi Camilo**
- **Jeremie Vandenplas**
- **Adrian Eliasson**

WAGENINGEN
UNIVERSITY & RESEARCH

Outline

12:30 - Intro and Ice Breaker

13:30 -High Performance Computing

12:45 - HPC Basics Review and Q&A

13:10 - Tips and Tricks

13:10 - SSH Keys

14:30 - Break

14:15 - Dynamic Job Arrays

15:00 - Types of Parallelism

15:30 - Types of Storage

15:35 - Apptainer + Exercise

16:00 - Bring Your Own Case Discussion

17:00 - End

WAGENINGEN
UNIVERSITY & RESEARCH

After this course, you should

- Be able to setup ssh keys and use them to connect to the cluster
- Be able to launch and monitor slurm jobs
- Understand different types of parallelism
- Be aware of apptainer containers and their basic usage
- Use storage resources effectively for your jobs
- Be able to interact with the community

Notation

bash shell commands start with $

```
$ echo "Hello"
```

Numbered lines denote a script

```
1 #!/bin/bash
2
3 echo -e "Hello, world
```

Ice Breaker

1. Go to **wooclap.com**
2. Enter code **WURHPC2**

WAGENINGEN
UNIVERSITY & RESEARCH

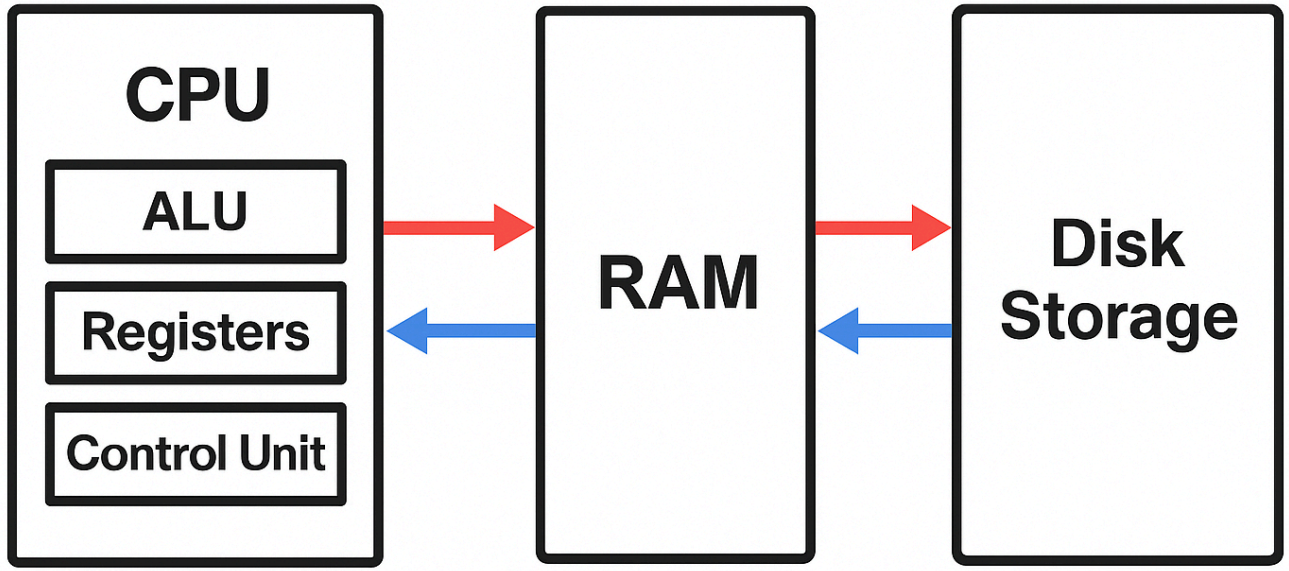
High Performance Computing

A Brief introduction into High performance computing concepts

Definition

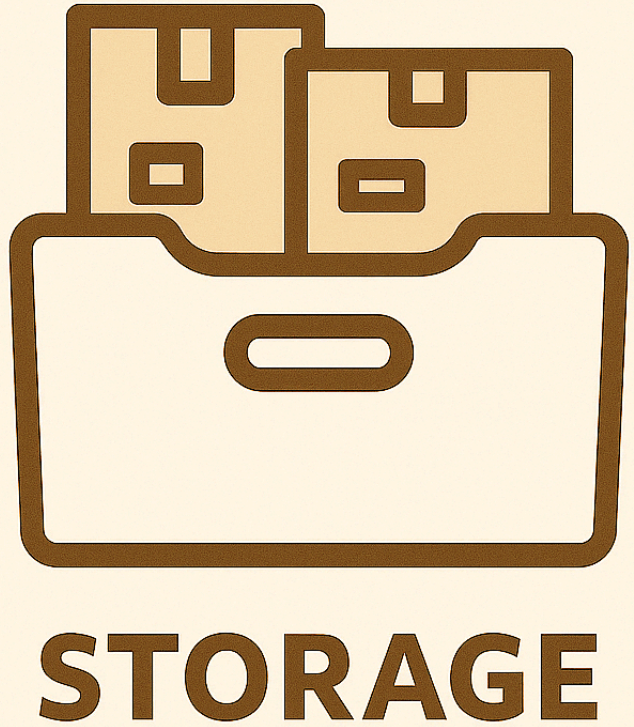
High Performance Computing (HPC) is the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation, in order to solve large problems in science, engineering, or business.

How Modern Computers Work



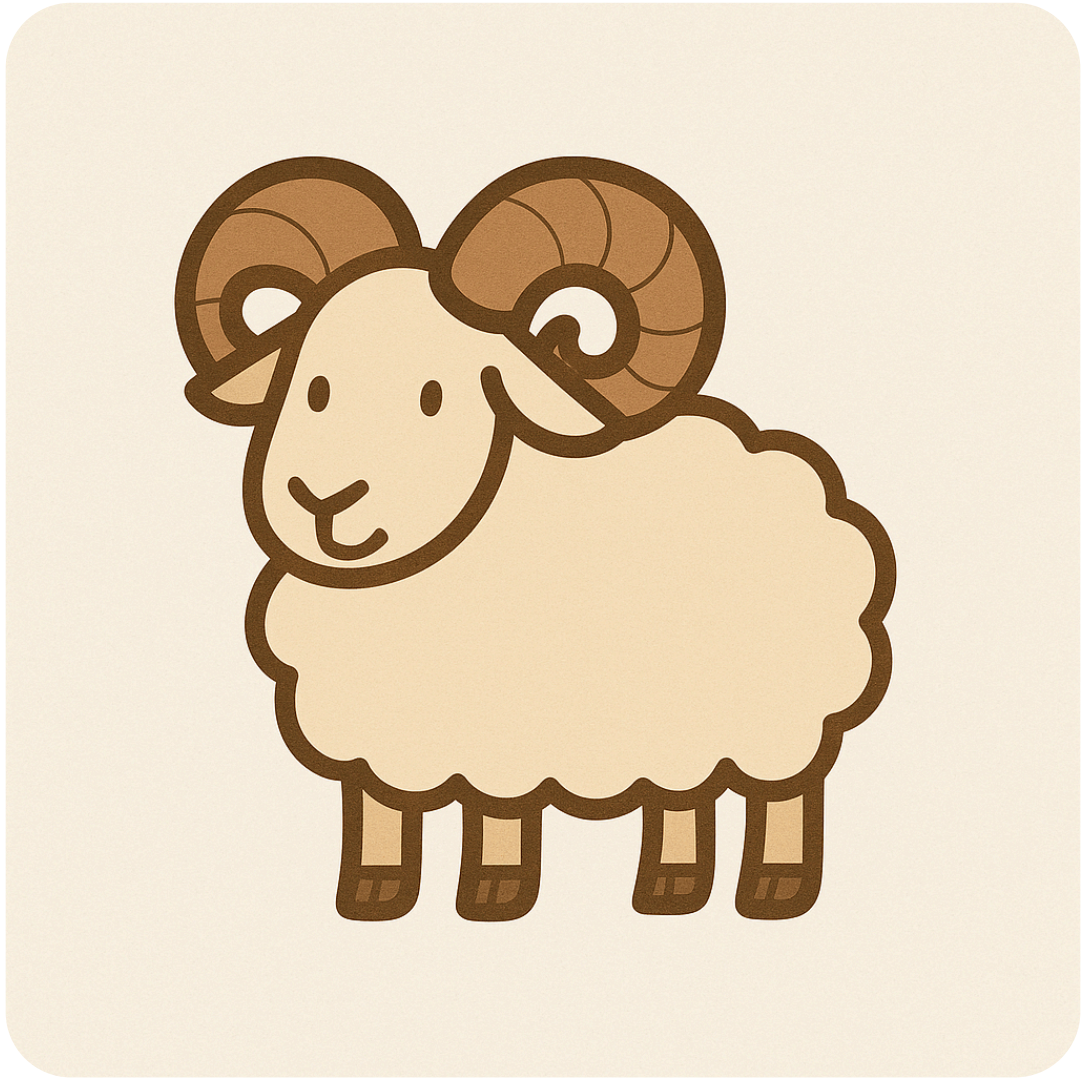
Persistent (Disk) Storage

- Usually where you keep your files, scripts and cat pictures
- Can be reasonably fast but not as fast as RAM
- Data is persistent, it does not get erased when powered off
- Examples
  - Flash drives (NVME, SSD, SDCard
  - Hard drives
  - Tapes



Random Access Memory (RAM)

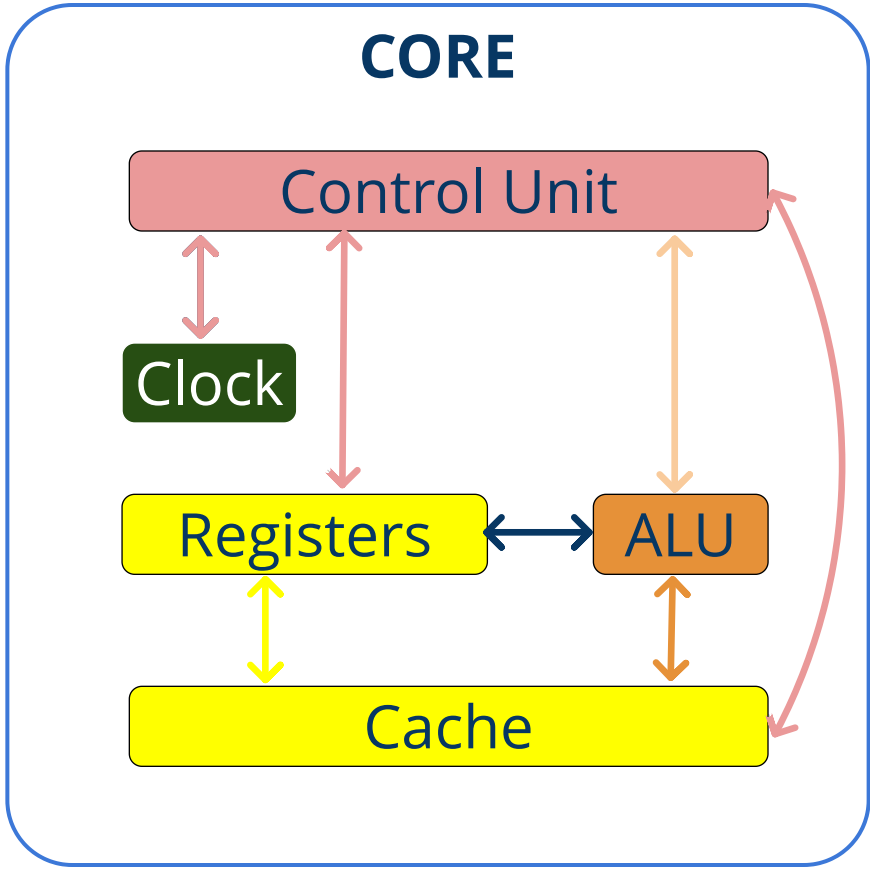
- Fast
- Used by processors
- Where active processes live
- Volatile
  - Cleared Often
  - Cleared after reboot



WAGENINGEN
UNIVERSITY & RESEARCH

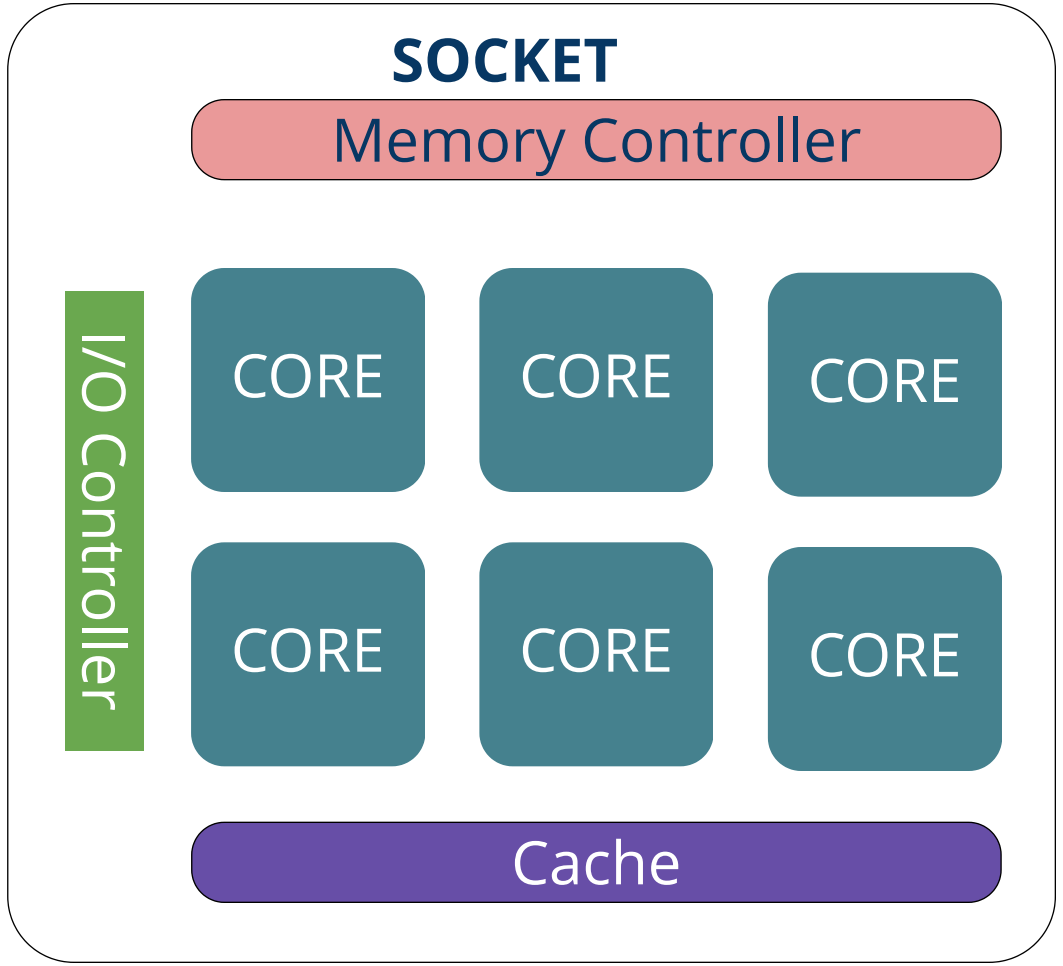
Cores

- Smallest computational unit
- Interchangeable* with cpu in HPC jargon
- Where (most of) the magic happens
- Dedicated memory (registers, cache)



Sockets

- Technically, it is the interface of a processor unit.
- Processor and socket can sometimes be used interchangeably.
- Collection of cores with shared cache and dedicated memory.
- Compute nodes usually have more than one.



Nodes

- A node is just a specialized computer in the network.
- Nodes are usually labelled after the main function they perform
- Login node - entry point, where users connect and submit jobs
- Compute node - Where computations take place.
  - Powerful - 2 Sockets
  - lots of RAM
  - interconnect with low-latency network (OmniPath, Infiniband)
- Master node - Control centre of the HPC.

WAGENINGEN
UNIVERSITY & RESEARCH

Resource Manager

Resources in an HPC are reserved by a scheduler, or resource manager.

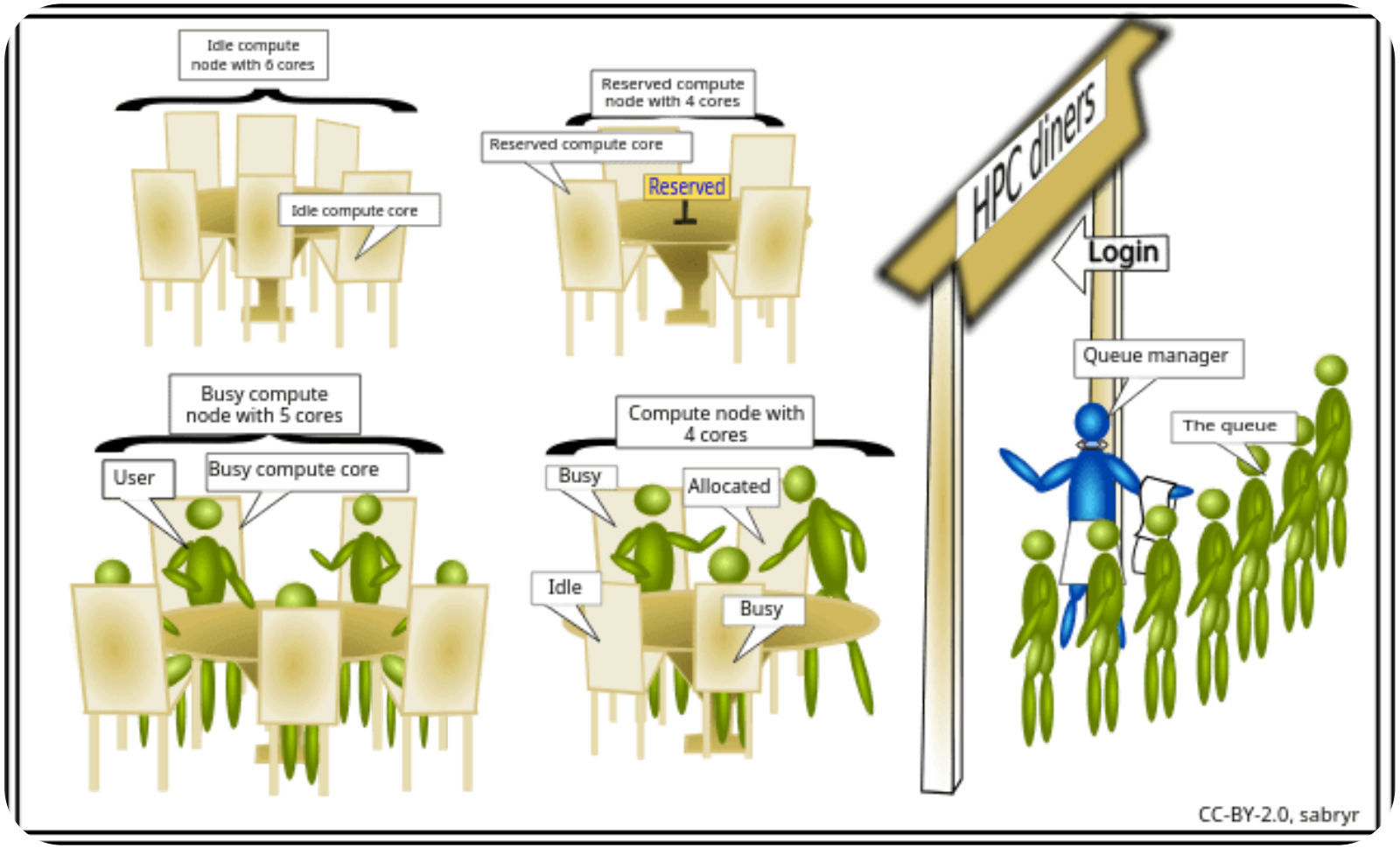
Resources:

- Cores (cpus)
- memory/memory per core
- Threads
- GPU

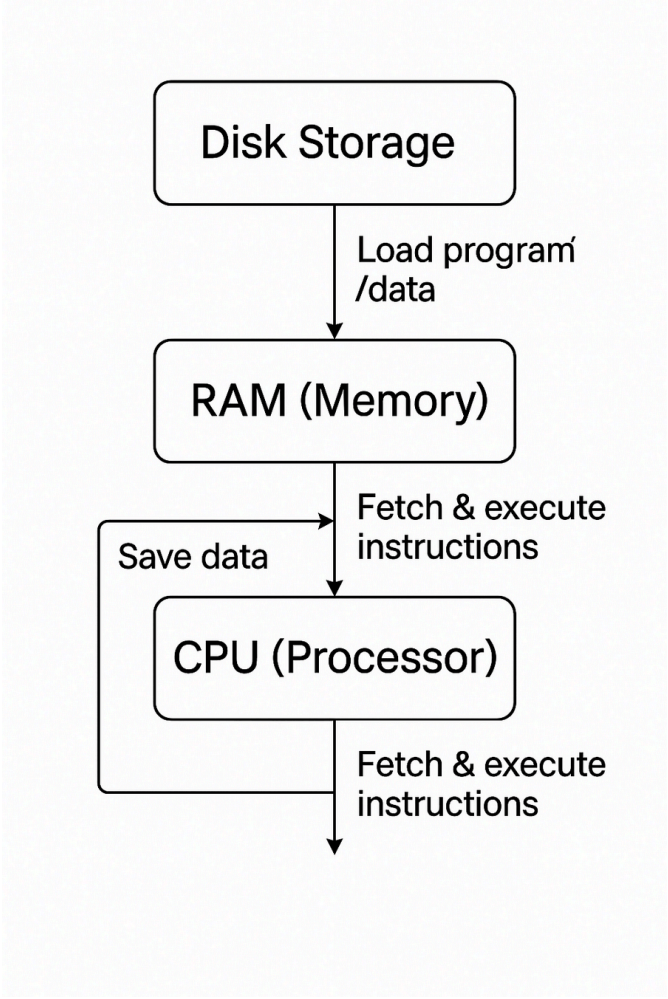
Examples:

- **Slurm**
- PBS Professional
- Torque
- LSF
- HTCondor

Restaurant Analogy



Processes



1. Code and data loaded into RAM when application is executed. This **process** gets an ID (pid)
2. CPU fetches instructions from RAM
3. Data is copied into the processor (caches and registers) to be operated on
4. Output from the CPU is written back into RAM
5. Persistent data is copied from RAM into disk storage
6. Data in RAM is erased when the process ends

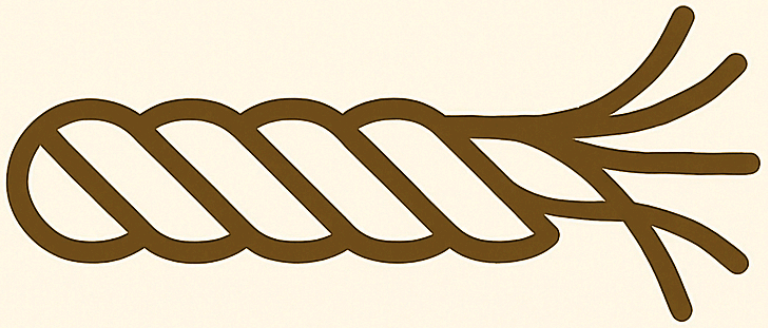
Fun Facts about Processes

- Processes copy the environment of their parent, which is usually a shell or script where their command was executed.
- Processes have their own memory address and environment.
- **Multi-process job**: Multiple **tasks** are assigned to multiple processes.

WAGENINGEN
UNIVERSITY & RESEARCH

Threads

- If a process is a rope, then a thread is one of its... threads.
- A process has at least one thread
- Threads share the same memory and environment can more easily "talk" to each other.
- Usually dedicated to simpler tasks
- **multi-threaded**: multiple cores assigned to multiple threads.
- Python and R are usually multithreaded languages.



THREADS

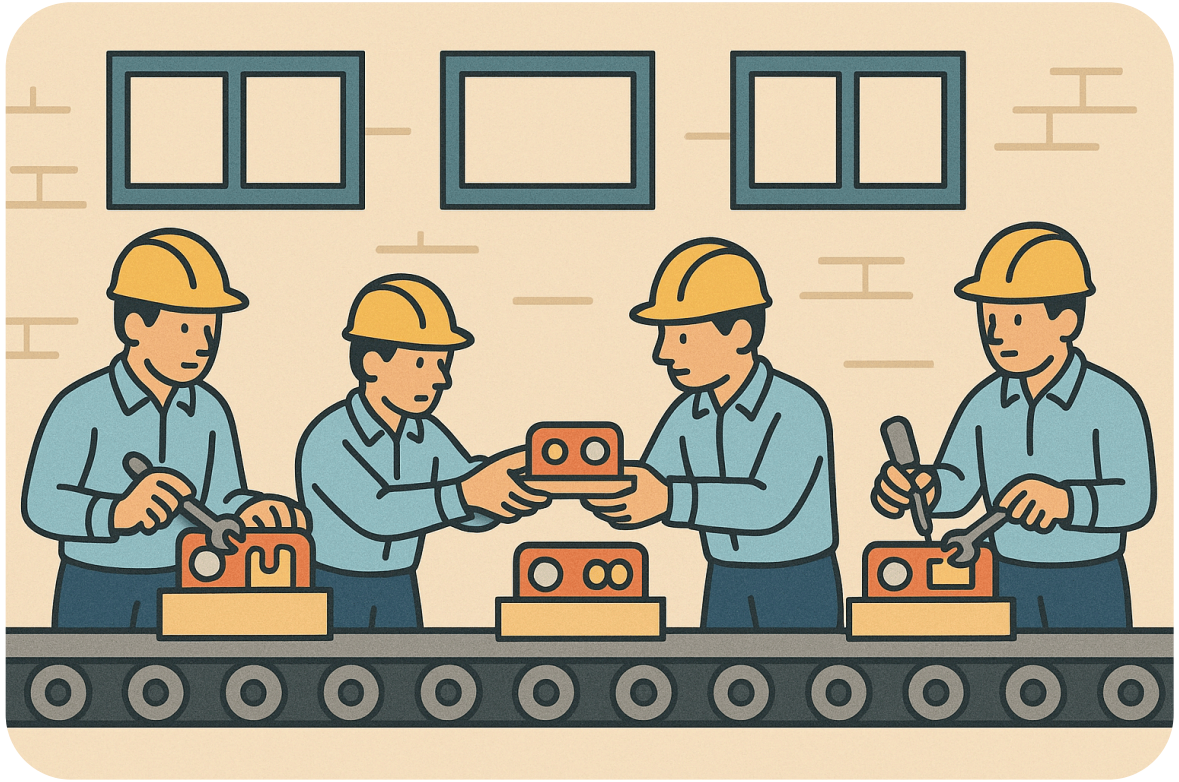
Tasks

- In broad terms, it is an allocation for a process.
- Each job has at least 1 task allocated.
- Each task corresponds to 1 process and at least 1 thread

WAGENINGEN
UNIVERSITY & RESEARCH

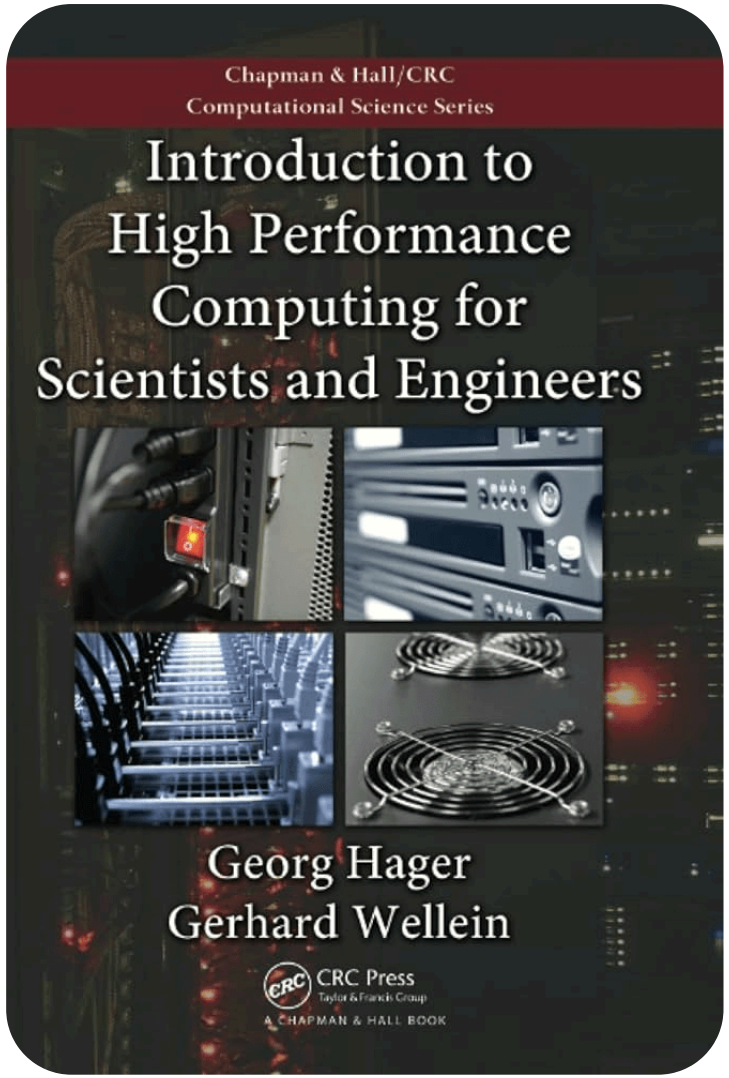
Assembly Line Analogy

- Assembly Line (**job**): determines the number of workers and number of stations.
- stations (**tasks/processes**): At least 1 worker, size (**RAM**) related to number of objects (**Data**) being serviced.
- Workers (**CPUS/threads):** at least 1 per workspace, can talk to each other or not.
- Multiple stations: **multi-processs.**
- Multiple workers in station: **multi-threaded**.



WAGENINGEN
UNIVERSITY & RESEARCH

Want to know more?

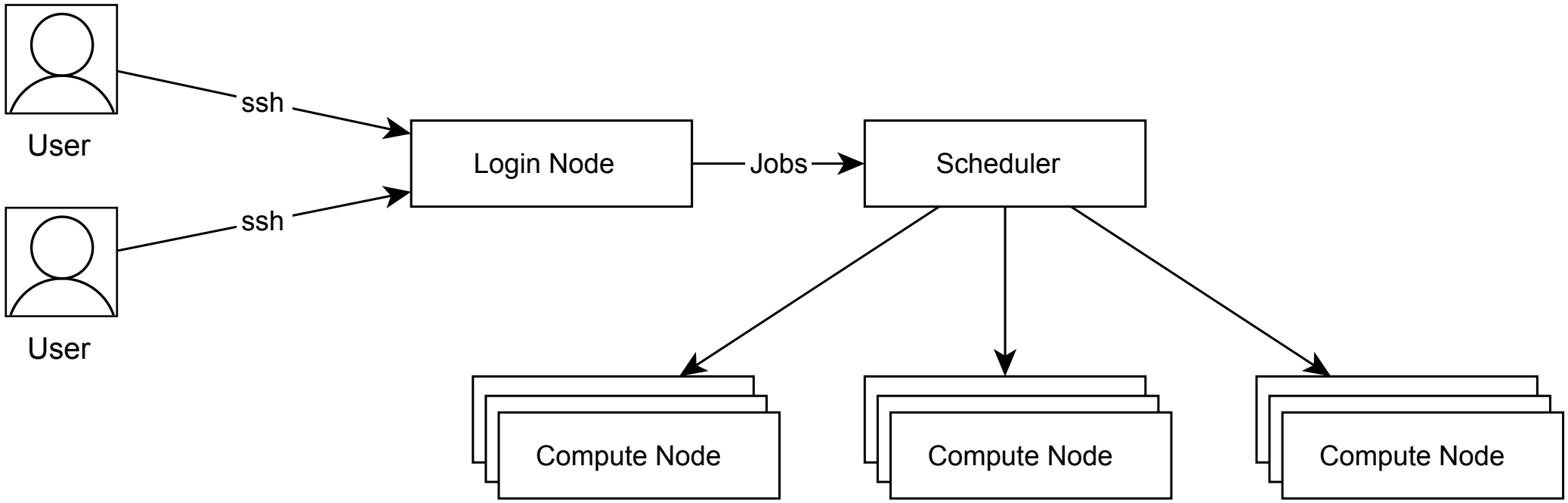


WAGENINGEN
UNIVERSITY & RESEARCH

HPC Basics Review

A Brief introduction into High performance computing concepts

HPC Diagram

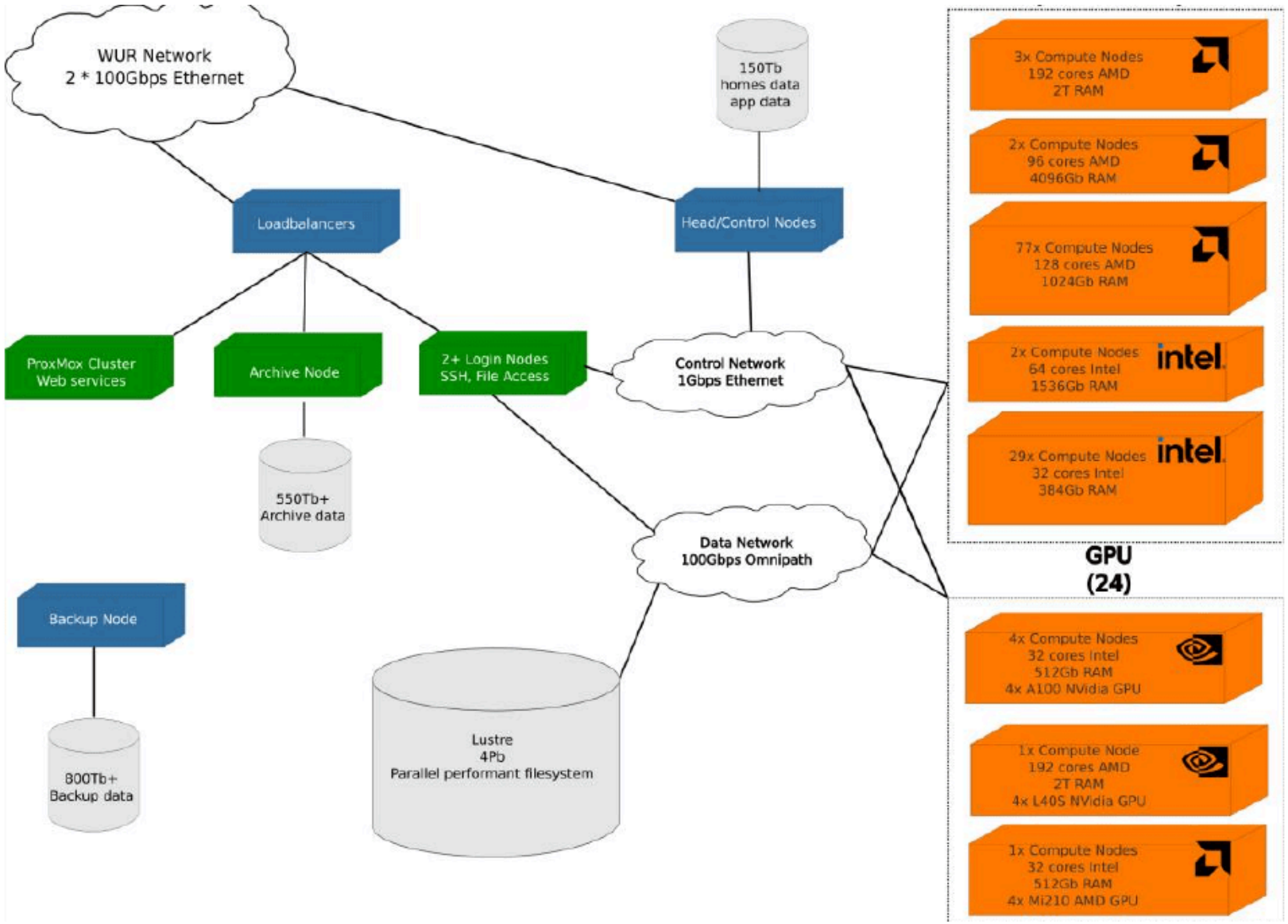

User
ssh
User
ssh
Login Node
Jobs
Scheduler
Compute Node
Compute Node
Compute Node

Overview

Cores: 11616

Nodes: 117

GPUs: 24

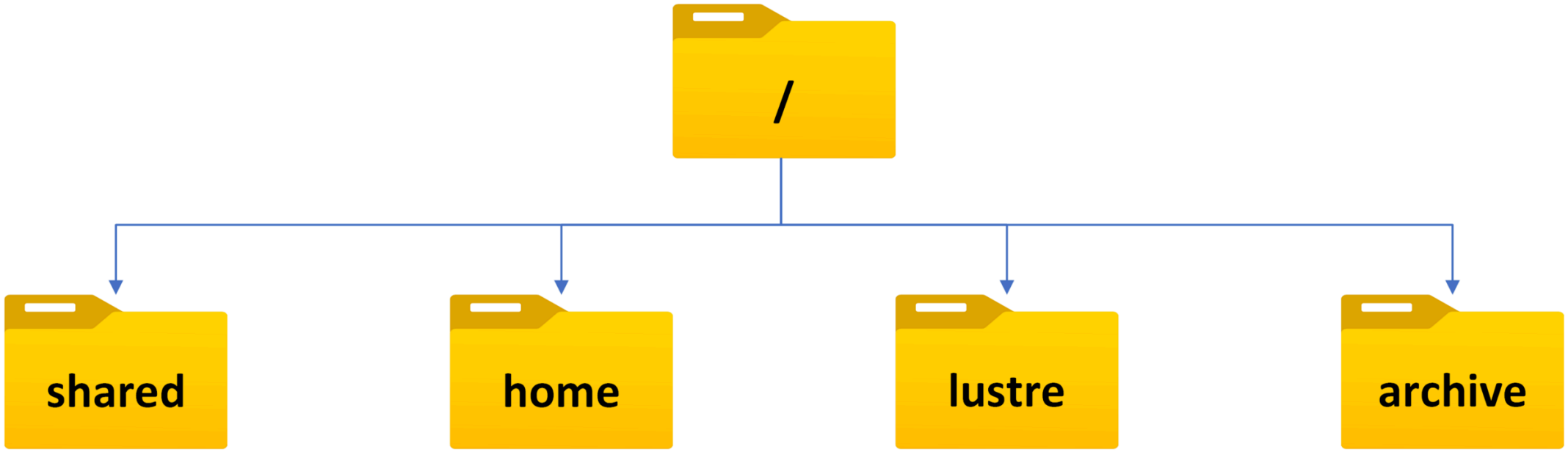


WAGENINGEN
UNIVERSITY & RESEARCH

Operating System

Ubuntu 24.04 LTS

Main Folders



Storage

Home:

- /home/[institution]/[username]
- limited to 200 GB (210GB hard limit)
- slow filesystem (not suitable for jobs)

Archive:

- /archive/[institution]/[username]
- Cheap
- Only for storage (no jobs)

shared:

- /shared
- Apps and modules

backup:

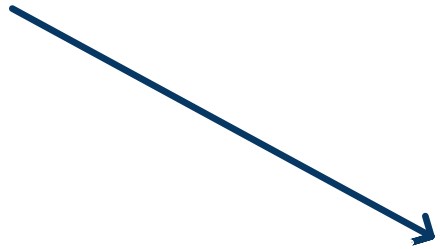
- /lustre/backup/[institution]/unit/[username]
- Extra costs for backup

nobackup:

- /lustre/nobackup/[institution]/unit/[username]

scratch:

- /lustre/scratch/[institution]/unit/[username]
- Can be cleaned up



Irods-Tape

WAGENINGEN
UNIVERSITY & RESEARCH

Beware

Irods and Tape

Irods

- There is an Irods server running at archive
- Files from Irods can be sent directly to tape
- In the future, every file in irods/archive will be sent directly to tape
- Federation with other Irods servers is in the plan

Tape

- Very cheap
- Suitable for projects that are no longer active
- itape command sends things directly to tape
- more info at our wiki: Wiki Link

WAGENINGEN
UNIVERSITY & RESEARCH

Jupyter

- Sandbox mode for small scripts running at the login0 node
- Larger jobs can run in the cluster
- Supports various versions of Julia, Python and R (**Ju****Py**te**R**)
- Reservations can be made for courses with support from the HPC team
  - Reservations for courses can be made via a form in the itsupport page
  - New request →Hosting → High Performance Computing

notebook.anunna.wur.nl

WAGENINGEN
UNIVERSITY & RESEARCH

Open OnDemand

- Graphical interface for the HPC
- Hosts GUI applications
- Buil-in:
  - File Browser
  - Jobs Manager
  - Annoucements page
  - etc...

ood.anunna.wur.nl

WAGENINGEN
UNIVERSITY & RESEARCH

Wiki

- General information about the cluster
  - Costs
  - Software
  - Workflows
  - Modules
- Everyone is welcome to contribute
- Powered by MediaWiki

wiki.anunna.wur.nl

WAGENINGEN
UNIVERSITY & RESEARCH

Software request and suggestions

ideas.anunna.wur.nl

- Request and upvote software to be installed
- Request and upvote features
- View work progress
- Most upvoted software/features will get prioritized.
- Experimental

WAGENINGEN
UNIVERSITY & RESEARCH

Best Practices and Policies

- We **never** remove user data
- No jobs should run on login node
- Maximum runtime of jobs is 3 weeks.
- No jobs should output/write to the home directory
- Do not add module commands to your **~/.bashrc** or **~/.bash_profile** files
- If you need to use anaconda, use mamba instead. Only modules are supported.

Env and Notable Variables

Display the variables in your session:

```
$ env
```

Notable:

HOME - stores the location of your home directory

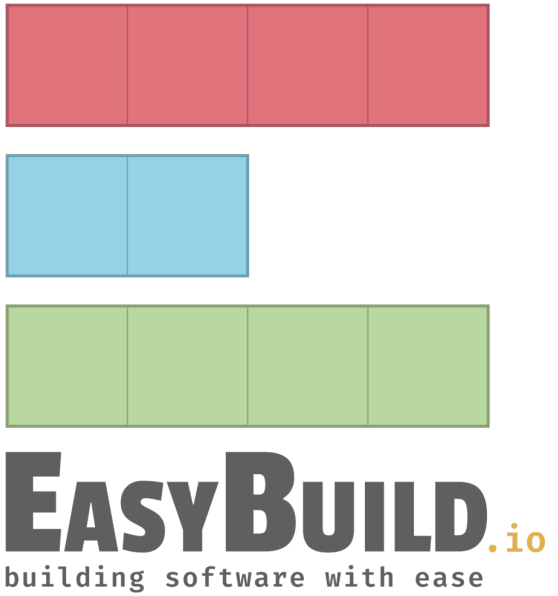
PATH - stores locations of your executable files (separated by :)

LD_LIBRARY_PATH - stores location of libraries

MODULEPATH - stores location of module files

Modules at Anunna

- Modules are arranged in "buckets" wrt to year
  - One version of software per bucket
  - Conveyor belt: software older than 3 years will be removed or moved to the **legacy** bucket.
  - A new bucket will be released each year.
- The building of the packages is being done with **EasyBuild**
  - Software tested across many HPCs worldwide
- If you need to a specific software
  - Submit a ticket
  - If you can provide an EasyBuild recipe, it will speed things up



WAGENINGEN
UNIVERSITY & RESEARCH

Module "Buckets"

- GPU
- groups
- utilities
- legacy
- 2023
- 2024
- 2025

Listing Available Modules

Get overview of available modules

```
$ module overview # ml ov
```

List available software:

```
$ module avail # ml av
$ module spider # ml spider
```

Search with a clear cache (slower):

```
$ module --ignore_cache av
$ module --ignore_cache spider
```

WAGENINGEN
UNIVERSITY & RESEARCH

Module Tips and Shorthand

ml is short for module

```
$ ml ov # module overview
```

Depending on the context it can mean different things

```
$ ml av # module avail
$ ml 2024 Python/3.12.3 # module load 2024
$ ml # module list
$ ml -Python/3.12 # module unload Python/3.12
```

ml key is a quick way to find if something is in the cluster

```
$ ml key terra
```

You can clear your cache by removing ~/.cache/lmod

WAGENINGEN
UNIVERSITY & RESEARCH

Collections

Multiple modules can be save in collections:

```
$ module save myModules
```

These collections can be retrieved in a later point with a single command

```
$ module restore myModules
```

List available collections

```
$ module savelist
```

List modules in collection

```
$ module describe myModules
```

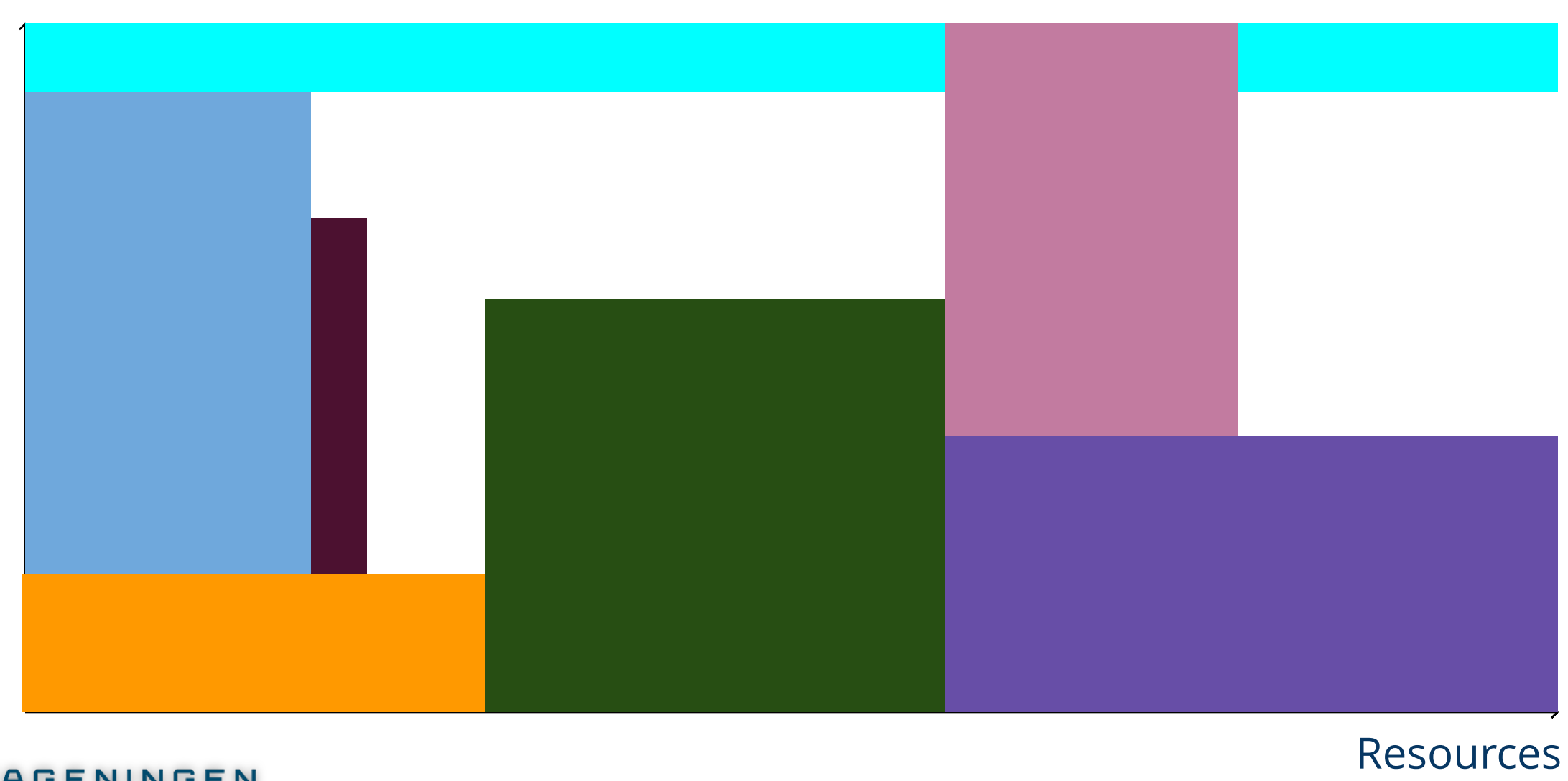
WAGENINGEN
UNIVERSITY & RESEARCH

SLURM

Simple Linux Utility for Resource Management

- Manages and **allocates resources** (compute nodes)
- Manages and **schedules jobs** on a set of allocated nodes
- **Sets up environments** for parallel and distributed computing

Tetris



WAGENINGEN
UNIVERSITY & RESEARCH

Costs

You pay for the resources you allocate for in time that you use.

Compute costs:

Queue	CPU core hour	GB memory hour	gpu hour	Node reservation day
Standard	€ 0.0150	€ 0.0011	€ 0.45	€ 50
High priority queue	€ 0.0195	€ 0.00143	€ 0.45	€ 50

Storage costs:

Lustre backup	Lustre Nobackup	Lustre Scratch	HOME	Archive	Tape
€ 150	€ 100	€ 100	€ 150	€ 65	€ 20

WAGENINGEN
UNIVERSITY & RESEARCH

The Anunna module

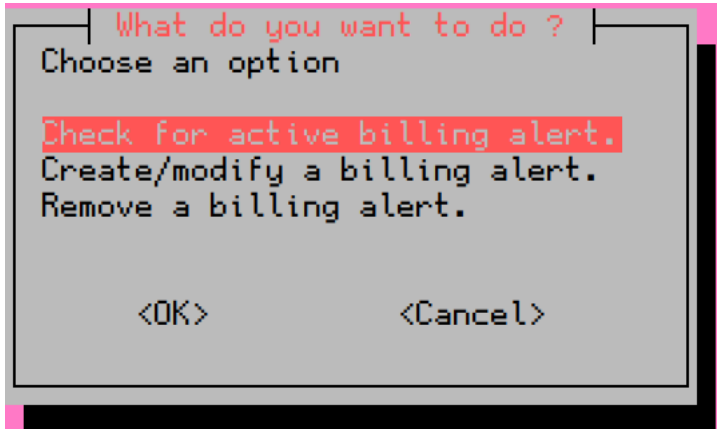
Collection of scripts and tools pertinent to anunna

```
$ module load anunna
```

get_my_bil : Estimate accumulated usage costs for the month

```
$ get_my_bill # estimates usage costs since begining of month
$ get_my_bill --start=2026-02-01 # estimated costs since specified date
```

alert_config : Sets up a usage cost email alert



WAGENINGEN
UNIVERSITY & RESEARCH

Good News, Everyone!

Free Anunna!

Upcoming change

- Fund shared research infrastructure (Anunna, Yoda, Archive Storage) collectively through the Research IT Budget
- End internal commitments and usage-based billing for WUR research groups
- Allocate costs annually through a WU/WR split based on actual usage → Annual recalibration at Management Director level
- Enable sustainable growth through fair-use policies

Expected impact

- Better alignment with national HPC/AI developments
- Lower administrative burden and simpler access
- Estimated savings of €0.5–1.0M per year
- Predictable and scalable funding model

WAGENINGEN
UNIVERSITY & RESEARCH

Interactive jobs

sinteractive template

```
$ sinteractive -c <num_cpus> --mem <amount_mem> --time <minutes> -p <partition>
```

Example

```
# sinteractive --cpus-per-task <cpus> --mem <MB>
$ sinteractive --cpus-per-task 1 --mem 2000
srun: job 19271078 queued and waiting for resources
srun: job 19271078 has been allocated resources
```

sinteractive is a wrapper for **srun**

```
1 #!/bin/bash
2 srun "$@" -I60 -N 1 -n 1 --pty bash -i
```

WAGENINGEN UNIVERSITY & RESEARCH

Note: sinteractive will fail if a job is not allocated in 60 seconds

Multi-process Interactive jobs

srun template

```
srun -N <#nodes> -n <#processes> \
        --ntasks-per-node=<#processes> \
        --time=<D-H:M> --constraint=<tag> \
        --mem-per-cpu<#G> --cpus-per-task=1 \
        --pty bash -i
```

interactive job on two node with 12 tasks, 6 per node

```
srun --nodes=2 --ntasks=12 --ntasks-per-node=6
        --time=1-0:0 --constraint=gen3 \
        --mem-per-cpu=4G --cpus-per-task=1 \
        --pty bash -i
```

WAGENINGEN
UNIVERSITY & RESEARCH

Experiment

Cores\Mem	1G	2G	4G	8G
1	FAIL	FAIL	FAIL	OK
2	FAIL	FAIL	FAIL	OK
4	FAIL	FAIL	OK	OK
8	FAIL	OK	OK	OK

- Multi-threaded languages like **Python** and **R** may have bottlenecks with single or few cores. More cores can sometimes lower the memory requirement of such jobs
- **Start small and increase as needed.**
- Measure efficiency: **seff**

SLURM Scripts

```
1  #!/bin/bash
2  ##---------- Name of the job ----------
3  #SBATCH --job-name=example01
4  ##---------- Mail address ----------
5  #SBATCH --mail-user=my.email@wur.nl
6  #SBATCH --mail-type=ALL
7  ##---------- Output files ----------
8  #SBATCH --output=output_%j.txt
9  #SBATCH --error=error_output_%j.txt
10 ##---------- Other information ----------
11 #SBATCH --comment='My project number'
12 ##---------- required resources ----------
13 #SBATCH --time=0-1
14 #SBATCH --nodes=1
15 #SBATCH --ntasks=1
16 #SBATCH --mem=4096
17 #SBATCH --cpus-per-task=1
18
19 ##---------- Environment, Operations and job step ----------
20 module purge
21 module load 2024
22 ml Python/3.12.3
23
24 echo "Hello $(hostname), my name is ${USER}!"
```

WAGENINGEN
UNIVERSITY & RESEARCH

Submitting SLURM scripts

sbatch

```
$ sbatch script_slurm.sh
Submitted batch job 19271078
```

Options specified in the command overrule the script

```
$ sbatch script_slurm.sh -N 2 -n 10 \
    --mem=32G --time=0-3:0:0 --cpu-per-task=2
```

WAGENINGEN
UNIVERSITY & RESEARCH

Extra Options

partition: main or gpu

```
--partition=main
```

or

```
--partition=gpu
```

Specific feature:

```
--constraint=gen3
```

Enable GPUs

```
--gres=gpu:1
```

CPUs per GPU:

```
--cpus-per-gpu=gpu
```

Constraints

Specifies the characteristics of the node selected for a job

Intel (gen2) - node1xx

gen2,intel,32cpus,384g,avx512,12gpercpu

AMD (gen3) - node2xx

gen3,amd,128cpus,1000g,8gpercpu

intel fat nodes - fat1xx

intel,24gpercpu,gen2,64cpus,1536g,avx512

intel fat nodes - fat1xx

gen3,amd,96cpus,4000g,41gpercpu

AMD (gen4) - node3xx

gen4,amd,192cpus,2304g,12gpercpu

Monitoring Jobs - sinfo

sinfo

View **information** about the **nodes** in the cluster

Information on nodes - exact long format

```
$sinfo -Nel
```

You can customize the information and format

```
$ sinfo --exact --format="%.10n %.11T %.4c %.8z %.10m %.10e
%.16C %.8d %.6w %.60f %20E"
```

WAGENINGEN
UNIVERSITY & RESEARCH

Monitoring Jobs -squeue

squeue

View **information** about jobs located in the SLURM **scheduling queue**

To report a list of users:

```
$ squeue -u <user_id>
$ squeue --me
```

To report a list of specific jobs:

```
$squeue -j <job_id_list>
```

To report the expected start time of pending jobs:

```
$squeue --start
```

Monitoring Jobs - sacct

sacct

- Displays accounting data for all jobs/steps
- Some information are available only at the end of the job

To get an overview of all jobs run in 2024

```
$ sacct -X -u <name> --starttime 2024-01-01 --endtime now
```

To get an overview of a job

```
$ sacct -j <job_id> --format=JobID%-20,AveRSS, MaxRSS,Elapsed
```

Monitoring Jobs - scontrol

scontrol

- Update job resource requests
- Work only for running jobs
- **Provides a lot of information…**

To get information of a job

```
$ scontrol show job <job_id>
```

To get information on nodes

```
$ scontrol show nodes
```

WAGENINGEN
UNIVERSITY & RESEARCH

check efficiency - seff

```
user001@login200:~$ seff 63358117
Job ID: 63358117
Cluster: anunna
User/Group: easybuilder/easybuilder
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 16
CPU Utilized: 09:45:44
CPU Efficiency: 2.59% of 15-16:49:04 core-walltime
Job Wall-clock time: 23:33:04
Memory Utilized: 5.20 GB
Memory Efficiency: 4.06% of 128.00 GB (128.00 GB/node)
```

Cancelling Jobs - scancel

Cancel a single job:

```
$scancel <job_id>
```

Cancel multiple jobs:

```
$scancel <job_id0> <job_id1> <job_id2> ...
```

Exercise - Create a Virtual Environment

Load the 2023 bucket and Python/3.11.3

```
$ module purge
$ module load 2024
$ module load Python/3.12.3
```

Create a virtual enviornment (in lustre scratch)

```
$ mkdir $myScratch/hpcCourse; cd $myScratch/hpcCourse
$ python -m venv $myScratch/hpcCourse/venv
$ source  $myScratch/hpcCourse/venv/bin/activate
```

Check:

```
$ which python
$ python -V
```

Install required libraries: datetime, matplotlib, pandas

```
$ pip install matplotlib pandas datetime
$ pip freeze # for checking
```

WAGENINGEN
UNIVERSITY & RESEARCH

Exercise - Slurm Script - 1/2

1. Make sure your $myScratch variable is set to the correct location

```
$ echo $myScratch
```

2. Create a folder for this course and a subfolder for your logs

```
$ mkdir -p $myScratch/HPCcourse/logs
```

3. Copy this course's **Resources** folder to your course directory

```
$ cp -r /lustre/shared/hpcCourses/Resources $myScratch/HPCcourse/
```

4.1 Copy **template.slurm** to your course folder as **weather.slurm**

```
$ cp $myScratch/HPCcourse/Resources/template.slurm  \
    $myScratch/HPCcourse/weather.slurm
```

4.2 From Resources, copy the python script we will be using

```
$ cp $myScratch/HPCcourse/Resources/weer_vanaf_2000.py $myScratch/HPCcourse/
```

WAGENINGEN
UNIVERSITY & RESEARCH

Exercise - Slurm Script -2/2

6. Editing weather.slurm:

6.1 Modify **weather.slurm** to allocate 1 core, 1GB of RAM for 10 min

Bonus: Set your error and logs to output to your newly created **logs** folder

6.2 load the same modules you used to create your venv

```
module load 2024
module load Python/3.12.3
```

6.3 Change directory to your course directory

```
cd $myScratch/HPCcourse
```

6.4 Activate your virtual environment

```
source $myScratch/HPCcourse/venv/bin/activate
```

6.5 Execute the python script with 240 as an argument

```
python $myScratch/HPCcourse/weer_vanaf_2000.py 240
```

7. Submit your job

```
sbatch $myScratch/HPCcourse/weather.slurm
```

WAGENINGEN UNIVERSITY & RESEARCH

Solution

```
#!/bin/bash
# -----------------------------Name of the job-----------------------
#SBATCH --job-name=hpcAdvanced
#-----------------------------Output files--------------------------
#SBATCH --output=logs/%j.log
#SBATCH --error=logs/%j.err
#-----------------------------Other information---------------------
#SBATCH --comment='01189998119991197253'
#-----------------------------Required resources--------------------
#SBATCH --time=0-0:10:0
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=1G
#-----------------------------Environment, Operations and Job steps----
module purge
module load 2024
module load Python/3.12.3

# Define Path of project folder
cd $myScratch/HPCcourse
# activate virtual environment
source ./venv/bin/activate
# Run python script with weather station address as argument
python ./weer_vanaf_2000.py 240
```

WAGENINGEN
UNIVERSITY & RESEARCH

Break - 10 min

Tips and Tools

Common tools and quality of life tips for using the HPC

TMUX - Terminal Multiplexer

- **Session Management**: Keep sessions alive over SSH connections.
- **Window Management**: Split terminals horizontally and vertically.
- **Productivity**: Run multiple programs in one terminal window.

Basic Workflow

1. Start a new session (usually in logins)

```
$ tmux                                  $ tmux new -s <whatever>
```

2. Run commands (e.g. copying) or start an interactive slurm session.
3. Detach the session and logout or go do something else

```
(ctrl + b) + d  # ctrl + b toggles command mode
```

4. Check what sessions you have active

```
$ tmux ls
```

5. Reattach your session and resume work

```
$ tmux a                                $ tmux a -t <whatever>
```

5. close the session (pane)

```
$ exit                                   (ctrtl + b) + x
```

WAGENINGEN
UNIVERSITY & RESEARCH

Commands

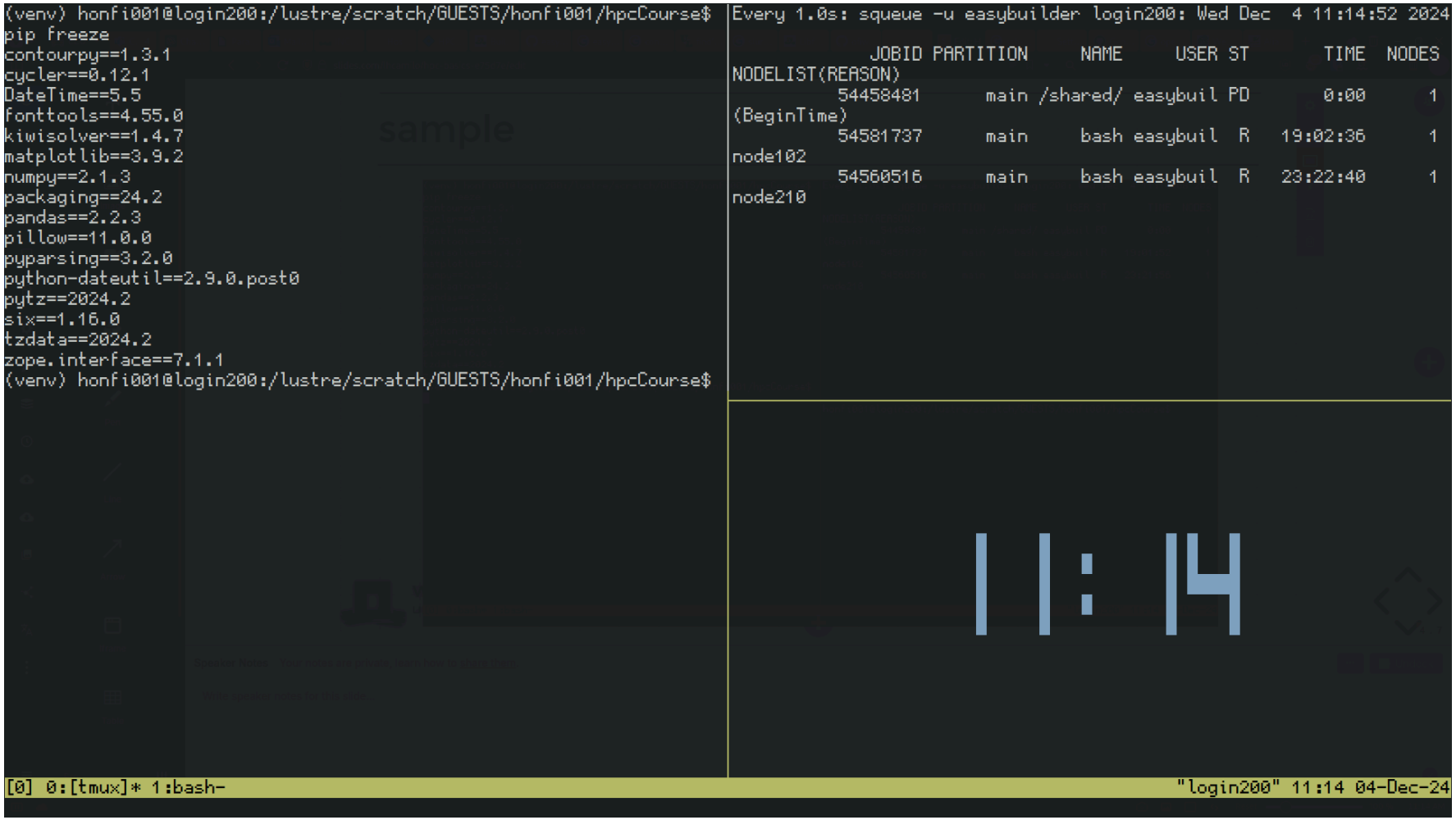
Starting and Managing Sessions

- **Start a new session**: `tmux new -s session_name`
- **List sessions**: `tmux ls`
- **Attach to a session**: `tmux attach -t session_name`
- **Detach from a session**: `Ctrl + b`, then `d`
- **Kill a session**: `tmux kill-session -t session_name`

Pane and Window Management

- **Split pane vertically**: `Ctrl + b`, then `"`
- **Split pane horizontally**: `Ctrl + b`, then `%`
- **Switch between panes**: `Ctrl + b`, then arrow keys
- **Close current pane**: `Ctrl + b`, then `x`
- **Create a new window**: `Ctrl + b`, then `c`
- **Switch between windows**: `Ctrl + b`, then `n` (next) or `p` (previous)

Example



WAGENINGEN
UNIVERSITY & RESEARCH

Symbolic Links

Similar to windows shortcuts, it links a location to another. Useful for mapping lustre folders to a folder in your home

Can be created with:

```
$ ln -s <locationSource> <linkLocation>
```

Creating a link to your scratch folder in your home

```
$ ln -s /lustre/scratch/<institution>/user001 ~/scratch
```

Check with **ls -l**

```
$ ls -l ~/scratch
lrwxrwxrwx 1 user001 domain users 31 Jun 12 10:15
/home/WUR/user001/scratch -> /lustre/scratch/GUESTS/user001
```

Remove a link with **unlink**

```
$ unlink ~/scratch
```

WAGENINGEN
UNIVERSITY & RESEARCH

Aliases

Shortcuts for long and habitual commands

Can be created with:

```
$ alias <myshortcut>="<my_long_command>"
```

Updating python packages

```
$ alias pipUpgrade="pip install -U pip && pip freeze --local | grep -v
^-e | cut -d = -f 1 | xargs -n1 pip install -U"
```

cd to scratch directory

```
$ alias cds="cd $/lustre/scratch/<institution>/$USER"
```

list available aliases in environemnt

```
$ alias
```

Exercise - Creating Variables - 1/2

If you do not have already, create an empty file in your home labelled .bash_aliases

```
$ touch ~/.bash_aliases
```

Load the anunna module

```
$ module load anunna
```

Run **getLustreDir** and append (>>) the output to your ~/.bash_aliases file

```
$ getLustreDir -h
$ getLustreDir -e >> ~/.bash_aliases
```

Check your ~/.bash_aliases file

```
# here we use cat, but your can also use less, more or nano
$ cat ~/.bash_aliases
```

Refresh your envionment

```
$ source ~/.bashrc
```

WAGENINGEN
UNIVERSITY & RESEARCH

Exercise - Creating Variables - 2/2

Test 1: Does running the command below show your scratch folder location

```
$ echo $myScratch
```

Test 2: Does running the command below take you to your scratch folder?

```
$ cd $myScratch
```

Bonus: add these lines to your ~/.bash_aliases and reload your environment again

```
1 alias cds="cd $myScratch"
2 alias cdb="cd $myBkp"
3 alias cdn="cd $myNBkp"
```

Note: the alias lines above should be place after the variables definitions

If it does not work, maybe you need to add these lines to the end of your ~/.bashrc

```
1 if [ -f ~/.bash_aliases ]; then
2     . ~/.bash_aliases
3 fi
```

WAGENINGEN
UNIVERSITY & RESEARCH

~/.bash_aliases - Example

```
1  #!/bin/bash
2
3  # location variables
4  export myScratch=/lustre/scratch/<institution>/<GROUP>/$USER
5  export myBkp=/lustre/backup/<institution>/<GROUP>/$USER
6  export myNBkp=/lustre/nobackup/<institution>/<GROUP>/$USER
7  # cd aliases
8  alias cds="cd $myScratch"
9  alias cdb="cd $myBkp"
10 alias cdn="cd $myNBkp"
11 # handy slurm aliases
12 alias sq="squeue --me"
13 alias wsq="watch -n 2 squeue --me"
```

Exercise - Creating Variables - 1/2

Using your text editor of choice (e.g. nano) create and edit a file labelled ~/.bash_aliases

```
$ nano ~/.bash_aliases
```

Fill in the location of your lustre folders with the syntax and save

```
1 export myScratch=<insert here the location of your scratch lustre folder>
2 export myBkp=<insert here the location of your backup lustre folder>
3 export myNBkp=<insert here the location of your nobackup lustre folder>
```

Open your ~/.bashrc. Make sure the following lines are there, if not add them.

```
1 if [ -f ~/.bash_aliases ]; then
2     . ~/.bash_aliases
3 fi
```

WAGENINGEN
UNIVERSITY & RESEARCH

Exercise - Creating Variables - 2/2

Reload your environment

```
$ source ~/.bashrc
```

Test 1: Does running the command below show your scratch folder location

```
$ echo $myScratch
```

Test 2: Does running the command below take you to your scratch folder?

```
$ cd $myScratch
```

Bonus: add these lines to your ~/.bash_aliases and reload your environment again

```
1 alias cds="cd $myScratch"
2 alias cdb="cd $myBkp"
3 alias cdn="cd $myNBkp"
```

Note: the alias lines above should be place below the variables definitions

SSH keys & SSH Config

How to configure SSH and use SSH keys

Config Features

- Set aliases and save addresses
- Persistent configuration easy to backup and restore (git)
- Associate configuration with addresses
- Usually located at:

Linux/Mac

```
~/.ssh/config
```

Windows

```
C:\Users\<YourUsername>\.ssh\config
```

WAGENINGEN
UNIVERSITY & RESEARCH

Example

To work with SSH more easily, you can put settings in your .ssh/config file:

```
Match host *.anunna.wur.nl
    User                        user001
    Compression                 no
    RequestTTY                  force
    IdentityFile                ~/.ssh/id_mykey
    IdentitiesOnly              yes
    ForwardX11                  yes
    ForwardX11Trusted           yes
    AddKeysToAgent              yes
    StrictHostKeyChecking       accept-new

Host anunna
        HostName login.anunna.wur.nl
```

SSH Keys

- Fingerprint for your machine
- Can be set with or without a password
- Can be associated with specific addresses in config
- Come in pairs: private and public keys
- Public key can be copied to other machines
- Private key stays in your workstation
- Can be used to with git(hub/lab)

Linux/Mac

```
~/.ssh/
```

Windows

```
C:\Users\<YourUsername>\.ssh\
```

WAGENINGEN
UNIVERSITY & RESEARCH

Procedure

At your laptop:

1. Generate ssh key on your laptop
    1. private key
    2. public key
2. Copy public key to hpc

At the remote machine:

1. Append key to ~/.ssh/authorized_keys (in the HPC)

Generating - ssh-keygen

Open a terminal/powershell in your machine

```
$ ssh-keygen -t ed25519
```

1. Confirm key name and location (press enter)
   - Windows: C:\Users\[USER]\.ssh
   - Linux/Mac: ~/.ssh
2. Type passphrase (optional but **very recommended**)
3. Type passphrase again

Generating - ssh-keygen

The procedure above generates 2 files:

```
~/.ssh/id_ed25519
~/.ssh/id_ed25519.pub    # public key
```

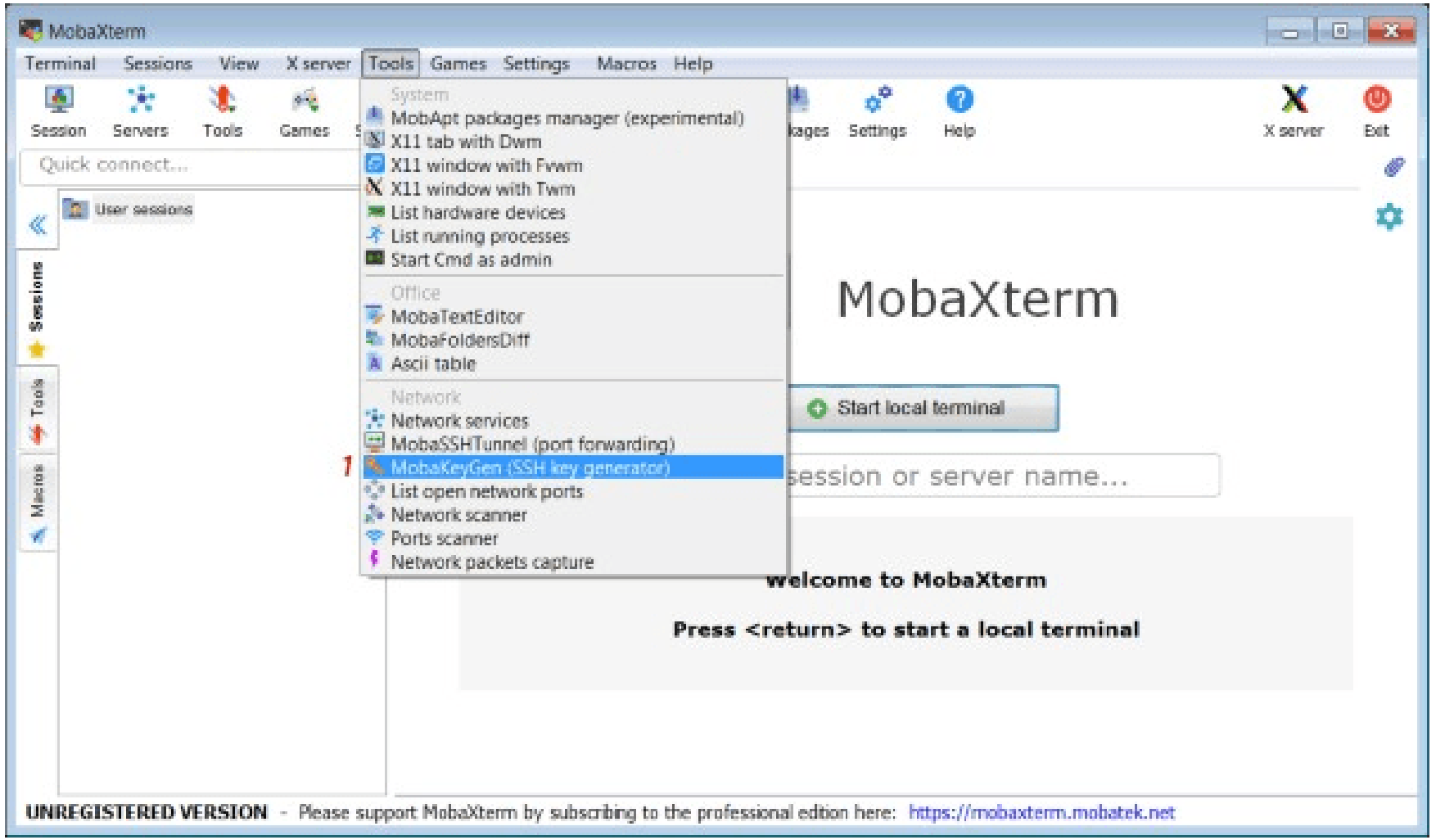
- **id_ed25519**: Private key, do NOT share it. This stays in your laptop
- **id_ed25519.pub**: Public key, copy to the computers you want to access.

Example

```
user001@myLaptop:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/user001/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user001/.ssh/id_ed25519
Your public key has been saved in /home/user001/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:pwYbiOWWkmVD+yawF1nwWYqO8A29FvxJstBYKefMYbk user001@myLaptop
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|                 |
|            .    |
|         . o     |
|  .     S . .    |
| o     . +   o   |
|. o . ..+ +.o.o. |
|=++E o.... ++=B.+|
|=@Boo++.    =*=@o|
+----[SHA256]-----+
```

MobaXterm - Key Generation

Surf's Wiki
click here



Upload Key to HPC

1. In your laptop: Copy the public key to your home folder in the HPC:

```
$ scp ~/.ssh/id_ed25519.pub [USER]@login.anunna.wur.nl:/home/WUR/[USER]/
```

2. Login to the HPC with your credentials (one last time)

3. In the HPC: Append (**>>**)the public key to *~/.ssh/authorized_keys*

```
$ cat id_ed25519.pub >> ~/.ssh/authorized_keys
```

1. In your laptop: Copy the public key to your home folder in the HPC:

```
ssh user001@login.anunna.wur.nl 'echo "$(cat ~/.ssh/id_ed25519.pub)" >> $HOME/test.file'
```

Alternatively

1. In your laptop: Copy the public key to your home folder in the HPC:

```
$ ssh user001@login.anunna.wur.nl  'echo "$(cat ~/.ssh/id_ed25519.pub)" \
 >> $HOME/.ssh/authorized_keys'
```

or

```
$ ssh-copy-id  -i ~/.ssh/id_ed25519.pub) user001@login.anunna.wur.nl
```

Multiple keys

If you have multiple keys you can give the key a custom name and a comment.
The following command creates **mykey** and **mykey.pub** at **$HOME/.ssh**

```
$ ssh-keygen -t ed25519 -f $HOME/.ssh/mykey -C "key for this and that"
```

With a custom key name you need to specify the key you want to use:

```
$ ssh-agent #start ssh-agent process
$ ssh-add $HOME/.ssh/mykey
$ Enter passphrase for /home/user/.ssh/mykey:
$ Identity added: /home/user/.ssh/mykey (key for this and that)
```

Additional Information

SSH Agents

Allows keys to remain active until you reboot

Additional Resources:

wiki.anunna.wur.nl

Surf's Wiki

WAGENINGEN
UNIVERSITY & RESEARCH

VSCode/Codium

To edit your code on the login node(s), install "Remote -SSH"

More info on usage in the Extensions tab in VSCode, but having SSH keys and agent set up makes life a lot easier

Please, use for editing purpose only.

and "Close Remote Connection" when done !

Break - 10 min

Dynamic Job Arrays

How to submit array of jobs in slurm

Embarrassingly Parrallel

Embarrassing Parallelism is obtained by launching the **same program** multiple times simultaneously

- `Every program does the same thing`
- `No inter-process communication`
- `Useful cases`
- `Multiple input/data files`
- `Random sampling`

Job Arrays - resource

Array with index values between **1 and 3**

--array=1-3

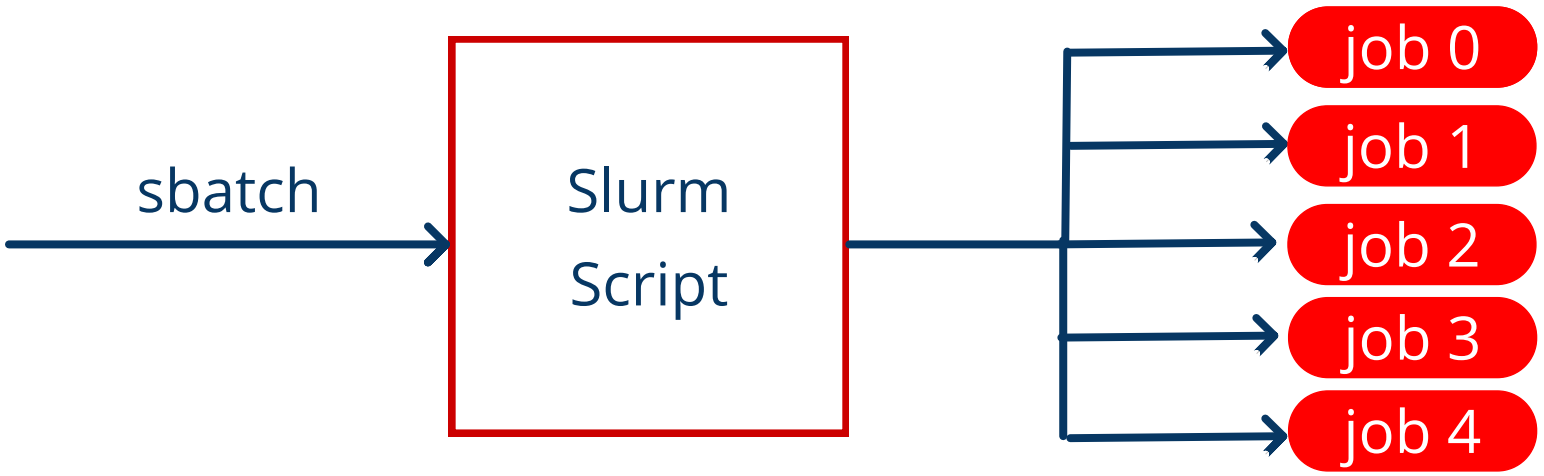
Array with index values of **1, 5, 10, 11, 12**

--array=1,5,10-12

Array with index values between **2 and 60, with a limited number (4) of simultaneous running jobs**

--array=2-60%4

The basic idea



SLURM Variables

Whenever you submit a job or start an interactive session, SLURM creates variables that are accesible from inside your job/script.

Allocate an interactive session and run the following command:

```
$ srun --ntasks=1 --cpus-per-task=3 --mem=3.5G --time=0-10   --nodes=1 --pty bash -i
$ env | grep SLURM_
```

Script Example

/lustre/shared/hpcCourses/Advanced/fruits.slurm

```
1  #!/bin/bash
2  # -----------------------------Name of the job-------------------------
3  #SBATCH --job-name=hpcAdvanced
4  #-----------------------------Output files-----------------------------
5  #SBATCH --output=logs/fruits.log
6  #SBATCH --open-mode=append
7  #-----------------------------Other information------------------------
8  #SBATCH --comment='01189998119991197253'
9  #-----------------------------Required resources-----------------------
10 #SBATCH --time=0-0:10:0
11 #SBATCH --ntasks=1
12 #SBATCH --cpus-per-task=1
13 #SBATCH --mem=100M
14 #-----------------------------Array
15 #SBATCH --array=0,1,2,3,4
16 #-----------------------------Environment and commands
17 cd $myScratch/HPCcourse
18 fruits=(banana apple orange grape berry tomato watermelon)
19 echo "${SLURM_JOB_ID} - $SLURM_ARRAY_TASK_ID: ${fruits[$SLURM_ARRAY_TASK_ID]}"
20 env | grep SLURM_
```

```
$ sbatch --chdir=$myScratch/hpcCourse ./fruits.slurm
```

Output Files

You can define the location of your output files.

Never point them to $HOME, put them in a log directory

```
1 #SBATCH --output=logs/output_%j.log
2 #SBATCH --error=logs/%j.err
```

If you do not care about logs, send them into the void.

Hard to troubleshoot

```
1 #SBATCH --output=/dev/null
2 #SBATCH --error=/dev/null
```

Exercise - Array Job

Copy the **weather.slurm** script from the previous exercise and save it as **weather_array.slurm** and adapt it to use arrays and run it.

- **SLURM option**: #SBATCH --array=240,260,270
- **SLURM variable**: $SLURM_ARRAY_TASK_ID

Use the variable **$SLURM_ARRAY_TASK_ID** as the argument for the script

```
python $myScratch/hpcCourse/weer_vanaf_2000.py $SLURM_ARRAY_TASK_ID
```

Solution

```
#!/bin/bash
# -----------------------------Name of the job-------------------------
#SBATCH --job-name=hpcAdvanced
#-----------------------------Output files-----------------------------
#SBATCH --output=logs/output_%j.log
#SBATCH --error=logs/%j.err
#-----------------------------Other information------------------------
#SBATCH --comment='01189998119991197253'
#-----------------------------Required resources-----------------------
#SBATCH --time=0-0:10:0
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2G
#SBATCH --array=240,260,270 # <--- This
#-----------------------------Environment, Operations and Job steps----
module reset
module load 2024
module load Python/3.12.3

cd $myScratch/hpcCourse
source $myScratch/hpcCourse/venv/bin/activate

python $myScratch/HPCcourse/weer_vanaf_2000.py $SLURM_ARRAY_TASK_ID
```

Takeways

Resources

- Each array job will allocate the same resources as its parent
  - If the parent allocates 1G and 1 core and the array has 100 elements, the entire series of jobs will allocate 100G and 100 cores

order

- The jobs will not be executed in order.

Array sizes are predefined

- Slurm won't be able to dynamically change the array inside the script.

Be sensible

Remember for HPC Basics

Template:

```
$ sbatch --<flags> </Path/To/Script.slurm>
```

Passing flags to sbatch will override the settings of the slurm script

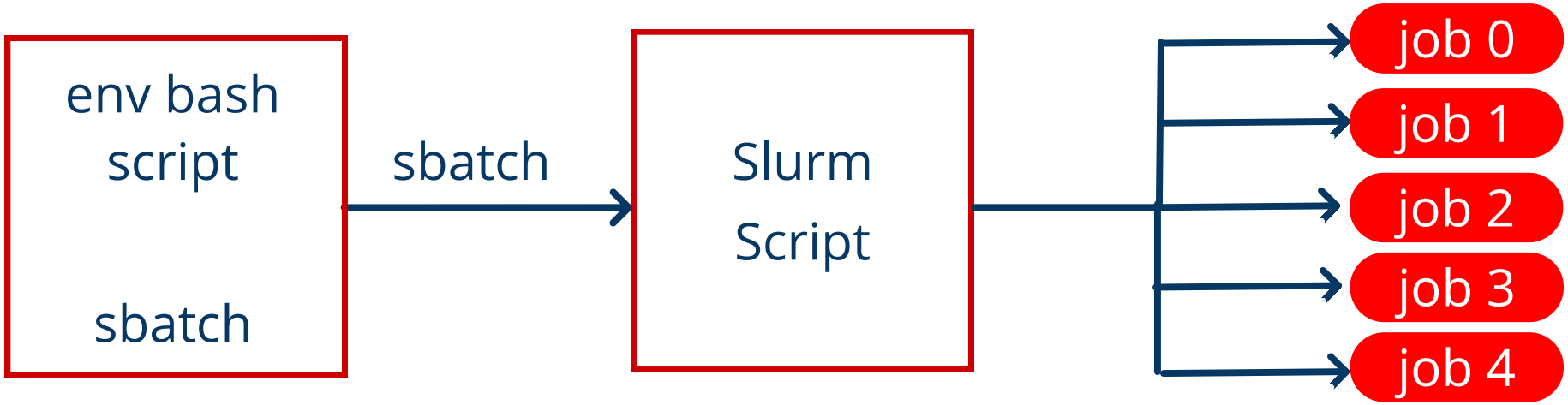
```
$ sbatch --mem=8G--cpus-per-task=6 script.slurm
```

The same can be done for arrays

```
$ sbatch --array=0-6%2 fruits.slurm
```

Dynamic Job Arrays

Arrays can be generated dynamically, outside of the slurm job script



What changed?

The array is defined directly at the sbatch command

```
sbatch --array=$MY_DYNAMIC_ARRAY
```

The array can be file with values

```
1 FILES=$(ls -1 $MY_DATA_LOCATION)
2 sbatch --array=$FILES process_files.slurm
```

The array can be file with indices

```
1 FILES=$(ls -1 $MY_DATA_LOCATION)
2 NUM_FILES=${#FILES[@]}
3 sbatch --array=0-$(($NUM_FILES-1)) mySlurmScript
```

Exercise - Dynamic Arrays Script

1. Make a folder inside your course directory labelled data

```
$ mkdir $myScratch/hpcCourse/data
```

2. Populate the newly created data folder with empty files lebelled 240, 260,270

```
$ touch $myScratch/hpcCourse/data/{240,260,270}
```

3. Create a bash script labelled **launcher.sh** and list the contents of the data folder and store in a variable labelled **stations**

```
stations=$(ls -1 $myScratch/hpcCourse/data | tr '\n' ',')
```

3.1 Add an sbatch command that uses the stations variable

```
sbatch --array=$stations $myScratch/hpcCourse/weather_array.slurm
sbatch --array=$stations $myScratch/HPCcourse/weather_array.slurm
```

4. Make you bash script executable and run it. Check your jobs with sacct

Solution

```
1 #!/bin/bash
2
3 stations=$(ls -1 $myScratch/hpcCourse/data | tr '\n' ',')
4 sbatch --array=$stations arrayStation.slurm
```

Parallelism

An overview of different job types

Job Types

- Serial
- Multi-threaded
- Multi-process
- GPU

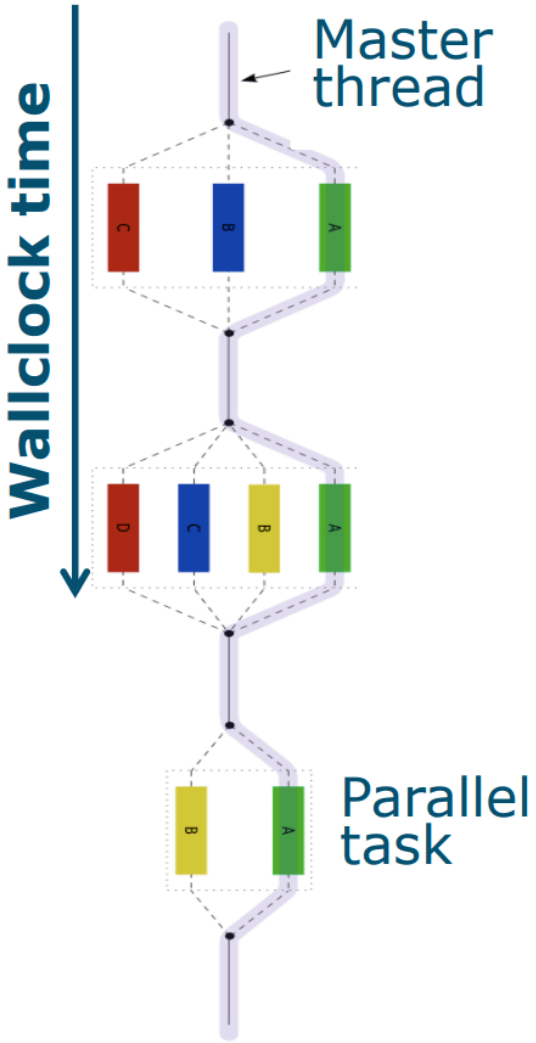
Serial

- Simplest type of job
- Runs on a single process (and single thread)
- Job runs through program sequentially

Example

```
##---------- required resources ----------
#SBATCH --time=0-0:10
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=4096
#SBATCH --cpus-per-task=1
##---------- Environment, Operations and job step ----------

module purge

module load myModule

myprog  # srun myprg
```

Multi-threaded



- Traditionally uses OpenMP or TBB
- Runs on a single node
- Programs spawns threads on the node
- Memory is shared between the threads

example

```
##---------- required resources ----------
#SBATCH --time=0-0:10
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=4000
#SBATCH --cpus-per-task=3
##---------- Environment, Operations and job step ----------

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

module purge
module load myModule

myprog # srun myprog
```

Note: Total memory used is 3 x 4 GB = 12 GB

Multi-Process

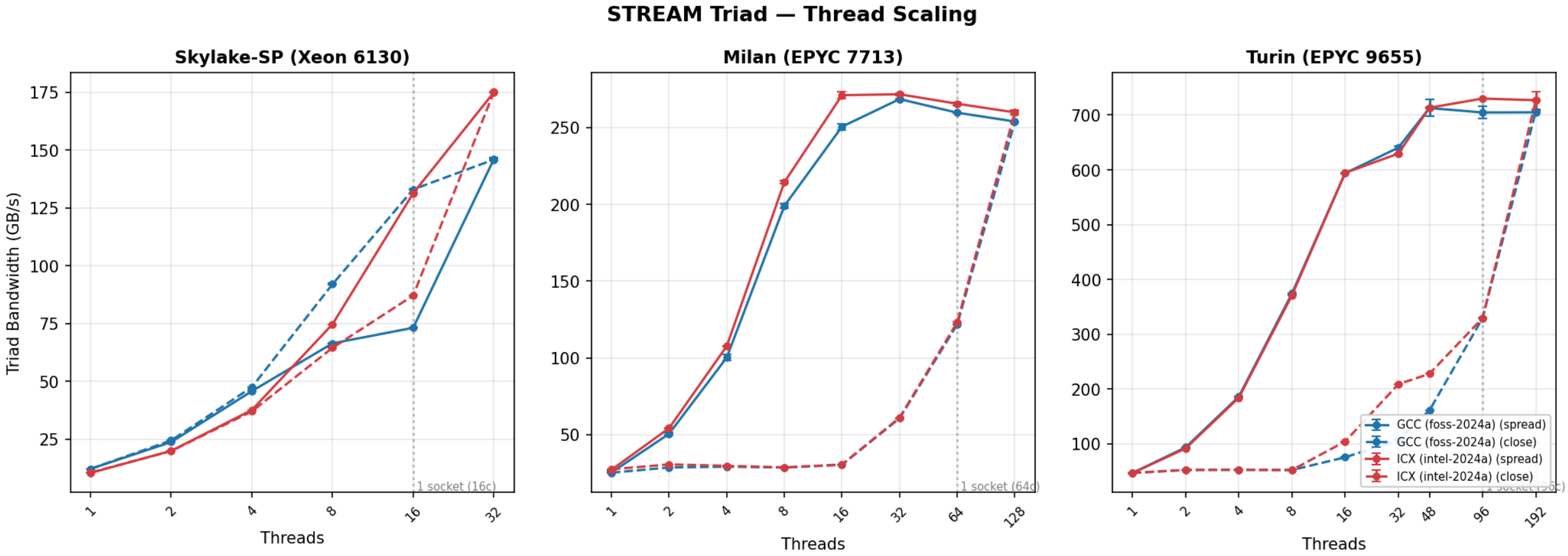
- Enabled by Message Passing
  - MPI
- One program (rank 0) spawns several copies across the computational nodes
  - These processes are labelled and organized by ranks
- Processes communicate with each other across the network ("passing messages")

WAGENINGEN
UNIVERSITY & RESEARCH

Example

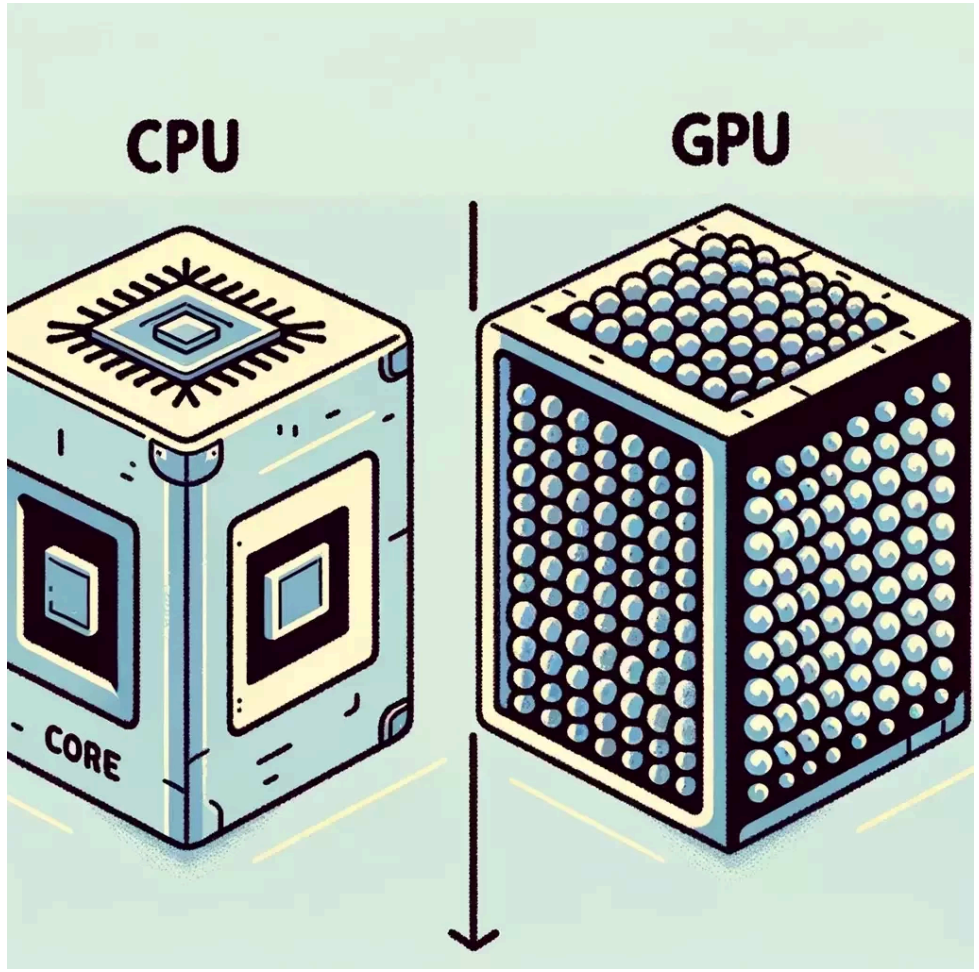
```
##---------- required resources ----------
#SBATCH --time=0-0:10
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --mem=4096
#SBATCH --cpus-per-task=1
##---------- Environment, Operations and job step ----------

module reset
module load 2024 OpenMPI

mpirun -np ${SLURM_NTASKS} myprog
```

Pinning for the Greater Good



WAGENINGEN
UNIVERSITY & RESEARCH

GPU



- Embarrassingly parallel on steroids
- Millions of simultaneous threads
- Relatively simple operations

Example

```
##---------- required resources ----------
#SBATCH --time=0-0:10
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=4096
##SBATCH --cpus-per-task=1
#SBATCH --cpus-per-gpu=4
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
##---------- Environment, Operations and job step ----------

module purge
module load myModule

myprog # srun myprog
```

GPU (and other parallel) Pitfalls

- Jobs in the gpu partition without the --gres=gpu:1 flag will not allocate a gpu
- If your job uses more than one GPU, make sure you also allocate node=1. GPUs a popular, and if you do not specify, slurm will select GPUs in different nodes and that will be SLOW
- Be mindful, if a node has 4 GPUs and 500GB RAM and 32 cores, try to limit your job to 125G and 8 cores per GPU, unless you **REALLY** need all the resources.
- If your gpu jobs need a lot of RAM and cores, maybe use node301 (192 cores and 2.5TB)
  - check the details of a specific gpu node with:

```
$ sinfo -p gpu,gpu_amd -o "%N %f"
```

GPU Types

- **NVIDIA V100 (16 GB HBM2)** – Strong general-purpose training GPU for FP32/FP16 workloads, well-suited for classic deep learning training and HPC compute.
- **NVIDIA A6000 (48 GB GDDR6)**– High end workstation GPU optimized for for high-throughput inference and small- to mid-scale training, rendering and AI workloads.
- **NVIDIA A100 80 GB (HBM2e)** – Top-tier large-scale AI GPU with excellent mixed-precision performance, ideal for massive model training and multi-GPU scaling.
- **NVIDIA L40S (48 GB GDDR6)** – Optimized for high-throughput inference and small- to mid-scale training, great performance-per-watt for production AI workloads.
- **AMD MI210 (64 GB HBM2e)** – Strong HPC-leaning training GPU with high FP64/FP32 and solid BF16/FP16 training performance, best when ROCm fits the software stack.

Exercise - Compute π

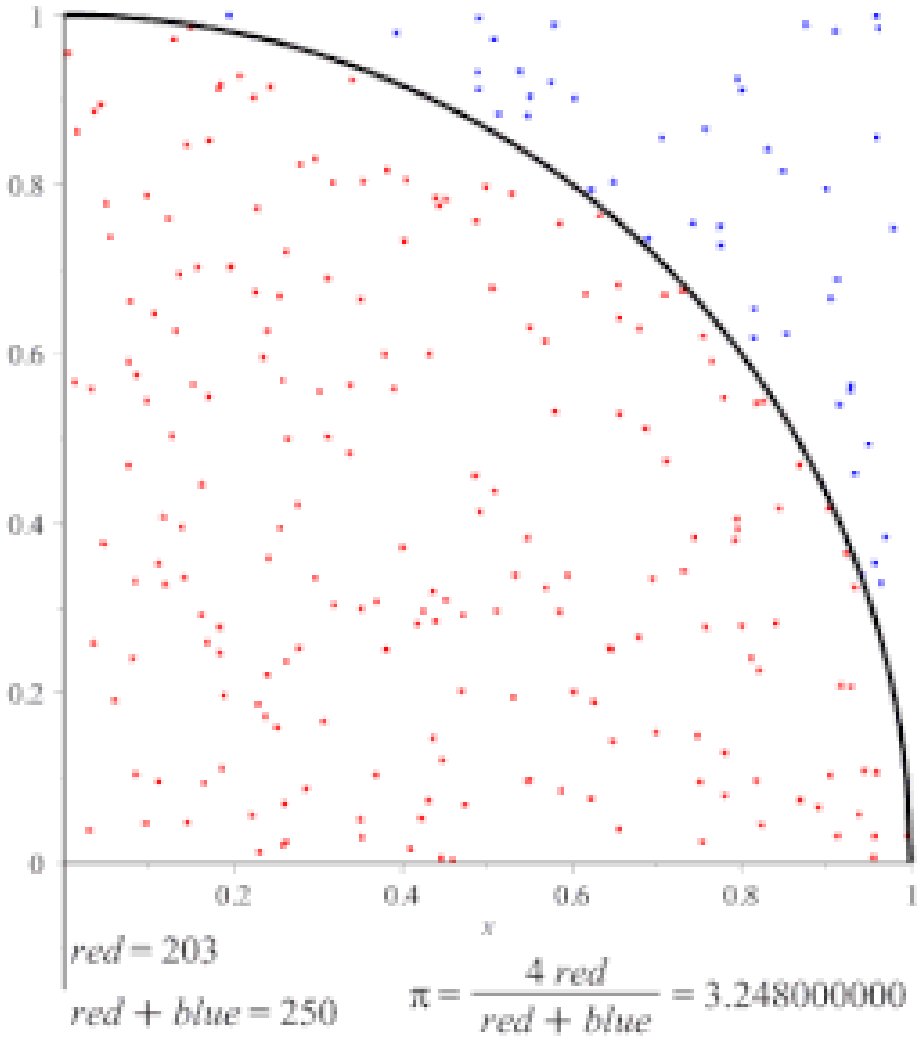
In this exercise, you will compute π with different job types and assess its performance

Exercise - Compute π - Method

Algorithm:

1. Generate a pair of random numbers between 0 and 1 and assign to x and y
2. Each number generated increases the total number of samples by 1
3. Each number generated that lies within the unit circle is registered in the variable N_in

$$\pi \approx 4 \frac{N_{in}}{N_{total}}$$



WAGENINGEN
UNIVERSITY & RESEARCH

Problem Statement

How to speed up this computation?

- Embarassingly parallel - each point is independent
- CPU bound - memory bandwidth is not the bottleneck

Experiment setup

- serial - 1 cpu
- multithread - 1, 2, 4, 8, ...,128 threads
- multiprocess - 1, 2, 4, 8, ..., 128, processes
- GPU - AMD and/or nvidia
- Hybrid - multithread + multiprocess

Exercise - Compute π - 1

Copy the exercise files

```
$ cp -r /lustre/shared/hpcCourses/pi-tutorial/exercise  $myScratch/hpcCourse/pi-tutorial
```

You should have:

```
$myScratch/hpcCourse/pi-tutorial
├── launch.sh
└── slurm
    ├── run_gpu_amd.slurm
    ├── run_gpu_nvidia.slurm
    ├── run_hybrid.slurm
    ├── run_multinode.slurm
    ├── run_multiprocess.slurm
    ├── run_multithread.slurm
    └── run_serial.slurm
```

Exercise - Compute π - 2

Find the files that need to be fixed

```
$ cd $myscratch/hpcCourse/pi-tutorial
$ grep -ri FIXME .
```

Substitute FIXME with the appropriate SLURM variables.

HINT: Look at the examples at the top of this section

Run launch.sh

```
$ launch.sh && watch squeue --me
```

Check the files in the results folder

```
$ cat  results/*.out
```

Are there any errors? If not proceed to the analyzes section

Exercise - Compute π - 3

Copy analyze.py

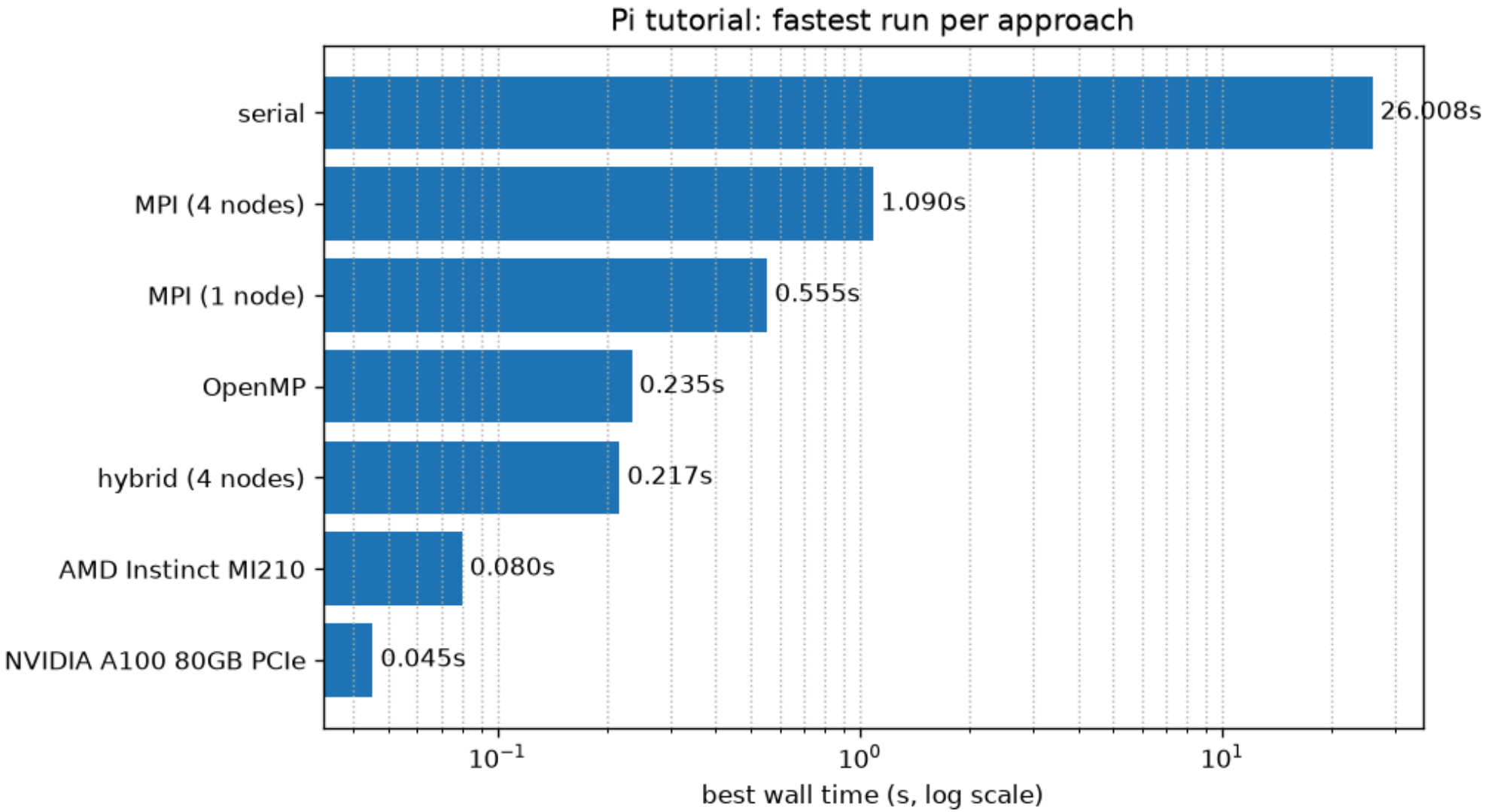
```
$ cp -r /lustre/shared/hpcCourses/pi-tutorial/analyze.py  .
```

Check the instructions inside the file.

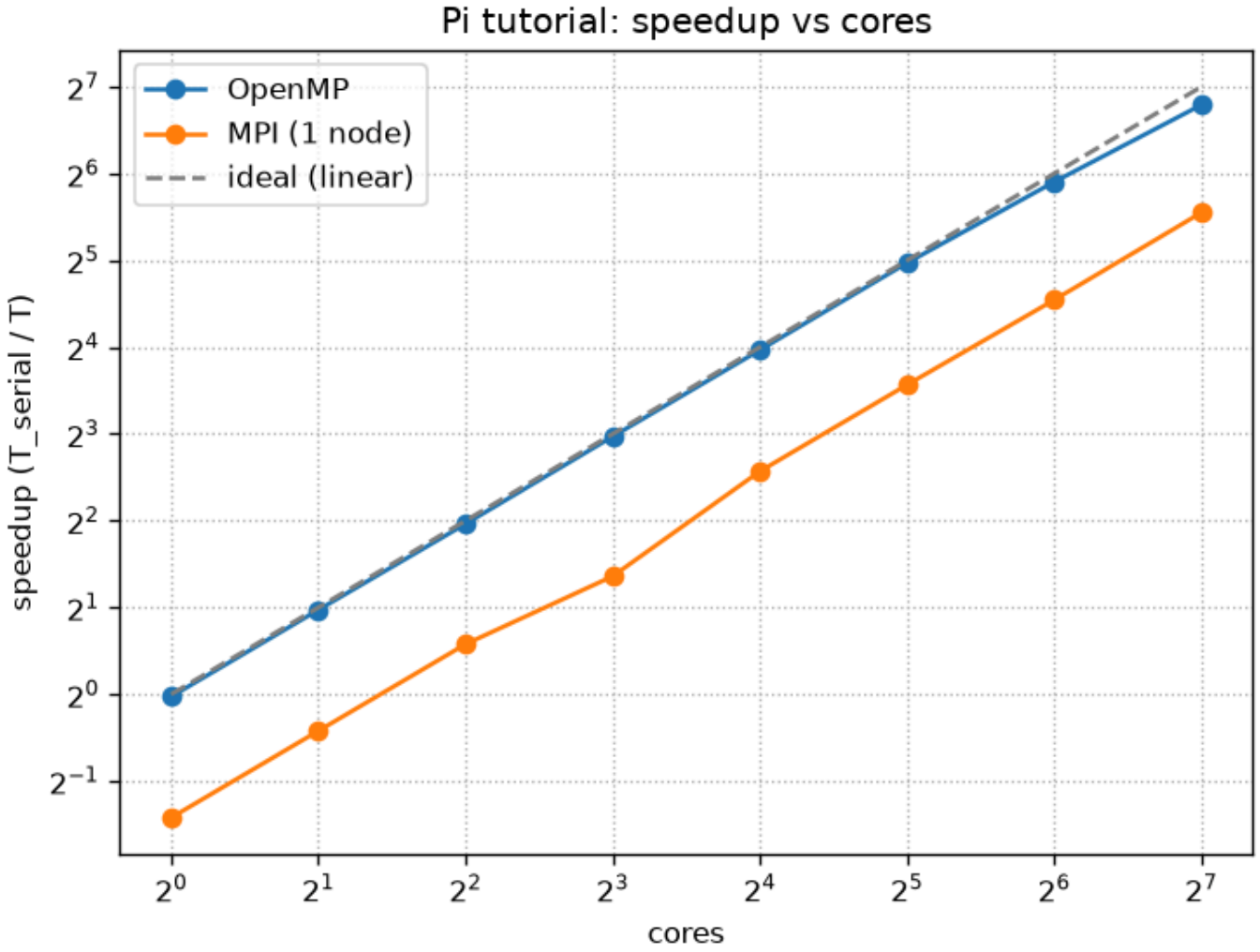
1. Depends on both pandas and matplotlib
2. Either create a venv or user modules. (You can also use the venv from a previous exercise)
3. create a plots subdirectory
4. Point the input of the script to your results folder and the output to the plots folder

```
$ python analyze.py ./results ./results/plots
```

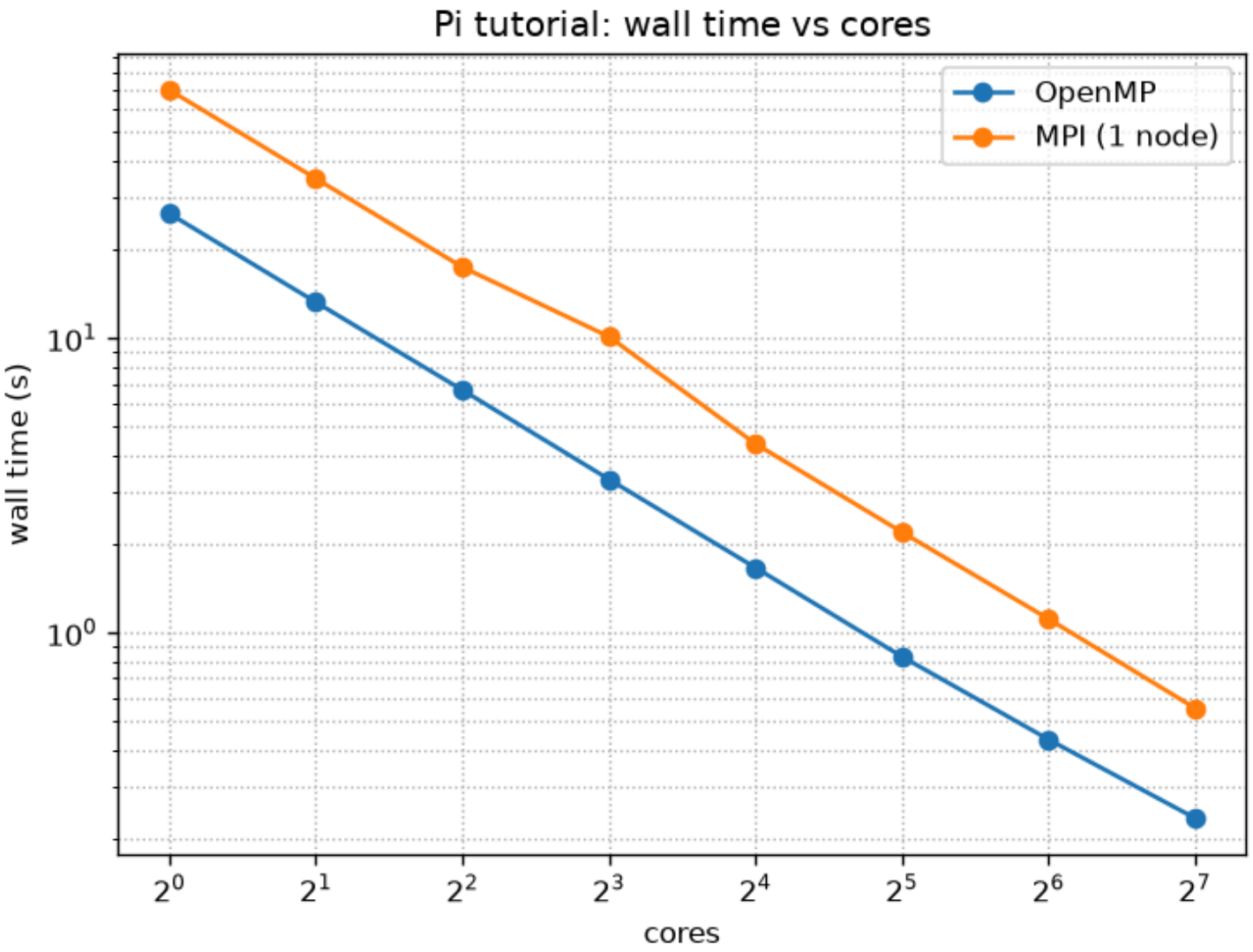
Results - Compute π - All



Results - Compute π - Speedup



Results - Compute π - WallTime



WAGENINGEN
UNIVERSITY & RESEARCH

Break - 10 min

Storage

Overview of different storage types available at the HPC

On the Node Storage

"I/O Performance"

RAM disk - /dev/shm

- Memory of the node, **It will count towards your slurm allocation.**
- Very fast, suitable for read/write intensive jobs
- Volatile, gets erased when job ends (or even before...)
- **Hint:** Make a folder **/dev/shm/$USER** and set permissions to 1755 (or 1777)
- Data needs to be copied to a permanent location

Local SSD - /tmp

- SSD on the node
- Volatile, gets erased when the node reboots
- Data needs to be copied to a permanent location

```
$ myDir=$(mktemp -d)
```

WAGENINGEN
UNIVERSITY & RESEARCH

Network Storage

Lustre - distributed filesystem

- scratch, backup, nobackup
- Built to be performant and to handle computations
- Low latency connection (OmniPath)

NFS

- /home & /shared
- Ethernet connection
- Mind the location of your logs

archive

- /archive
- Outside cluster

"I/O Performance"

WAGENINGEN
UNIVERSITY & RESEARCH

Sequential Data Bandwidth

```
================ Rung 1: Sequential Bandwidth ================
tier         write GB/s     read GB/s
-------------------------------------
shm                9.65          7.81
nvme               1.53          6.22
lustre             2.25          5.95
home               0.11          0.11
-------------------------------------
GB/s = 1e9 bytes/s. shm is buffered (RAM); others are O_DIRECT.
```

Archiving - Tapes

/lustre/

Hot storage for data in active use or short term storage. Datasets you run jobs on, or will run a job on next week? These data belong on /lustre/

€ 100-150 / TB / year

/archive/

Data you might need again soon? These data belong on /archive/. A fast storage solution with limited space. Only available on the HPC logins

€ 65 / TB / year

Tape via iRODS

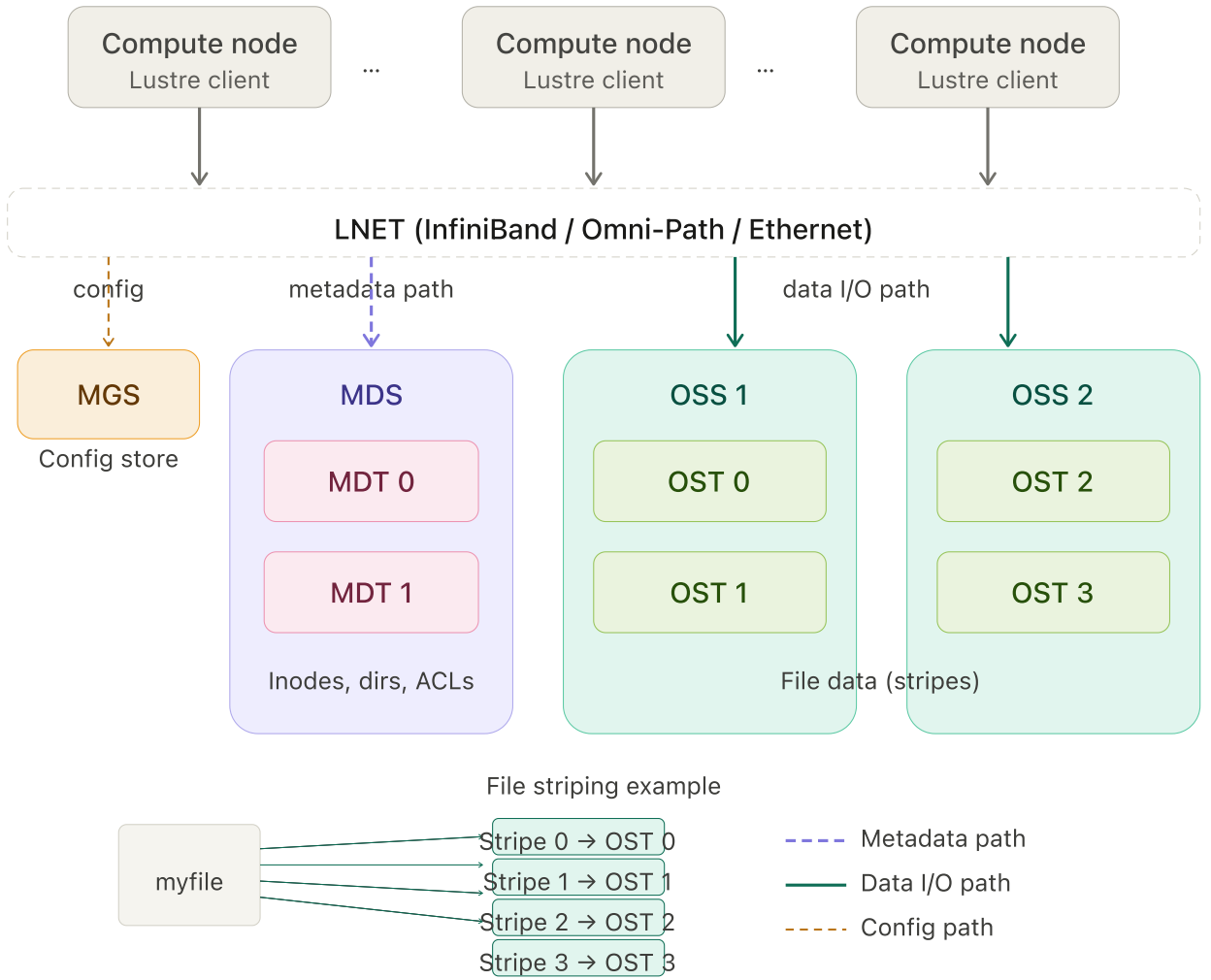
Finished your analyses, but the data needs to be stored and maybe accessed in the future? These data belong on tape. Virtually unlimited space!

€ 20 / TB / year

WUR iRODS docs

WAGENINGEN
UNIVERSITY & RESEARCH

We need to talk about Lustre



WAGENINGEN
UNIVERSITY & RESEARCH

Striping and Replicas

You can distribute your files accross OSTs

```
$ lfs getstripe .
$ lfs df -h
$ lfs setstripe -c -5 .
$ lfs mirror create -N3 -c 6 /lustre/scratch/my_project/important_data.dat
```

Apptainer Containers

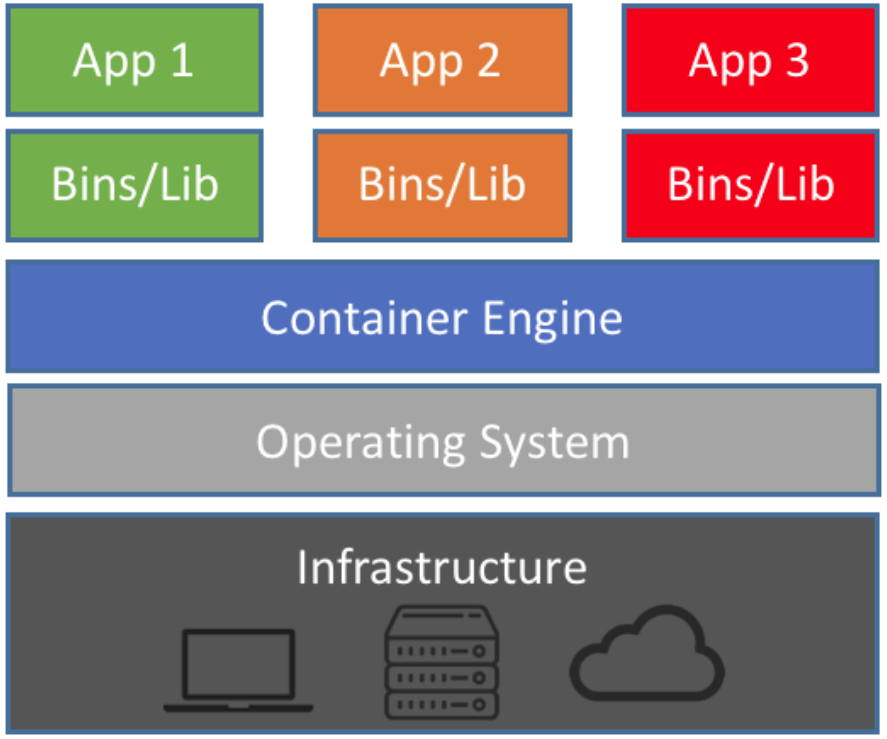
How to launch jobs with apptainer

WAGENINGEN
UNIVERSITY & RESEARCH

Containers

A **container** is an entity providing an isolated software environment (or filesystem) for an application and its dependencies.

Containers share the resources and kernel of the host machine.



Containers Examples

Docker:

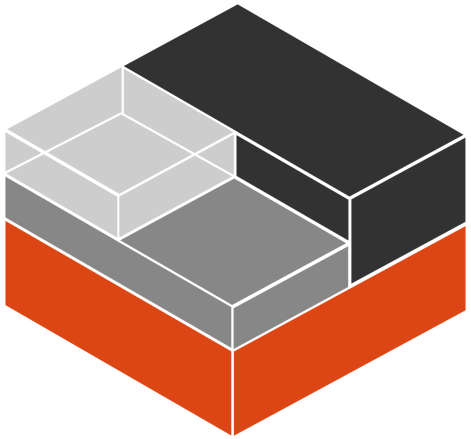
The first engine to gain popularity, still widely used in the IT industry. Not very suitable for HPC as it requires *root* privileges.

Apptainer:

Under free and open source software (FOSS) guarantee.

A simple, powerful *root*-less container engine for the HPC world.

Container Technologies



Workflow

- Create or download image
  - dockerhub
  - shub
- Create a recipe: definition files (.def)
- Execute image
  - **exec** - executes a command in the image environment
  - **shell -** opens interactive shell
  - **run** - executes a script built into an image, usually an application
- Publish/Distribute

Example - Checking OS Version - 1/2

Use apptainer to download the another version of ubuntu

1. Open a **sinteractive** session with 4 GB of memory
2. Load **Apptainer** under the 2023 modules
3. Go to your scratch folder
4. Pull an **ubuntu 24.04** image from dockerhub and save it under **ubuntu-2404.sif**
5. Check the version of ubuntu running on the node
6. Run a shell session on the **ubuntu-2404.sif** image.
7. Check the version of ubuntu running on the container.

Example - Hello World

Please **do not** run apptainer directly on login nodes

```
$ sinteractive --mem 1G
$ module load 2024
$ module load Apptainer/1.4.0
$ apptainer pull ./hello-world.sif shub://vsoch/hello-world
$ apptainer run hello-world.sif
```

Example - Checking OS Version - 2/2

```
$ sinteractive --mem 4G
$ module load 2024
$ module load Apptainer/1.4.0
$ cat /etc/lsb-release
$ cd /lustre/scratch/WUR/user001
$ apptainer pull ubuntu-2404.sif docker://ubuntu:24.04
$ apptainer shell ubuntu-2404
Apptainer> cat /etc/lsb_release # or cat /etc/os-release
Apptainer> exit
```

Exercise 1/3 - Play in the Sandbox

Please **do not** run apptainer directly on login nodes

Get an interactive session with 1 core and 10GB of RAM for 1 hour

```
$ sinteractive -c 1 --mem=10G --time=0-1
```

Load the apptainer module

```
$ ml utilities Apptainer/1.4.5
```

Create a temporaty directory and pull the image in sandbox mode

```
$ myTemp=$(mktemp -d)
$ apptainer pull --sandbox  $myTemp/ubuntu2404 docker://ubuntu:24.04
```

Access the image in a writable shell as (fake) root

```
$ apptainer shell -C  --writable --fakeroot  $myTemp/ubuntu2404
```

Exercise 2/3 - Play in the Sandbox

Please **do not** run apptainer directly on login nodes

Update and install the fortune-mod, lolcat, cowsay and nano

```
$ apt update && apt upgrade -y
$ apt install lolcat fortune-mod cowsay nano # or vim
```

Edit /.singularity/env/90-environment and append /usr/games to PATH

```
$ nano /.singularity.d/env/90-environment.sh
export PATH=/usr/games:${PATH}
```

Exit your container and run fortune | cowsay| lolcat with exec

```
$ exit
$ apptainer exec -C $myTemp/ubuntu2404 fortune | cowsay | lolcat
```

Exercise 3/3 - Play in the Sandbox

Please **do not** run apptainer directly on login nodes

Make your image portable by converting into a sif

```
$ apptainer build lolcat.sif $myTemp/ubuntu2404
```

You can go back and forth between sifs and sandbox images

More Tutorials

Reproducible computational environments using containers: Introduction to Apptainer

Bring Your Own Case

Discuss your use case and/or problem with the group

Closing Remarks

Support, wiki, community and whatever else

Communication and Support

Message of the day

- Announcements often posted

Annuna Users at Teams

- Feel free to post there if you have any questions or problems

Tickets

- For more involved support and questions, create a ticket.
  - https://support.wur.nl/
  - New Requests
  - Create a ticket
  - Type HPC at the field "*What is your question about?*"

WAGENINGEN
UNIVERSITY & RESEARCH

Wiki - wiki.anunna.wur.nl

- Slides from courses
- Instructions on how to run jobs
- Tips about python and R
- ...

Feedback Form

WAGENINGEN
UNIVERSITY & RESEARCH

So long, and thanks for all the fish!

WAGENINGEN
UNIVERSITY & RESEARCH