



VLAAMS
SUPERCOMPUTER
CENTRUM



Vlaanderen
is supercomputing



Best Practices for HPC

or

How to get your work done faster?



24 Oct 2023 - VSC User Day 2023

Alex Domingo (VUB-HPC), Kenneth Hoste (HPC-UGent)


https://docs.vscentrum.be/contact_vsc.html

Read the documentation(s)

docs.vscentrum.be

VLAAMS
SUPERCOMPUTER
CENTRUM




- **VSC docs have recently been revamped** 
 - Improved navigation, responsive design, sexier!
- VSC sites have extra documentation/info pages

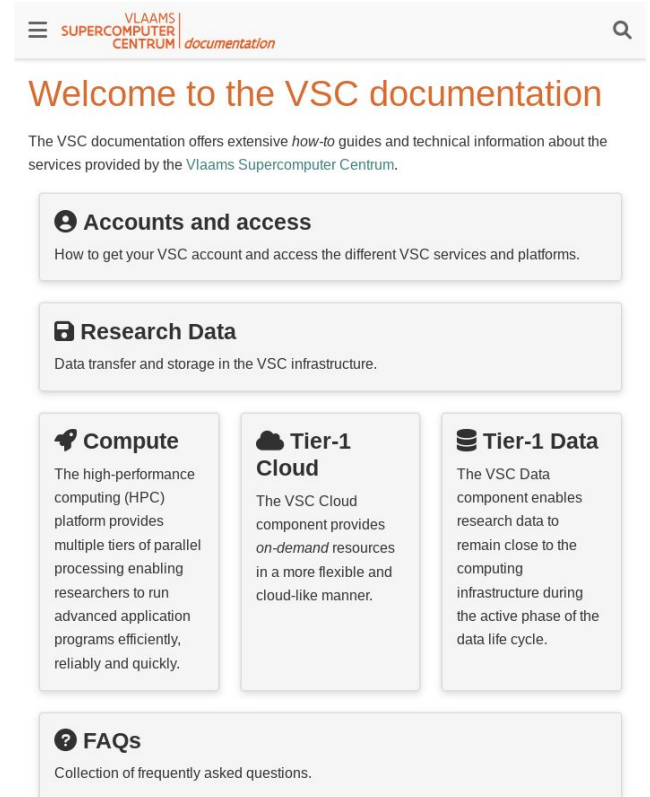
KU Leuven: hpcleuven.github.io/HPC-intro

UGent: docs.hpc.ugent.be

UAntwerp: hpc.uantwerpen.be/support/documentation



VUB: hpc.vub.be

- Software (usually) has its own documentation:
read it! 



The screenshot shows the VSC documentation website. At the top, there is a navigation bar with the VLAAMS SUPERCOMPUTER CENTRUM logo and a search icon. Below the navigation bar, the main heading reads "Welcome to the VSC documentation". A sub-heading states: "The VSC documentation offers extensive *how-to* guides and technical information about the services provided by the Vlaams Supercomputer Centrum." The page features several content blocks: "Accounts and access" (How to get your VSC account and access the different VSC services and platforms.), "Research Data" (Data transfer and storage in the VSC infrastructure.), "Compute" (The high-performance computing (HPC) platform provides multiple tiers of parallel processing enabling researchers to run advanced application programs efficiently, reliably and quickly.), "Tier-1 Cloud" (The VSC Cloud component provides on-demand resources in a more flexible and cloud-like manner.), "Tier-1 Data" (The VSC Data component enables research data to remain close to the computing infrastructure during the active phase of the data life cycle.), and "FAQs" (Collection of frequently asked questions.).

Participate in trainings

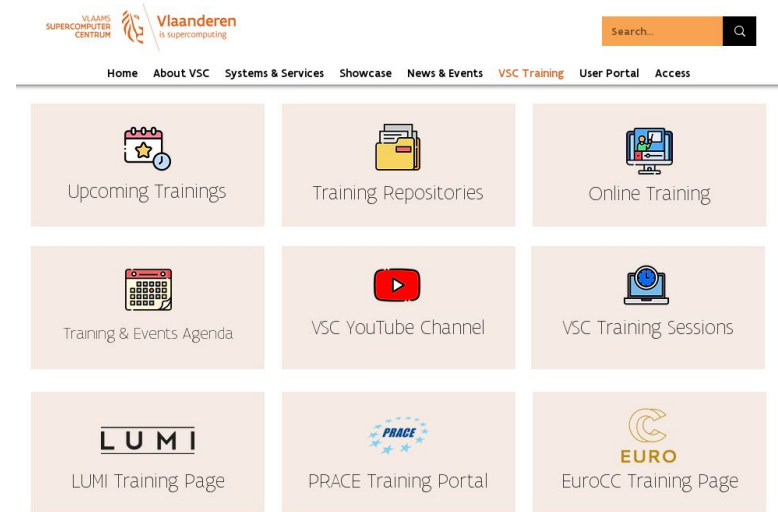
-  **There is a training for you!**
 - Introductions to get started on the HPC
 - Lessons on specific technologies, methodologies and software
 - Virtual or on-site
-  **Our partners also provide trainings**

PRACE: <https://events.prace-ri.eu/category/2>

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

HPC-Portal.eu: <https://hpc-portal.eu/node/88?category=11>

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



The screenshot shows the VSC Training website interface. At the top, there is a navigation bar with links for Home, About VSC, Systems & Services, Showcase, News & Events, VSC Training, User Portal, and Access. A search bar is located on the right. Below the navigation bar, there is a grid of nine tiles, each with an icon and a label: Upcoming Trainings (calendar icon), Training Repositories (folder icon), Online Training (computer monitor icon), Training & Events Agenda (calendar icon), VSC YouTube Channel (YouTube icon), VSC Training Sessions (laptop icon), LUMI Training Page (LUMI logo), PRACE Training Portal (PRACE logo), and EuroCC Training Page (EURO logo).

Upcoming Training

The list below shows the currently scheduled VSC training courses. Please follow the links in the list to find out more about the courses and to register.

16 ^{Mon}_{Oct} HPC@UAntwerp intro... / UAntwerp | Campus Middelheim L...

We are here to help you!



Questions or problems? **Send us an email!**



- Tier-2 support teams at each VSC site:

hpc@ugent.be

hpcinfo@kuleuven.be

hpc@vub.be

hpc@uantwerpen.be

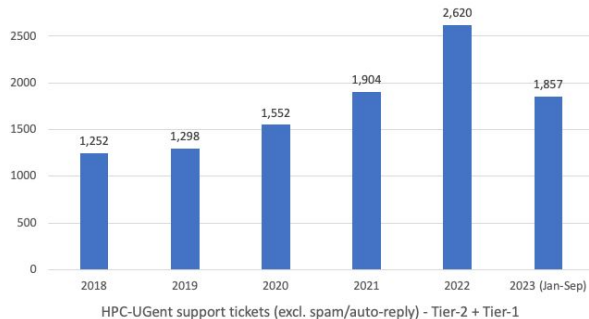
- Tier-1 compute: compute@vscentrum.be
- Tier-1 cloud: cloud@vscentrum.be
- Tier-1 data: data@vscentrum.be
- General questions: info@vscentrum.be




Please read the documentation first...

Don't contact individual people directly!



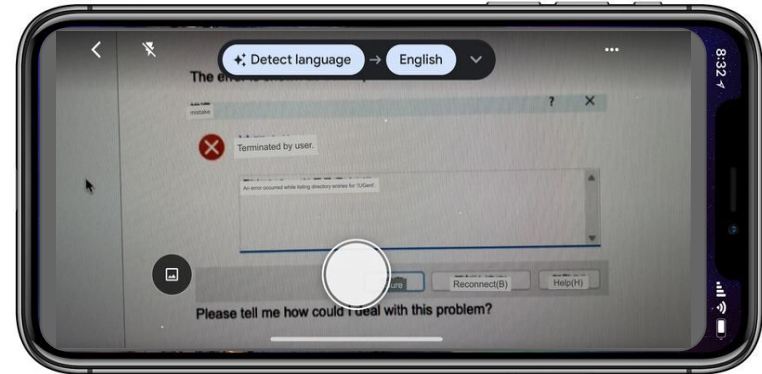
Contact us, but please be patient...



-  **Understanding and troubleshooting issues takes time...**
- VSC support teams may get over 10 “tickets” per day 
- Please be polite & **patient** when contacting us!
- Stick to a single problem/question per support ticket
- If something is **urgent** for you, mention it in email subject, and provide clear deadlines - **We will do all we can.** 
- If you don't hear back in a reasonable time, **don't hesitate to send a reminder** (in the same email thread!)
- We have to look into general problems & tasks first, user-specific tickets get lower priority due to time constraints...

Help us help you

There's an art to opening a good support request... 🤔



Don't make us jump through hoops to help you... 😓

Help us help you



Include **all relevant information** when contacting us... 

Connection problems


- Mention your VSC account + OS
- Where are you connecting from?
- Home vs work, VPN, ...
- Client software: PuTTY, WinSCP, SSH, MobaXterm, Cyberduck, ...

Did you use a troubleshooting checklist?

Did you consider using the web portal?




Software errors, failing jobs

- Mention your VSC account
- Provide job IDs + path to job script + submit cmd
- Keep job output files in their original location
- Mention other relevant output or error files
- **Don't send files or scripts as attachments, mention location in your VSC account!**
- **We prefer to look at the problem "in context"**
- Ideally explain how to reproduce the problem (which cluster, which commands did you run, ...)


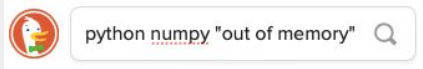

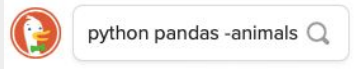
Always copy-paste error messages (if you can) instead of sending screenshots! 

Pro tips when searching for answers



- You are usually not the first person to run into a problem... 
-  Start with **reading** (and trying to understand) the error message you see
- **Try to use a search engine** (Google, DuckDuckGo, ...) 

Secrets to success:

- Use a **good search query**: English, handful of keywords, ...
- **Use quotes** around error messages to avoid searching for individual words
 Example: 
- **Filter out irrelevant hits** by using negative terms (-).  Example: 
- Take into account the **date & context of solution or answer** you found (you do not have administrator rights (sudo) in your VSC account)
- **Be careful with what ChatGPT tells you**, it may be *hallucinating*...

Plan your strategy



Design a plan of execution for your research project on the HPC

Resources

How many simulations will you need to run?
For how long? Do you need compute credits?
What is the optimal number of CPU cores to use?
How much memory? Can I use GPUs?

Software

What software packages do you need?
Are they available in the system?

Data

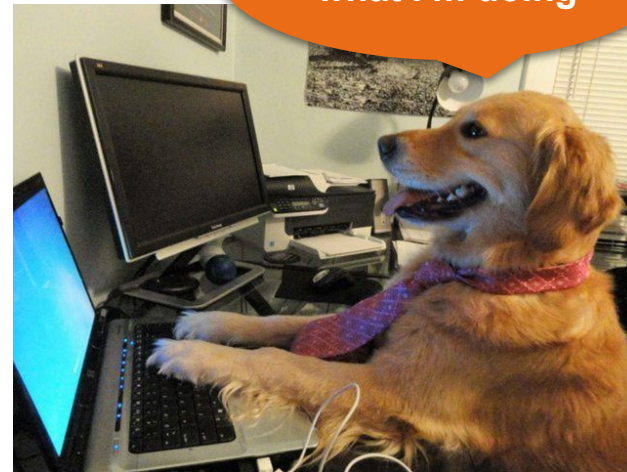
How much data is needed as input?
How much is generated as output?
How many files? Need to transfer/share data?

Extras

What other tasks are needed in your project?
Processing of data files? Analysis of results?

Booooooring...

I have no idea
what I'm doing





Parallel computational jobs in batches

Resources

Few number of jobs using a lot of resources

- ⇒ Performance depends on network speed
- ⇒ Execution time improves with parallelization
- ⇒ Think big! Use Tier-1 if Tier-2 becomes too small

Software

Software with support for MPI

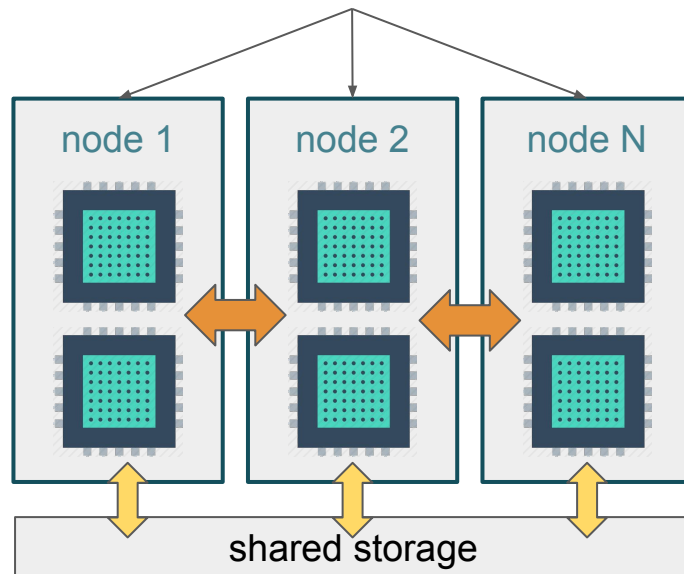
- ⇒ Not all software has this capability
- ⇒ Better installed by the HPC team
- ⇒ Might take some time to get ready...

Data

Fast shared storage space

- ⇒ Check quota in \$VSC_SCRATCH
- ⇒ Is your MPI application I/O-intensive?

single job script
single application



Plan your strategy: “Embarrassingly” Parallel



Task farming or job arrays

Resources

Many independent tasks

- ⇒ Serial or weak-parallelization tasks (few cores)
- ⇒ High overhead due to resource allocation
- ⇒ Memory loading can be a bottleneck

Soft.

Extra tools to automatise task distribution

- ⇒ GNU Parallel, worker, workflow manager tools, ...

Data

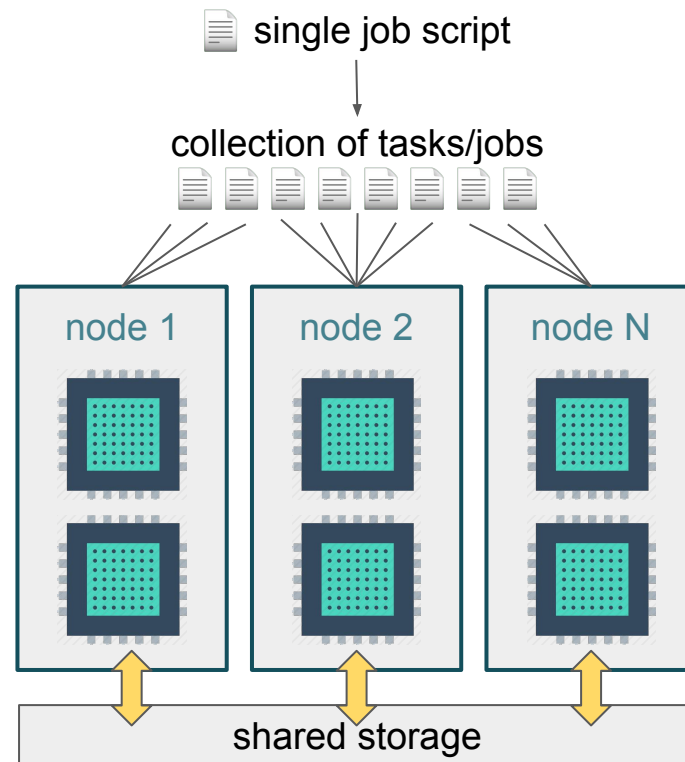
Organization of many input/output files

- ⇒ Check limits on number of files in the storage
- ⇒ Use a Hierarchical Data Format (HDF) or similar

Extras

Post-processing

- ⇒ Wait for last task completion



Plan your strategy: Bioinformatic Pipelines



I/O intensive jobs

Resources

Jobs handling a lot of data

- ⇒ Weak-parallelization, adding cores does not help
- ⇒ Benefits from memory bandwidth

Software




Software stack with hundreds of packages

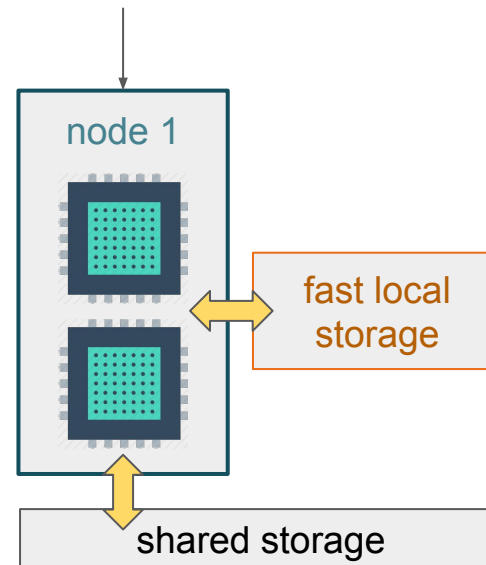
- ⇒ Installation of pipelines is time consuming
- ⇒ Finding right combination of versions can be tricky
- ⇒ Pipelines commonly rely on software wrappers

Data

Storage with very fast random reads

- ⇒ Access to large genomic DBs is the bottleneck
- ⇒ Check if DBs are already available in the cluster
- ⇒ Use systems with fast local storage (SSDs)

 single job script
 many steps
 many applications





Jobs on GPUs

Resources

Single simulation run on a GPU per job

- ⇒ GPUs are more limited resources than CPUs
- ⇒ GPU memory matters, but not handled by job
- ⇒ CPU power still matters to feed the GPU
- ⇒ Multi-GPU should be considered as experimental

Software

Software needs specific support for GPUs

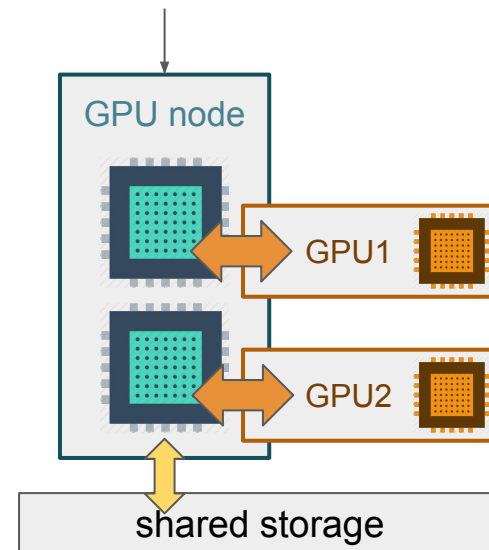
- ⇒ Build on top of Nvidia CUDA or AMD ROCm
- ⇒ Better if installed by the HPC team
- ⇒ Might take some time to get ready

Extras

Pre-processing and monitoring

- ⇒ Preparation of data can be the bottleneck of the job
- ⇒ Solutions to monitor the GPUs in real-time:
TensorBoard, wandb.ai, neptune.ai

- single job script
- pre-process on CPU
- simulation on GPU



- + docs.vscenrum.be/jobs/job_submission.html#requesting-gpus
- + hpc.vub.be/docs/job-submission/gpu-job-types

Plan your strategy: Interactive Workflow



Graphical interface: Jupyter, RStudio, Matlab, ...

- ✓ Tier-1 Hortense (OoD)
- ✓ Tier-2 UGent (OoD)
- ✓ Tier-2 KULeuven (OoD)
- ✓ Tier-2 VUB (JupyterHub)

Resources

Interactive session for non-intensive compute

- ⇒ Resources might be shared – oversubscription
- ⇒ Time limits might be shorter
- ⇒ GPU may be used for visualisation (not compute)

Software

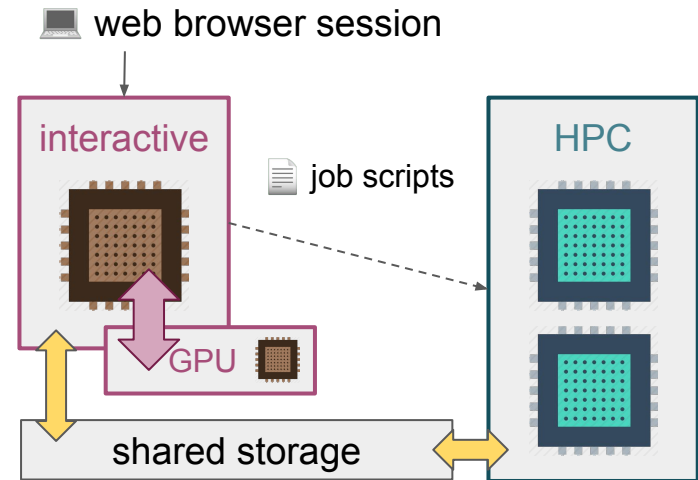
Centrally installed software is available

- ⇒ Software modules can be loaded as usual
- ⇒ Additional graphical tools installed by HPC team

Extras

Compute intensive simulations

- ⇒ Launch non-interactive job from interactive session on dedicated resources
- ⇒ Start interactive session on dedicated resources and run simulations directly on it



- + docs.vscentrum.be/leuven/services/openondemand.html
- + docs.hpc.ugent.be/web_portal
- + hpc.vub.be/docs/notebooks

🧐 Tips & tricks in Linux shell environment

Try to be more like a computer geek! 

(no, you don't need to move into a cave or stop taking showers...)

The Linux shell environment (usually `bash`)
is a **powerful** instrument.

Don't be intimidated by it, **use it to your advantage!**

You can use these tricks **interactively**, and (most) also in **job scripts!**



```
vsc40000@Hortense $ module load compute_power  
vsc40000@Hortense $ |
```

See also [VSC](#) and [HPC-UGent](#) documentation + [VSC training events!](#)

🧐 Tips & tricks in Linux shell environment

1) 📖 Use the **built-in documentation** (man pages)

2) Use the **(shell) history**, Luke 📜

- **Shell keeps history of last 1,000 commands**
(can be increased via \$HISTSIZE)
- Access previous commands via up arrow (↑)
- Run history command to see full history
- **Use Ctrl-R to search through history!**

3) Type like a 🥷 by cheating via **tab completion!**

(TAB is the key above CAPS lock key on your keyboard)

```
$ man cp # read docs for cp command
```

```
$ history
```

```
...
```

```
997 echo "Hello VSC users"
```

```
998 mkdir demo
```

```
999 cd demo
```

```
1000 vim job.sh
```

```
# use Ctrl + R to search history  
(reverse-i-search)`echo':  
echo "Hello VSC users"
```

```
# use tab completion to avoid typos
```

```
$ cat file_<TAB>
```

```
$ cat file_with_very_long_name.txt
```




Tips & tricks in Linux shell environment

4) Impress your colleagues by **piping commands** together to do more complex things.

The output of command N is “streamed” via pipe (|) as input for command N+1.

Easy to combine many simple commands to get a complex task done quickly!



```
# Example:
# - take first 5 text file (*.txt) that have 'input' in the filename
# - sort corresponding *.dat files based on time they were last changed + show metadata
$ ls *.txt | grep input | head -5 | sed 's/\.txt/\.dat/g' | xargs ls -lrt
-rw-r--r--  1 vsc40000 users  123 Oct 17 11:37 input1.dat
-rw-r--r--  1 vsc40000 users  431 Oct 18 03:19 input12.dat
-rw-r--r--  1 vsc40000 users  351 Oct 18 21:01 input231.dat
-rw-r--r--  1 vsc40000 users  829 Oct 19 09:48 input45.dat
-rw-r--r--  1 vsc40000 users  641 Oct 19 11:17 input6.dat
```



Tips & tricks in Linux shell environment


```
$ alias x="echo 'Hello VSC Users!'"
$ x
Hello VSC Users!

# avoid expanding $TEST when defining alias,
# ensure it's expanded when alias is used
$ alias t='echo "$TEST"'
$ export TEST=test123
$ t
test123

$ function f(){
# create your own commands with functions
}

$ ml # module list
$ ml foss/2023a # module load foss/2023a
```

5) Define your own **custom aliases and functions**

- Useful for long commands that you run a lot
- Also useful if you can't type, jsut liek em 
- `alias x="echo 'Hello VSC Users!'"`
- Careful with single vs double quotes!
- For more complex things, use shell functions
- Usually in shell startup script like `~/ .bashrc`

6) **Use pre-defined aliases/functions**, like `ml`

- Shorthand for both `module load` and `module list`!
- Also works for other subcommands, like `ml swap`



Tips & tricks in Linux shell environment

7) **Set up your environment** just like you want by extending the **shell startup script** in your home dir

- Usually `~/.bashrc` (but there are others)
- Set environment variables, define aliases, ...
- Startup script runs every time you log in, and/or every time a job starts running
- **Don't *load* modules in your startup script!** (for a variety of reasons)

```
$ cat ~/.bashrc
# set some extra environment variables
export MY_FAVOURITE_CLUSTER=hortense
export SPB=$VSC_SCRATCH_PROJECTS_BASE
# define my aliases
alias m='ml swap cluster/dodrio/cpu_milan'
alias p="cd $SPB/2023_000"

$ echo $MY_FAVOURITE_CLUSTER
hortense

$ echo $SLURM_PARTITION
cpu_rome
$ m
$ echo $SLURM_PARTITION
cpu_milan

$ p; pwd
/dodrio/scratch/projects/2023_000
```

Organising your data

Not all directories in the HPC clusters are equal

They belong to different storage systems with different capabilities and purposes:

Depends!
Check site docs

When in doubt,
pick `$VSC_SCRATCH!`

Folder	Capacity	Availability	Performance	Reliability	Backups
<code>\$VSC_HOME</code>	< 10 GB	All VSC sites	Low	High	Yes
<code>\$VSC_DATA</code>	< 100 GB	All VSC sites	Low	High	Yes
<code>\$VSC_SCRATCH</code>	< 500 GB	Local cluster	High	Mid-High	No
<code>\$VSC_SCRATCH_NODE</code> <code>\$TMPDIR</code>	<i>Depends</i>	Local node	<i>Depends</i>	Low	No

+ docs.vscenrum.be/data/storage_locations.html

General considerations for all storage systems


- The **storage is the slowest memory** in the system, minimize its access

Use a temporary ramdisk

```
#!/bin/bash
#SBATCH --mem=250G

RAMDISK=/dev/shm/$SLURM_JOB_ID
mkdir -p $RAMDISK

./high-IO-app $RAMDISK
./postprocess.sh $RAMDISK > $VSC_SCRATCH
```

 not applicable in general

Stage in/out data to/from scratch on-the-fly


```
#!/bin/bash
```

```
DATADIR="${VSC_DATA}/my-project/dataset01"
WORKDIR="${VSC_SCRATCH}/${SLURM_JOB_ID}"
```

```
# Stage in data to working directory
echo "Populating work directory: $WORKDIR"
mkdir -p "$WORKDIR"
rsync -av "$DATADIR/" "$WORKDIR/"
```

```
# Run program on scratch WORKDIR
cd "$WORKDIR"
./high-IO-app data.inp > results.out
```

```
# Save output and clean the WORKDIR
# (these steps are optional, you can also
perform these manually once the job ends)
cp -a results.out "$SLURM_SUBMIT_DIR/"
rm -r "$WORKDIR"
```

 can be done systematically



General considerations for all storage systems

- **Number of files matters**, putting millions of files in a single folder will slow down all filesystem operations on that folder
 - Organize large numbers of files in subdirectories
 - Use a *Hierarchical Data Format* (HDF*) or similar
 - Pack files together in a tarball with the tar command
- Use the **Globus platform** to move data in/out or between VSC clusters
 - All VSC sites have their own endpoints in Globus
 - **Best transfer performance** thanks to dedicated resources for the Globus agent on the cluster

Rule of thumb:
1,000 files per folder

+ docs.vscenrum.be/globus

Managing your own software stack



You can install the software you need yourself in your VSC account ...

... but there are some things you should be aware of, and take into account.

In general, the software you use **should be compiled for the specific system** on which it will be used (w.r.t. CPUs, interconnect, OS, ...).

If not, you may observe a **significant reduction in performance**.



Managing your own software stack



You can **ask the HPC support team** 🧐 🧐 🧐 to install the software you need.

Recommended for:

- Standard releases of software
- Software that is (partially) implemented in a compiler programming language, like C, C++, Fortran, Rust, ...
- Software that requires performance-sensitive libraries like MPI, CUDA, ...
- Software that you can not get installed yourself (even after swearing a lot...)

Be patient, we get a lot of installation requests!



Managing your own software stack



Sometimes **installing the software yourself** in your VSC account is feasible, even with limited experience (or patience)!

Recommended for:

- Software implemented in (only) an interpreted programming language, like Python, Perl, Java, ...
- Software supported in EasyBuild, doesn't require administrator rights (see also docs.hpc.ugent.be/easybuild)
- Compiled software that you know well, or that you are developing or changing

In case of problems: contact the HPC support team 🧐 , **we are happy to help!**

Managing your own software stack



Pro tips 🧐 for installing software yourself



- Compile software **on a worker node of the cluster where you will be running it**
- Use `-march=native` (GCC) or `-xHost` (Intel compilers) to **target specific CPU**
(but don't use `-xHost` on a system with AMD CPUs! 🧐)
- Use `$VSC_ARCH_LOCAL + $VSC_OS_LOCAL` to install in cluster-specific subdirectory
- Don't install “complex” software packages yourself (PyTorch, OpenFOAM, ...)

Be careful with using (pre-built) container images or conda/mamba to install software, because that often implies running **generic binaries** (not optimized for specific CPUs)...



Managing your own software stack



Example: Python virtual environment **on top of centrally installed software**

interactive session

```
# load module for Python, PyTorch, ...
$ ml PyTorch/1.13.1-foss-2022a-CUDA-11.7.0

# create Python virtual environment
$ export VENV_DIR=$VSC_DATA/vsc-demo
$ python3 -m venv $VENV_DIR

# activate virtual env + install Poutyne
$ source $VENV_DIR/activate
$ pip install Poutyne

# exit virtual env
$ deactivate
```

job script


```
#!/bin/bash
# (resource requirements go here)

# set up job environment:
# load PyTorch + activate virtual env.
ml PyTorch/1.13.1-foss-2022a-CUDA-11.7.0
source $VSC_DATA/vsc-demo/activate

# run your Python script that uses Poutyne
python3 pytorch_poutyne_example.py
```

Optimization of job resources

! Knowing in advance the optimal amount of resources (CPU cores, memory, time) needed by jobs can be hard!

- Adding more **cores** doesn't *automagically*  make programs run faster, software needs to support multi-threading (OpenMP, ...), or multi-processing (MPI, ...)
- Adding more **nodes** makes no difference unless program uses **multi-processing**
- Adding **GPUs** makes no difference unless program has support for **CUDA/ROCm**
- Adding more **memory** will not make any difference unless your job needs it (*i.e.* OOM errors)

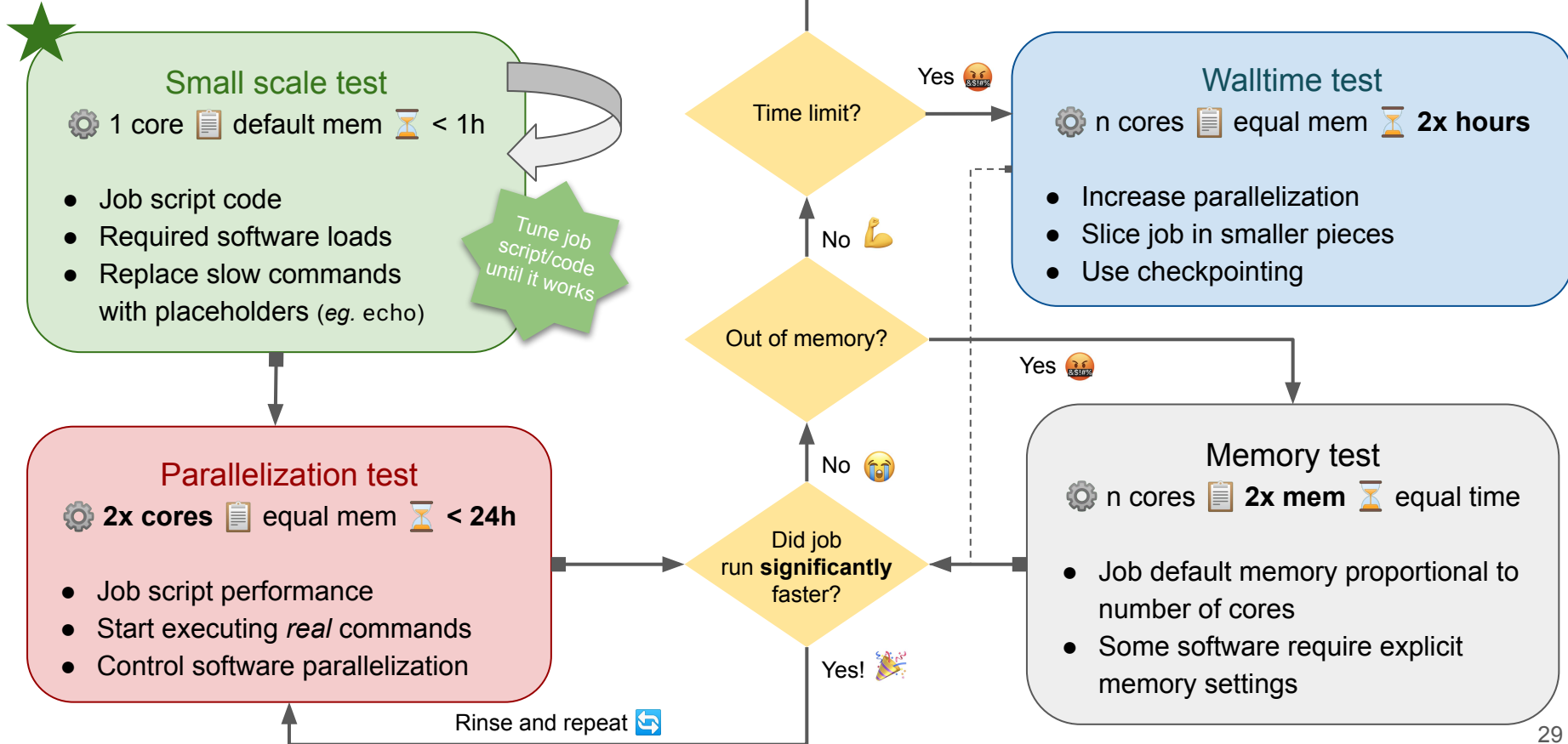
Beware of **oversubscribing** allocated cores, the software stack used in the job might have **multiple layers of parallelization** which renders the calculation of needed cores complex

- OpenBLAS starts 1 thread/core on top of program that starts 1 thread/core
- PyTorch starts 1 thread/core on top of your own Python script starting 1 thread/core

**N^2 threads
on N cores!**



Optimization of job resources



Monitoring resource usage



Resource usage in real time:

- `srun --jobid=<SLURM_JOBID> --pty bash`

Resources of completed jobs:

- XDMoD: xdmod.hpc.kuleuven.be
- `slurm_jobinfo` ----->
- `sacct` ----->
- `seff`

✓ Tier-2 UAntwerp
✓ Tier-2 KUL

★ All VSC sites

```
$ slurm_jobinfo 8627024
Name           : cpu-pin
User           : vsc10122
Partition      : ivybridge_mpi
Nodes          : node[112,115]
Cores          : 8
State          : COMPLETED
Submit        : 2023-10-17T16:18:00
Start         : 2023-10-17T16:18:14
End           : 2023-10-17T16:18:26
Reserved walltime : 00:05:00
Used walltime  : 00:00:12
Used CPU time  : 00:00:05
% User (Computation) : 67.42%
% System (I/O)   : 32.58%
Mem reserved   : 36000M
Max Mem used   : 5.36M (node112,...)
Max Disk Write : 20.48K (node112,...)
Max Disk Read  : 5.52M (node112,...)
Working directory : /vscmnt/...
```

✓ Tier-1 Hortense
✓ Tier-2 UGent
✓ Tier-2 KUL
✓ Tier-2 VUB

```
$ SACCT_FORMAT="jobid%-16,jobname%-10,user%8,state,nnodes%6,ncpus%5,elapsed,timelimit,maxrss,reqmem,totalcpu,cputime"
$ sacct -j 8627024
```

JobID	JobName	User	State	NNodes	NCPUS	Elapsed	Timelimit	MaxRSS	ReqMem	TotalCPU	CPUTime
8627024	cpu-pin	vsc10122	COMPLETED	2	8	00:00:12	00:05:00		36000M	00:05.565	00:01:36
8627024.batch	batch		COMPLETED	1	4	00:00:12		0		00:04.042	00:00:48
8627024.extern	extern		COMPLETED	2	8	00:00:12		0		00:00.001	00:01:36
8627024.0	nodesused		COMPLETED	2	8	00:00:03		5492K		00:00.949	00:00:24
8627024.1	orted		COMPLETED	1	4	00:00:03		0		00:00.571	00:00:12

Tricks to get your jobs started faster

We often get the question: "**When will my job start?**" (a.k.a. "Can you predict the future?" )

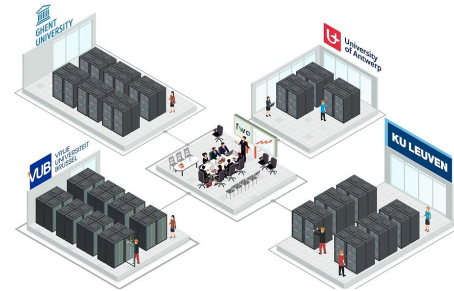
Short answer: "**It depends**" (when running jobs will finish, which additional jobs will be submitted, ...)



Total waiting (turnaround) time for jobs: **waiting time in the queue** + **time it takes to run**

Pro tips:

- **Consider all available resources:**
 - **Multiple clusters** per VSC site
 - Clusters at **other VSC sites** (very similar setup, but with minor differences)
 - **Tier-1 compute project proposal** to access Hortense (~100k CPU cores + 160 GPUs, fewer users)
 - Oversubscribed **debug partitions/clusters** where a job requesting limited resources **starts in seconds** (but may run slower)
docs.vscenrum.be/gent/tier1_hortense.html#interactive-and-debug-partition



Tricks to get your jobs started faster





We often get the question: "**When will my job start?**" (a.k.a. "Can you predict the future?" )

Short answer: "**It depends**" (when running jobs will finish, which additional jobs will be submitted, ...)



Total waiting (turnaround) time for jobs: **waiting time in the queue** + **time it takes to run**

Pro tips:

- **Don't waste time over-optimizing**, there's a ~25% chance you are sleeping  when your job completes...
- **Consider requesting *less* resources** (#cores/nodes, walltime, memory, ...), fill the gaps in the cluster! 
- Rule of thumb: jobs that request a **quarter node** (or less) usually start very quickly (but no guarantees) 
- **Balancing act** w.r.t. requested walltime & cores/nodes/memory & number of jobs 
 - Break up large long-running job into multiple independent smaller/shorter jobs (if possible)
 - Don't submit thousands (or more) of tiny jobs, pack them together (at least 15min, handful of cores)