High Performance Computing - Basic Course

09/06/2026

WAGENINGEN
UNIVERSITY & RESEARCH

Instructors

- **Jeremie Vandenplas**
- **Leonardo Honfi Camilo**
- **Fons Prinsen**
- **Jan van Haarst**

After this course, you should

- Understand basic concepts of High Performance Computing.
- Be aware of the Anunna supercomputer and its features.
- Understand user environments and modules
- Be able to create, deploy and monitor slurm jobs.
- Be able to interact with other anunna users and seek out support

Notation

bash shell commands start with $

```
$ echo "Hello"
```

Numbered lines denote a script

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

Feel free to try it yourself or follow along

Ice Breaker

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

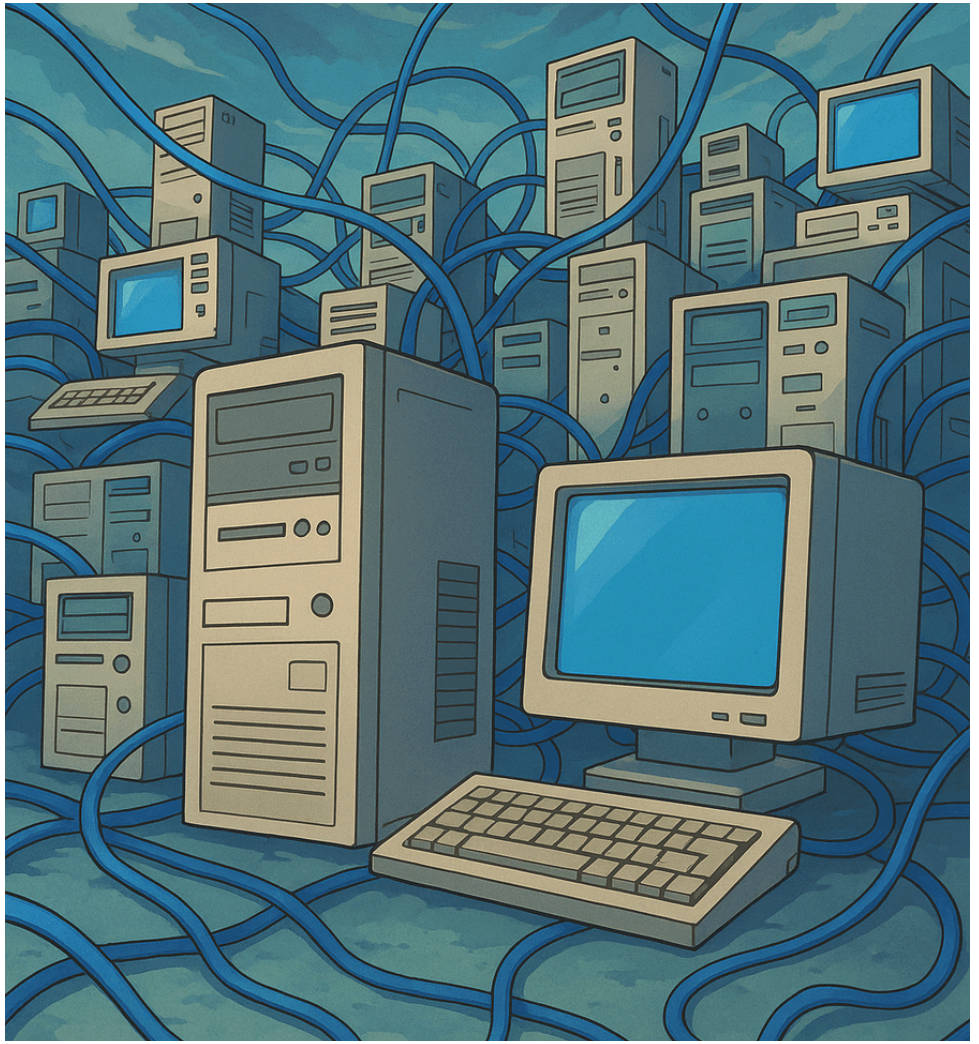
WAGENINGEN
UNIVERSITY & RESEARCH

Outline

- Ice Breaker
- Why Use an HPC
- High Performance Computing
- Connecting to the HPC
- Anunna
- WebApps
- Transferring Files
- Break
- Shell Environments
- LMOD - Environment Modules
- SLURM Jobs
- Jupyter at Anunna
- Monitoring jobs
- Closing remarks

WAGENINGEN
UNIVERSITY & RESEARCH

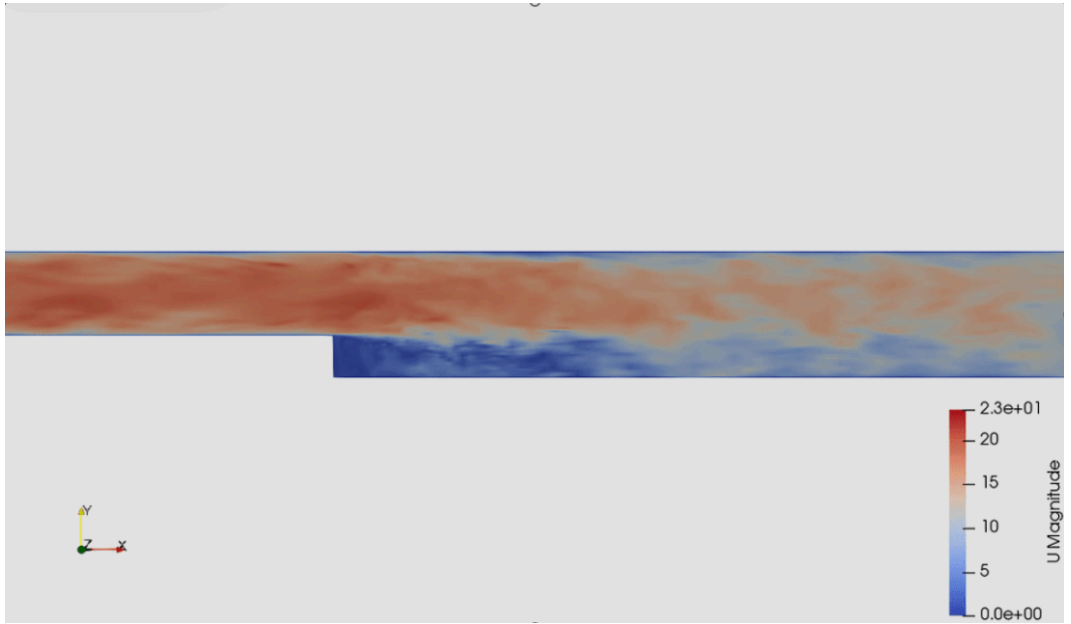
why should I use an HPC



WAGENINGEN
UNIVERSITY & RESEARCH

My first laptop

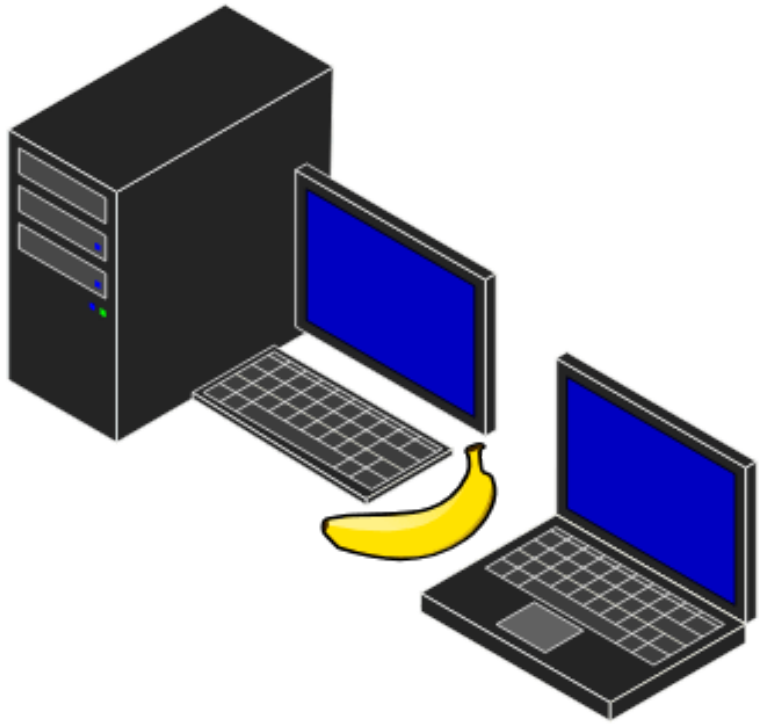
- AMD Turion 64 X2 Dual-Core
- 2 GB of RAM, 160GB of storage
- NVIDIA GeForce Go 6150
- Ubuntu 7.10 - Gutsy Gibbon
- Matlab, LaTeX, OpenFOAM



WAGENINGEN
UNIVERSITY & RESEARCH

Your Laptop/Workstation

- Flexible, private and convenient
- Good for prototyping and small tasks
- Portable (somewhat)



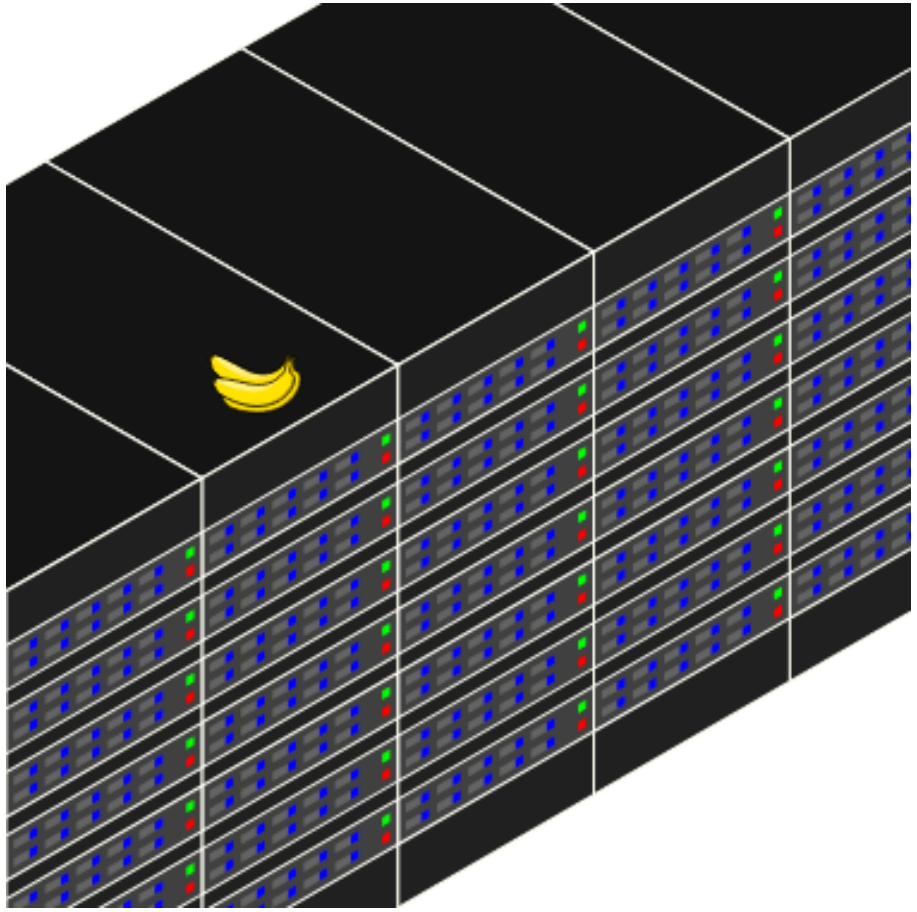
- Limited Resources
- Heat and noise
- Device is "locked" while task is running.
- Takes up space

WAGENINGEN
UNIVERSITY & RESEARCH

Supercomputers

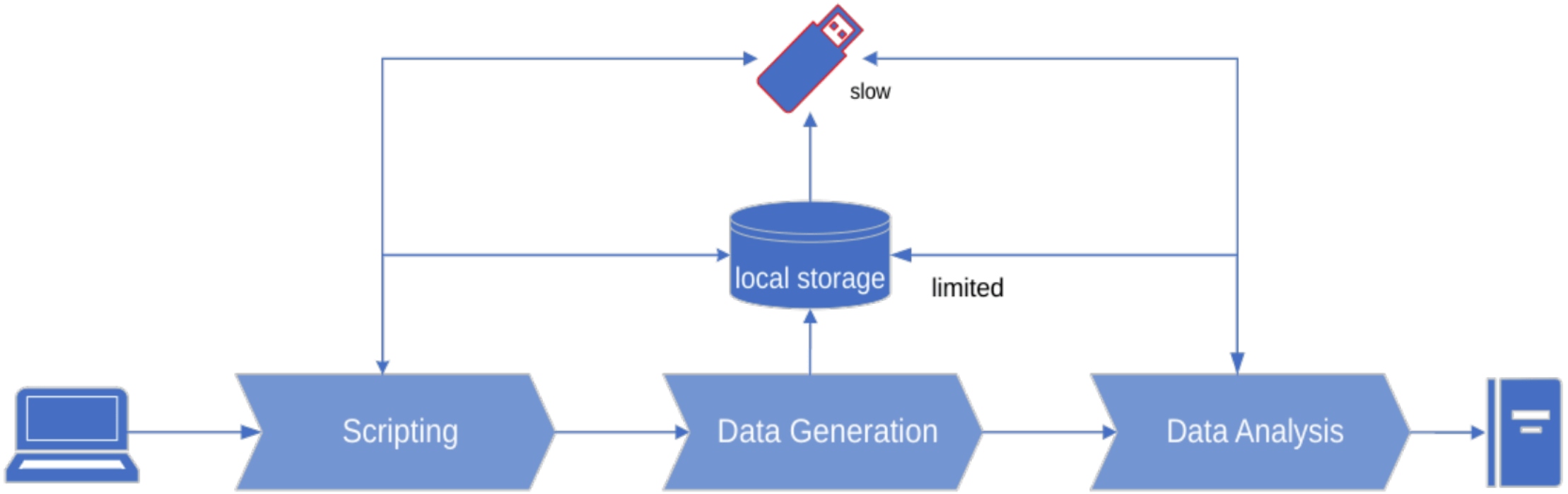
An aggregate, or cluster, of computers interconnected in a high speed low-latency network controlled by a resource manager and shared by multiple users.

- Extensible resources
- Housed in a datacentre
- Efficient
- Scalable workflows



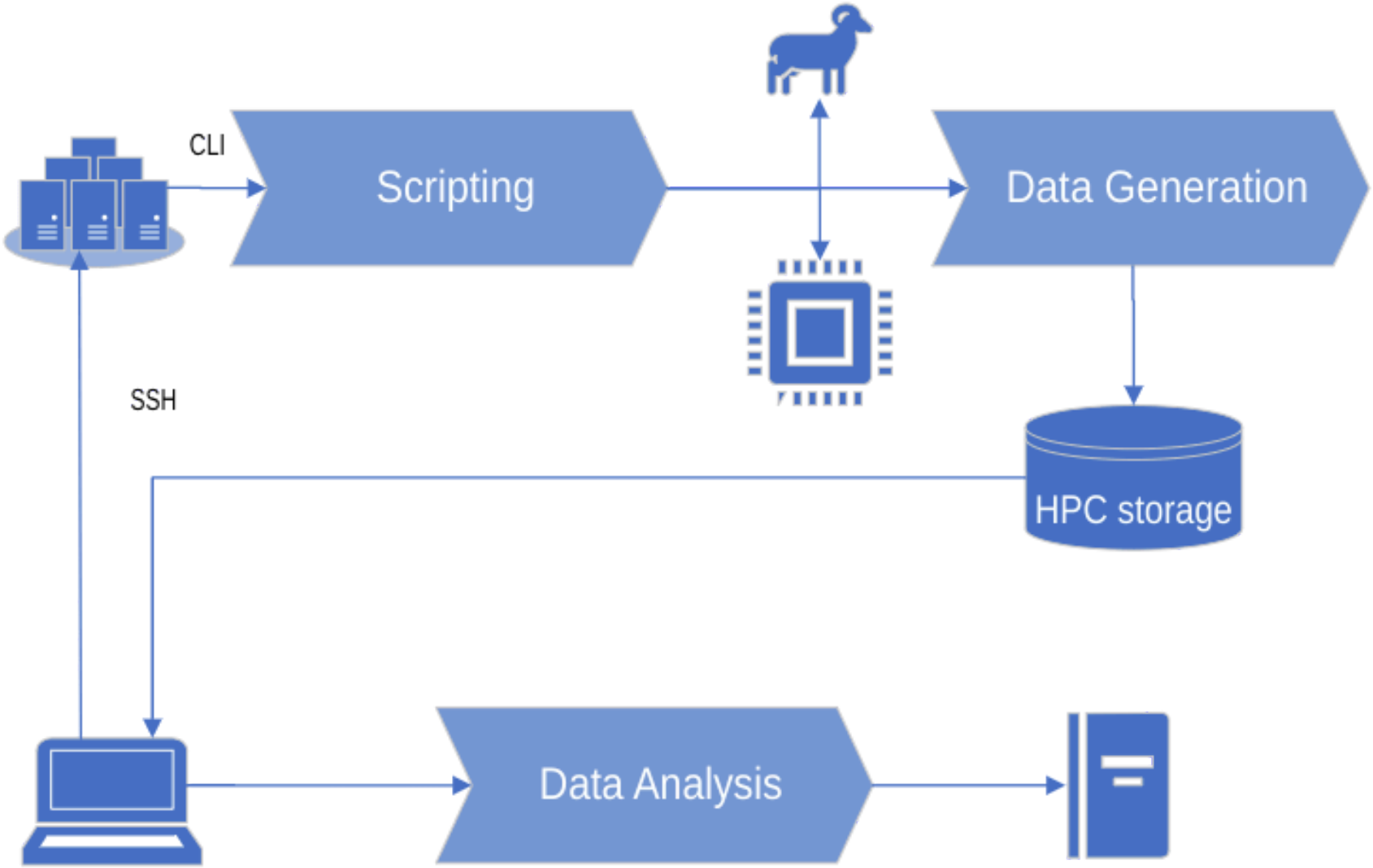
WAGENINGEN
UNIVERSITY & RESEARCH

Typical Workstation workflows



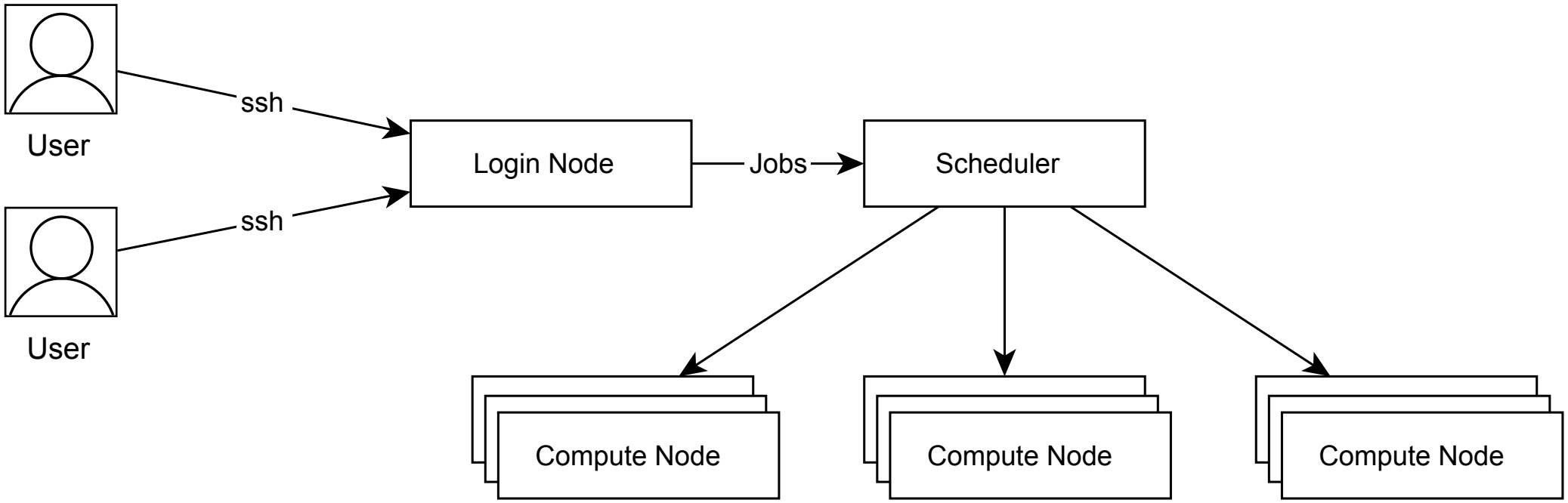
WAGENINGEN
UNIVERSITY & RESEARCH

Typical HPC workflows



WAGENINGEN
UNIVERSITY & RESEARCH

Concept Diagram



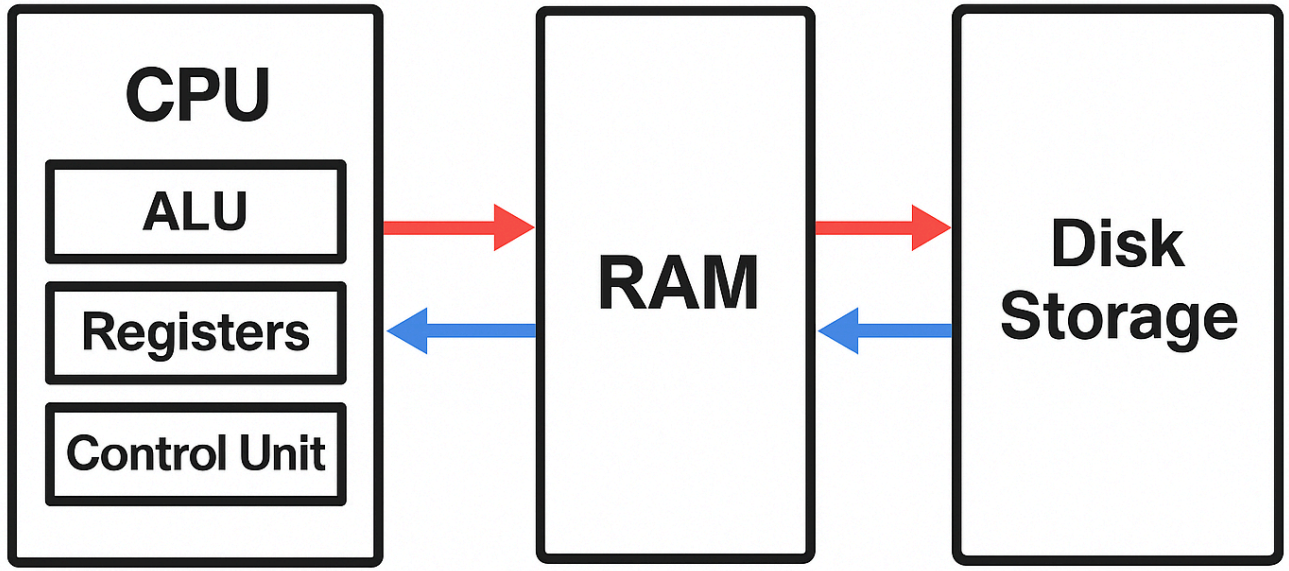
High Performance Computing

A Brief introduction into High performance computing concepts

Definition

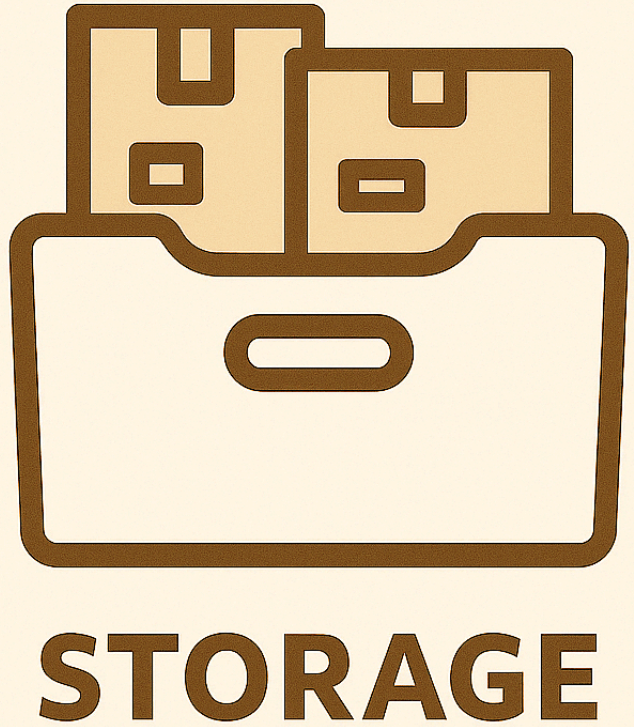
High Performance Computing (HPC) is the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation, in order to solve large problems in science, engineering, or business.

How Modern Computers Work



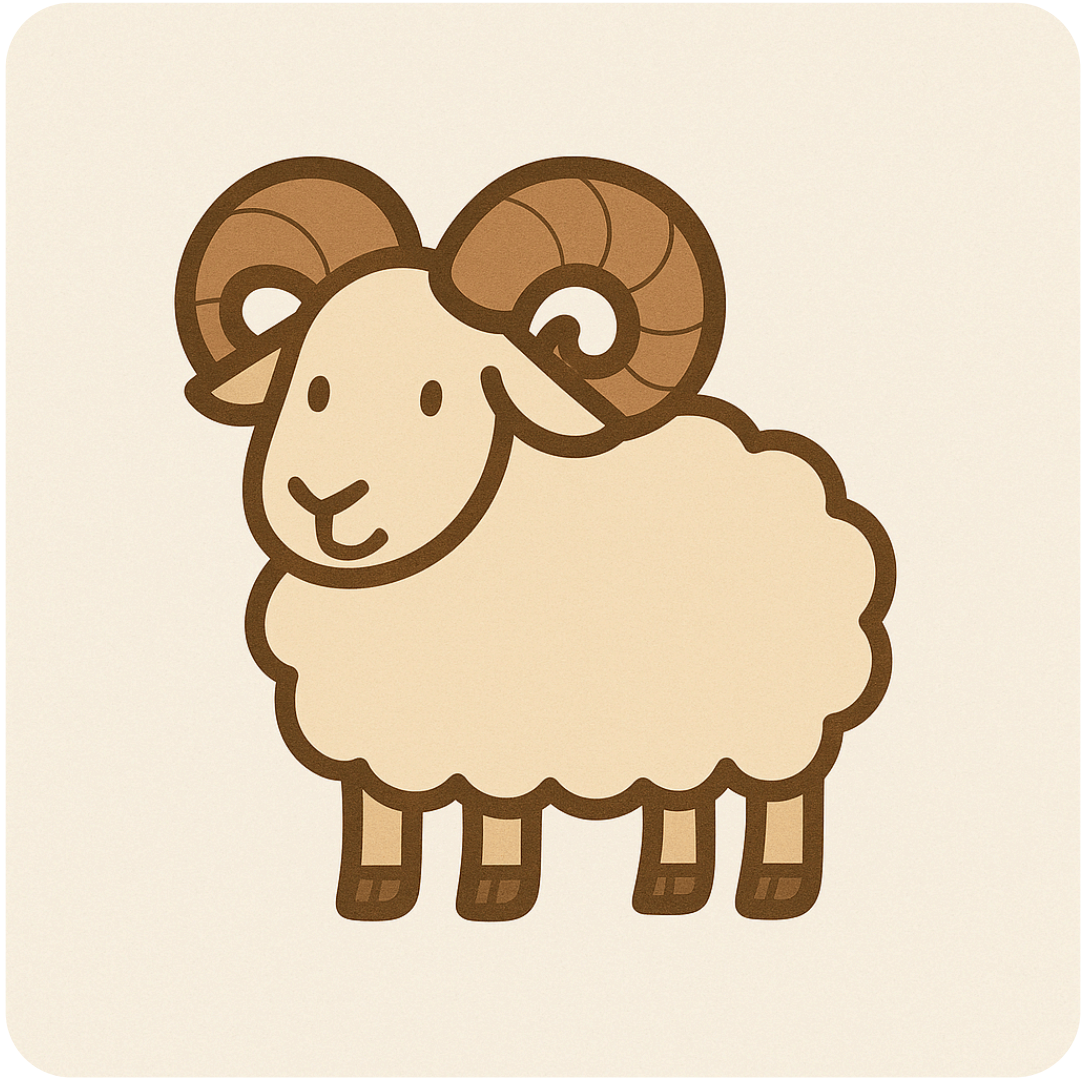
Persistent (Disk) Storage

- 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)

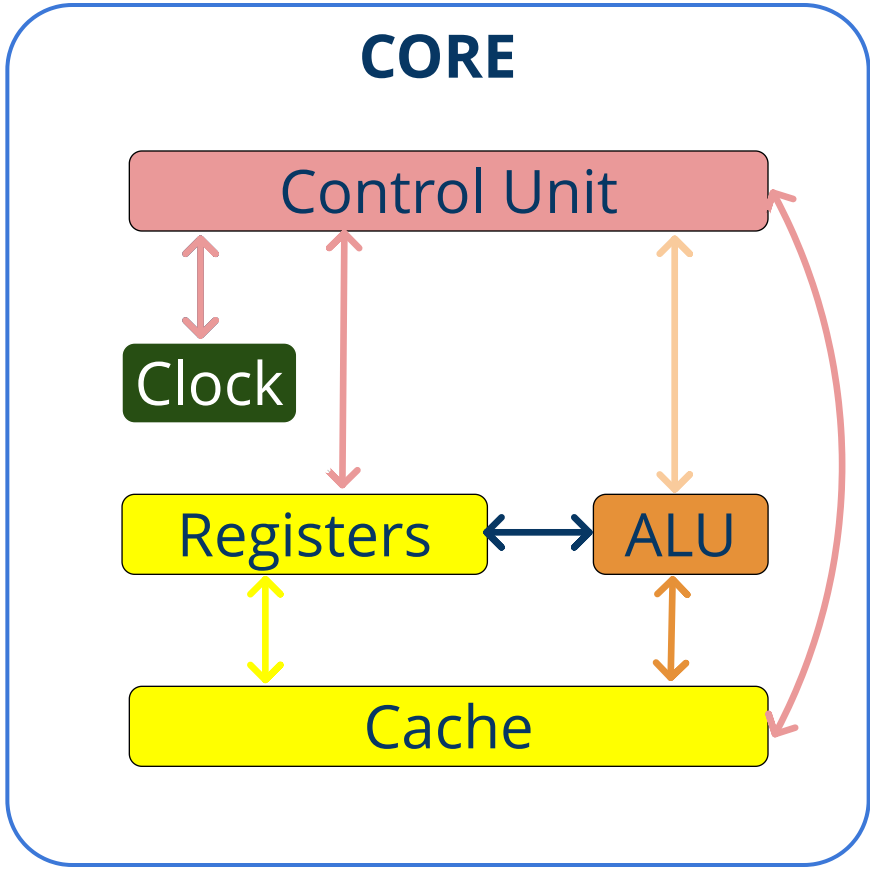
- Fast
- Used by processors
- Where active processes live
- Volatile
  - Cleared Often
  - Cleared after reboot



WAGENINGEN
UNIVERSITY & RESEARCH

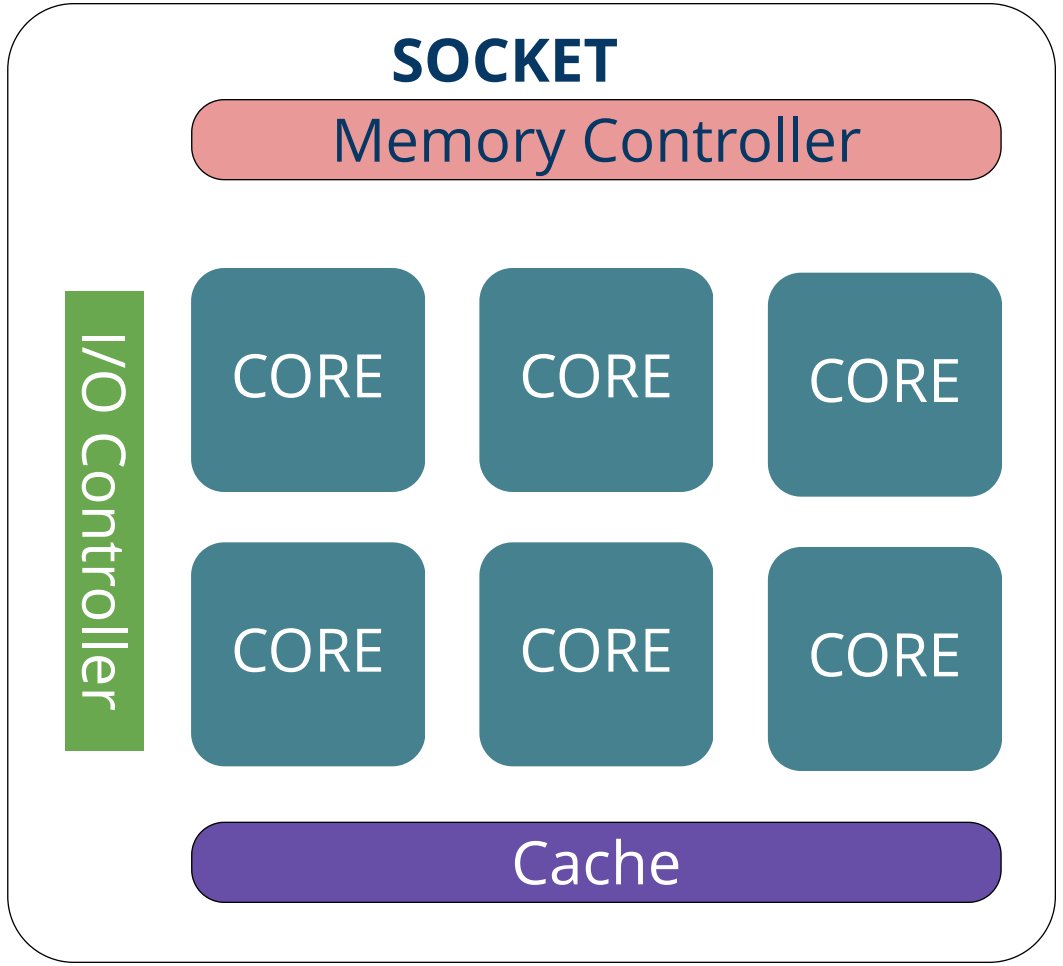
Cores

- Smallest computational unit
- Interchangeable* with cpu in HPC jargon
- Where (most of) the magic happens
- Dedicated memory (registers, cache)
- Registers and cache are small but very fast



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.

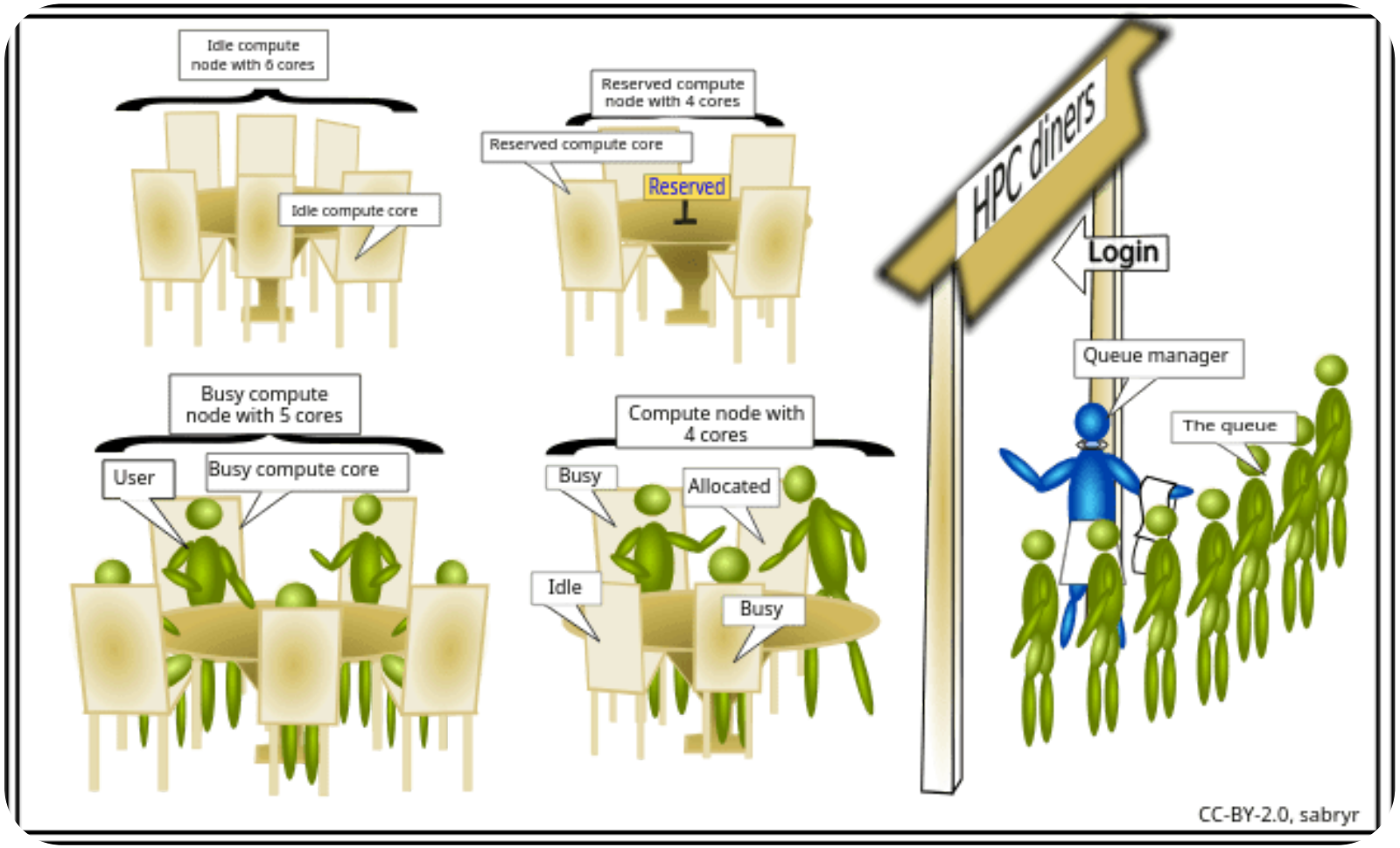
Resources:

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

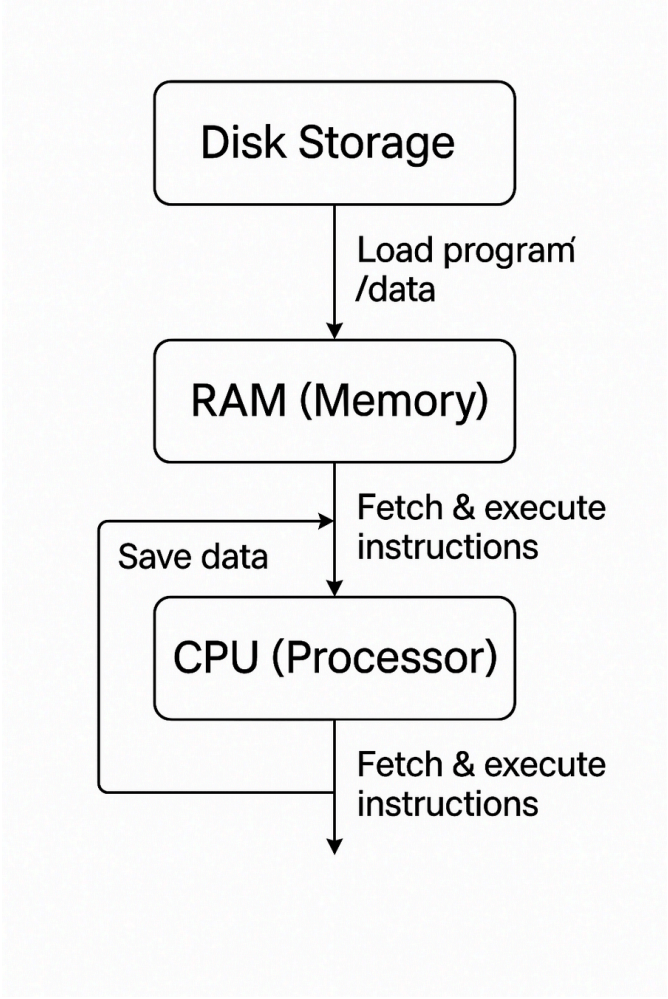
Examples:

- **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

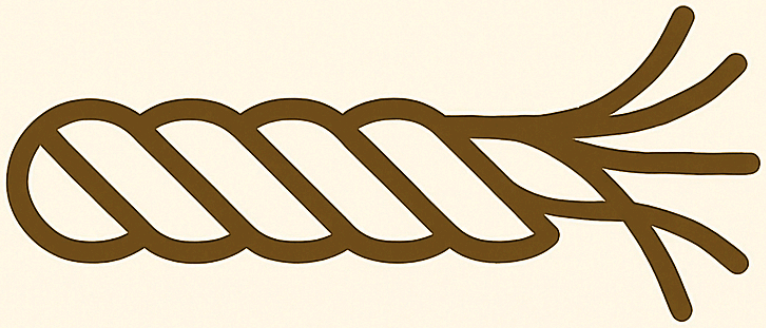
Fun Facts about Processes

- 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

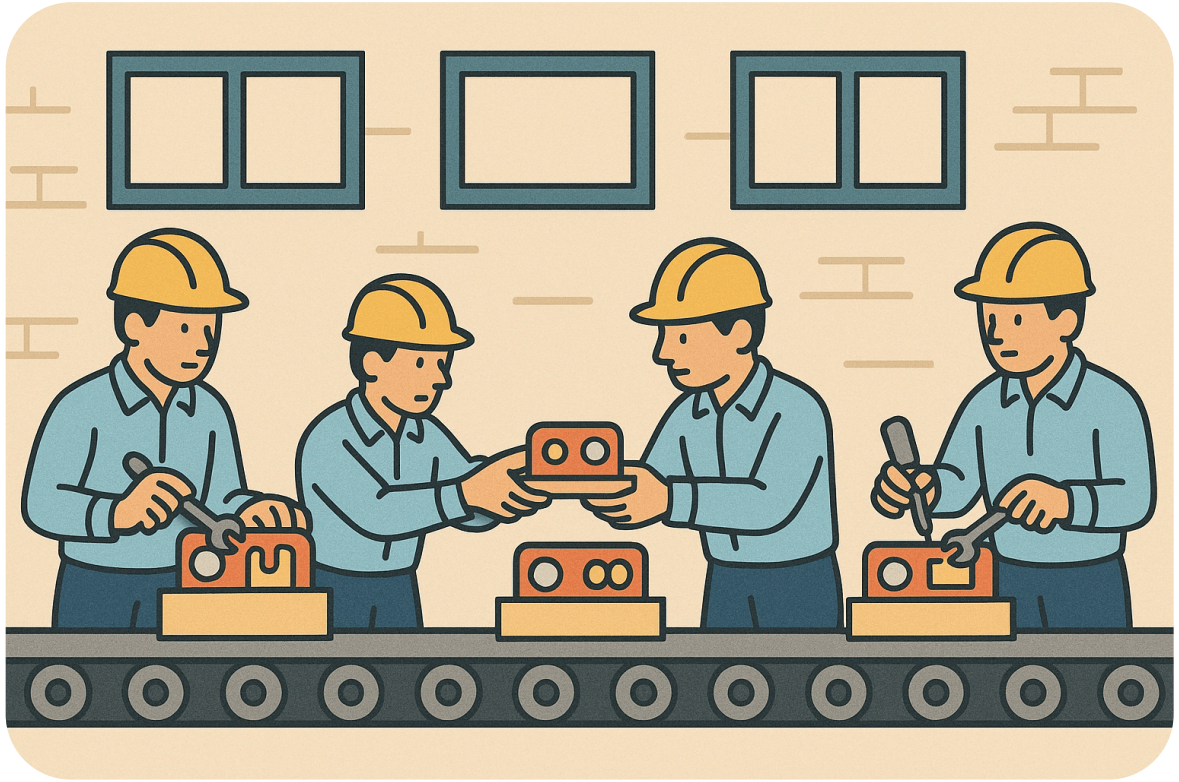
Tasks

- 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

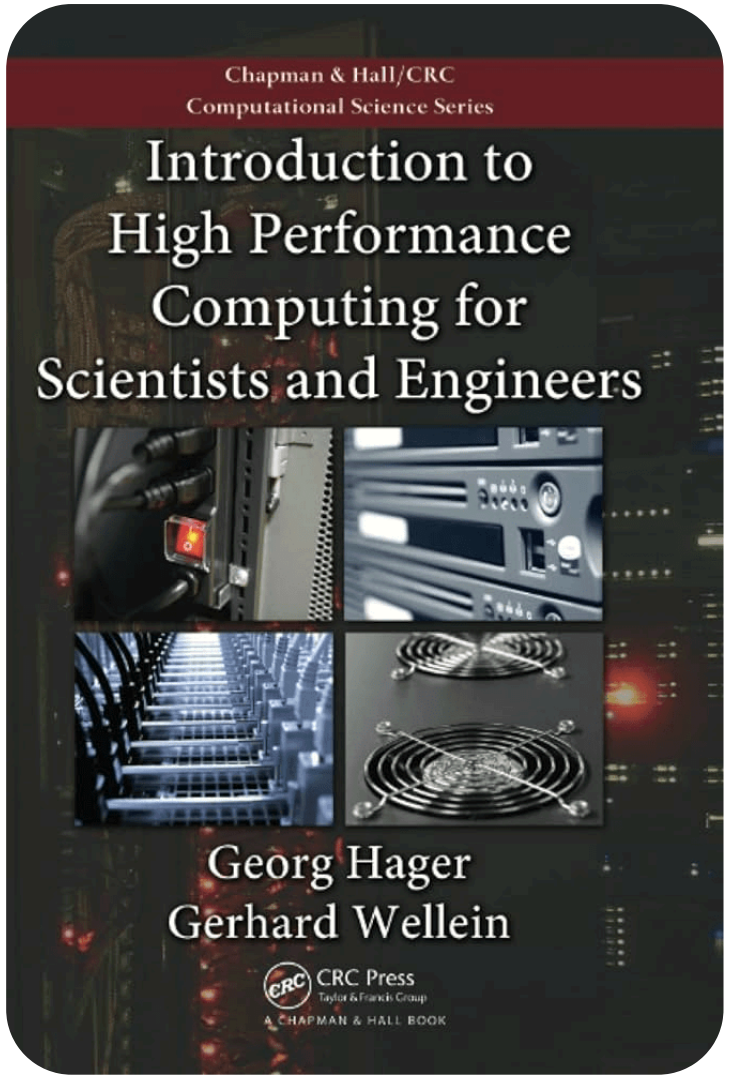
Assembly Line Analogy

- 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

Connecting to the WUR HPC

Guiding users on connecting to the HPC from Linux, Mac, and Windows.

Before You Connect

- The connection to the HPC is enabled by the Secure Shell (SSH) protocol
- On Linux and macOS, SSH is preinstalled.
- On Windows, we recommend the use of **MobaXTerm**
- If not, then you can use
  - Putty
  - Windows Subsystem for Linux (WSL)
  - PowerShell

WARNING: If you mistype the password, your IP may be blocked for 24h.

Windows: PuTTY

1. Install the SSH Client PuTTY
    - Available at the WUR software library
2. Configure PuTTY as follows
    - Host name: login.anunna.wur.nl
3. Hit the [Yes] button
    - A terminal opens with the prompt "Login as:"
4. Type your WUR username followed by <Return>
5. Type your WUR password followed by <Return>

Connecting to the Anunna HPC

Open a terminal and run the command:

```
$ ssh username@login.anunna.wur.nl
```

WARNING: no characters are displayed when typing your WUR password!!!

MobaXTerm: go to Session => SSH and under remote host type fill in

```
login.anunna.wur.nl
```

Anunna

An overview of the Wageningen University's supercomputer and its services

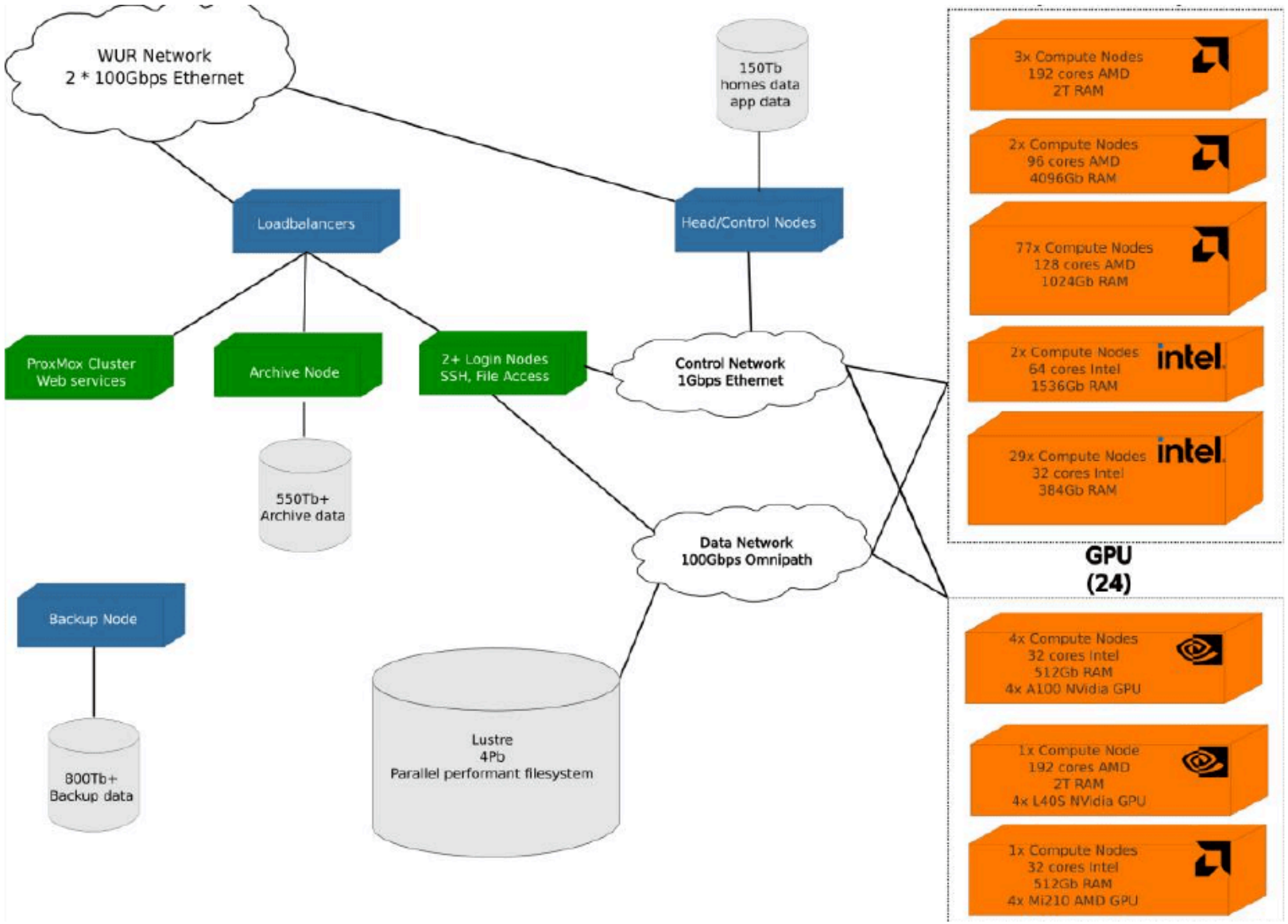
WAGENINGEN
UNIVERSITY & RESEARCH

Overview

Cores: 11616

Nodes: 117

GPUs: 24



WAGENINGEN
UNIVERSITY & RESEARCH

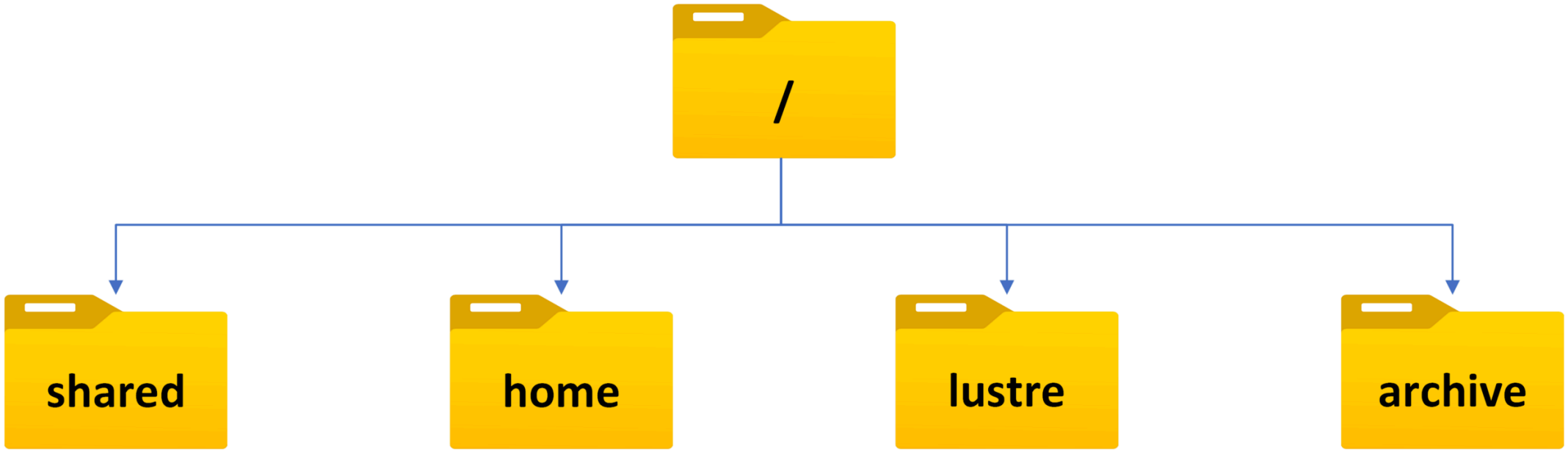
Operating System

Ubuntu 24.04 LTS

This Is a Transition Slide

WAGENINGEN
UNIVERSITY & RESEARCH

Main Folders



Shared - /shared

/shared/apps

- location of the applications installed in the cluster
- dedicated directories for software for a specific institution and group
  - /shared/apps/[institution]/[unit]/[username]

/shared/easybuild/modules/groups

- location of the module files in the cluster
- dedicated directories for modules for a specific institution and group

/home

Home:

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

Running jobs in your home directory may lead your jobs to be cancelled.

Archive

Archive:

- /archive/[institution]/[username]
- A lot cheaper than lustre
- Only for storage (no jobs)
- Slooooooooooooow

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

Storage - Lustre

backup:

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

nobackup:

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

scratch:

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

WAGENINGEN
UNIVERSITY & RESEARCH

Beware

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 $myNoBkp"
```

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

Best Practices and Policies

- We **never** remove user data
- No jobs should run on login node, allocate a compute 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
- Anaconda installations are not supported

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

Web Services

Jupyter

notebook.anunna.wur.nl

- Jobs 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

WAGENINGEN
UNIVERSITY & RESEARCH

Anunna (web)Apps

apps.anunna.wur.nl

- Graphical interface for the HPC
- Built-in:
  - File Browser
  - Jobs Manager
  - Annoucements page
  - Terminal
  - etc...

- R-Studio
- Jupyter
- Desktop
- Metashape Pro*
- Many more to come.

WAGENINGEN
UNIVERSITY & RESEARCH

Wiki

wiki.anunna.wur.nl

- General information about the cluster
  - Costs
  - Software
  - Workflows
  - Tutorials
  - Modules
- Everyone is welcome to contribute

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.

Transfering Files

How to transfer file to and from the HPC

Linux/Mac/Powershell - scp

Template:

```
$ scp -<flags>  user@addressSource user@addressTarget
```

Copying files to the HPC:

```
$ scp myfile user@login.anunna.wur.nl:destination
```

Copying directories from the HPC :

```
$ scp -r user@login.anunna.wur.nl:directory destination
```

STFP

SSH File Transfer Protocol

- Protocol for transfering files via ssh
- Has several graphical clients that implemented
- Often presented in a file browser interface

Linux

File browser

Windows

MobaXterm, Winscp

Mac

Cyberduck

Exercise - Transferring files

Copy file from the location below to your laptop and open it on your laptop.

- **file location:** /lustre/shared/hpcCourses/Basics/picture.jpg
- **Bonus:** Try using both methods
- **Hints:**
  - Do the copying from a terminal inside your laptop not connected to the hpc
  - if you are executing the command from a local terminal, you do not need to use the IP address of your laptop, just the location of the file.

Solution

From a terminal inside your laptop.

```
$ picLocation=/lustre/shared/hpcCourses/picture.jpg
$ scp user001@login.anunna.wur.nl:$picLocation ~/Pictures/
```

WAGENINGEN
UNIVERSITY & RESEARCH

Downloading Files into the HPC

curl

```
curl -JOL <linkAddress>
```

wget

```
wget <linkAddress>
```

Exercise - Download files

1. Go to **github.com/Code-Hex/Neo-cowsay**
2. Go to releases (right side of the screen)
3. Under **assets** look for **cowsay_2.0.4_Linux_x86_64.tar.gz**
4. right click into the link and copy the link address
5. user wget (or curl) to download the file into your home directory
6. create the folder ~/**apps/neocowsay**
7. Extract the file with the command **tar xvfz file.tar.gz -C <TargetFolder>**
8. **run:**

```
$ ~/apps/neocowsay/cowsay "Hello!"
```

Exercise - Downloading Files

Grab the files from the github project and download them at your ~

```
$ cd ~
$ wget https://github.com/Code-Hex/Neo-cowsay/releases/download/v2.0.4/cowsay_2.0.4_Linux_x86_64.tar.gz
```

make a directory for the app in the apps folder

```
$ mkdir -p ~/apps/neocowsay
```

Extract the tarball into the newly created directory

```
$ tar xvfz ~/cowsay_2.0.4_Linux_x86_64.tar.gz -C ~/apps/neocowsay/
```

Run the following command

```
$ ~/apps/neocowsay/cowsay "Hello!"
```

Solution

```
$ cd ~
$ wget https://github.com/Code-Hex/Neo-cowsay/releases/download/v2.0.4/cowsay_2.0.4_Linux_x86_64.tar.gz
$ mkdir -p ~/apps/neocowsay
$ tar xvfz cowsay_2.0.4_Linux_x86_64.tar.gz -C ~/apps/neocowsay
$ cd
$ ~/apps/neocowsay/cowsay "Hello!"
 _________
< Hello! >
 ---------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     |
```

Environments

Bash environment variables, how to manage them and some examples

Definitions

What are variables?

Bash environment variables are key-value pairs stored within the Bash shell that influence the behaviour of software on the system.

What are they used for?

They can be used to configure shell settings, store data like paths to executables or directories, and control the operation of scripts and applications.

Env and Notable Variables

Display the variables in your session:

```
$ env
```

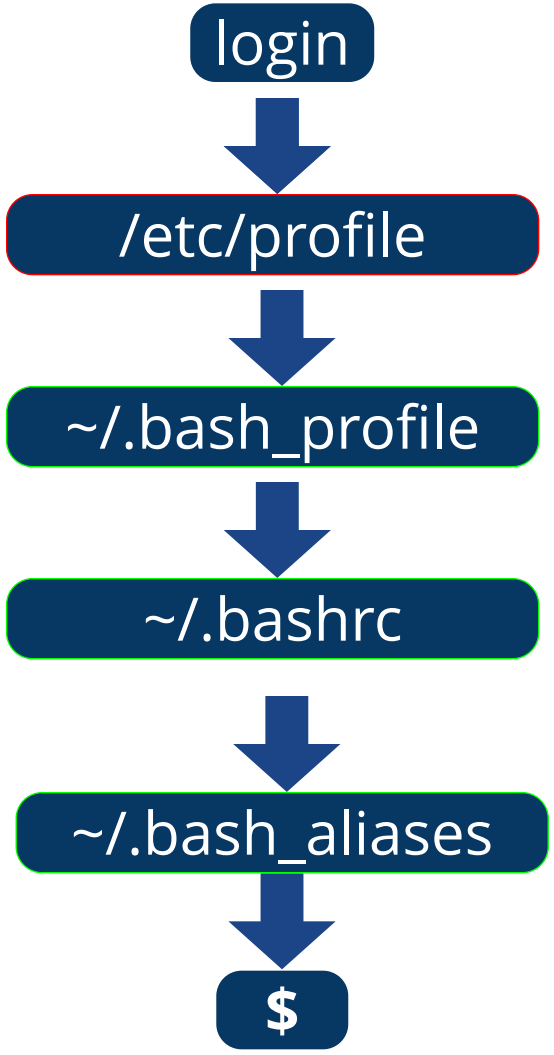
Notable:

HOME - stores the location of your home directory

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

LD_LIBRARY_PATH - stores locations of libraries

WHERE DO THESE VARIABLES COME FROM?



PATH

Environment variable containing a list of location separated by colons ":".
It defines the location of executables in a linux system.

```
$ echo $PATH

/shared/apps/slurm/24.05.4/sbin:/shared/apps/slurm/24.05.4/bin:/usr/lo
cal/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/home/
WUR/user001/.local/bin:/home/WUR/user001/bin
```

When using a variable use the expansion operator $ in front of the variable

Printing the contents of a Variable

Environment variable containing a list of location separated by colons ":".
It defines the location of executables in a linux system.

You can print the contents of the variable with the echo command

```
$ echo HOME

HOME
```

When using a variable in a command, the expansion operator $ must be put in front of the variable (as a prefix)

```
$ echo $HOME

/home/WUR/user001
```

WAGENINGEN
UNIVERSITY & RESEARCH

Assigning Variables

Variables can assigned by using the following template

```
$ var="Hello"
```

Export it to make it visible from all child processes of shell session

```
$ var="Hello"
$ export var
```

```
$ export var="Hello"
```

The export command transforms a shell variable into an environment variable

Example

```
$ PATH=$HOME/apps/neocowsay:$PAT
$ echo "Hello!" | cowsay  # "cat
$ cowsay "Hello"
```

WAGENINGEN
UNIVERSITY & RESEARCH

Exercise - Appending Variables

1. Add the **~/apps/neocowsay** folder (from the previous exercise) to your **PATH** variable
2. Change your working directory to your home
3. Run the command

```
$ echo "Hello" | cowsay
```

Workalong

Check your current PATH variable

```
$ echo $PATH
```

Check your current PATH variable has the neocowsay directory

```
$ echo $PATH | grep neocowsay
```

Add the neocowsay location to your PATH variable (append). The order matters

```
$ PATH=$HOME/apps/neocowsay:$PATH
#or
$ PATH=$PATH:$HOME/apps/neocowsay
```

Check your current PATH variable has the neocowsay directory

```
$ echo $PATH | grep neocowsay
```

Go to your ~ and run

```
$ cd ~
$ cowsay "Hello"
```

WAGENINGEN
UNIVERSITY & RESEARCH

Aliases

Aliases subtitute a longer command with a short one

Change directory to my scratch directory

```
alias cds='cd $(getLustreDir -s)'
```

To have an overview of existing aliases

```
$ alias
```

Aliases can be stored in ~/.bash_aliases (see example in wiki)

Symbolic Links (shortcuts)

It is handy to create links to your lustre folders in your home folder (jupyter)

```
$ ln -s $myScratch ~/scratch
```

Can be removed with the following command

```
$ unlink <linkToBeRemoved>
```

Lmod - Environment Modules

An introduction to environment modules and some usage examples

What does it do?

Lmod

A flexible, Lua-based **environment module** system that enables the dynamic modification of a user's environment via module files, simplifying the management of software and library versions on HPC systems.

Environment module

A system that allows for the dynamic configuration of a user's shell environment variables, facilitating easy management of application environments and versions on UNIX-like systems.

MODULEPATH

- A bash environment variable that tell **lmod** where the modules are located.
- Multiple locations can be specified, divided by a colon (**:**)
- Can be modified with an **export** command
- Can be appended with the **module use** command

Find the default MODULEPATH location:

```
$ echo $MODULEPATH
```

WAGENINGEN
UNIVERSITY & RESEARCH

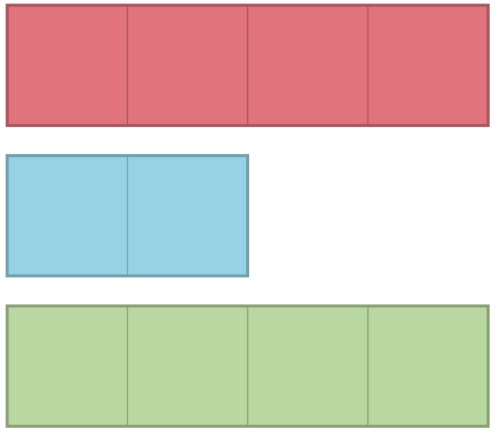
Let's talk about buckets

WAGENINGEN
UNIVERSITY & RESEARCH

Modules at Anunna

- Modules are to be 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 from modules is optimized for the HPC
- Contact us if you need some specific software
- More information in the wiki:

Link to wiki Article



EASYBUILD.io
building software with ease

WAGENINGEN
UNIVERSITY & RESEARCH

Module "Buckets"

- A collection of modules made with group with common dependencies (toolchains).
- Buckets are just module files that modify the **MODULEPATH**
  - **GPU:** CUDA and GPU related packages independent of toolchains
  - **utilities:** Common utilities that are independent of toolchains
  - **2023:** Software built around the 2023 toolchains
  - **2024:** Software built around the 2024 toolchains
  - **2025*:** Software built around the 2025 toolchains
  - **groups:** User maintained, also includes private software modules
  - **legacy:** Old software, **no longer supported** and are due to be removed

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

Loading Modules

load module (e.g. python)

```
$ module load 2023
$ module load Python/3.11.3-GCCcore-12.3.0
```

load module (e.g. python)

```
$ ml 2023
$ ml Python-bundle-PyPI/2023
```

List loaded modules

```
$ module list
```

Searching for Modules

Searching for modules

```
$ module avail R/ # ml av R/
$ module spider R/4 # ml spider R/
```

Search for a keyword inside a module's description

```
$ module key numpy # ml key numpy
```

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

Removing Modules

Remove individual module (e.g. python)

```
$ module unload Python/3.11.3
```

Remove all modules

```
$ module reset
```

Swap a module with another

```
$ module swap 2023 2024
```

WAGENINGEN
UNIVERSITY & RESEARCH

Exercise - Searching for Modules

1. Search for the **terra** package
2. Load the module containing that package
    1. (Choose the appropriate bucket)
3. Check the modules loaded
4. Clean up (reset/purge) the modules
5. Check your loaded modules again

Solution

```
$ module key terra
-----------------------------------------

The following modules match your search criteria: "terra"
-----------------------------------------

  R-bundle-CRAN: R-bundle-CRAN/2023.12-foss-2023a, R-bundle-CRAN/2024.11-foss-
2024a
    Bundle of R packages from CRAN

  terra: terra/1.7-55 (E), terra/1.7-83 (E)

Names marked by a trailing (E) are extensions provided by another module.

$ module load 2023 R-bundle-CRAN/2023
$ module list
$ module purge
$ ml # abreviation for module list
```

Getting Additional Information

Get the description of a module

```
$ module whatis Python-bundle-PyPI
```

Get the entire module file

```
$ module show Python-bundle-PyPI # ml show Python-bundle-PyPI
```

MODULEPATH

Defines the location of your module files. Like path it can also be modified

Append MODULEPATH:

```
$ module use $HOME/myFolder
```

Entries can also be removed

```
$ module unuse $HOME/myFolder
```

Redefine MODULEPATH directly

```
$ export MODULEPATH=/my/Directory/Location
```

Exercise - Create Your Own Modules

Do This exercise in a new bash session. Either login again or type "bash".

1. Create a folder called **modules** in your home directory
2. Create a subfolder inside of **modules** named **cowsay** (**~/modules/cowsay**)
3. Copy the file **2.0.4.lua** from **/lustre/shared/hpcCourses** to the cowsay module subfolder
4. Redefine your MODULEPATH to point to **~/modules**
5. Use the spider command to search for the cowsay module
6. At your home directory, load the **cowsay** module
    1. Check your PATH variable, before and after loading the module (you can always unload)
7. Execute the command: **cowsay "Hello"**

Workalong - 1/2

Grab a fresh shell session , either login again or run

```
$ exec bash
```

Create a folder for your modules in your home directory

```
$ mkdir -p ~/modules/cowsay
```

Copy the module file from the course folder to your newly created folder

```
$ cp /lustre/shared/hpcCourses/Basic/2.0.4.lua ~/modules/cowsay/
```

Reset your modules and check your MODULEPATH and PATH variables

```
$ module reset
$ echo $MODULEPATH
$ echo $PATH
```

Add the module directory

```
$ module use ~/modules
```

WAGENINGEN
UNIVERSITY & RESEARCH

Workalong - 1/2

Check your MODULEPATH variable again, notice anything different?

```
$ echo $MODULEPATH
```

List your available modules

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

load the cowsay module and list it

```
$ ml cowsay/2.0.4 # module load cowsay/2.0.4
$ ml # module list
```

Check your PATH vartiable again. Do you undertand what the module did?

```
$ echo $PATH | grep neocowsay
```

Now you can run cowsay from wherever you want

```
$ cd $myScratch
$ cowsay "I am in a module"
```

WAGENINGEN
UNIVERSITY & RESEARCH

The Anunna module

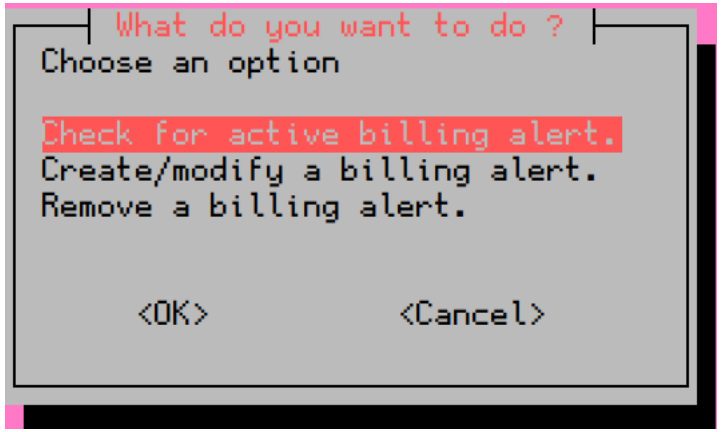
Collection of scripts and tools pertinent to anunna

```
$ 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

Jobs

How to submit and monitor a job at the WUR HPC

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

Cluster Overview

Get an overview of the current state of a cluster/partition

```
$ sinfo -Nel -p <partition>
```

```
user001@login200:~$ sinfo -Nel -p gpu
Tue Jun 09 11:09:14 2026
NODELIST   NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE
REASON
gpu100         1       gpu       mixed 32     2:16:1 384000        0    168 intel,nv none
gpun200        1       gpu      mixed- 32     2:16:1 512000        0    108 intel,nv none
gpun201        1       gpu       mixed 32     2:16:1 512000        0    112 intel,nv none
gpun202        1       gpu       mixed 32     2:16:1 512000        0    118 intel,nv none
gpun203        1       gpu       mixed 32     2:16:1 512000        0    116 intel,nv none
gpuxn200       1       gpu       mixed 64     2:32:1 512000        0    151 intel,nv none
node301        1       gpu      mixed- 192    2:96:1 230400        0     44 gen4,amd none
```

Get an overview of the entire cluster (pipe to less, q to quit)

```
$ sinfo -Nel | less
```

WAGENINGEN
UNIVERSITY & RESEARCH

Queue Overview

Get an overview of the entire queue

```
$ squeue -a
```

Get an overview of a partition

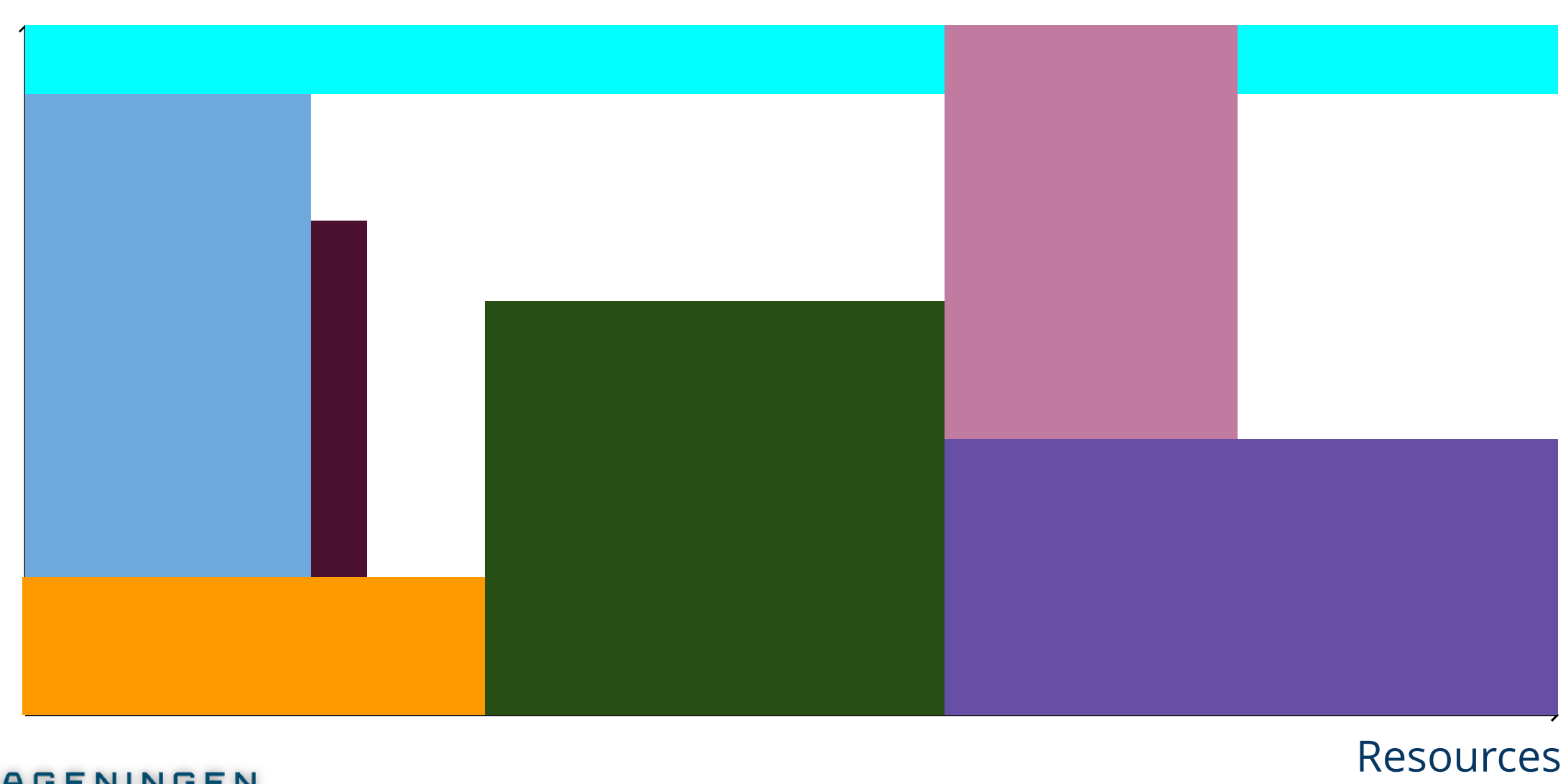
```
$ squeue -p gpu
```

More about squeue later...

Requirements for a job

- Sequential or parallel
- If parallel: **multi-threaded** or **multi-process**?
- Resource requests:
  - Number of CPUS
  - Amount of RAM
  - Expected computing time
  - …
- Job steps
  - Job steps can be created with the command **srun** (advanced)

Tetris



WAGENINGEN
UNIVERSITY & RESEARCH

Resources - How to determine

- Read documentation
- Read the source code
- Test it! Run it in a safe environment
  - use and interactive session: srun or sinteractive

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**

Interactive Sessions

- Bash session in a computational node
- Clean environment, you will need to load modules again
- You can specify the resources available:
  - Number cores
  - Amount of RAM
  - Time
  - ...
- If resources are exceeded, e.g. RAM or time, the session will close
- Two ways to do it:
  - **srun**
  - **sinteractive**

sinteractive

sinteractive template

```
$ sinteractive -c <num_cpus> --mem <amount_mem> --time <minutes> -p <partition> --x11
```

Example

```
# sinteractive -c <cpus> --mem <MB> --time=D-H:M
$ sinteractive -c 1 --mem 2000 --time=0-0:10
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

Srun

srun template

```
srun -N <nodes> -n <processes> \
        --ntasks-per-node=<processes_per_node> \
        --time=<D-H:M> --constraint=<tag> \
        --mem=2G --cpus-per-task=1 \
        --x11 --pty bash -i
```

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

Most jobs (R and Python) will need only 1 node (**-N**) and 1 task (**-n**) and multiple cpus-per-task (threads/cores). For these jobs we can use **sinteractive**

check efficiency

```
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)
```

Exercise - Virtual Environment - Step 0

Load the 2024 bucket and Python/3.12.3

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

Create a virtual enviornment (in lustre scratch)

```
$ mkdir $myScratch/hpcCourse; cd $_
$ 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 - bash Script - Step 1

Application: Weather Data Analyser

- Python script that loads weather data from a Dutch weather station and draws some figures
- /lustre/shared/hpcCourses/Basics/weer_vanaf_2000.py

Tasks:

1. Copy the python script above into **$myScratch/hpcCourse/**
2. Write a bash script that loads the modules and activates the virtual environment you created. Call it **weather.sh** (make sure it is executable)
3. The script should load the necessary modules and activate your virtual environment you prepared in the last slide
4. Your script should end with this command (**hint:** use variables for paths)

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

Solution

```
#!/bin/bash

module purge
module load 2024
module load Python/3.12.3

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

Exercise - Step 2

Find the job requirements

Tasks

1. Find the job requirements (e.g., memory) of the Python script by using **sinteractive,** increase the amount of RAM until the job no longer crashes

```
$ sinteractive --mem 100M --time=0-0:10 -c 1
$ cd $myScratch/hpcCourse
$ ./weather.sh
```

Takeaways

- Interactive sessions
  - Are quick ways to experiment with the settings of a job
  - Provide rapid feedback
  - Can be used with applications that require input.
  - Will be terminated as soon as you disconnect. (unless you use tmux/screen)

Short break?

Example SLURM Script

```
#!/bin/bash
#---------Name of the job--------------
#SBATCH --job-name=example1
#---------Mail address -----------------
#SBATCH --mail-user=my.email@wur.nl
#SBATCH --mail-type=ALL
#---------Output files-----------------
#SBATCH --output=result_%j.txt
#SBATCH --error=error_result_%j.txt
#---------Other information------------
#SBATCH --comment='Some comments'
#---------Required resources-----------
#SBATCH --time=0-1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem=4000
#---------Environment, Operations and Job step -------
echo 'This is my job'
```

WAGENINGEN
UNIVERSITY & RESEARCH

SLURM options

Job name:

```
--job-name="job1"
```

To set the name of output files:

```
--output=result_%j.txt
--error=error_result_%j.txt
```

Set comment/label (for billing):

```
--comment="abcd"
```

SLURM options - required resources

To require **1 task**:

```
--ntask=1
```

To require **1 task that can use up to 8 cpus:**

```
--ntasks=1
--cpus-per-task=8
```

To require **memory** (in MB):

```
--mem=4G
OR
--mem-per-cpu=4G
```

SLURM options - Time

To require **15 minutes**:

```
--time=15
```

To require **1 day**:

```
--time=1-00:00:00
```

To require **1 day, 5 hours and 30 minutes:**

```
--time=1-05:30:00
```

Maximum time is set to 3 weeks

WAGENINGEN
UNIVERSITY & RESEARCH

Partitions

Main partition:

```
--partition=main
```

GPU partition:

```
--partition=gpu          --partition=gpu_amd
```

If a gpu partition is used, remember to allocate a gpu

```
--gres=gpu:1 #allocates 1 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

GPU Types

- **NVIDIA V100 (16 GB HBM2) - gpu100** – Strong general-purpose training GPU for FP32/FP16 workloads, well-suited for classic deep learning training and HPC compute.
- **NVIDIA A100 80 GB (HBM2e) - gpun20[0 - 3]** – 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) - node301** – Optimized for high-throughput inference and small- to mid-scale training, great performance-per-watt for production AI workloads.
- **AMD MI210 (64 GB HBM2e) - gpua201** – Strong HPC-leaning training GPU with high FP64/FP32 and solid BF16/FP16 training performance, best when ROCm fits the software stack.

You need to use constraints to select the node where each GPU lives

Inspecting a node

```
$ scontrol show node <nodename>
```

```
user001@login200:~$ scontrol show node node301
NodeName=node301 Arch=x86_64 CoresPerSocket=96
   CPUAlloc=17 CPUEfctv=192 CPUTot=192 CPULoad=4.15
   AvailableFeatures=gen4,amd,zen5,avx512,192cpus,2304g,12gpercpu,nvidia,L40S,aruru
   ActiveFeatures=gen4,amd,zen5,avx512,192cpus,2304g,12gpercpu,nvidia,L40S,aruru
   Gres=gpu:4
   NodeAddr=node301 NodeHostName=node301 Version=24.11.6
   OS=Linux 5.15.0-88-generic #98-Ubuntu SMP Mon Oct 2 15:18:56 UTC 2023
   RealMemory=2304000 AllocMem=458752 FreeMem=2209849 Sockets=2 Boards=1
   State=MIXED+PLANNED ThreadsPerCore=1 TmpDisk=0 Weight=44 Owner=N/A MCS_label=N/A
   Partitions=gpu
   BootTime=2026-06-04T09:32:19 SlurmdStartTime=2026-06-09T02:00:47
   LastBusyTime=2026-06-04T09:50:27 ResumeAfterTime=None
   CfgTRES=cpu=192,mem=2250G,billing=192,gres/gpu=4
   AllocTRES=cpu=17,mem=448G,gres/gpu=4
   CurrentWatts=0 AveWatts=0
```

Submitting a script to SLURM

Template:

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

Executing a slurm script

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

SLURM assigns an ID to the job ($JOBID)

Executing a slurm script overriding the memory and core allocation

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

WAGENINGEN
UNIVERSITY & RESEARCH

View the status of your jobs

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
```

Cancelling a job

Cancel a single job:

```
$ scancel <job_id>
```

Cancel multiple jobs:

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

Cancel all my jobs

```
$ scancel --me
```

Cancel all my pending jobs:

```
$ scancel --me --state=PENDING
```

Exercise - Step 3

Write and submit a slurm script

1. **Write** a SLURM script for the job with the resource requirements determined at Step 2, label it **weather.slurm**
2. **Submit** the job to SLURM
   - Use the **sbatch** command
3. **Check the status** of your job and cancel it if needed.
4. **Check the figures** generated by the Python script.

TIP - Template of a SLURM script:

```
/lustre/shared/hpcCourses/Basics/script_slurm.sh
```

Solution

hpcbasics_jobs.sh

```
2  # -----------------------------Name of the job-------------------------
3  #SBATCH --job-name="hpcCourse"
4  #-----------------------------Output files-----------------------------
5  #SBATCH --output=output_%j.txt
6  #SBATCH --error=error_output_%j.txt
7  #-----------------------------Other information------------------------
8  #SBATCH --comment='Some comments'
9  #-----------------------------Required resources-----------------------
10 #SBATCH --time=0-0:10:00
11 #SBATCH --ntasks=1
12 #SBATCH --cpus-per-task=1
13 #SBATCH --mem=2G
14 #-----------------------------Environment, Operations and Job steps----
15
16 module load 2024
17 module load Python/3.12.3
18 #myScratch=/lustre/scratch/[institution]/user001/
19 source $myScratch/hpcCourse/venv
20 python $myScratch/hpcCourse/weer_vanaf_2000.py 240
```

Monitoring Jobs

Introduction to monitoring slurm jobs, and the anunna module

Monitoring Active and Finished Jobs

Commonly used SLURM commands to monitor and control jobs

- squeue
- scancel
- sacct
- scontrol

scontrol - Slurm configuration and state

- Used to update job resource requests
- Works 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
```

To get information on a specific node

```
$ scontrol show node node201
```

sacct

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

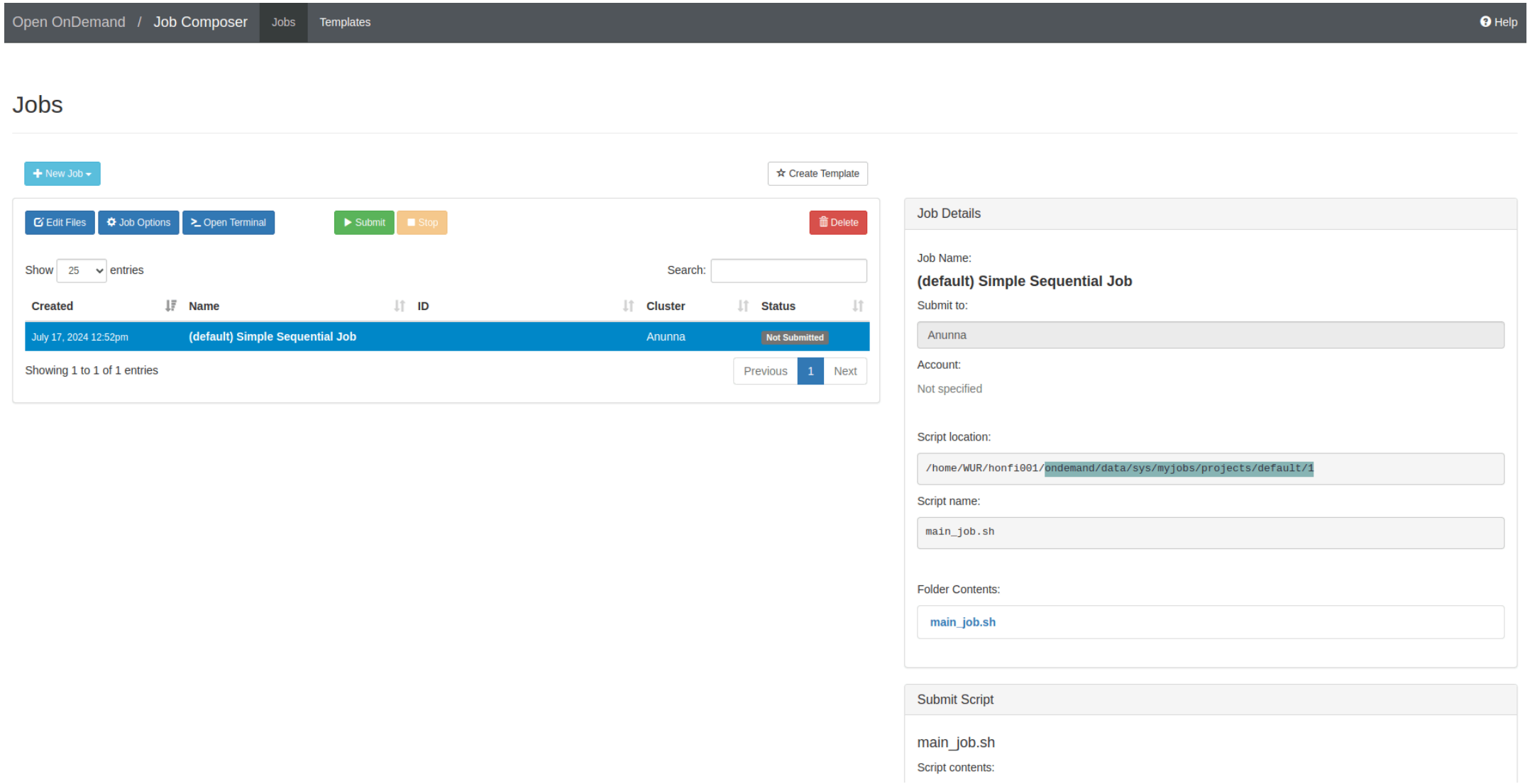
To get an overview of all jobs run in 2024

```
$ sacct -X -u $USER --starttime 2024-01-01 --endtime now
```

To get an overview of a job for troublshooting

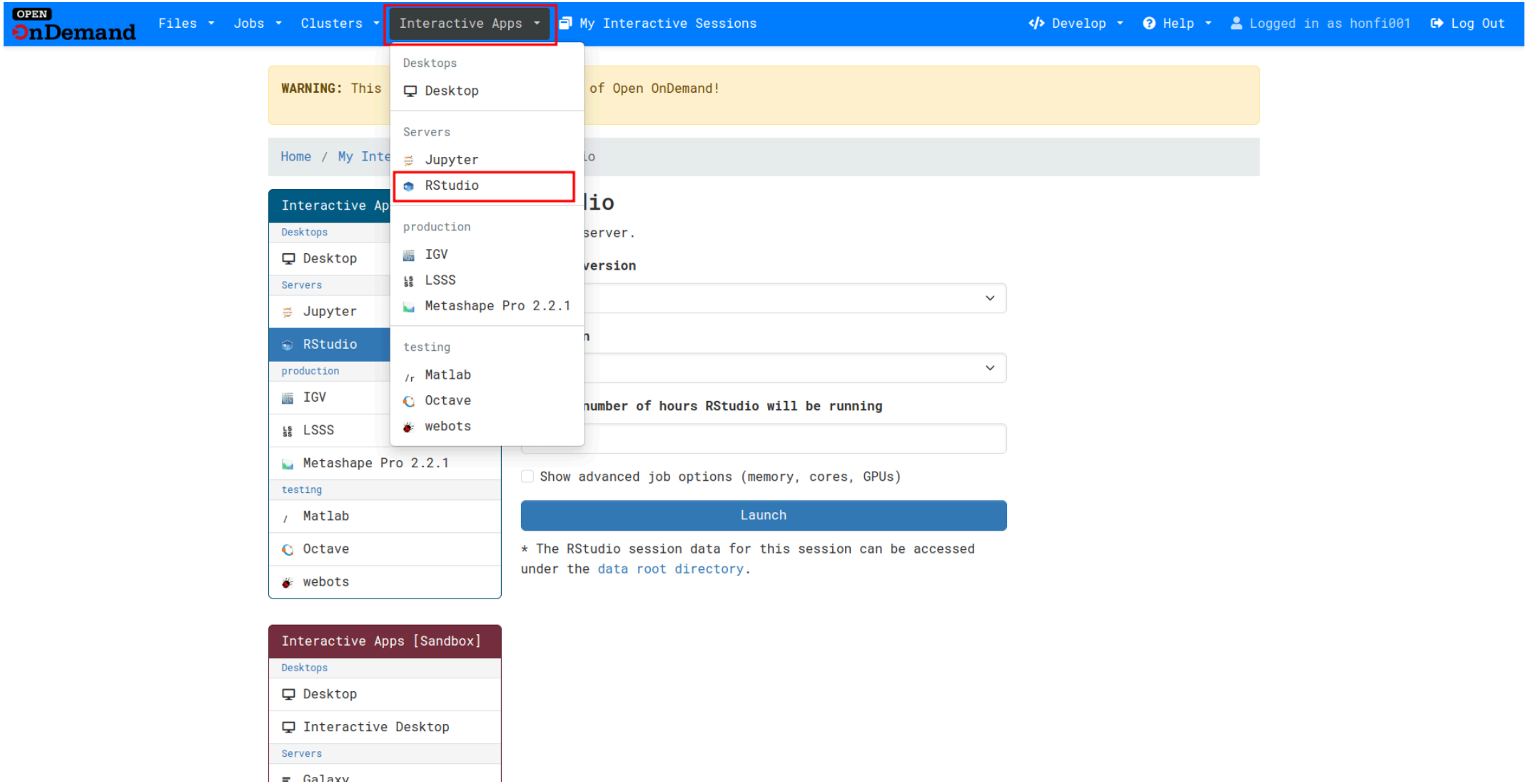
```
$ sacct -j <job_id> -X --
format=JobID,JobName,Partition,State,ExitCode,Start,End,Elaps
ed,MaxRSS,NodeList
```

Jobs Composer



WAGENINGEN
UNIVERSITY & RESEARCH

Interactive Apps



WAGENINGEN
UNIVERSITY & RESEARCH

Demonstration?

Job Arrays (Optional)

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

Job Arrays - environment variable

The array is accessed in the a SLURM variable

The job array index value

$SLURM_ARRAY_TASK_ID

Exercise - Step 4

Aim

- **Submit** an array of jobs to run the same program simultaneously using different inputs

Tasks

- **Write** a SLURM script for a **job array** that loads and generate figures for **multiple Dutch weather stations**, with IDs 240, 260 and 270
  - **SLURM option**: --array=240,260,270
  - **SLURM variable**: $SLURM_ARRAY_TASK_ID

Solution

hpcbasics_array.sh

```
1  #!/bin/bash
2  # -----------------------------Name of the job-----------------------
3  #SBATCH --job-name="hpcCourse"
4  #-----------------------------Output files--------------------------
5  #SBATCH --output=output_%j.txt
6  #SBATCH --error=error_output_%j.txt
7  #-----------------------------Other information---------------------
8  #SBATCH --comment='Some comments'
9  #-----------------------------Required resources--------------------
10 #SBATCH --time=0-0:10:00
11 #SBATCH --ntasks=1
12 #SBATCH --cpus-per-task=1
13 #SBATCH --mem=2G
14 #SBATCH --array=240,260,270
15 #-----------------------------Environment, Operations and Job steps----
16
17 module load 2023
18 module load Python/3.11.3
19 myscratch=/lustre/scratch/[institution]/user001/
20 cd $mycratch/hpccourse
21 python weer_vanaf_2000.py $SLURM_ARRAY_TASK_ID
```

Closing Remarks

Support, wiki, community and whatever else

HPC Advanced Overview

12:30 - Introduction and IceBreaker

12:45 - SSH Keys

13:00 - Tools and tips

13:15 - HPC Basics Review

13:45 - Dynamic Job Arrays + exercise

14:10 - Types of Parallelism + exercise

14:45 - Break

15:00 - Types of Storage

15:30 - Apptainer Containers

16:00 - Bring Your Own Problem

17:00 - End

WAGENINGEN
UNIVERSITY & RESEARCH

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.
  - 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