

Introduction to HPC-UGent

18 March 2026

GHENT UNIVERSITY

ugent.be/hpc

docs.hpc.ugent.be

hpc@ugent.be

VLAAMS SUPERCOMPUTER CENTRUM

Vlaanderen is supercomputing

Agenda

- [10:00 - 12:00] *Introduction to HPC-UGent* presentation + Q&A
  - Overview of available hardware, getting a VSC account,
    using the HPC-UGent Tier-2 clusters, getting support, demos and examples, …

Only for on-site attendees

- [12:00 - 13:00] Sandwich lunch
- [13:00 - 14:00] Guided tour of UGent datacenter 10,
  incl. visit to HPC-UGent Tier-2 and VSC Tier-1 cluster
- [14:00 - 17:00] Hands-on session: Getting started with HPC-UGent
  - Login + submitting example jobs
  - Getting started with your own workloads + Q&A

(Kenneth)

Documentation

- **Extensive documentation on using the HPC-UGent infrastructure is available**
- https://docs.hpc.ugent.be
- We will specifically use information from these sections:

Introduction to HPC

Getting an HPC account

Connecting to the HPC infrastructure

Running batch jobs

Running jobs with input/output data

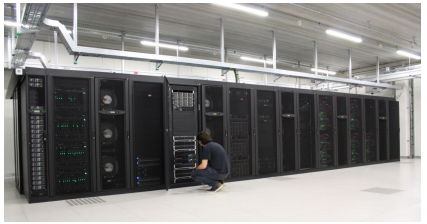
Using the HPC-UGent web portal

Fine-tuning job specifications

Interactive and debug cluster

(Kenneth)

HPC-UGent in a nutshell



- Part of central UGent Functional Domain ICT (formerly DICT)
- Our mission:

  *HPC-UGent provides centralised **scientific computing** services, training, and support for researchers from Ghent University, industry, and other knowledge institutes.*

- Our core values:

  Empowerment - Centralisation - Automation - Collaboration

GHENT UNIVERSITY

(Kenneth)

The HPC-UGent teams: HPC operations

Wouter Depypere
Team lead

Álvaro Simón García
System administration (cloud)

Kenneth Waegeman
System administration (storage)

Jonathan De Loght
System administration (cloud)

Andy Georges
System administration

Stijn De Weirdt
AI

(Kenneth)

The HPC-UGent team: HPC support

Ewald Pauwels
Team lead

Balázs Hajgató
User support

Eliza Depoorter
User support

Godfried Borremans
User support/Documentation

Kenneth Hoste
User support & training
Software Installations
EuroHPC projects

Lara Peeters
Project employee
scientific software

Pavel Tomanek
Software installations

(Kenneth)

What is High-Performance Computing (HPC)?

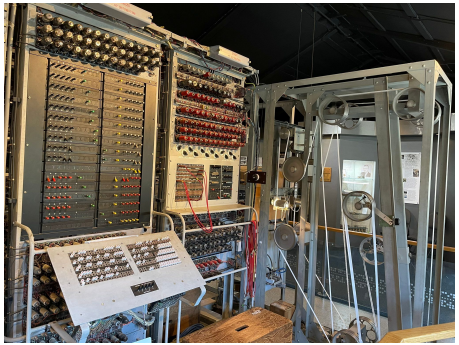
- **High Performance Computing (HPC)** is running computations on a supercomputer, a system at the frontline of contemporary processing capacity – particularly in terms of size, supported degree of ***parallelism***, network interconnect, and (total) available memory & disk space.
- A **computer cluster** consists of a set of loosely or tightly connected computers (also called (worker)nodes) that work together so that in many respects they can be viewed as a single system.
- HPC is also known as “supercomputing”, or more broadly “scientific computing”

(Lara)

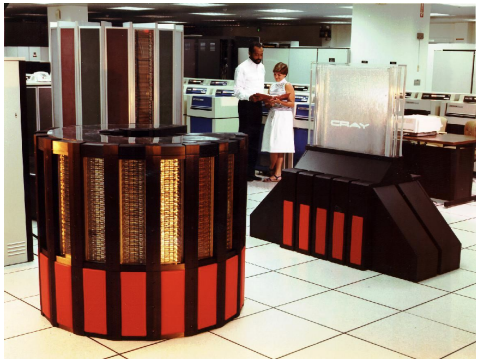
What is High-Performance Computing (HPC)?

(Lara)

harnessing the power of multiple interconnected cores/nodes/processing units



Colossus - first digital computer (1944)



Cray-2 supercomputer (1985)

"Rack" with desktop PCs in a basement

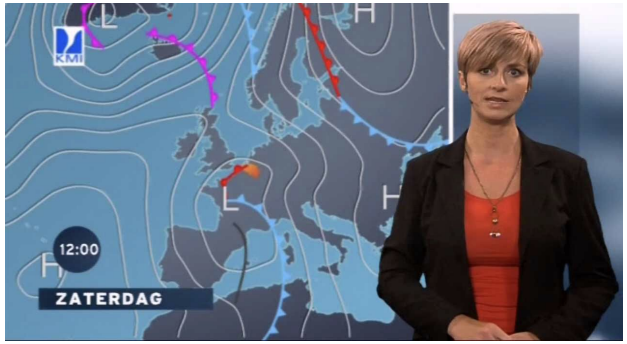
HPC-UGent Tier-2 infrastructure (2018)

IBM Blue Gene/P in Argonne National Lab (2007)

MareNostrum 4 in Barcelona (2017)

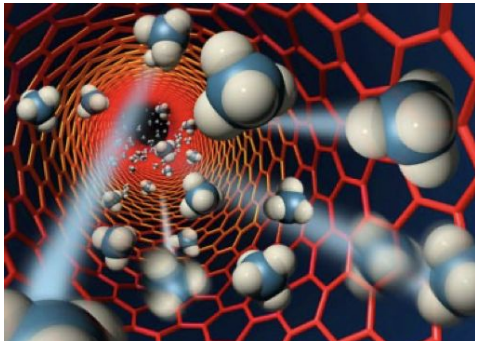
What are supercomputers used for?

(Lara)

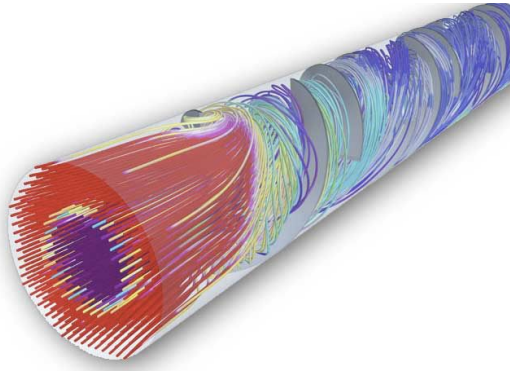


Weather prediction & modelling

Animation rendering



Molecular modelling, materials research, …



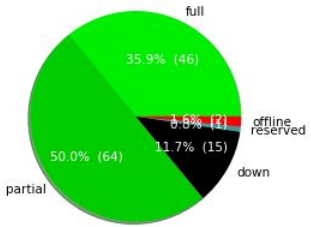
Computational Fluid Dynamics (CFD)
Aerodynamics, studying flow of gasses & liquids in volumes, etc.

***Simulation** of atomic weapons, etc.*

Terminology: (worker)nodes

- Example cluster from the HPC-UGent Tier-2 infrastructure: doduo *(current default cluster)*
- 128 **(worker)nodes**, also referred to as “servers”
- 1 (worker)node is the equivalent of 1 computer (but with more cores, memory, faster network, …)
- Check other HPC-UGent Tier-2 clusters

doduo

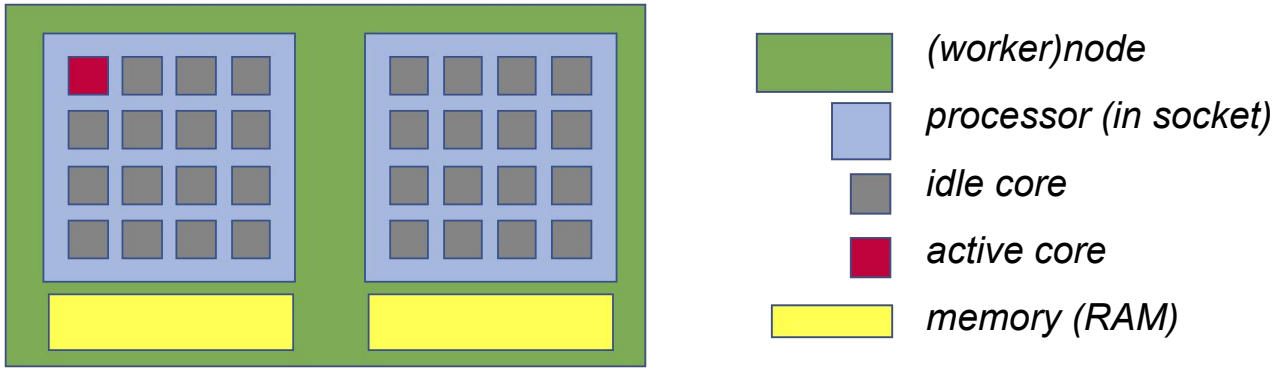


(Lara)

Terminology: cores, CPUs, processors

Modern servers, also referred to as **(worker)nodes** in the context of HPC, include one or more *sockets*, each housing a **multi-core processor** (next to memory, disk(s), network cards, …). A modern (micro)processor consists of **multiple cores** that are used to execute computations.

Example:
a single workernode with two 16-core processors running a single core job

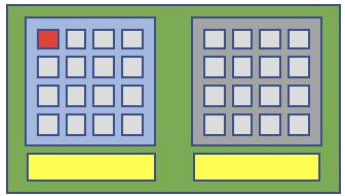


Not shown here: local disk, network cards, GPUs, …

(Lara)

Parallel vs **sequential** software (single-core)

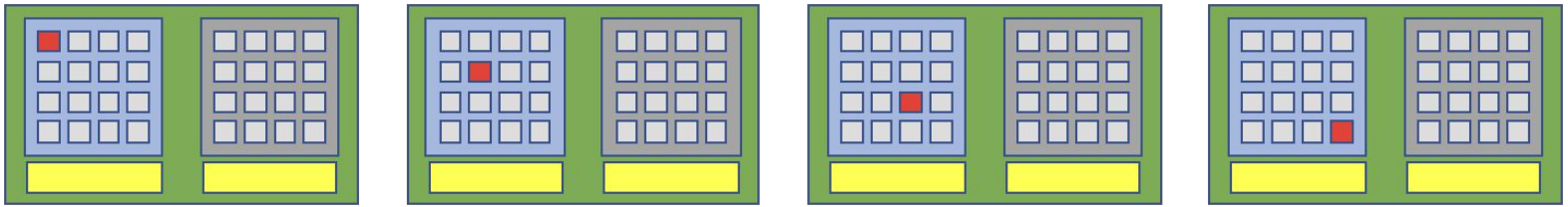
Sequential (a.k.a. serial) software does not do calculations in parallel, it only uses one **single core** of a single workernode.



This type of software does not run faster by just throwing cores (or nodes) at it...

But, you can run multiple instances of the same program at the same time!

Example: running a Python script 100 times, each on 1 core, to quickly analyse 100 datasets



(Lara)

Parallel computing

- Long-running calculations can be done faster through **parallelization**
- Split up large problem into smaller problems, divide and conquer
- Challenge is to **keep all cores busy** (doing useful work)
  - Avoid bottlenecks: slow memory/disk, network latency, limited bandwidth
  - Avoid load imbalance: waiting for longest running part to be done
  - Avoid overhead (extra work) in breaking up problem + composing final result

(Lara)

Parallel vs sequential software (single-node or multi-node)

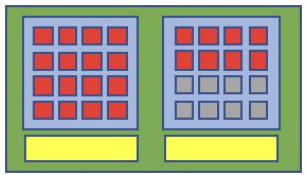
In **parallel** software, many calculations are carried out simultaneously.
This is based on the principle that large problems can often be divided into smaller tasks, which are then solved concurrently (“in parallel”).

Example: OpenFOAM can easily use 160 cores at the same time to solve a CFD problem.

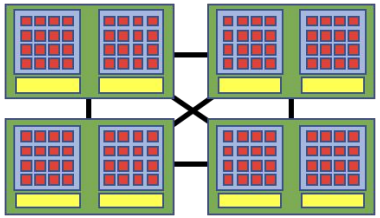
There are two common parallel programming paradigms (among others):

- **OpenMP** for shared memory systems (multi-threading) → using cores of a *single* node
- **MPI** for distributed memory systems (multi-processing) → using cores of *multiple* nodes

OpenMP software can use multiple or all cores in a single node



MPI software can use (all) cores in multiple nodes



(Lara)

Centralised hardware in UGent datacenter (S10 @ Sterre)

(Lara)

Different “tiers” of supercomputers

eurohpc-ju.europa.eu

EURO

EuroHPC
Joint Undertaking

Tier-0
Europe

Tier-1
Regional / National

Tier-2
University

Tier-3
Personal workstation (desktop, laptop)

VLAAMS
SUPERCOMPUTER
CENTRUM

Vlaanderen
is supercomputing

vscentrum.be

GHENT
UNIVERSITY

ugent.be/hpc

(Kenneth)

HPC-UGent Tier-2 infrastructure

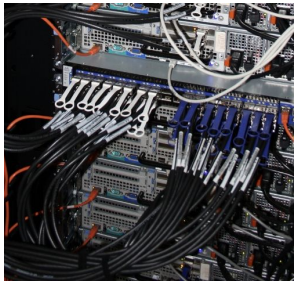
- HPC-UGent Tier-2 infrastructure currently consists of **7 clusters** (+ login nodes, shared storage, …)
- Different types of clusters:
  - 3 CPU-only compute clusters (+ one in pilot mode)
  - 3 GPU clusters
  - 1 CPU-only interactive + debug cluster (oversubcribed resources + strict user limits)
- **Available for academic researchers free of charge**, funding through FWO; usage by industry via a pay-as-you-use contract (after free exploratory period)
- All Tier-2 clusters are running Red Hat Enterprise Linux 9 (RHEL9) as operating system

HPC-UGent Tier-2 compute clusters

- `doduo`: 128 nodes, each with 96 cores (AMD Zen 2Rome) + ~250GiB of memory **(default)**

- `gallade`: 16 nodes, each with 128 cores (AMD Zen3 Milan) + ~940GiB of memory
- `shinx`: 48 nodes, each with 192 cores (AMD Zen4 Genoa) + ~370GiB of memory

- `skiddo`: 28 nodes, each with 256 cores (AMD Zen5 Turin) + ~710GiB of memory **(pilot)**
- All with:
  - High-speed Infiniband network between nodes
  - Fast access to shared file systems
  - Fast local disk (SSD or NVMe)



docs.hpc.ugent.be/infrastructure

(Kenneth)

HPC-UGent Tier-2 GPU clusters



- `joltik`: 10 nodes,
  each with 32 CPU cores (Intel Cascade Lake),
  4 NVIDIA V100 GPUs (32GB of GPU memory),
  ~250GB of system memory
- `accelgor`: 9 nodes,
  each with 48 CPU cores (AMD Zen3 Milan),
  4 NVIDIA A100 GPUs (80GB of GPU memory),
  ~500GB of system memory
- `litleo`: 8 nodes,
  each with 48 CPU cores (AMD Zen4 Genoa),
  2 NVIDIA H100 GPUs (96GB of GPU memory),
  ~315GB of system memory

- All with high-speed network, fast access to shared filesystems, fast local disk (SSD)

docs.hpc.ugent.be/infrastructure

(Kenneth)

HPC-UGent Tier-2 interactive + debug cluster: donphan

- 16 nodes, each with 36 CPU cores (Intel Cascade Lake) + ~738GB of memory

  1 *shared* NVIDIA Ampere A2 GPU (16GB of GPU memory)
- Incl. high-speed network, fast access to shared storage, local disk (NVMe)
- Recycled hardware from old `kirlia` cluster (retired in May 2023)
- **Heavily oversubscribed!** More running jobs => All jobs run slower (due to CPU sharing)
- **Strict user limits:**
  - Max. 3 jobs running, 5 jobs in queue
  - Max. 8 cores + 27GB of memory in use (in total)
- ⇒ **No waiting time for jobs to start**! Perfect for debug jobs, or interactive use (web portal)
- See also dedicated section in HPC-UGent documentation

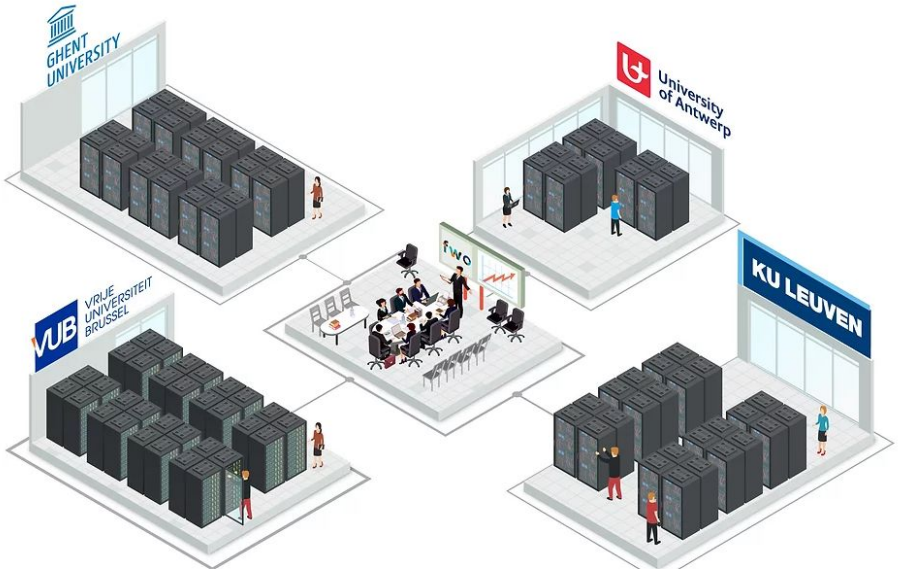
docs.hpc.ugent.be/infrastructure

(Kenneth)

VSC Tier-2 infrastructure

VLAAMS SUPERCOMPUTER CENTRUM
Vlaanderen is supercomputing

- You can use your VSC account to access HPC infrastructure provided by other VSC hubs
- Your `$VSC_HOME` and `$VSC_DATA` directories are available on each of these systems



docs.vscentrum.be/en/latest/hardware.html

(Kenneth)

VSC Tier-1 compute cluster “Hortense”

(a.k.a. `dodrio`)

VLAAMS SUPERCOMPUTER CENTRUM

Vlaanderen is supercomputing

compute@vscentrum.be

- Hosted, operated, and supported by HPC-UGent team since 2021
- 2x 384 CPU-only nodes (128-core AMD Rome or Milan CPUs) + 40 GPU nodes (4x NVIDIA A100)
- **Over 100,000 CPU cores in total!**
- High-speed Infiniband network (HDR-100) + 6PB of dedicated scratch storage

- **Project-based access** (free of charge, funded by FWO)
- 3 cut-off dates per year for submitting project proposals
- Project duration is typically 8 months
- 500k - 5M core hours (CPU-only) or 1k - 25k GPU hours

vscentrum.be/compute

docs.vscentrum.be/en/latest/gent/tier1_hortense.html

(Kenneth)

Next VSC Tier-1 compute cluster

VLAAMS SUPERCOMPUTER CENTRUM
Vlaanderen is supercomputing

compute@vscentrum.be

- Coming soon (fall 2025) at Green Energy Park in Zellik
- Operated by Vrije Universiteit Brussel (VUB)
- Investment of 8.6 million Euro
- ~25,000 CPU cores (with 2GB or 8 GB RAM per core)
- 168 NVIDIA H200 GPUs
- NVIDIA Quantum-2 InfiniBand network at 200 Gbps
- 4 visualisation nodes
- 4460 TB of scratch storage + 24 TB NVME
- Peak power consumption of 315 kW,200 kW water-cooled for improved energy efficiency

vscentrum.be/compute
vscentrum.be/post/vsc-inaugurates-sofia-the-new-flemish-tier-1-supercomputer

(Kenneth)

Next VSC Tier-1 compute cluster

VLAAMS SUPERCOMPUTER CENTRUM
Vlaanderen is supercomputing

compute@vscentrum.be

(Kenneth)

VSC Tier-1 cloud

VLAAMS SUPERCOMPUTER CENTRUM
Vlaanderen is supercomputing

- **Project-based access**
- Free of charge
- **Self-managed virtual machines**
- For use cases that are not a good fit for compute clusters
- More info: vscentrum.be/cloud
- Contact: cloud@vscentrum.be

(Kenneth)

Getting a VSC account

VLAAMS SUPERCOMPUTER CENTRUM
Vlaanderen is supercomputing

- All members of UGent association can request a VSC account
  - Researchers & staff
  - Master/Bachelor students
- **VSC account can be used to access HPC infrastructure on all VSC sites**
- Subscribed to `hpc-announce` and `hpc-users` mailing lists
- Beware of using HPC for teaching/exam purposes!
  - No guarantee on HPC availability (due unexpected power outage, maintenance, …)
  - Have a backup plan at hand
  - Advisable teaching/exam formula: project work
- See also HPC-UGent documentation

ugent.be/hpc/en/access/faq/access

(Lara)

Managing your VSC account

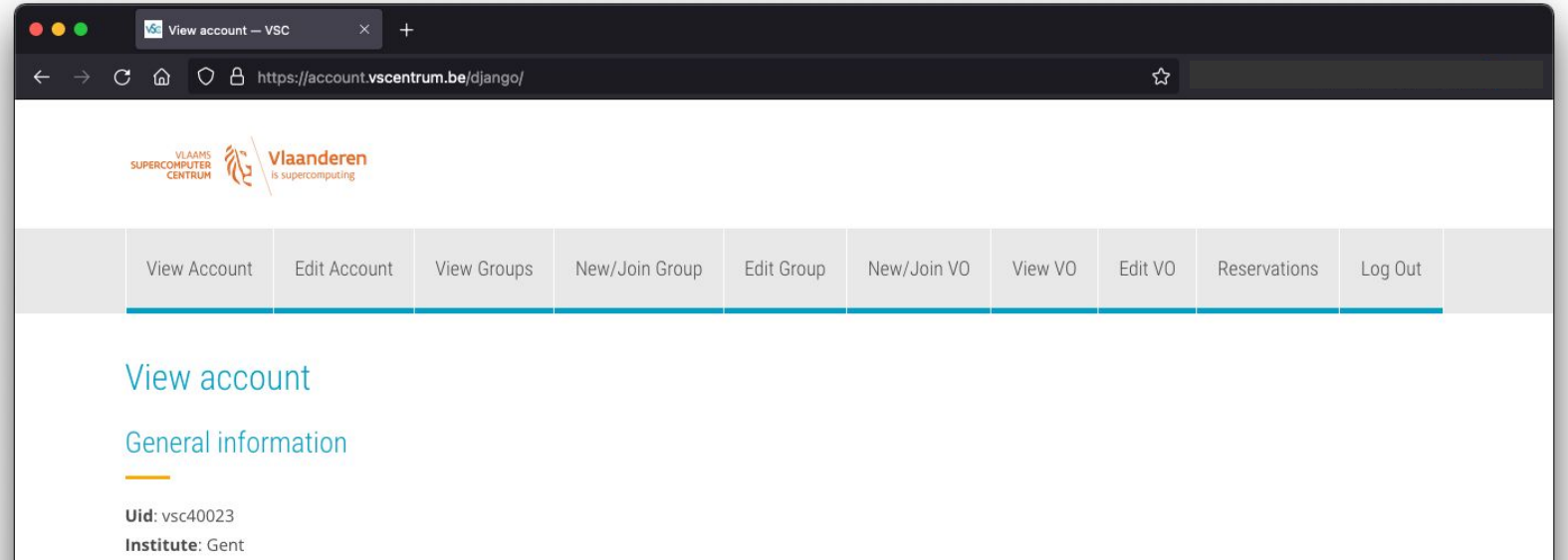
VLAAMS SUPERCOMPUTER CENTRUM
Vlaanderen is supercomputing

You can manage your VSC account via the VSC account page

account.vscentrum.be

Can be used to join/leave user groups, consult storage usage, request more storage quota, …

manage your Virtual Organisation (VO), …



(Lara)

Workflow on HPC-UGent infrastructure

1. Connect to login nodes
2. Transfer your files
3. (Compile your code and test it)
4. Create a job script
5. Submit your job
6. Be patient
   - Your job gets into the queue
   - Your job gets executed
   - Your job completes
7. Inspect and/or move your results

(Lara)

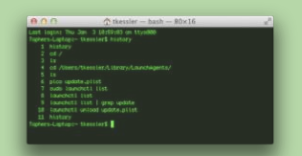
High-level overview of HPC-UGent infrastructure

Getting access

https://login.hpc.ugent.be
Web Portal (No SSH key required)

OR

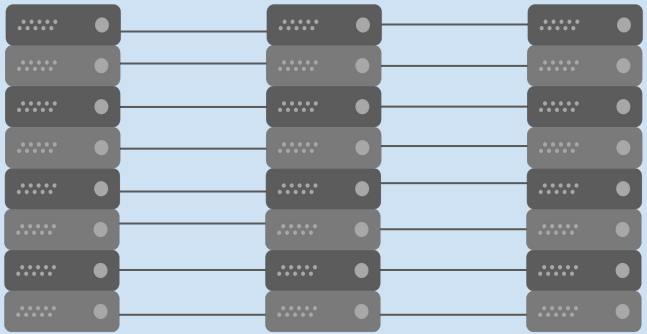
ssh login.hpc.ugent.be
Terminal (requires SSH key)



Login nodes

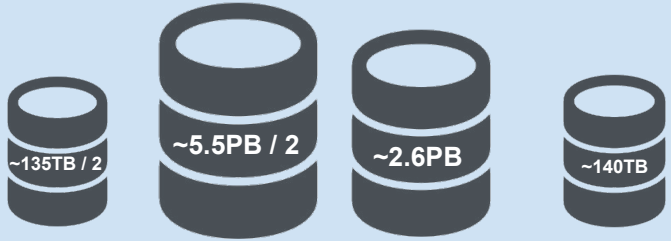
gligar09 or gligar10

Compute nodes (where your jobs will run)



Data

File system



Home | Data | Scratch | Arcanine Scratch (SSD)

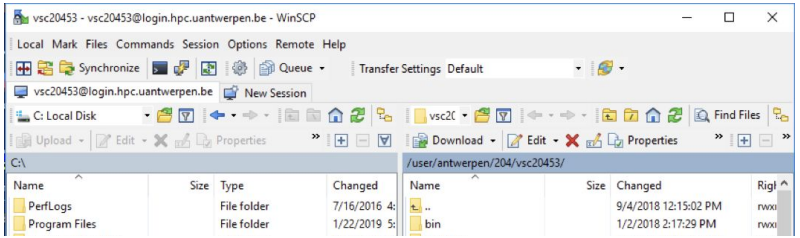
(Lara)

Option 1: Connecting to the HPC-UGent login nodes with SSH

```
$ ssh vsc40023@login.hpc.ugent.be

STEVIN HPC-UGent infrastructure status on Thu, 17 Nov 2022 20:30:01
     cluster     - full - free -  part - total - running - queued
                   nodes  nodes   free   nodes   jobs      jobs
-------------------------------------------------------------------
      slaking          0      6      2      10          3         4
      swalot         19       0     94     118       1659      1172
      skitty         33       0     33      72       1391       439
     victini          7       0     87      96        588     30080
      joltik          3       1      6      10         26       833
      kirlia          7       0      9      16         23        40
        doduo        50       0     67     128       1983     12695
    accelgor          6       0      2       9         31      1267

Documentation is available at:
 https://www.ugent.be/hpc/en/support/documentation.htm
For a full view of the current loads and queues see:
 https://hpc.ugent.be/clusterstate/
Updates on current system status and maintenance can be found on:
 https://www.ugent.be/hpc/en/infrastructure/status
To contact the support team, send an email to hpc@ugent.be

Last login: Thu Nov 17 20:30:34 2022 from 10.141.10.60
[vsc40023@gligar07 ~]$
[vsc40023@gligar07 ~]$ hostname
gligar07.gastly.os
```

- See dedicated section in HPC-UGent documentation
- `login.hpc.ugent.be`
- Requires SSH client + SSH private key
- Windows: PuTTy - macOS/Linux: `ssh` command

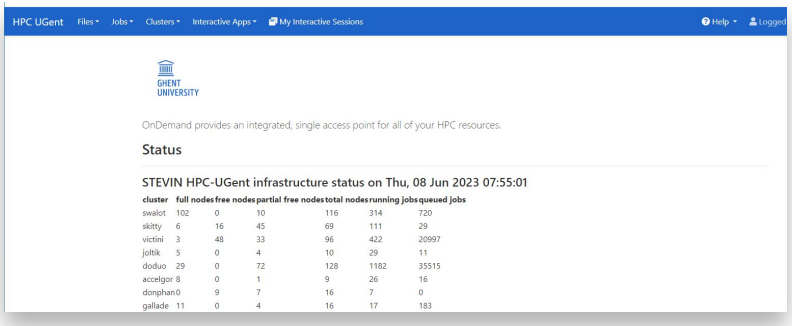
- Transferring files to/from HPC-UGent infrastructure
  - **Done via the login nodes**
  - Options:
    - On Linux or macOS:
      - Using `scp` or `rsync` command in terminal window
      - Using a graphical like the built-in file manager or Cyberduck
    - On Windows: using WinSCP (left: own system, right: HPC; drag-and-drop)



(Lara)

Option 2: Connecting to the HPC-UGent login nodes with web portal

Recommended!



- See dedicated section of HPC-UGent docs
- https://login.hpc.ugent.be
- Powered by Open OnDemand

- **Works with a standard internet browser** (Firefox, Chrome, …)
- **Does not require SSH key pair** (only login via UGent account)
- Provides file browser, shell session, desktop environment, interactive apps, …

(Lara)

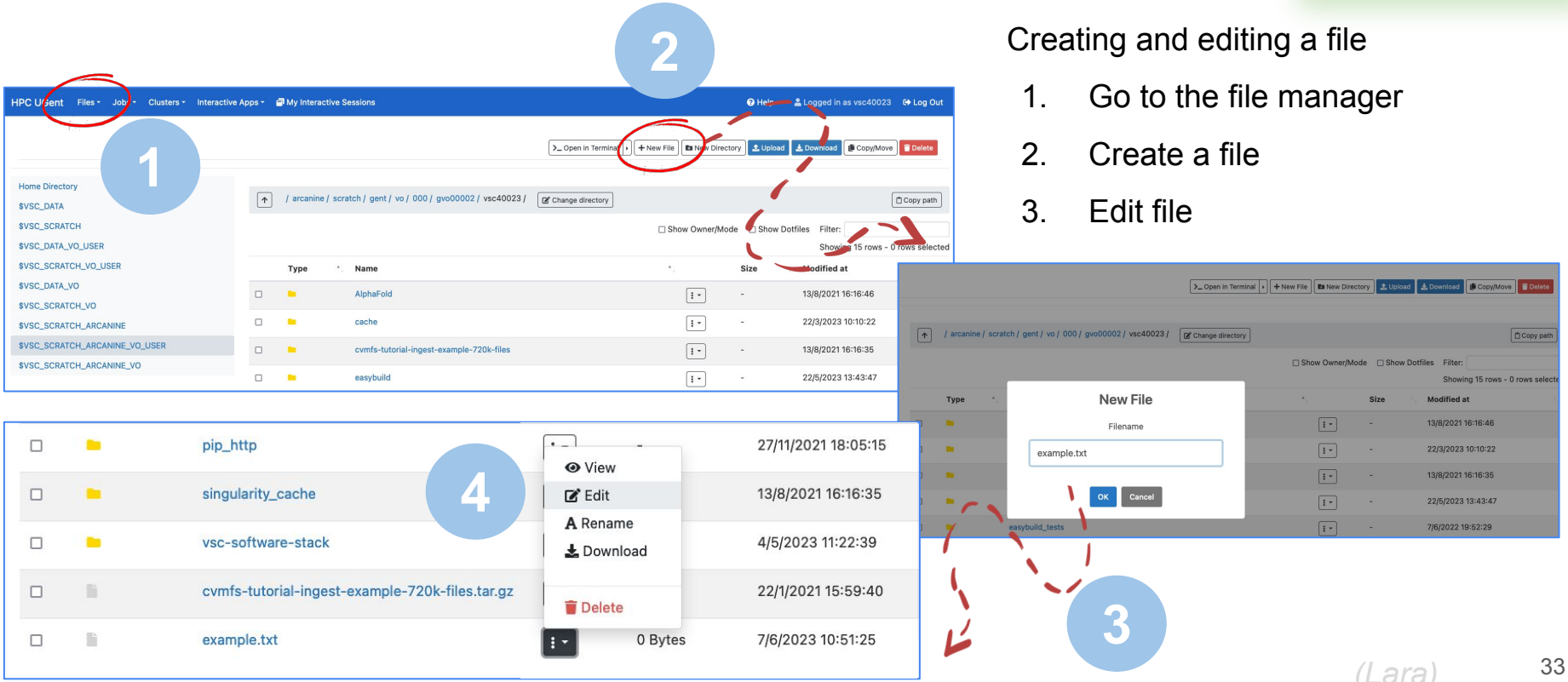
Getting shell access via web portal

1. Click on the ‘Clusters’ tab
2. Click on Login Shell Access
3. Use the shell environment

(Lara)

Option 2: Using the web portal file browser to view, edit, manage files

Recommended!



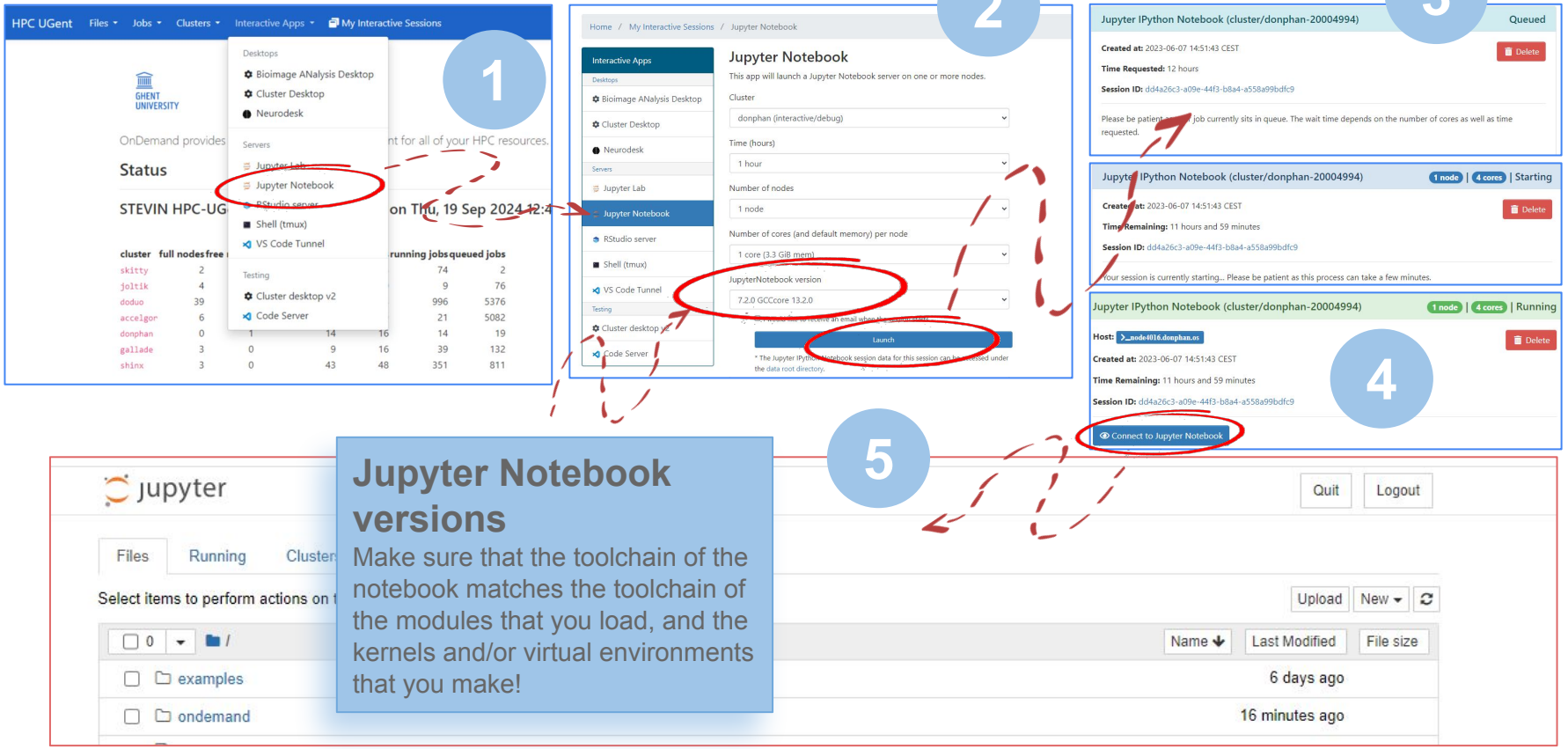
Creating and editing a file

1. Go to the file manager
2. Create a file
3. Edit file

(Lara)

UGent web portal: interactive apps (Jupyter notebook)

(Lara)

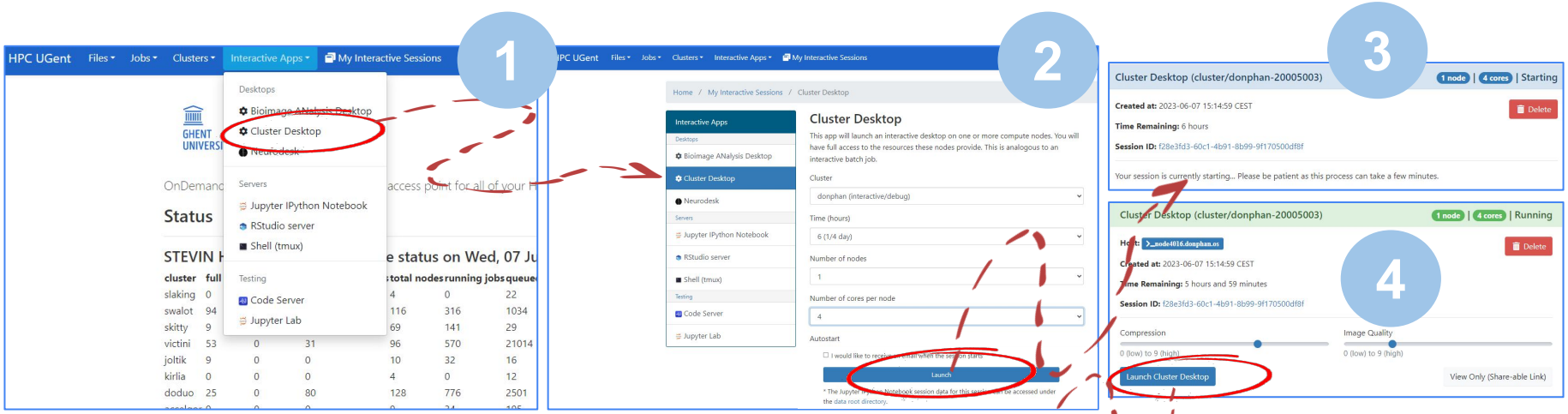


Jupyter Notebook versions

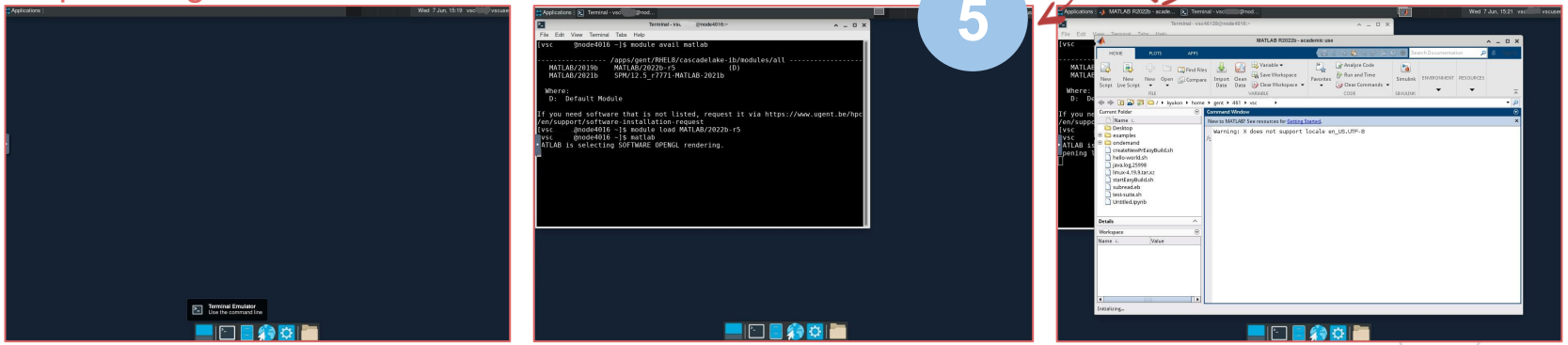
Make sure that the toolchain of the notebook matches the toolchain of the modules that you load, and the kernels and/or virtual environments that you make!

(Lara)

UGent web portal: interactive apps (desktop for GUI apps)



Example Using MATLAB



Connection restrictions

For security reasons, some connection restrictions have been put in place.

Connecting to the HPC-UGent login nodes is only possible when if one of the following applies:

- Using a university network (WiFi in UGent building, UGent VPN, …)
- Using SSH + a Belgian commercial internet provider (take this into account when you're travelling!)
- Your IP address has been whitelisted
  - Automatically (and temporary) via the VSC firewall app: https://firewall.vscentrum.be
  - By exception (for example for corporate networks)

You need to connect to the firewall app in new tab and wait up to 30s.

Keep the tab open while you are connected.

GHENT UNIVERSITY

Sign in

Email

Can't access your account?

Back Next

Authorize hpc-firewall?

Application requires following permissions

- Read scope

Cancel Authorize

(Lara)

Linux command line interface (shell)

```
$ mkdir hpc_demo
$ cd hpc_demo
$ ls
$ echo science > results.txt
$ ls -l
total 1
-rw-rw-r-- 1 vsc40023 vsc40023 8 Nov 17 20:57 results.t
$ cat results.txt
science
$ echo $VSC_DATA
/data/gent/400/vsc40023
$
```

- **Linux shell environment** is standard way of using HPC systems
- Involves typing to run shell commands, or using (bash) scripts
- Example commands: `ls`, `cd`, `mkdir`, `cp`, `mv`, `rm`, `export`, `echo`, …
- Commands can be “piped” together to do more complex operations
- May feel arhaic, but is actually **very powerful**…
- Same scripting language (bash) is used in job scripts
- **Learning the basics of the Linux shell is strongly recommended!**
- See separate basic Linux tutorial at docs.hpc.ugent.be/linux-tutorial

(Lara)

Submitting and managing jobs on HPC-UGent clusters

- HPC-UGent clusters run Slurm as resource manager + job scheduler
- **Torque (PBS) frontend is (still) available and recommended** (via *jobcli* project)
  - `qsub` command to submit jobs, `qdel` command to delete jobs
  - `qstat` command to list queued + running jobs
  - `qalter` command to change jobs (before they start running)
  - `qhold` command to put jobs on hold, qrls to release them again
- Use `--help` option to get list of available options for each command
- Use `--debug` option to get more information about what's going on behind the scenes
- Use `--dryrun` option to inspect what will be done (without actually doing it)

docs.hpc.ugent.be/torque_frontend_via_jobcli

(Kenneth)

What is a job script?

```
#!/bin/bash
echo "I am a minimal job script"
```

A job script is (bash) shell script, a text file that includes shell commands, that specifies:

- The **resources** that are required by the calculation
  (number of nodes & cores, amount of memory, how much time is required, …)
- The **software** that is used for the calculation (usually via `module load` commands)
- The steps that should be done to execute the calculation (starting from home dir.), specified as **shell commands**, typically:
  - 1) Staging in of input files
  - 2) Running the calculation
  - 3) Staging out of results

(Kenneth)

Required resources are specified via `#PBS` directives

```
#!/bin/bash
#PBS -N solving_42          # job name
#PBS -l nodes=1:ppn=4       # single-node job, 4 cores
#PBS -l walltime=10:00:00   # max. 10h of wall time
#PBS -l vmem=50gb           # 50GB of (virtual) memory required
# rest of job script goes here ...
```

- Required resources can be specified via `#PBS` lines in job script
- Or via options to job submission command (`qsub -l ...`)
- **Maximum walltime of jobs on HPC-UGent clusters: 72 hours (3 days)**
- For longer calculations: break it up in shorter jobs, use a different (faster) cluster, use more cores (if software scales), use some form of “checkpointing”, …

(Kenneth)

Central software stack via modules [1/2]

- **Scientific software is made available via *environment modules***
- An env. module prepares the environment for using a particular software application
- Module naming scheme: `<name>/<version>-<toolchain>[-<suffix>]`
- Interacting with module files is done via the `module` command (Lmod)
- Load a module to prepare the session or job environment for using the software:

```
module load SciPy-bundle/2023.07-gfbf-2023a
```

- Modules that are required as dependencies will be loaded automatically
- To see list of currently loaded modules, run `module list` (or `ml`)

(Kenneth)

Central software stack via modules [2/2]

- To get an overview of *all* available modules, run `module avail` (or `ml av`)
- To see available versions for specific software, run `module avail soft_name/`
- To unload all currently loaded modules, run `module purge`
- Modules are installed using a particular toolchain (`foss`, `intel`, ...), which includes C/C++/Fortran compilers, MPI library, BLAS/LAPACK/FFT libraries
- **You should only combine modules that were installed with the same toolchain**, or a subtoolchain thereof (for example `foss/2023a` + `GCC/12.3.0`)
- See also dedicated section in HPC-UGent documentation

(Kenneth)

Central software stack via modules (example)

```
$ python -V; which python
Python 3.9.21
/usr/bin/python
$ python -c 'import numpy; print(numpy.__version__)'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'numpy'
$ module load SciPy-bundle/2025.06-gfbf-2025a
$ python -V; which python
Python 3.13.1
/apps/gent/RHEL9/zen2-ib/software/Python/3.13.1-GCCcore-14.2.0/bin/python
$ python -c 'import numpy; print(numpy.__version__)'
2.3.1
```

(Kenneth)

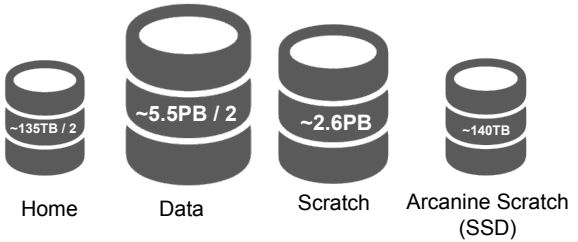
Useful environment variables for job scripts

(these are only defined in the context of a running job!)

- **`$PBS_JOBID`**: job id of running job
- **`$PBS_O_WORKDIR`**: directory from which job was submitted on login node
  - It is common to use `cd $PBS_O_WORKDIR` at beginning of a job script
- **`$PBS_ARRAYID`**: array id of running job
  - Only relevant when submitting array jobs (`qsub -t`)
- **`$TMPDIR`**: unique *local* directory specific to running job
  - Cleaned up automatically when job is done, so make sure to copy result files!
- **`$EBROOTXYZ`, `$EBVERSIONXYZ`**: root directory/version for software package XYZ
  - Only available when module for XYZ is loaded

(Kenneth)

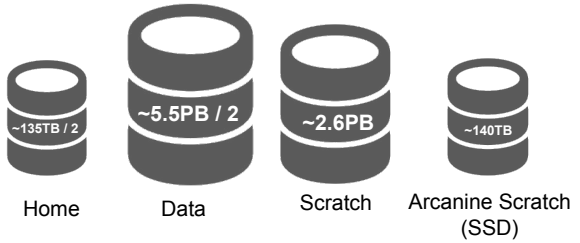
Input/output data and shared filesystems



- See dedicated section in HPC-UGent documentation
- Think about input/output:
  - How and where will you *stage in* your data and input files?
  - How and where will you *stage out* your output and result files?
- Manually (on login nodes) vs automatically (as a part of job script)
- **Home filesystem** (`$VSC_HOME`): only for limited number of small files & scripts
- **Data filesystem** (`$VSC_DATA*`): 'long-term' storage, large files
- **Scratch filesystems** (`$VSC_SCRATCH*`): for 'live' input/output data in jobs

(Kenneth)

Storage quota (disk space)



- Home directory (`$VSC_HOME`): 3GB (fixed!)
- Personal data directory (`$VSC_DATA`): 25GB (fixed!)
- Personal scratch directory (`$VSC_SCRATCH`): 25GB (fixed!)
- Current quota usage can be consulted on VSC accountpage
- **More storage quota (100s of GBs, even TBs) available for *virtual organisations (VOs)***; see dedicated section on VOs in HPC-UGent documentation
- Additional quota can be requested via VSC accountpage (“Edit” tab)
- Shared directories with VO members: `$VSC_DATA_VO`, `$VSC_SCRATCH_VO`
- Personal VO subdirectories: `$VSC_DATA_VO_USER`, `$VSC_SCRATCH_VO_USER`

(Kenneth)

Current storage usage - personal directories

See *"View Account"* tab on VSC accountpage (account.vscentrum.be)

(for now, only data volumes, not number of files (inode quota))

Usage

⚠ Values can be outdated up to 1 hour.

Personal

Storage name	Used	Quota	%	Inodes used (files)
VSC_HOME	2.25 GiB	5.7 GiB	39.39%	77521
VSC_DATA	13.28 GiB	23.75 GiB	55.93%	234842
VSC_DATA_SHARED	n/a	972.8 MiB	n/a	n/a
VSC_SCRATCH_ARCANINE	0 B	974.0 MiB	0.00%	1
VSC_SCRATCH_KYUKON	15.17 GiB	23.75 GiB	63.88%	27467

(Kenneth)

Current storage usage - own VO directories

See *"View Account"* tab on VSC accountpage (account.vscentrum.be)

(for now, only data volumes, not number of files (inode quota))

Virtual Organisation

Storage name	Virtual Organisation	Used	Quota	%	Inodes used (files)
VSC_DATA_VO	gvo00002	0 B	7.67 TiB	0.00%	1
VSC_SCRATCH_ARCANINE_VO	gvo00002	0 B	7.25 TiB	0.00%	1
VSC_SCRATCH_KYUKON_VO	gvo00002	0 B	10.35 TiB	0.00%	1

(Kenneth)

Current storage usage - total usage in VO directories

- See *"View VO"* tab on VSC accountpage

  (for now, only data volumes, not number of files (inode quota))

- **Detailed info per VO member can only be consulted by VO administrators!**

Virtual Organisation quota

⚠ Values can be outdated up to 1 hour.

Name	Used	Quota	%
VSC_DATA_VO	7.75 TiB	15.34 TiB	50.55%
VSC_DATA_SHARED_VO	0 B	1.9 GiB	0.00%
VSC_SCRATCH_ARCANINE_VO	8.23 TiB	14.5 TiB	56.72%
VSC_SCRATCH_KYUKON_VO	14.73 TiB	20.7 TiB	71.19%

VSC_DATA_VO

User	Used	Quota	%
vsc40023	1.22 TiB	1.73 TiB	70.69%
vsc40002	146.76 GiB	1.73 TiB	8.29%
vsc41206	0 B	1.73 TiB	0.00%

(Kenneth)

Multi-threaded “Hello World” with OpenMP (in C)

```
#include <stdio.h>
#include <omp.h>
int main() {
    #pragma omp parallel
    {
        int thread_id = omp_get_thread_num();  // Get the thread ID
        int n_threads = omp_get_num_threads(); // Get total # threads
        printf("Hello, World! from thread %d of %d\n", thread_id, n_threads);
    }
    return 0;
}
```

Compile with: `gcc -O2 example_openmp.c -fopenmp -o example_openmp`

Run in job script with: `./example_openmp`

(Kenneth)

Distributed “Hello World” with MPI (in C)

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);  // Initialize MPI
    int world_rank, world_size;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank); // Get process rank
    MPI_Comm_size(MPI_COMM_WORLD, &world_size); // Get # processes
    printf("Hello, World! from process %d of %d\n", world_rank, world_size);
    MPI_Finalize(); // Finalize MPI
    return 0;
}
```

Compile with: `mpicc -O2 example_mpi.c -o example_mpi`

Run in job script with: `mpirun -np 4 ./example_mpi` `# or use mympirun (recommended!)`

(Kenneth)

Full example job script (single-core job)

```
#!/bin/bash
#PBS -N count_example         # job name
#PBS -l nodes=1:ppn=1         # single-node job, single core
#PBS -l walltime=2:00:00      # max. 2h of wall time

# load Python 3.13, with batteries included (extra PyPI packages)
module load Python-bundle-PyPI/2025.07-GCCcore-14.3.0
# copy input data from location where job was submitted from
cp $PBS_O_WORKDIR/input.txt $TMPDIR
# go to temporary working directory (on local disk) & run Python code
cd $TMPDIR
python -c "print(len(open('input.txt').read()))" > output.txt
# copy back output data, ensure unique filename using $PBS_JOBID
cp output.txt $VSC_DATA/output_${PBS_JOBID}.txt
```

(Kenneth)

Full example job script (multi-node MPI job)

```
#!/bin/bash
#PBS -N mpi_hello             # job name
#PBS -l nodes=2:ppn=4         # 2 nodes, 4 cores per node
#PBS -l walltime=2:00:00      # max. 2h of wall time

module load OpenMPI/5.0.8-GCC-14.3.0 # or foss/2025b
module load vsc-mympirun

# go to working directory, compile and run MPI hello world program
cd $PBS_O_WORKDIR
# C code for MPI Hello: https://mpitutorial.com/tutorials/mpi-hello-world
mpicc mpi_hello.c -o mpi_hello
mympirun ./mpi_hello
```

(Kenneth)

Job output files

- **Your job script may produce informative, warning, and/or error messages.**
  - Two output files are created for each job: stdout (`*.o*`) + stderr (`*.e*`)
  - Located in directory where job was submitted from (by default)
  - Messages produced by a particular command in the job script

    can be "caught" and redirected to a particular file instead:

    ```
    example > out.log 2> err.log
    ```

    (see dedicated section of our Linux tutorial for more details)

- In addition, the software used for the calculation may have generated additional output or result files (which is very software-specific).

(Kenneth)

Job submission and management workflow

- Submit job scripts from a login node to a cluster for execution using `qsub` command:

```
$ module swap cluster/donphan
$ qsub example.sh
12345
```

- An overview of the active jobs is available via the `qstat` command:

```
$ qstat
Job ID    Name        User            Time Use S Queue
------- ----------- -------------- -------- - -------
12345   example      vsc40000        1:32:57  R donphan
```

- To remove a job that is no longer necessary, use the `qdel` command: `qdel 12345`

(Kenneth)

Job scheduling

- All HPC-UGent clusters use a **fair-share scheduling** policy.
- No guarantees on when job will start (and impossible to predict), so **plan ahead**!
- Job priority is determined by various factors:
  - Historical usage
    - Aim is to balance usage over users
    - Infrequent/frequent users => higher/lower priority
  - Requested resources (# nodes/cores, walltime, memory, ...)
    - Larger resource request => lower priority
  - Time waiting in queue
    - Queued jobs get higher priority over time
  - User limits
    - Avoid that a single user fills up an entire cluster

(Kenneth)

Checking resource usage: `slurm_jobinfo`

```
$ slurm_jobinfo 12345678
Name                : example
User                : vsc40000
Partition           : shinx
Nodes               : node4201.shinx.os
Cores               : 8
State               : COMPLETED
Submit              : 2025-11-10T10:58:47
Start               : 2025-11-10T11:01:58
End                 : 2025-11-10T18:49:44
Reserved walltime   : 3-00:00:0.0
Used walltime       :   07:47:46.0
Used CPU time       :   07:38:54.0
% User (Computation): 82.03
% System (I/O)      : 17.97
Mem reserved        : 15G
Max Mem used        : 14.97G (node4201.shinx.os)
```

Specify job ID
(make sure correct `cluster` module is loaded)

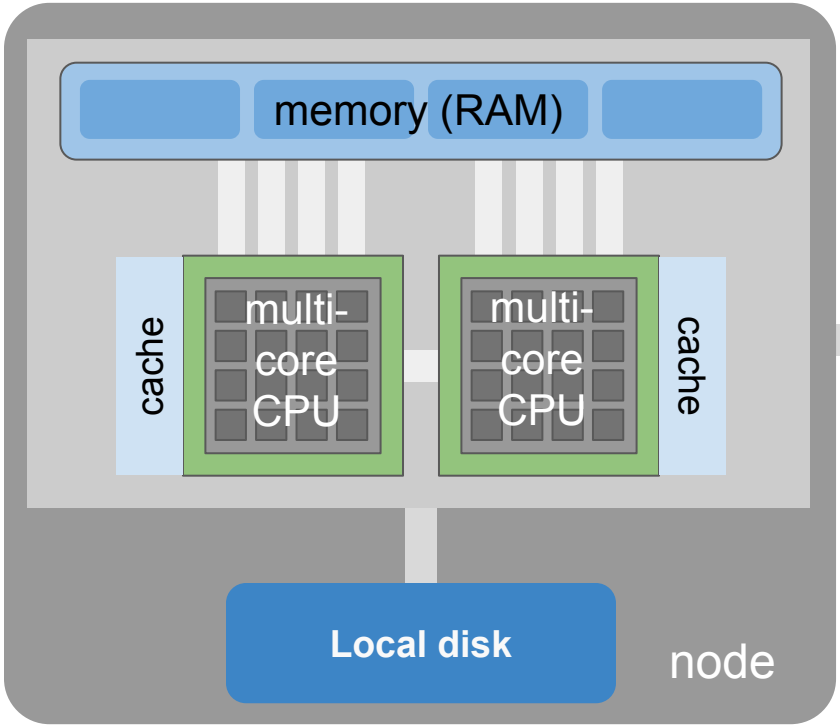
Used CPU time should be roughly equal to # cores times used walltime.

For this job it is <u>not</u>, only a single core was used?!

Large ratio of system CPU is not ideal, can maybe be improved

(Kenneth)

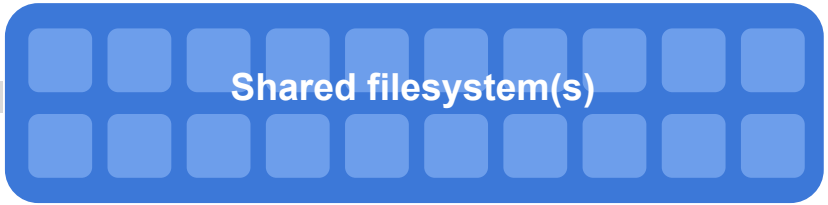
High-level overview of a (worker)node + shared filesystems



Types of disk:

HDD | SSD | NVME

slow → fast



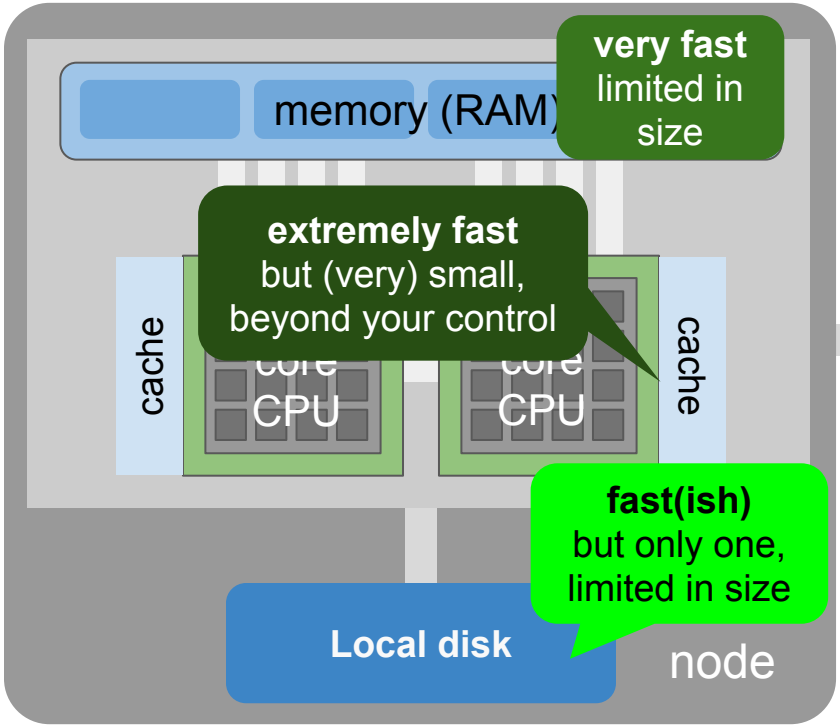
Filesystems @ HPC-UGent Tier-2:

$VSC_HOME	“slow” & small
$VSC_DATA	“slow” & large (+ safe)
$VSC_SCRATCH	fast & large
$VSC_SCRATCH_ARCANINE	faster & medium size

(Kenneth)

Speed in which data can be read (or written)

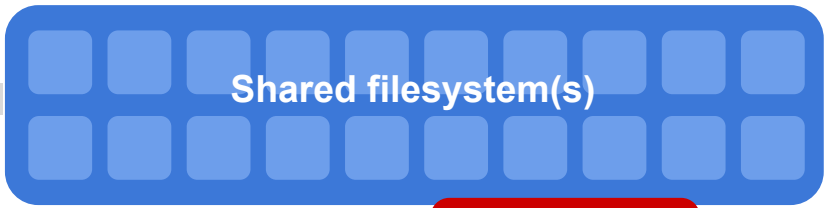
VLAAMS SUPERCOMPUTER CENTRUM



Types of disk:

HDD | SSD | NVME

slow → fast



Filesystems @ HPC-UGe

Don't use this for data

$VSC_HOME

"slow" & small

slow(ish)

$VSC_DATA

"slow" & large (+ safe)

fast (& large)

$VSC_SCRATCH

st & large

$VSC_SCRATCH_ARCANINE

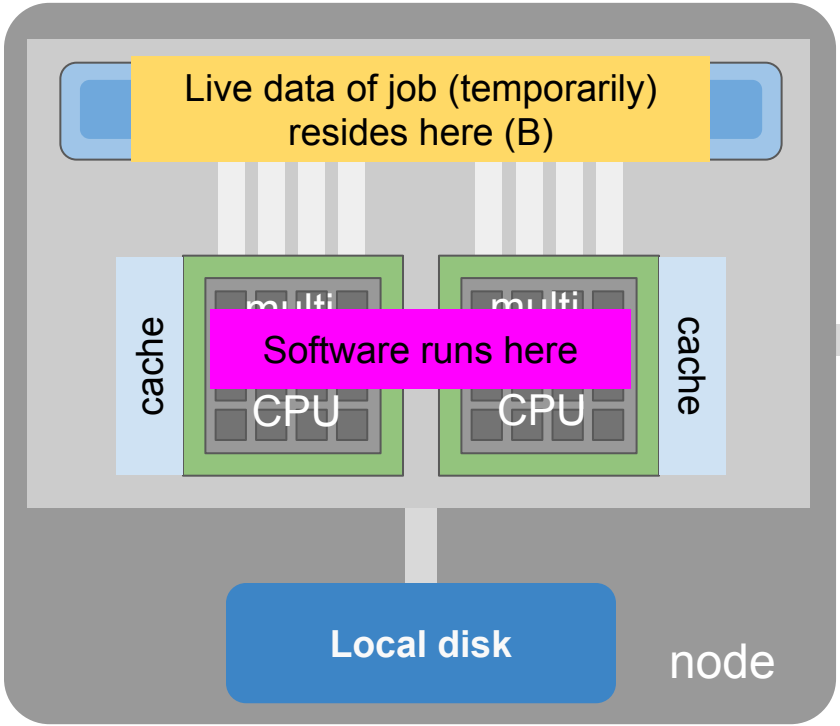
edium size

faster

(Kenneth)

Where does your data come from?

VLAAMS SUPERCOMPUTER CENTRUM

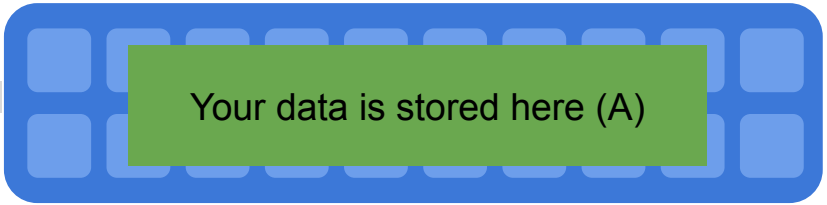


Types of disk:

HDD | SSD | NVME

slow → fast

How the (input) data for your jobs gets from A to B can matter (a lot), especially when it's large volume (GBs), lots of (small) files, or is accessed "randomly".



Filesystems @ HPC-UGent Tier-2:

$VSC_HOME	"slow" & small
$VSC_DATA	"slow" & large (+ safe)
$VSC_SCRATCH	fast & large
$VSC_SCRATCH_ARCANINE	faster & medium size

Embarrassingly parallel jobs

- Use case: lots of (very) short single-core tasks (hours, or even minutes/seconds)
- Submitting lots of tiny jobs (minutes of walltime) is not a good idea
  - Overhead for each job (node health checks), lots of bookkeeping (output files, etc.)
- Better options:
  - Array jobs
    - Single job script, each (sub)job is assigned a unique id (via `$PBS_ARRAYID`)
  - GNU parallel
    - General-purpose tool to easily run commands in parallel with different inputs
  - Worker tool (see dedicated section in HPC-UGent documentation)
    - One single job that processes a bunch of tasks (multi-core or even multi-node)
    - Job script is parameterized, submit with `wsub` rather than `qsub`

(Kenneth)

Python virtual environments (venv's)

- Isolates project dependencies from the system-wide Python installation
- Makes local installation of Python packages (which are not available via modules) easier
- Recommendations & caveats on HPC clusters:
  - Prefer using **Python software installed as module** (if available): better performance
  - **Activate** the Python virtual environment with the **same modules loaded**
  - Virtual environment built on one cluster **will likely not work** on another cluster (diff. OS / CPU)
- Making a virtual environment (manually):

```
$ module load SciPy-bundle/2025.06-gfbf-2025a
$ python -m venv example-venv
$ source example-venv/bin/activate   # activate virtual environment (similar to 'module load')
$ pip install example                # install package not available as module
$ python script.py                   # use virtual environment to run Python script
$ deactivate                         # deactivate virtual environment (like 'module unload')
```

docs.hpc.ugent.be/setting_up_python_virtual_environments

(Lara)

vsc-venv: Python virtual environment wrapper script

- Encapsulates the creation and management of Python virtual environments
- Facilitates working with Python virtual environments across HPC-UGent clusters
- It makes sure that:
  - The same modules are loaded on each activation
  - A new virtual environment is created for each cluster OS & CPU microarchitecture
- Arguments for the `vsc-venv` tool:
  - ***modules.txt*** (optional): file with list of names of environment modules to load
  - ***requirements.txt*** (required): file with list of Python packages to install

docs.hpc.ugent.be/setting_up_python_virtual_environments

(Lara)

vsc-venv: example + demo

modules.txt:

```
SciPy-bundle/2025.06-gfbf-2025a
Pillow/11.3.0-GCCcore-14.2.0
```

requirements.txt:

```
beautifulsoup4==4.14.3
```

```
$ module load vsc-venv
$ source vsc-venv --activate --requirements requirements.txt --modules modules.txt
$ python --version              # Python provided by dependency of SciPy-bundle module
Python 3.13.1
$ python -c "import PIL”        # import PIL from Pillow module
$ python -c "import numpy”      # import numpy from SciPy bundle
$ python -c "import bs4”        # import bs4 from beautifulsoup4 package
```

docs.hpc.ugent.be/setting_up_python_virtual_environments

(Lara)

Software installations - caveats with Conda & co

- ***How*** the software was compiled/installed can impact **performance** a lot
- Software should be **optimized** for the specific CPUs on which it will be run
  - This is usually not the case when using container images or Conda environments…
  - Generic binaries are used that can run anywhere
  - Performance is sacrificed for mobility of compute…
- Vast majority of software installed in central software stack on HPC-UGent infrastructure is optimized for the specific CPUs of each cluster
- Additional caveats with Conda: messes up your home directory very easily…

Software installations

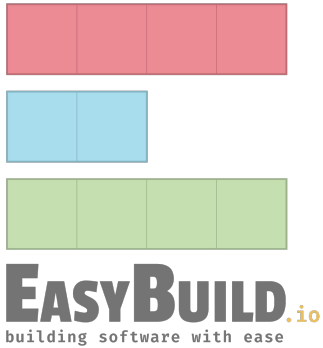
- To submit a request for software installation, use the request form:

  docs.hpc.ugent.be/software_installation_requests

- Requests may take a while to process (especially for new software), so **be patient**…
- Make the request sooner rather than later!

- All software installations are done using EasyBuild
- Originally developed by HPC-UGent,

  now a worldwide community of experts!
- See also easybuild.io



(Lara)

What if you no longer have to install

*a **broad range of scientific software***

from scratch on every laptop, HPC cluster,

or cloud instance you use or maintain,

without compromising on performance?

Lara

EESSI shared software stack

- European Environment for Scientific Software Installations (EESSI)
- **Shared repository of (optimised) installations of scientific software**
- Uniform way of providing software to users, regardless of the system they use
- Should work on any Linux system (with Intel, AMD, or Arm CPU)
- From laptops and personal workstations to HPC clusters and cloud
- Support for different CPU (micro)architectures, interconnects, GPUs, etc.
- Already available on clusters at UGent + ~50 systems across Europe, see https://eessi.io/docs/systems

EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

https://eessi.io

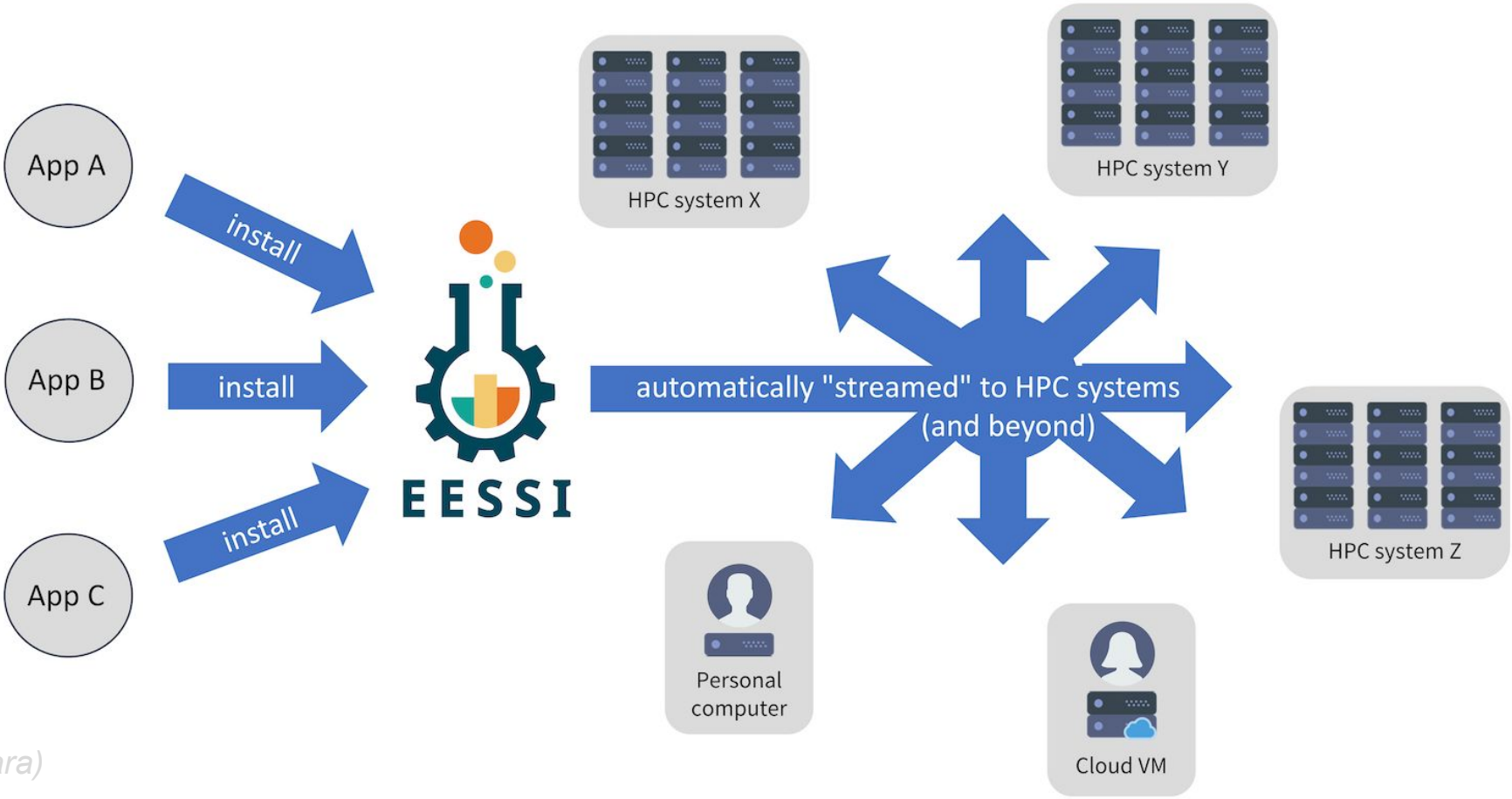
(Lara)

Major goals of EESSI

- Providing a truly **uniform software stack**
  - Use the (exact) same software environment everywhere
  - **Without sacrificing performance** for “mobility of compute” (like is typically done with containers/conda)
- **Avoid duplicate work** (for researchers, HPC support teams, sysadmins, …)
  - Tools that automate software installation process (EasyBuild, Spack) are not sufficient anymore
  - Go beyond sharing build recipes => work towards a shared software stack
- Facilitate HPC training, development of (scientific) software, …

Lara

EESSI shared software stack

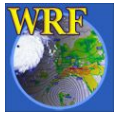


(Lara)

Overview of software provided by EESSI

Over 1,200 software software installations available
per CPU target via software.eessi.io CernVM-FS repository;
increasing every day

- Spread across 2 versions of EESSI: 2023.06, 2025.06
- 15 CPU microarchitectures supported (Intel, AMD, Arm)
- Over 700 different software packages
- + additional extensions: Python packages, R libraries
- Including ESPResSo, GROMACS, LAMMPS, OpenFOAM, PyTorch, R, QuantumESPRESSO, TensorFlow, waLBerla, WRF, …
- Compiler toolchains used in EESSI versions:
  - 2023.06: `foss/2023a`, `foss/2023b`
  - 2025.06: `foss/2024a`, `foss/2025a`, `foss/2025b`
- Detailed overview: eessi.io/docs/available_software



(Lara)

Using EESSI is… easy!

- Step 0: Make sure that EESSI is available (if not, install it, see https://eessi.io/docs)

```
ls /cvmfs/software.eessi.io/
```

- Step 1: Initialize your shell environment to use EESSI (pick a version)

```
source /cvmfs/software.eessi.io/versions/2025.06/init/lmod/bash
```

- Step 2: Activate the software you want to use by loading the corresponding module(s)

```
module load GROMACS/2025.2-foss-2025a
```

- Step 3: Run the software!

```
gmx ...
```

The workflow is *exactly* the same no matter which system you use…

Tip: on Windows or macOS, you can create a Linux VM using Lima: https://lima-vm.io

(Lara)

Demo: Running LAMMPS via EESSI @ HPC-UGent


```
#!/bin/bash
#PBS -N "EESSI_Demo_LAMMPS_lj"
#PBS -l nodes=1:ppn=4
#PBS -o EESSI_demo.out
#PBS -e EESSI_demo.err
#PBS -l walltime=00:30:00

module --force purge
export OMPI_MCA_orte_keep_fqdn_hostnames=1
unset SLURM_EXPORT_ENV

source /cvmfs/software.eessi.io/versions/2023.06/init/bash
module load LAMMPS/29Aug2024-foss-2023b-kokkos
mkdir /tmp/$USER && cd /tmp/$USER
curl -o in.lj https://raw.githubusercontent.com/lammps/lammps/refs/heads/develop/bench/in.lj
export OMP_NUM_THREADS=1
mpirun -np 4 lmp -in in.lj
rm -r /tmp/$USER
```

```
$ cd /apps/gent/tutorials/Intro-HPC/examples
$ cp Running-batch-jobs/EESSI_example.pbs $HOME
$ cd $HOME
$ qsub EESSI_example.pbs
$ qstat
$ cat EESSI_demo.out
```


(Lara)

On which systems is EESSI available?

- On VSC systems:
  - **Tier-2 infrastructure at UGent + VUB + Tier-1 Hortense (+ new Tier-1 soon)**
  - **Tier-1 cloud**: can deploy it yourself and have access to a full software stack in minutes
- EESSI is already available on various other European systems (and beyond)
  - EuroHPC JU systems incl. Vega, Karolina, MareNostrum 5, Deucalion, Discoverer, …
  - Snellius @ SURF, EMBL, Univ. of Stuttgart, Sigma2 in Norway, etc.
- EESSI can be used in virtual machine in European Open Science Cloud (EOSC),

  see also https://www.eessi.io/docs/blog/2025/10/22/eosc
- **Overview of (known) systems that have EESSI available at https://eessi.io/docs/systems**

Lara

Getting Scientific Software Installed training

VLAAMS SUPERCOMPUTER CENTRUM

- In-person VSC training held on 3 Nov 2025 in Brussels
- Slides + recordings: https://github.com/vscentrum/gssi-training
- Topics:
  - EESSI
  - Using containers (Apptainer)
  - Conda, Mamba, Pixi
  - Python: pip, venv
  - R, Julia
  - Software compilation
  - Building container images
  - EasyBuild + Spack
  - General overview (pros & cons)
- Will be organised again in Fall 2026 (more info soon)

(Kenneth)

Questions, problems, getting help

Don't hesitate to contact the HPC-UGent Tier-2 support team via hpc@ugent.be

- Always include:
  - VSC login id
  - Clear description of the problem or question, include error messages, …
  - Location of job script and output/error files in your VSC account
  - Preferably don’t send files in attachment, we prefer to look at it “in context”
  - Also mention job IDs, which cluster was used, how job was submitted, etc.
- Preferably use your UGent email address
- Alternatives:
  - Short (Teams) meeting (for complex problems, big projects)
  - `hpc-users` mailing list

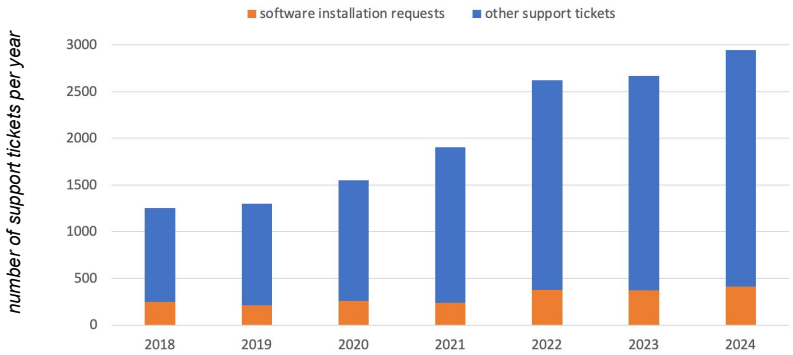
(Kenneth)

Questions, problems, getting help (be patient…)

Don't hesitate to contact the HPC-UGent support team via hpc@ugent.be,

but be patient…

- We're doing what we can to keep up with incoming support questions + software installation req.
- We have been getting **50-100 new tickets per week** recently, and have a backlog of ~250 tickets
- **Help us help you**: read the docs, provide sufficient details (like job IDs, output files, etc.), …
- Feel free to send a reminder in the same ticket or mail thread, especially if your work is blocked



(Kenneth)

HPC Best Practices (@ VSC User Day 2024)

VLAAMS SUPERCOMPUTER CENTRUM | Vlaanderen is supercomputing

HPC Best Practices and Some Magic Tricks

22 Oct 2024 - VSC User Day 2024

Alex Domingo (VUB-HPC)

Ehsan Moravveji (KU Leuven)

docs.vscentrum.be/contact_vsc.html

vscentrum.be/ud24

[direct link to slides (PDF)]

Recording available via https://youtu.be/ltSWg-vHHXs

(Kenneth)

Agenda

- [10:00 - 12:00] *Introduction to HPC-UGent* presentation + Q&A
  - Overview of available hardware, getting a VSC account,
    using the HPC-UGent Tier-2 clusters, getting support, demos and examples, …

Only for on-site attendees

- [12:00 - 13:00] Sandwich lunch
- [13:00 - 14:00] Guided tour of UGent datacenter 10,
  incl. visit to HPC-UGent Tier-2 and VSC Tier-1 cluster
- [14:00 - 17:00] Hands-on session: Getting started with HPC-UGent
  - Login + submitting example jobs
  - Getting started with your own workloads + Q&A

(Kenneth)