



VLAAMS
SUPERCOMPUTER
CENTRUM



Vlaanderen
is supercomputing

HPC training

Part1: Introduction

VUB-HPC

5 November 2024

hpc@vub.be

<https://hpc.vub.be>

Intro slides are available here:

<https://hpc.vub.be/hpctraining/intro.pdf>

Free (online) training sessions

VUB-HPC <https://hpc.vub.be/posts/category/event/>



VSC <https://www.vscentrum.be/training>

VLAAMS
SUPERCOMPUTER
CENTRUM



Vlaanderen
is supercomputing

PRACE <https://events.prace-ri.eu/>



EuroCC <https://encs.se/events/>



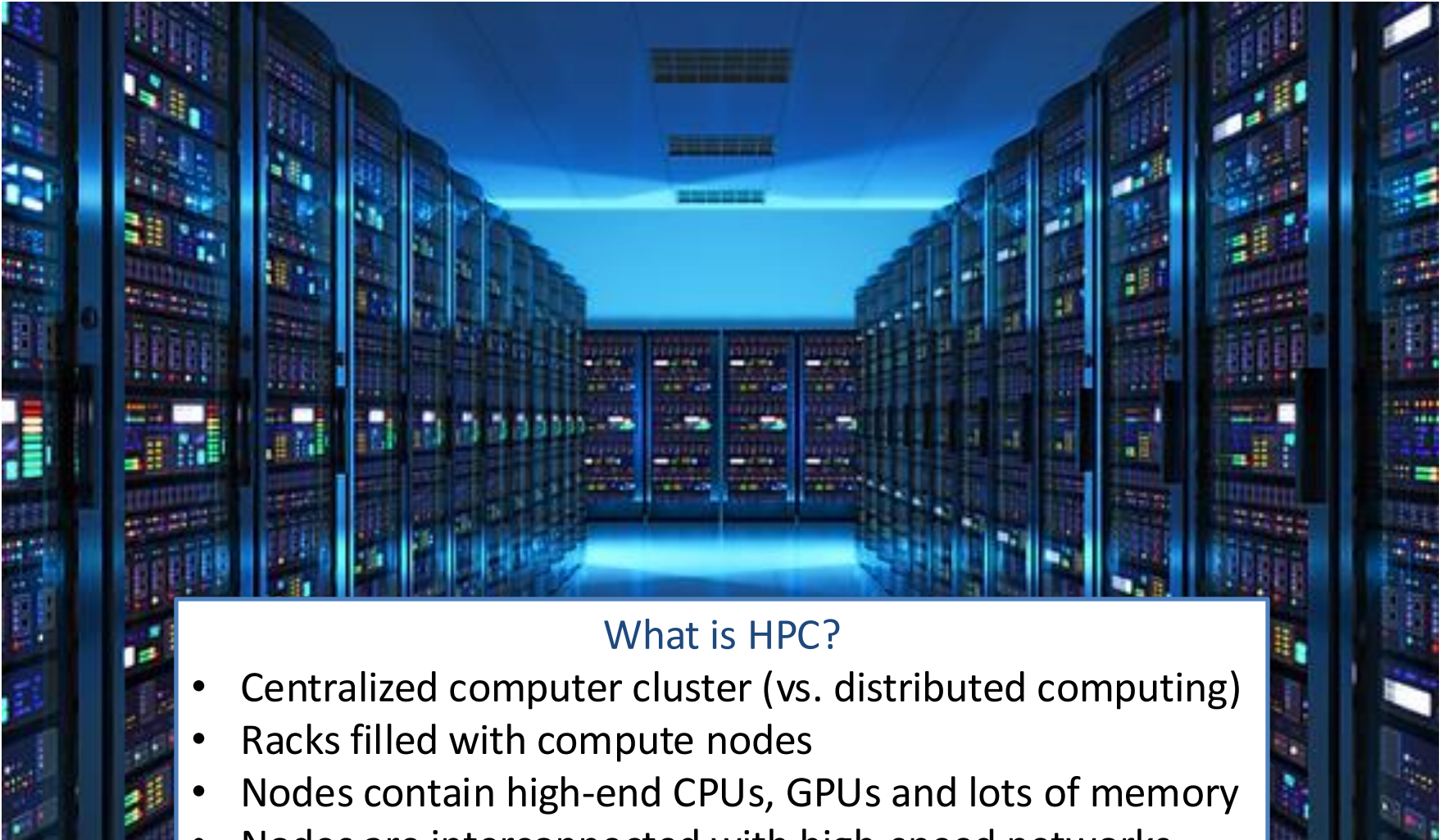
LUMI <https://lumi-supercomputer.eu/events/>



portal for European HPC <https://hpc-portal.eu/node/88?category=11&init=1>

Scientific Python, MPI, OpenMP, GPU, Code optimization, ... and much more

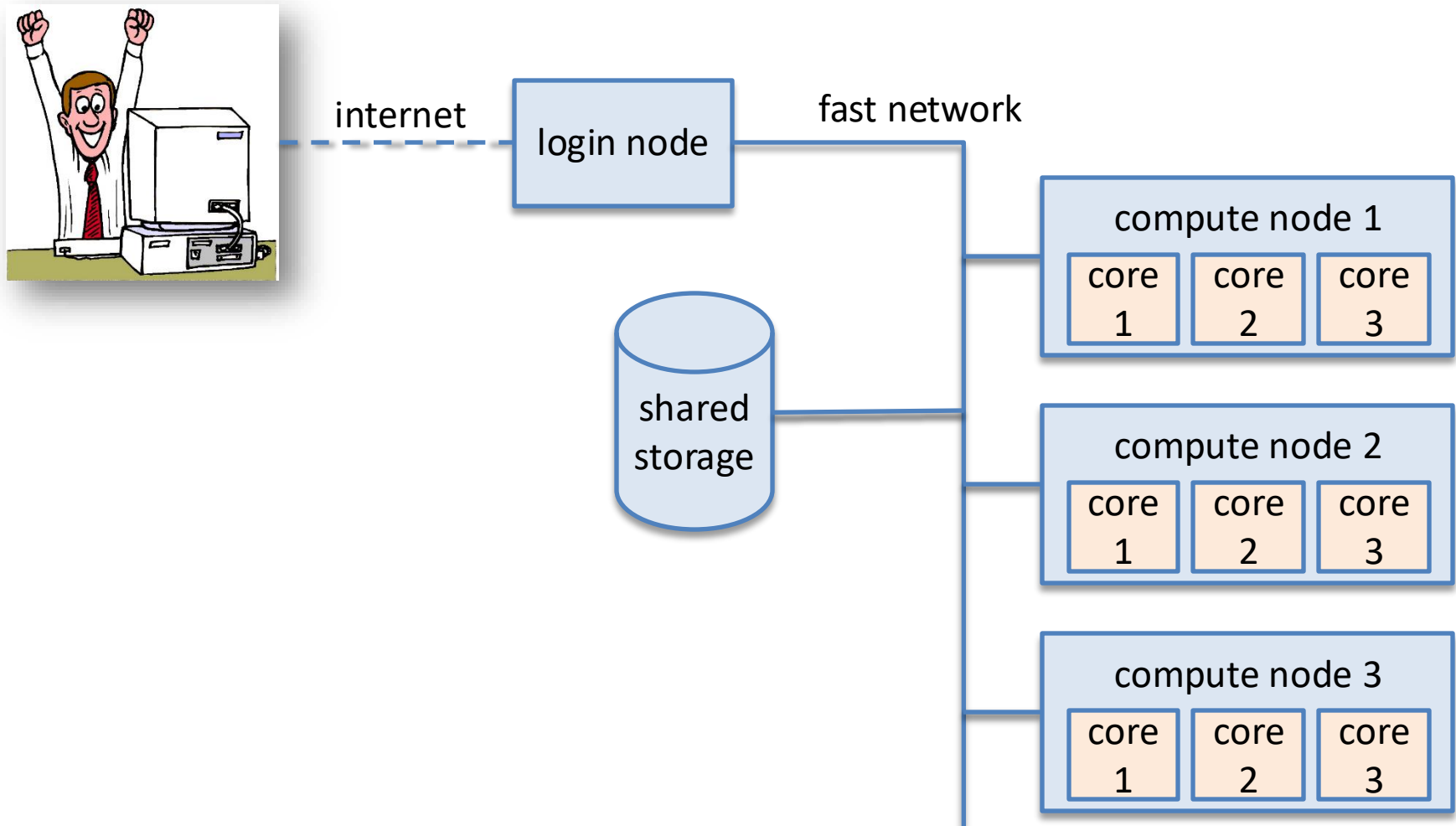
What is high performance computing (HPC)?



What is HPC?

- Centralized computer cluster (vs. distributed computing)
- Racks filled with compute nodes
- Nodes contain high-end CPUs, GPUs and lots of memory
- Nodes are interconnected with high-speed networks
- Runs software designed for HPC

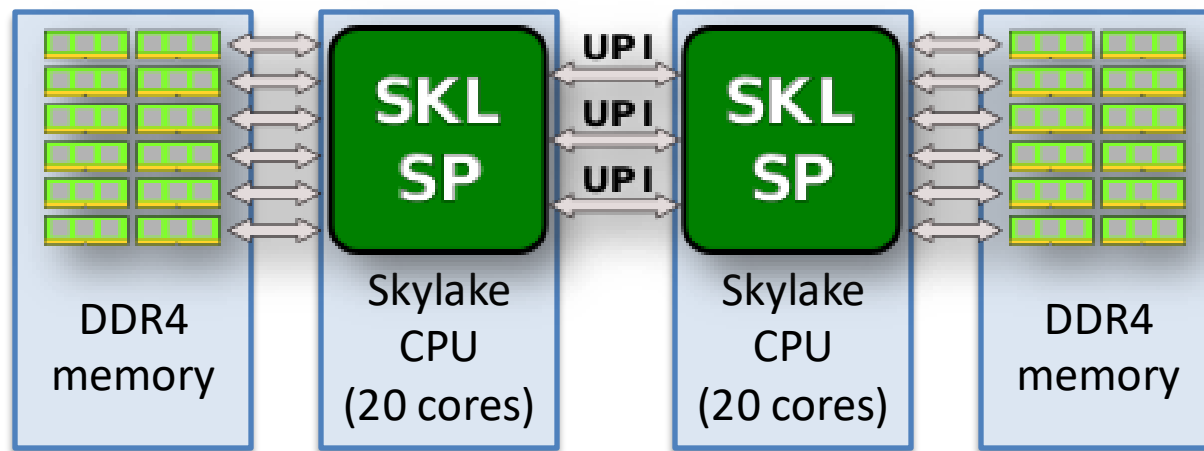
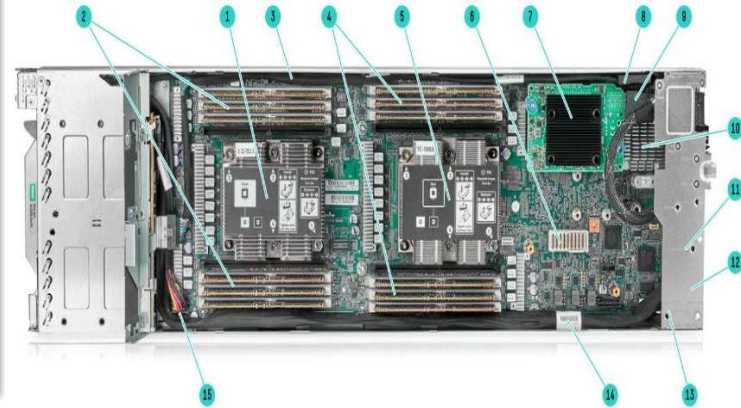
HPC architecture



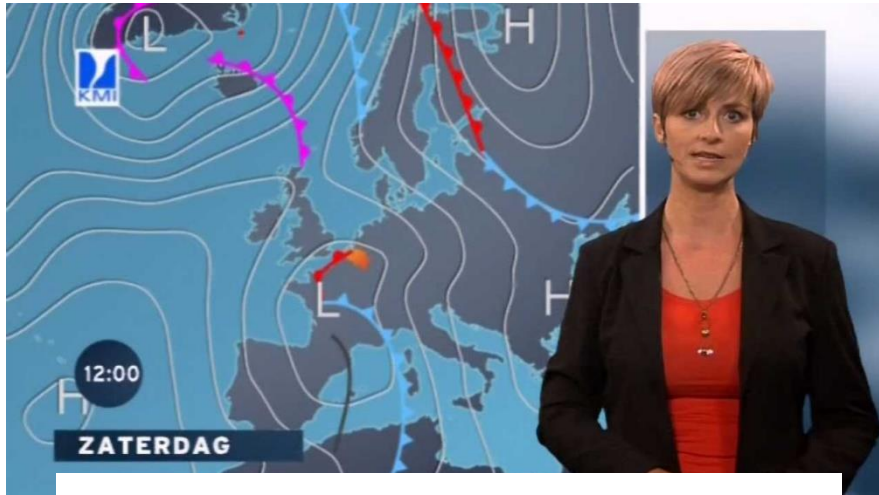
HPC architecture

node vs. CPU vs. core

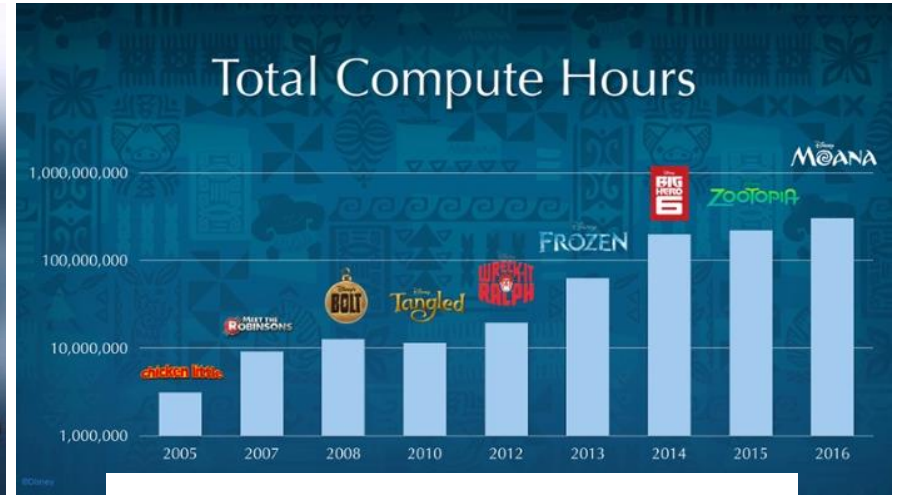
- 1 **node** can have multiple **CPU**s (sockets)
 - each CPU has its own memory
 - Hydra: currently 2 CPUs/node
- 1 **CPU** can have multiple CPU **cores**
 - Hydra: currently 24-64 cores/node



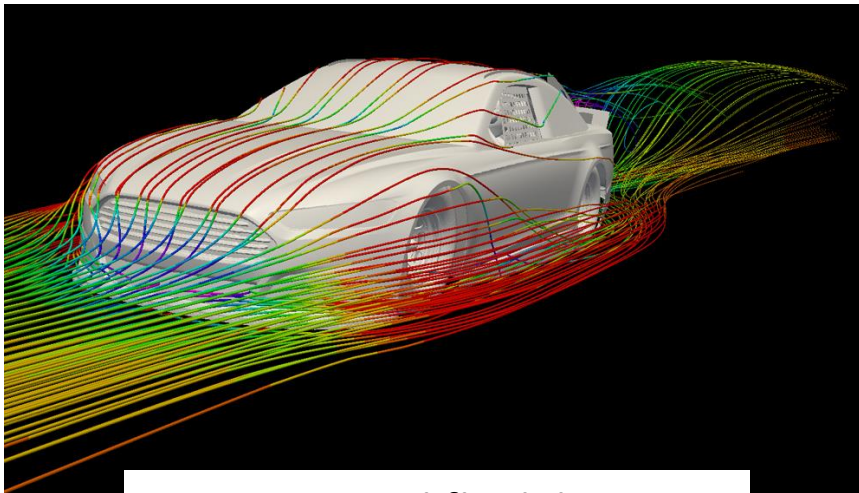
Everyday applications of HPC



Weather prediction, climate modeling



Animation movies: CGI rendering

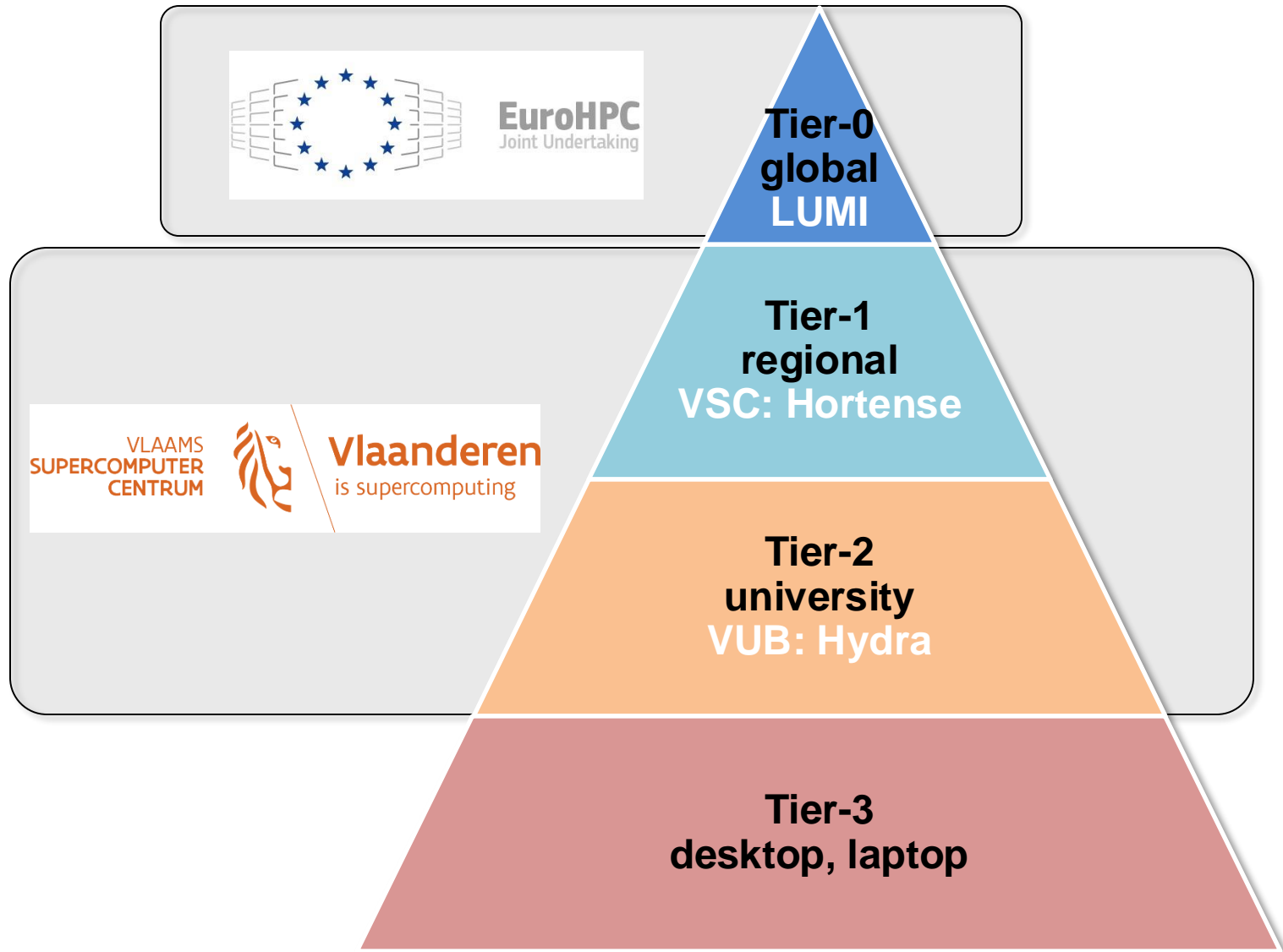


Computational fluid dynamics



High-frequency trading

HPC tiers: scaling from local to global



Tier-0 in Europe

Project-based access



LUMI (CSC, Kajaani, Finland)

<https://www.lumi-supercomputer.eu>

Consortium: Finland, Belgium, the Czech Republic, Denmark, Estonia, Iceland, Norway, Poland, Sweden, and Switzerland.

€ 200 million

Production: 2022

375 petaflops (1.5 million laptops)

- >200,000 AMD CPU cores
- ~10,000 AMD GPUs

Tier-1 in Flanders: VSC

Project-based access



VLAAMS
SUPERCOMPUTER
CENTRUM



Vlaanderen
is supercomputing

Hortense (Gent)

https://docs.vscentrum.be/gent/tier1_hortense.html

Production: 2022

- ~100,000 CPU cores
- 160 NVIDIA A100 GPUs
- Interconnect: ~12.5GB/sec



Tier-1 in Flanders: not only compute



VLAAMS
SUPERCOMPUTER
CENTRUM



Vlaanderen
is supercomputing



<https://www.vscentrum.be/compute>

<https://www.vscentrum.be/data>

<https://www.vscentrum.be/cloud>

Tier-1 and Tier-2 in Flanders: VSC

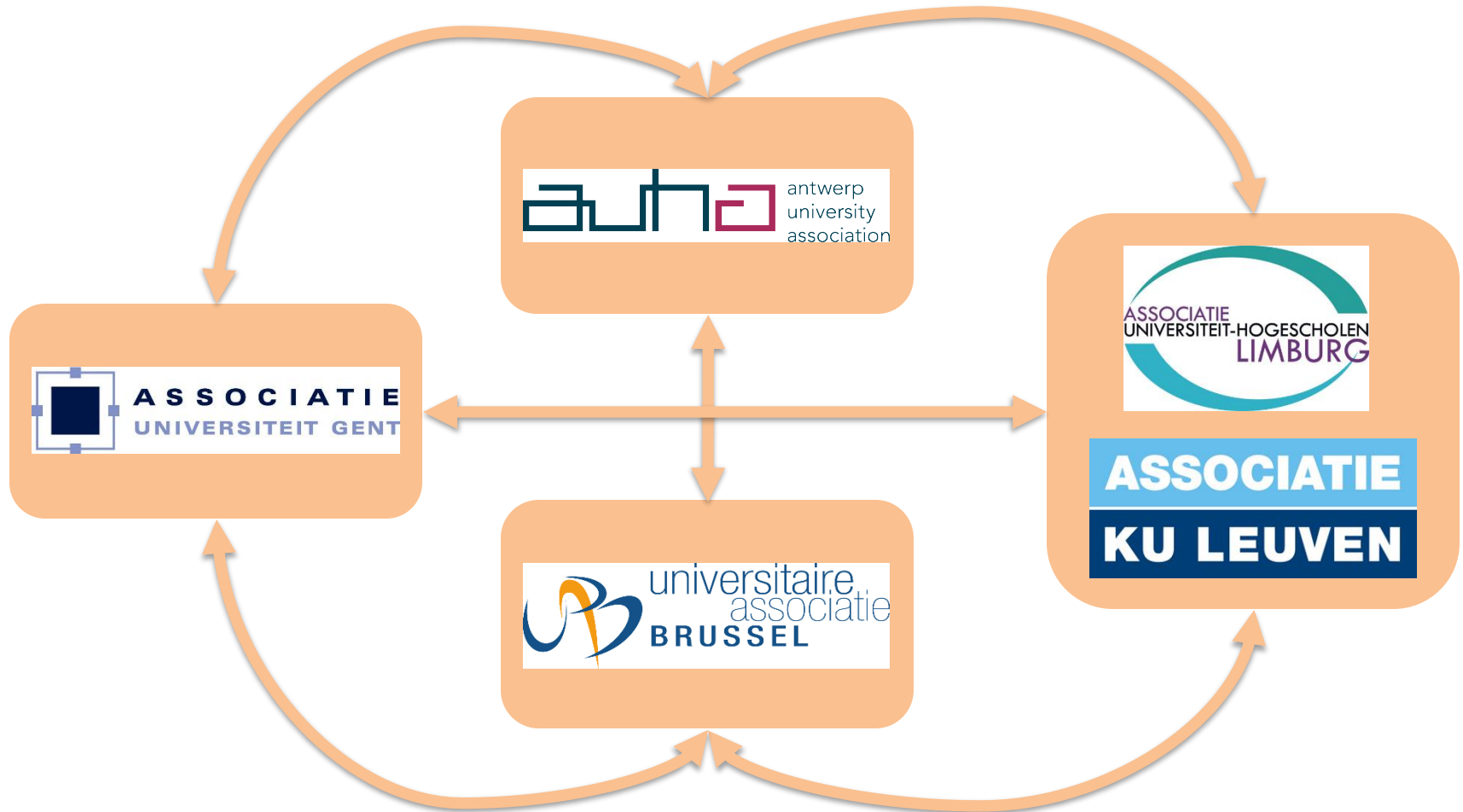


VLAAMS
SUPERCOMPUTER
CENTRUM



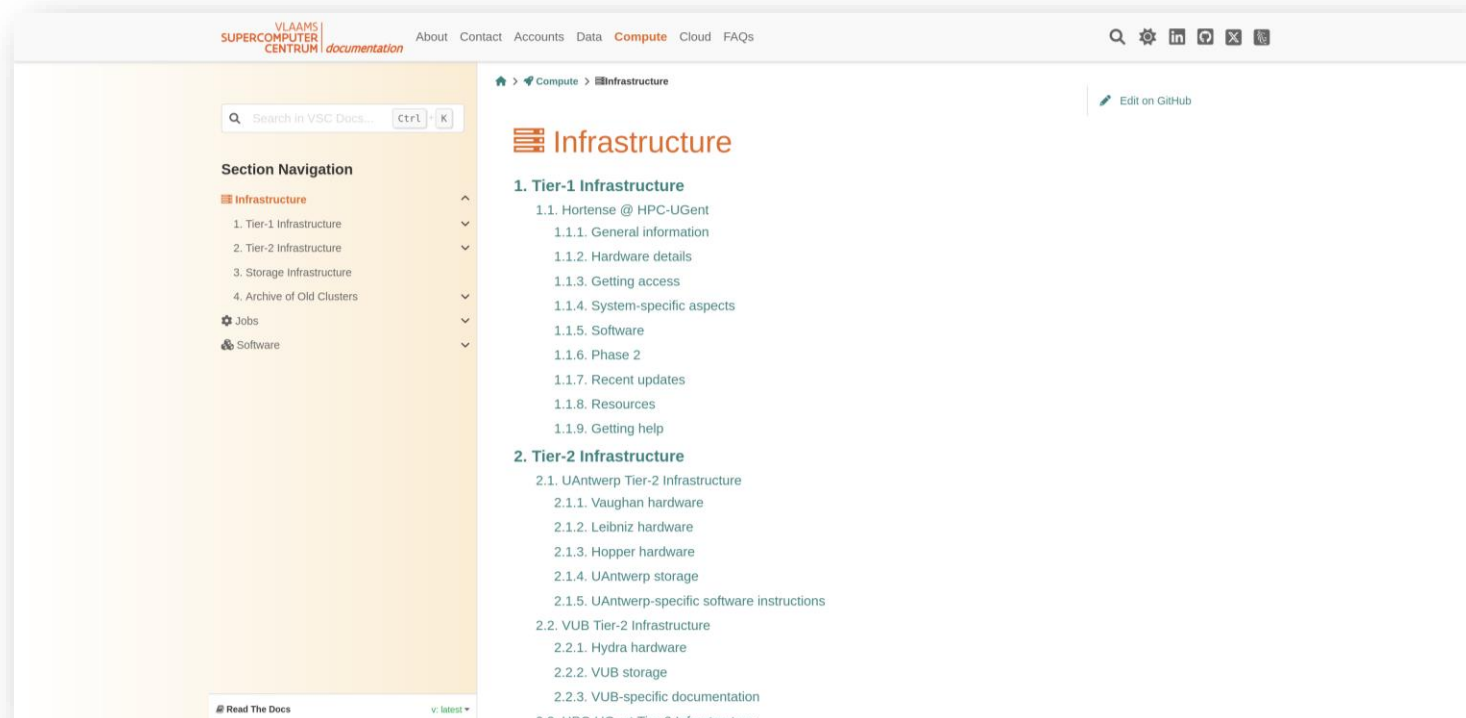
Vlaanderen
is supercomputing

Interconnected HPC hubs



Tier-1 and Tier-2 in Flanders: hardware details

<https://docs.vscentrum.be/hardware.html>



Say hello to Hydra



Hydra: a multi-headed serpentine water monster in Greek and Roman mythology. For every head chopped off, the Hydra would regrow a couple of heads.

Say hello to Hydra



Hydra: the multi-headed HPC cluster that eats your science problems for breakfast.

The SDC and VUB-HPC team

SDC: Scientific Data &
Compute

<https://hpc.vub.be/about/#the-sdc-team>

The VUB-HPC team will
fiercely protect you from
multi-headed monsters and
ugly water creatures (no
guarantees).

Ward Poelmans

- HPC Team Lead
- System Administrator
- VSC Operational Team

Background

Ph.D. Physics

Contact

✉ Ward.Poelmans@vub.be

Samuel Moors

- User Support
- Research Software Engineer
- HPC Teaching

Background

Ph.D. Chemistry

Contact

✉ Samuel.Moors@vub.be

Alex Domingo

- User Support
- Research Software Engineer
- System Architect

Background

Ph.D. Computational Chemistry

Contact

✉ Alex.Domingo.Toro@vub.be

Stéphane Gerard

- Platform Development
- Member of IIHE

Background

Master in Physics

Contact

✉ Stephane.Gerard@vub.be

Eric Luyten

- Object storage manager
- Object storage user support

Background

Master in Computing Science

Contact

✉ Eric.Luyten@vub.be

Cintia Willemyns

- Non-academic user support

Background

Ph.D. Physics

Contact

✉ Cintia.Willemyns@vub.be

Carl Van Poyer

- Object storage manager
- Object storage user support

Background

Master in Electronics-ICT

Contact

✉ Carl.Van.Poyer@vub.be

Jan Turek

- Business developer
- Outreach

Background

Ph.D. Chemistry

Contact

✉ Jan.Turek@vub.be

Xavier Deraet

- Business developer
- EuroCC liaison officer

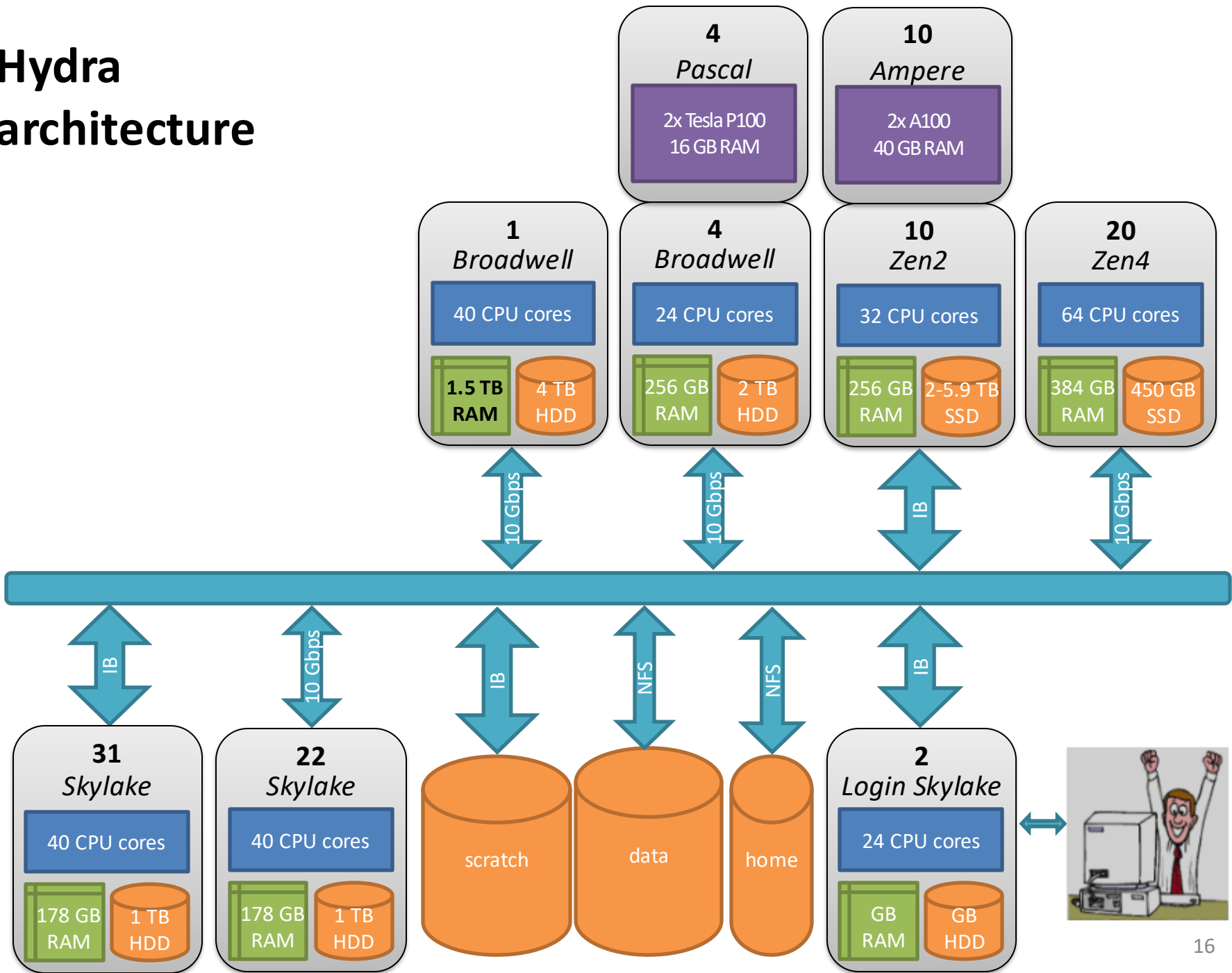
Background

Ph.D. Chemistry

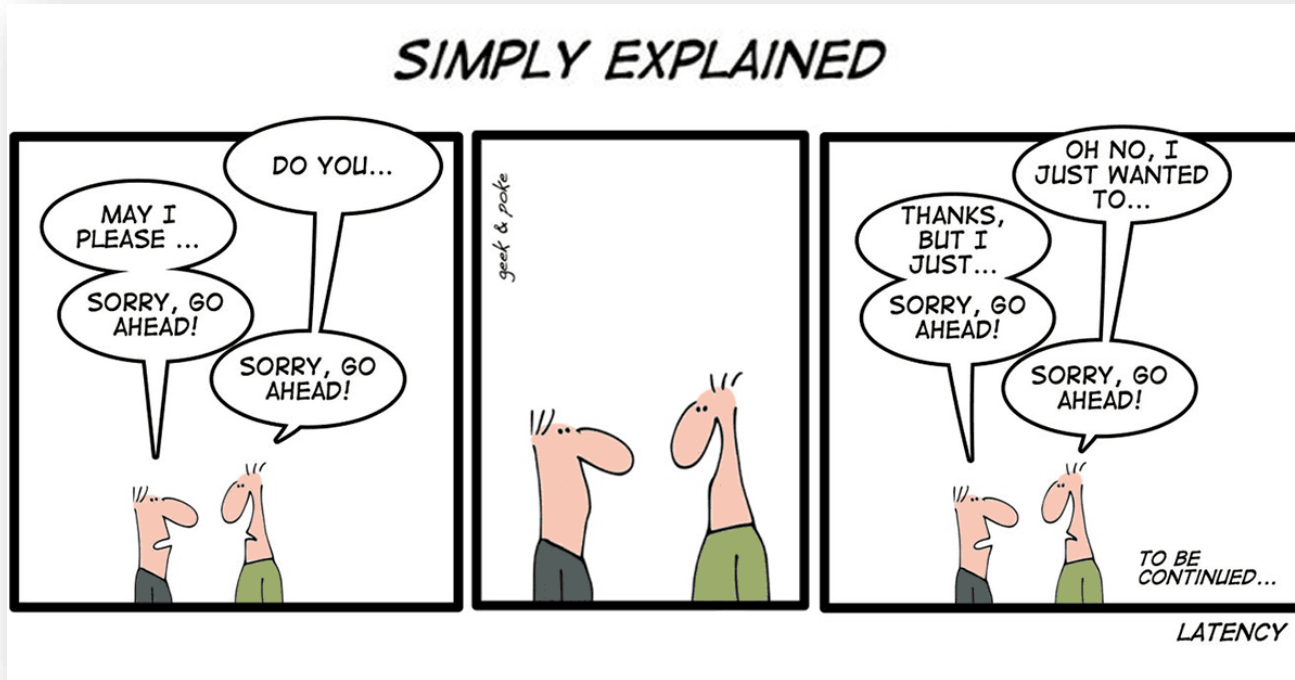
Contact

✉ Xavier.Deraet@vub.be

Hydra architecture



Some HPC concepts



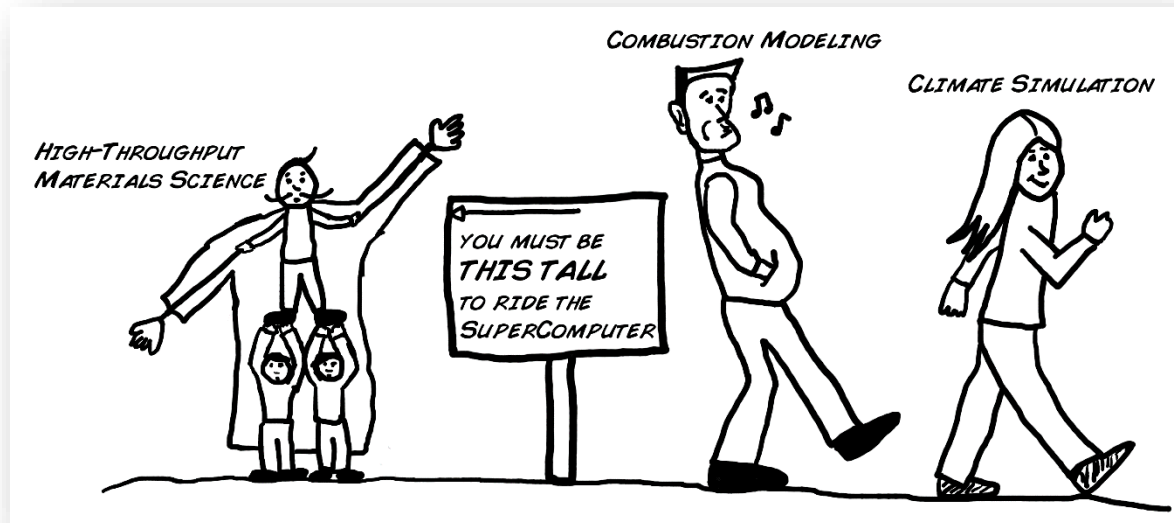
Running software in HPC: sequential vs. parallel execution

Sequential

- Single-node, single-core
- No scaling ...
- ...or 100% scaling efficiency (embarrassingly parallel)
- High-throughput computing: **HTC**

Parallel

- Divide large problem into smaller ones, which are solved **simultaneously** on multiple cores
- more difficult to code: communication, synchronization
- **scaling efficiency** depends on #threads, #processes



Why should I run my code in parallel?

1. Solve your problems faster
2. Solve bigger problems
 - Compute time limited problems
 - Memory limited problems
 - More cores, nodes -> more memory, higher total memory bandwidth

Parallel execution: multiprocessing vs multithreading

multiprocessing

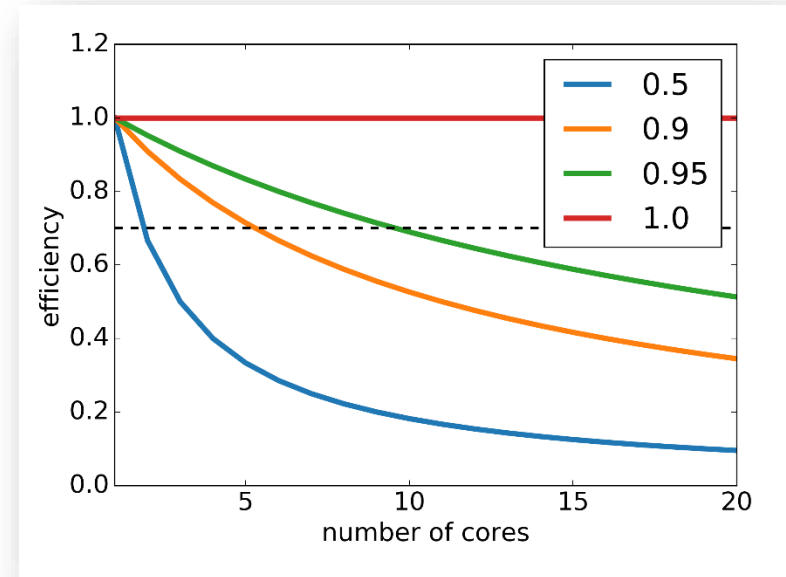
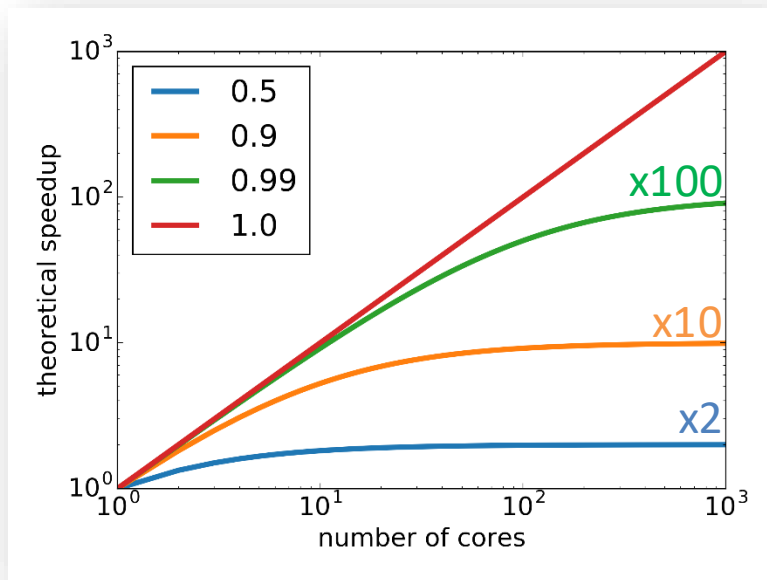
- process = stream of instructions being executed
- each process has its own process ID
- processes **do not share memory**
- processes communicate via system calls
- main types:
 - message passing: MPI, RPC, ...
 - sub-processing: Python multiprocessing, R doParallel, ...

multithreading

- thread = lightweight process (LWP)
- threads are created by parent process or parent thread
- each thread has its own thread ID
- threads **share memory** with each other
- methods: OpenMP, POSIX threads (pthreads), C++11 threads, thread building blocks (TBB), ...

Benchmarking Parallel efficiency: Amdahl's law (a.k.a. strong scaling)

Theoretical speedup is limited by the fraction of the task that can be parallelized

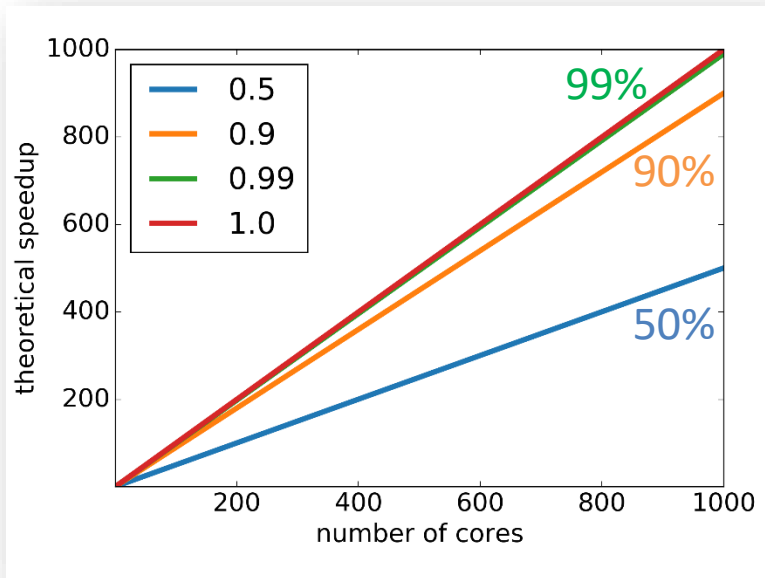


Parallel efficiency =
speedup / number of cores

- optimal efficiency: >70%

Benchmarking Parallel efficiency: Gustafson's law (a.k.a. weak scaling)

Theoretical speedup scales
linearly with number of cores



- Amdahl's law:
 - fixed problem size
 - reducing walltime
- Gustafson's law:
 - fixed walltime
 - increasing problem size

Working in Hydra: terminal vs. graphical interface

Terminal interface

- Typing commands in a shell (command line interface)
- Writing text/code in a text editor

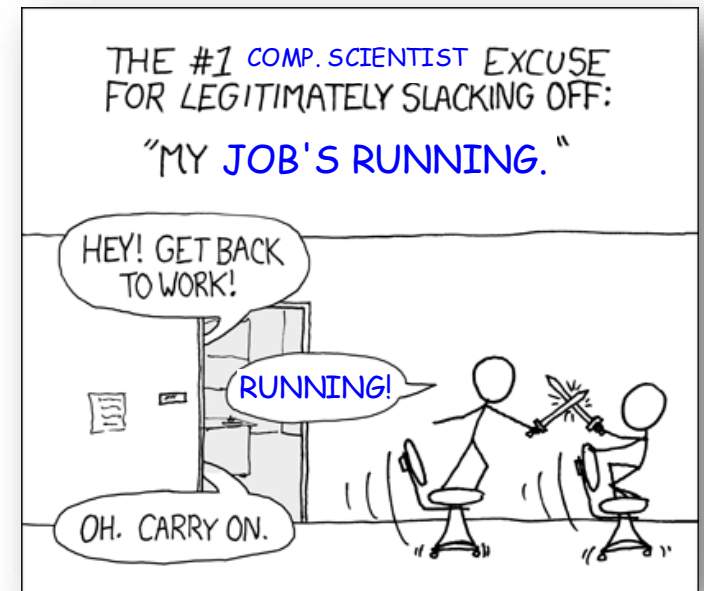
Graphical interface

- X11 forwarding (slow)
 - https://hpc.vub.be/docs/software/graphical_apps/
- Notebooks platform:
 - <https://notebooks.hpc.vub.be>
 - <https://hpc.vub.be/docs/notebooks/>
- Open OnDemand (OOD) Web portal:
 - <https://portal.hpc.vub.be>

A typical terminal based workflow

1. Connect to Hydra
2. Optionally: transfer input files to Hydra
3. Check if a module for your software is available
4. Create a batch job script
5. Submit job
6. Wait while ...
 - job sits in the job queue
 - job runs
 - job finishes
7. Optionally analyze results, transfer output files to different location

We'll guide you through the different steps in the following slides.



Connecting to Hydra (1/3)

Connecting via SSH client

- Linux, macOS: Terminal
- Windows:
 - MobaXterm (recommended)
 - home edition = free
 - File browser (upload/download)
 - X11 client, VNC client, ...https://docs.vscentrum.be/access/access_using_mobaxterm.html
 - Terminal via Windows subsystem for Linux (WSL)
= Linux VM integrated in Windows
 - PuTTY (open source)
https://docs.vscentrum.be/access/text_mode_access_using_putty.html

<https://hpc.vub.be/docs/login.html>

Connecting to Hydra (2/3)

Connecting via the OOD web portal

- Web interface
<https://portal.hpc.vub.be>
- Terminal on login node or in interactive job
- Light-weight graphical desktop environment (Xfce)
- File browser (upload/download)

[https://hpc.vub.be/docs/\(to be added\)/](https://hpc.vub.be/docs/(to be added)/)

Connecting to Hydra (3/3)

Connecting via the notebooks platform

- Web interface
<https://notebooks.hpc.vub.be>
- Terminal in interactive job
- File browser (upload/download)
- Jupyter notebooks
- VSCode-web

<https://hpc.vub.be/docs/notebooks/>

Transferring files to/from Hydra

Terminal-based transfer

- rsync, scp, sftp, sshfs (remote file access)

Web-based transfer

- OOD web portal, notebooks platform
- Globus (recommended for high volumes)

<https://hpc.vub.be/docs/data/management/#globus>

Transfer with graphical interface

- MobaXterm, Filezilla, FireFTP, WinSCP

https://docs.vscentrum.be/access/access_using_mobaxterm.html#copying-files-to-and-from-the-cluster

<https://hpc.vub.be/docs/data/management/#data-transfer>

<https://docs.vscentrum.be/data/transfer.html>

Using software on Hydra

Software modules

- Large collection of end-user software is provided to all users via modules (Lmod)
- How does it work?
 - preparing the environment for using the software
`$PATH`, `$LD_LIBRARY_PATH`, `$PYTHONPATH`, ...
- Why modules?
 - different versions of programs/libraries can be installed
 - easy loading and unloading
 - independent from your shell settings
 - dependencies are automatically loaded as well
 - scientific reproducibility:
 - each software module will always run the same executables and in the same environment

Compiler toolchains

- Toolchain = collection of tools to build and run software:
 - compilers for C/C++ and Fortran
 - a communications library (MPI)
 - mathematical libraries (linear algebra, FFT)
- Full toolchains:
 - **intel** Intel compilers (icc, ifort), Intel MPI, Intel MKL
 - **foss** GCC, OpenMPI, BLACS, OpenBLAS, FFTW, ScaLAPACK
 - **foss-CUDA** foss (with CUDA-aware OpenMPI) + CUDA
- Modules compiled with **intel** and **foss** should **not** be loaded together.
- Some programs run faster when compiled with **intel** (usually Fortran code), others with **foss**.

https://docs.vscentrum.be/en/latest/software/software_stack.html

My software module is not available: what now? (1/3)

Ask the HPC team to install it (recommended)

- Optimized for **performance** on all available CPU architectures
- Built **reproducibly** with EasyBuild
- Exact same version available for **all users**
- How to request?
 - Provide usage info:
why do you need it, which version, which toolchain, ...
 - Open source software (preferred):
provide link to software sources
 - Proprietary software:
provide necessary licenses, keys, software sources
- No guarantees: users should test, benchmark

My software module is not available: what now? (2/3)

Build/install/port software yourself

- Use an interactive job session for compiling in a node with the right CPU architecture, for example:

```
$ srun -p zen4 --pty bash -l
```
- Load a suitable **buildenv** module
 - loads compiler toolchain
 - defines environment variables for building with optimal performance and finding the right libraries
 - CFLAGS, FFLAGS, CXXFLAGS, LIBBLAS, LIBLAPACK, LIBFFT, LD_LIBRARY_PATH, LDFLAGS, ...
 - example :

```
$ module load buildenv/default-foss-2023a
```
- Test, benchmark!

https://hpc.vub.be/docs/software/additional_software/#compiling-and-testing-your-software-on-the-hpc

My software module is not available: what now? (3/3)

Use an Apptainer container (f.k.a. Singularity)

- Allows running programs in a highly controlled and isolated environment
- Less optimized → **not recommended for production runs**
- Ideal for quickly **testing new software** before creating an optimized build
- No root permissions required for running
- How to obtain an Apptainer image:
 - import Docker image
(e.g. from <https://hub.docker.com>)
 - create your own
- More info: <https://apptainer.org/docs/user/latest/>
- Contact HPC support for assistance: hpc@vub.be

<https://docs.vscentrum.be/software/singularity.html>

Running jobs in Hydra

- Running calculations on Hydra requires submitting **jobs** to the **job queue**
- Jobs are handled by **Slurm** workload manager
<https://slurm.schedmd.com/>

Slurm provides a suite of commands for:

- submitting jobs
- altering properties of waiting jobs (reordering, deleting)
- monitoring job progress
- killing problematic or no longer needed jobs
- checking job resource usage

The queue system

- The Slurm Job scheduler decides which job will start next
- Job priority is determined by **fair-share** policy:
 1. historical usage
 - aim = balancing usage between users:
 - infrequent users get higher priority
 - (recent) frequent users get lower priority
 2. requested resources (nodes, cores, time limit, memory)
 - more resources => lower priority
 - exception: multi-node > multi-core > single-core
 3. queue waiting time
 - queued jobs get higher priority over time
- User limits: avoids that a single user fills up an entire cluster
- No guarantees on when job will start, so plan ahead!

Maximum time limit in Hydra

- max time limit: 5 days
 - my job needs more time, what to do:
 - request more cores?
 - request more memory?
 - request faster nodes?
 - Use GPUs?
 - check restarting options of the software
 - use DMTCP checkpointing? (no MPI support)
<https://hpc.vub.be/docs/job-submission/restart-checkpoint/#checkpoints-with-dmtcp>

<https://hpc.vub.be/docs/faq/troubleshooting/#how-can-i-run-a-job-that-takes-longer-than-the-time-limit>

Can I use Hydra for personal data?

As backup storage for your own personal data

- Answer: NO

Analyzing personal research data

- Answer: it depends
- Data protection officer: dpo@vub.be
- Research data management: dmp@vub.be
- More info: <https://hpc.vub.be/docs/data-protection/>

Acknowledgements

Acknowledgements

- It is important for the funding of the VSC clusters that you acknowledge the VSC in all your scientific publications:

The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

[https://docs.vscentrum.be/how do i acknowledge the vsc in publications.html](https://docs.vscentrum.be/how_do_i_acknowledge_the_vsc_in_publications.html)