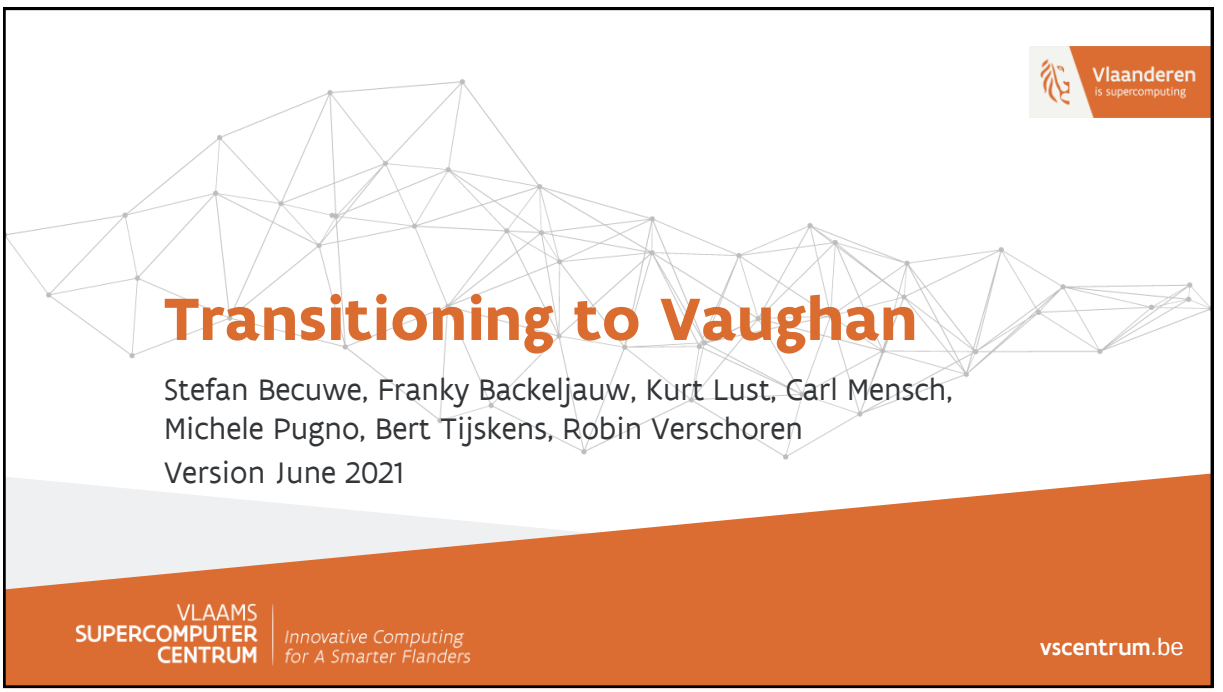


1

Recent changes

- New faces and new tasks
- Storage was swapped out for a very different system
- Hopper decomissioned, replaced with Vaughan
  - Intel-compatible but a different design philosophy
- As Torque/Moab support has been bad and development slow the past few years, the scheduler is being replaced with SLURM
  - Already on Vaughan, soon also on Leibniz
  - You'll need to change your job scripts!
- New data transfer service: Globus

VLAAMS
SUPERCOMPUTER
CENTRUM

2

New faces and new tasks

- Three people joined recently:

Carl Mensch (2020)	Robin Verschoren (2021)	Michele Pugno (60%, 2021)
		

- New tasks
  - Several team members participate in the EuroCC National Competence Centre Belgium (Carl, Bert, Stefan)
  - Kurt Lust now works for 50% for the LUMI consortium in the LUMI User Support Team. LUMI is a EuroHPC pre-exascale computer based on AMD CPUs and GPUs.

VLAAMS SUPERCOMPUTER CENTRUM

3

New storage

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

VLAAMS SUPERCOMPUTER CENTRUM

4

New storage: design principles

- Installed February 2020
- Rather than using a single file system and hardware technology for all volumes, we decided this time to go for a mixed setup
  - Traditional file systems exported over NFS may be a better choice to deal with the metadata operations overload and inefficient volume use caused by packages that install tons of small files (e.g., Python, R and MATLAB)
  - Parallel file systems offer a much better price/performance and price/volume ratio
  - Limited use of SSD
    - SSDs have poor lifespan when used in the wrong way (i.e, high write/erase load and lots of small write operations)
    - Data center quality long-life SSDs are up to 20 times more expensive per TB than hard disks
- Tried to make our storage even more independent from the cluster to make it easier to keep the storage available during system maintenance periods
  - So you'd still be able to run on other VSC-clusters

VLAAMS SUPERCOMPUTER CENTRUM

5

New storage: home and apps

- Home directories (/user volume)
  - On very expensive hardware (mirrored SSD) to improve application startup times
  - Should only be used for configuration files
  - But not:
    - Install software
    - Job output
  - Smallest volume (3.5 TB) hence small disk quota (3 GB) and you can't get more
  - Goes on backup
  - Mounted via NFS
- Applications volume (/apps)
  - On very expensive hardware (RAID SSD) to improve application startup times
  - Not user-writable
  - Mounted via NFS

VLAAMS SUPERCOMPUTER CENTRUM

6

New storage: data volume

- Data volume
  - Hard disk RAID array, but still fairly expensive technology
  - Conventional file system (XFS)
  - Should be used for fairly static data
    - So not the volume to write lots of job output to unless explicilty asked to run a particular type of jobs on this volume
    - Back-up as long as the data remains static enough
      - We may choose not to backup certain accounts or directories
  - Good place for installing your own software
    - Hint: For R and Python it is better to extend an existing module rather than do your own installation with Conda (i.e., using pip and/or easy_install in Python)
  - Exported via NFS

VLAAMS SUPERCOMPUTER CENTRUM

7

New storage: Parallel scratch file system

- Scratch
  - Parallel file system: switched to BeeGFS rather than GPFS / SpectrumScale
  - Storage:
    - Metadata on redundant SSDs (mirroring)
    - Data on 7 hard disk pools of 14 drives
  - Highest capacity of all our storage systems: 0.6 PB and room to grow
  - Highest bandwidth: up to 7 GB/s combined over all users and nodes
    - But this requires the right access profile in software
  - You can request a very high block quota but the number of files that you can store will be limited as HPC storage is designed to work with large files
  - Technologies such as HDF5 and netCDF are designed to store data in a portable and structured way that is more efficient than using tons of small files in a directory structure…

VLAAMS SUPERCOMPUTER CENTRUM

8

Quota

- Shown at login
  Or run myquota
- Current defaults:

	block quota	file quota
/home	3 GB	20,000
/data	25 GB	100,000
/scratch	50 GB	100,000

- For scratch the file quotum is actually not the number of files that you see, but the number of files occupied on the storage system. Large files can be split in multiple files (called chunk files in BeeGFS), but this is hidden from the user (and in fact increases performance).

VLAAMS SUPERCOMPUTER CENTRUM

9

You can't beat physics...

- Central storage is crucial to keep storage manageable and data safe
  - It is hard to clean a drive after a job unless we would have a single job per node policy
  - Bad experience with local drives in the past
- Hard drives and SSDs are not very compatible with the environment of a densely packed hot-running supercomputer
  - While that dense packaging is a necessity to keep communication lines short
- But a shared file system will always have a higher latency:
  - Move through more software layers
  - Some network latency also; even if you could eliminate the software layers you'd still have a considerably higher latency than with a SSD in a laptop mounted directly next to the CPU
- Which is why networked file systems are bad at small single-threaded IOPs
  - On a PC it will cost you a factor of 10 or more on a SSD
  - On networked storage it will cost you a lot more performance...
- Scaling capacity is relatively cheap, scaling bandwidth is already more expensive and scaling IOPS is extremely expensive.

VLAAMS SUPERCOMPUTER CENTRUM

10

New compute cluster

VLAAMS SUPERCOMPUTER CENTRUM

11

Cluster overview

Leibniz (2017)	Vaughan (2020-2021)
• 152 regular compute nodes	• 152 compute nodes
○ 144*128GB, 8*256GB	○ 256GB
○ 2 2.4GHz 14-core Broadwell CPUs (E5-2680v4)	○ 2 2.35GHz 32-core AMD Rome CPUs (7452)
○ No swap, small local tmp for OS	○ No swap, small local tmp for OS
• 24 256 GB nodes from Hopper still attached	
• 2 GPU compute nodes (dual Tesla P100)	
• 1 NEC SX Aurora node	
• 2 login nodes (identical CPU to the compute nodes)	• 2 login nodes (2 16-core AMD EPYC 7282 CPUs each)
• 1 visualisation node (hardware accelerated OpenGL)	

➢ The storage is a joint setup for Leibniz and Vaughan with more than 650TB capacity spread over a number of volumes

VLAAMS SUPERCOMPUTER CENTRUM

12

Why AMD?

- Similar theoretical peak performance with AMD (Rome) and Intel (Cascade Lake)
  - AMD reaches this by using more cores but sticking with AVX2 vector instructions
  - Intel reaches this by using less cores but a vector instruction set with longer vectors (AVX512)
    - Added bonus is some initial support for operations that are popular in neural networks
  - Performance per core for scalar operations similar on both
- AMD offers significantly more memory bandwidth per socket
  - AMD: 8 64-bit channels per socket, 3200 MHz data rate
  - Intel: 6 64-bit channels per socket, 2666 MHz data rate
  - However, AMD has worse memory latency but compensates this partially with larger caches
- Looking at benchmarks, we concluded that for the majority of our users AMD was the better choice
  - Codes are often memory bandwidth constrained
  - Several codes don't offer much vectorisation so more net performance on AMD
  - Benchmark mix we looked at: AMD node can do 60-80% more work than Intel node
- Diversity in the VSC: Intel Cascade Lake available elsewhere

VLAAMS SUPERCOMPUTER CENTRUM

13

The processor

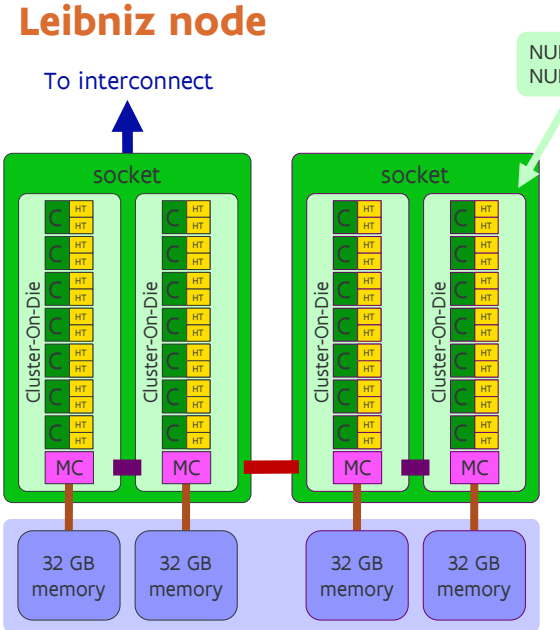
	Leibniz	Vaughan	
CPU	Intel Broadwell E5-2680v4	AMD Rome 7452	
Base clock speed	2.4 GHz	2.35 GHz	-2%
Cores/node	28	64	+130%
Vector instructions	AVX2+FMA	AVX2+FMA	
DP flops/core/cycle	16 (e.g., FMA instruction)	16 (e.g., FMA instruction)	+0%
Memory speed	8*64b*2400MHz DDR4	16*64b*3200MHz DDR4	+166%
Theoretical peak/node	851/1075/1254 Gflops (base/2.4GHz/turbo)	2400 (base)	+182% - +152%
DGEMM Gflops 50k x 50k	995 Gflops	Roughly 2 TFlops	

- Note that core speed didn't increase anymore; the number of cores did
- Expect similar to slightly higher IPC on Vaughan
- No major changes to the instruction set this time
  - But check our documentation if you are an Intel compiler user!

VLAAMS SUPERCOMPUTER CENTRUM

14

Leibniz node

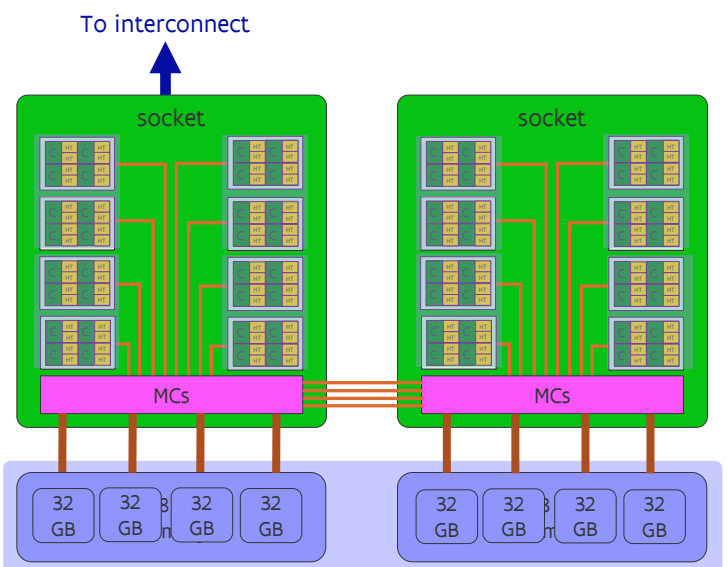


- BrENIAC Tier-1 Broadwell section
- Leibniz @ UAntwerpen

VLAAMS SUPERCOMPUTER CENTRUM

15

Vaughan (AMD Rome) node



- Vaughan @ UAntwerpen
- Hortense @ UGent (but 16 core complexes per socket)
- L3 caches shared per 4-core core complex (CCX), not per socket
  - Fast access to memory cached by another core of the same CCX, but almost as slow as main memory access otherwise.
- As on Leibniz, slower access to memory on the other socket.

VLAAMS SUPERCOMPUTER CENTRUM

16

Need to know more?

- All this hardware stuff sounds like Chinese to you?
  - Follow the "Supercomputers for Starters" of our introduction courses
  - See recorded session on the CalcUA web site, in particular the "Processors in supercomputers" part

VLAAMS SUPERCOMPUTER CENTRUM

17

Getting on Vaughan

VLAAMS SUPERCOMPUTER CENTRUM

18

Login infrastructure

- 2 login nodes similar to the compute nodes but less cores
- External names:
  - login-vaughan.hpc.uantwerpen.be: rotates between the two login nodes
  - login1-vaughan.hpc.uantwerpen.be (and login2-): For a specific name
- Internal names (between VSC-clusters)
  - login1.vaughan.antwerpen.vsc, ln1.vaughan.antwerpen.vsc (and login2/ln2)
- Reminder:
  - Accessible by ssh from within Belgium
  - Outside Belgium, the UAntwerp VPN should be used
    - UAntwerp service, not a VSC service, so support via the UA-ICT helpdesk
    - Use the Cisco AnyConnect Client on vpn.uantwerpen.be or iOS/Android stores. OS-builtin IPsec clients fail to connect from some sites.

VLAAMS SUPERCOMPUTER CENTRUM

19

Module system

VLAAMS SUPERCOMPUTER CENTRUM

20

Modules

- Basis for proper software management on a multi-user machine
  - If you're not loading modules, you're doing something wrong (unless you installed optimized software yourself)
  - And a reminder: We cannot install software from non-relocatable RPMs or that requires sudo rights to install or run
- Some software needs specific settings of environment variables, we try to hide those in the modules
- Some evolution since the installation of Leibniz
  - Moving slowly towards organising the software more in software stacks to ease retiring software that is no longer compatible with the system

VLAAMS SUPERCOMPUTER CENTRUM

21

Building your environment:
Software stack modules and application modules

- Software stack modules:
  - calcua/2020a: Enables only the 2020a compiler toolchain modules (one version of the Intel compiler and a compatible version of the GNU compilers), software built with those compilers, software built with the system compilers and some software installed from binaries
    - Older software stacks have not been reinstalled as the 2019 Intel compilers were transitional and as the 2018 compilers are not supported on CentOS 8.
  - calcua/supported: Enables all currently supported application modules: up to 4 toolchain versions and the system toolchain modules
    - Easy shortcut, but support for this software stack module may disappear
  - So it is a good practice to always load the appropriate software stack module first before loading any other module!
  - Moving away from leibniz/… and vaughan/… to calcua/… which works on all CalcUA clusters.

VLAAMS SUPERCOMPUTER CENTRUM

22

Building your environment:
Software stack modules and application modules

- 3 types of application modules
  - Built for a specific software stack, e.g., 2020a, and compiler (intel-2020a, GCCcore-9.3.0, …)
    - Modules in a subdirectory that contains the software stack version in the name
    - Compiler is part of the module name
    - Try to support a toolchain for 2 years
  - Applications installed in the system toolchain (compiled with the system compilers)
    - Modules in subdirectory system
    - For tools that don't take much compute time or are needed to bootstrap the regular compiler toolchains
  - Generic binaries for 64-bit Intel-compatible systems
    - Typically applications installed from binary packages
    - Modules in subdirectory software-x86_64
    - Try to avoid this as this software is usually far from optimally efficient on the system

VLAAMS SUPERCOMPUTER CENTRUM

23

Building your environment:
Discovering software

- `module av` : List all *available* modules
  - Depends on the software stack module you have loaded
- `module av matlab` : List all *available* modules whose name contains matlab (case insensitive)
- `module spider` : List *installed* software packages
  - Does not depend on the software stack module you have loaded
- `module spider matlab` : Search for all modules whose name contains matlab, not case-sensitive
- `module spider MATLAB/R2020a` : Display additional information about the MATLAB/R2020a module, including other modules you may need to load
- `module keyword matlab` : Uses information in the module definition to find a match
  - Does not depend on the software stack module you have loaded
- `module help` : Display help about the module command
- `module help baselibs` : Show help about a given module
  - Specify a version, e.g., `baselibs/2020a` to get the most specific information
- `module whatis baselibs` : Shows information about the module, but less than help
  - But this is the information that module keyword uses
- `module -t av |& sort -f` : Produce a sorted list of all available modules, case insensitive

VLAAMS SUPERCOMPUTER CENTRUM

24

Building your environment:
Discovering software

```
$ module spider parallel/20180422
…
 You will need to load all module(s) on any one of the lines below before the
"parallel/20180422" module is available to load.

      calcua/2016b
      calcua/2017a
      calcua/2018a
      calcua/2018b
      …
```

- It does not mean that you need to load all those calcua modules before you can load the parallel module. You have to chose one of those lines and which one depends also on other software that you want to use with parallel.

VLAAMS SUPERCOMPUTER CENTRUM

25

Building your environment:
Enabling and disabling packages

- Loading software stack modules and packages:
  - `module load calcua/2020a` : Load a software stack module (list of modules)
  - `module load MATLAB/R2020a` : Load a specific version
    - Package has to be available before it can be loaded!
  - `module load MATLAB` : Load the default version (usually the most recent one)
    - But be prepared for change if we change the `calcua/supported`!
- Unloading packages:
  - `module unload MATLAB` : Unload MATLAB settings
    - Does not automatically unload the dependencies
  - `module purge` : Unload all modules (incl. dependencies and cluster module)
- `module list` : List all loaded modules in the current session
- Shortcut:
  - `ml` : Without arguments: shortcut for `module list`
  - `ml calcua/2020a` : With arguments: Load the modules

VLAAMS SUPERCOMPUTER CENTRUM

26

Need a bigger refresh on modules?

- Check the "Preparing for your job"-part in the recordings of the HPC@UAntwerp introduction.

VLAAMS SUPERCOMPUTER CENTRUM

27

Slurm workload manager

VLAAMS SUPERCOMPUTER CENTRUM

28

Slurm workload manager

- Torque and Moab has served us well for over 10 years
- However, in 2016 Adaptive Computing was acquired by another company and since then development has nearly halted and support is bad
- New resource manager and scheduling software: Slurm
  - At the moment the dominant scheduler in academic computing centers
  - Developed at Lawrence Livermore National Lab as an extensible resource manager (2002)
  - Sophisticated scheduling features have been added since 2008
  - Development now largely coordinated by a spin-off, SchedMD LLC.
- Single package for resource management and scheduling ensures more logical command line options.
- There is a plugin that provides some compatibility with Torque, but since it is buggy and limited we prefer to fully switch to Slurm job scripts
  - We've been preparing for this for over 2 years by stressing those features in Torque that resemble those of Slurm during the introductory courses

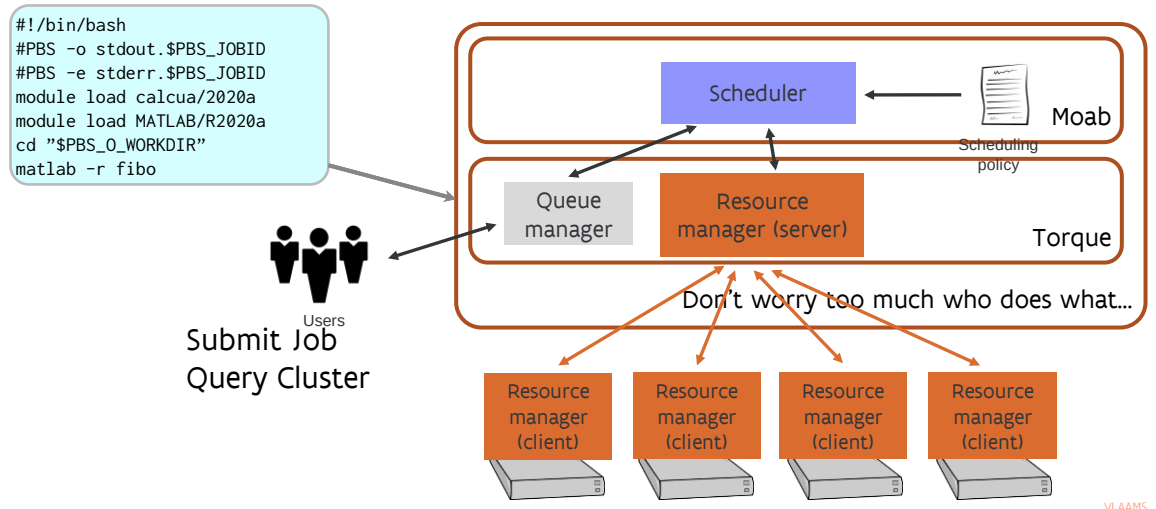
VLAAMS SUPERCOMPUTER CENTRUM

29

Job workflow with Torque & Moab

What happened behind the scenes

```
#!/bin/bash
#PBS -o stdout.$PBS_JOBID
#PBS -e stderr.$PBS_JOBID
module load calcua/2020a
module load MATLAB/R2020a
cd "$PBS_O_WORKDIR"
matlab -r fibo
```



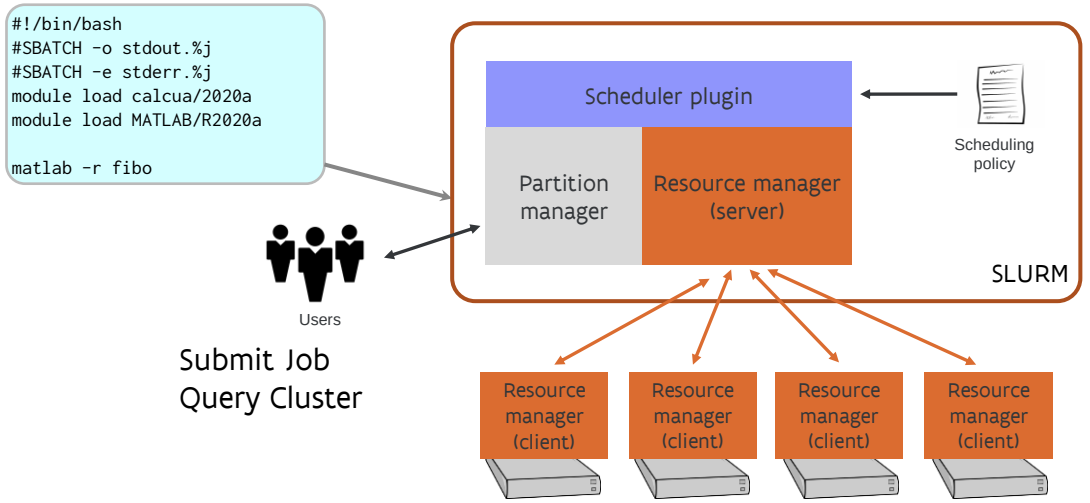
VLAAMS SUPERCOMPUTER CENTRUM

30

Job workflow in Slurm
What happens behind the scenes

```
#!/bin/bash
#SBATCH -o stdout.%j
#SBATCH -e stderr.%j
module load calcua/2020a
module load MATLAB/R2020a

matlab -r fibo
```



VLAAMS SUPERCOMPUTER CENTRUM

31

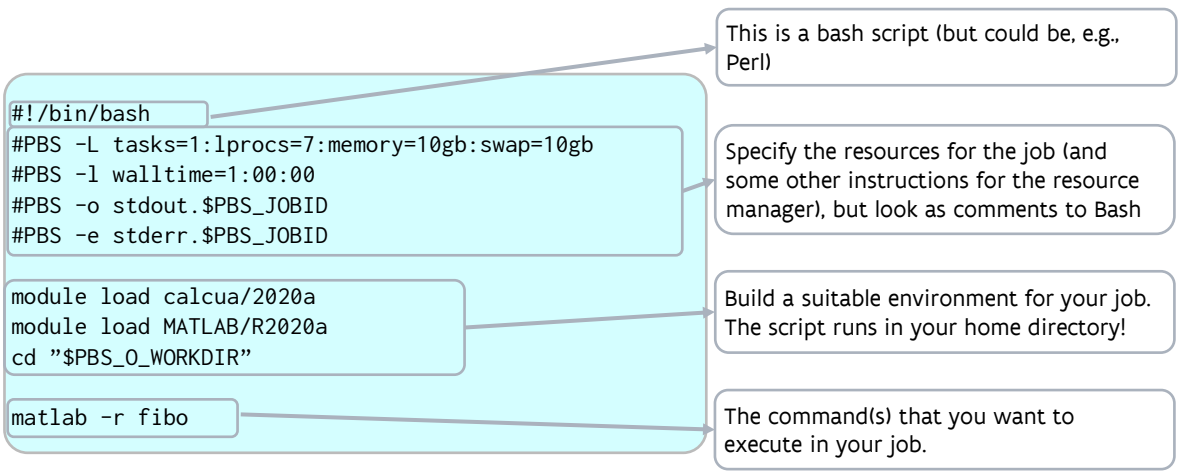
Important concepts

- Node: As before, the hardware that runs a single operating system image
- Core: Physical core in a system
- CPU: A virtual core in a system (hardware thread). On CalcUA clusters: core = CPU.
- Partition: A job queue with limits and access control
- Job: A resource allocation request
- Job step: A set of (possibly parallel) tasks within a job
  - The job script itself is a special step called the batch job step
  - A MPI application typically runs in its own job step
- Task: Executes in a job step and corresponds to a Linux process:
  - A shared memory program is a single task
  - MPI application: Each rank (=MPI process) is a task
    - Pure MPI: Each task uses a single CPU (also single core for us)
    - Hybrid MPI/OpenMP: Each task uses multiple CPUs
  - A single task can not use more CPUs than available in a single node

VLAAMS SUPERCOMPUTER CENTRUM

32

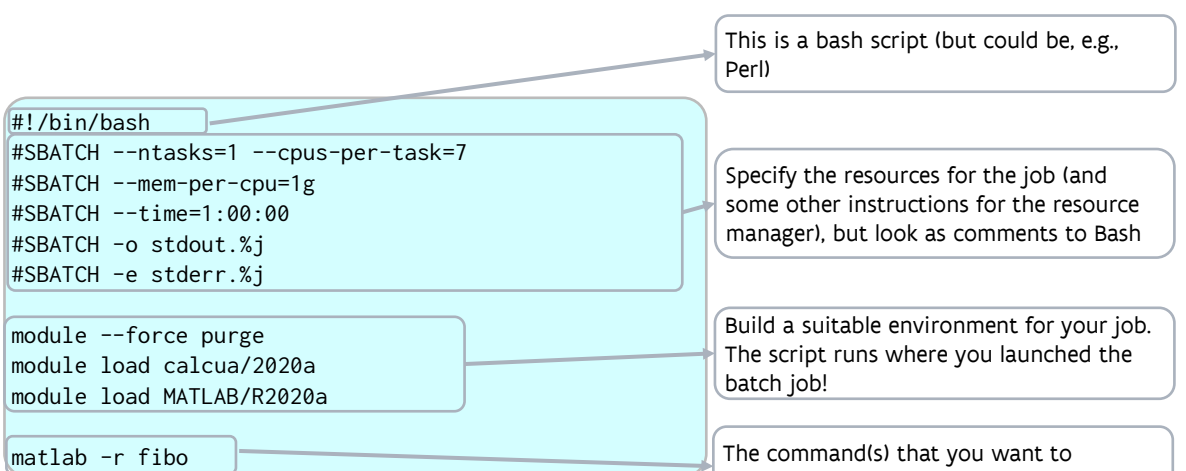
The job script (Torque)



VLAAMS SUPERCOMPUTER CENTRUM

33

The job script (Slurm)



VLAAMS SUPERCOMPUTER CENTRUM

34

Specifying resources in job scripts (Torque and Slurm)

example.sh

```
#!/bin/bash
#PBS -N name_of_job
#PBS -L tasks=5:lprocs=1:memory=1gb:swap=1gb
#PBS -l walltime=01:00:00

#PBS -m e
#PBS -M <your e-mail address>

#PBS -o stdout.file
#PBS -e stderr.file

cd "$PBS_O_WORKDIR"
module load calcua/2020a vsc-tutorial

mpirun mpi_hello
```

```
#!/bin/bash
#SBATCH --jobname=name_of_job
#SBATCH --ntasks=5 --cpus-per-task=1
#SBATCH --mem-per-cpu=1g
#SBATCH --time=01:00:00

#SBATCH --mail-type=END
#SBATCH --mail-user=<your e-mail address>

#SBATCH -o stdout.file
#SBATCH -e stderr.file

module --force purge
module load calcua/2020a vsc-tutorial

srun mpi_hello
```

$ qsub example.sh

$ sbatch example.sh

VLAAMS SUPERCOMPUTER CENTRUM

35

The job environment
Some important differences

Torque/MOAB	Slurm
Job starts in a clean login environment (unless –V is used)	Job copies the environment from which the job was submitted ➢ Risk: Not everything in the environment makes sense on a compute node
Job starts in the home directory ➢ `cd $PBS_O_WORKDIR` to move to the directory from which the job was submitted	Job starts in the directory from which the job was submitted ➢ Remove `cd $PBS_O_WORKDIR` or you'll end up in your home directory!
No process starter for MPI programs included ➢ Made starting of hybrid MPI/OpenMP jobs cumbersome	Comes with a very versatile process starter, `srun` ➢ If you use `I_MPI`-environment variables in your job script, you'll have to change some!

VLAAMS SUPERCOMPUTER CENTRUM

36

Slurm resource requests

37

Slurm resource requests

- Options can be specified in various ways:
  - Lowest priority: In a batch script on lines starting with `#SBATCH` at the top of your script.
    - This should be the first block of comment lines
  - Several options can be passed through environment variables also.
    - Overwrite values on `#SBATCH` lines
    - Dangerous as you can easily forget they exist…
    - See the sbatch manual page.
  - Highest priority: On the command line of `sbatch` and two related commands (`srun` and `salloc`)
    - Specify **before** the job script as otherwise they will be interpreted as arguments to the job script!
    - Overwrites values on `#SBATCH` lines and environment variables.

38

Slurm resource requests

- Two variants for the options:
  - Long variant (with double dash): `--long-option=value` or `--long-option value`
  - Sometimes there is a short, single-letter variant (with single dash): `-S value` or `-Svalue`

VLAAMS SUPERCOMPUTER CENTRUM

39

Slurm resource requests
Requesting processing resources

- Very similar to the Torque version 2 syntax (with `-L`)
- You don't request processing as physical resources such as nodes and cores, but logical ones: task slots and CPUs (= hyperthreads = cores on our current setup)
  - Task: A space for a single process
  - CPUs for a task:
    - The logical processors from Torque
    - In most cases the number of computational threads for a task
- Specifying the number of tasks:
  - Long option: `--ntasks=10` or `--ntasks 10` will request 10 tasks
  - Short option: `-n 10` or `-n10` will request 10 tasks
- Specifying the number of CPUs per task:
  - Long option: `--cpus-per-task=4` or `--cpus-per-task 4` will request 4 CPUs for each task
  - Short option: `-c 4` or `-c4` will request 4 CPUs for each task

VLAAMS SUPERCOMPUTER CENTRUM

40

Slurm resource requests
Requesting wall time

- Time formats:
  - `mm` : Minutes
  - `mm:ss` : Minutes and seconds (and not hours and minutes!)
  - `hh:mm:ss` : Hours, minutes and seconds
  - `d-hh` : Days and hours
  - `d-hh:mm` : Days, hours and minutes
  - `d-hh:mm:ss` : Days, hours and minutes
- Maximum on Vaughan is 3 days
- Requesting wall time:
  - Long option: `--time=12:00:00` or `--time 12:00:00` will request 12 hours of walltime
  - Short option: `-t 12:00:00` or `-t12:00:00` will request 12 hours of walltime

VLAAMS SUPERCOMPUTER CENTRUM

41

Slurm resource requests
Requesting memory

- This is different from Torque
  - Torque version 2 syntax: physical (RAM) and virtual memory per task
  - Slurm
    - Specified per CPU and not per task
    - Just a single parameter corresponding to RAM
- Specifying the amount: Only integers allowed!
  - `10k` or `10K` is 10 kilobyte
  - `10m` or `10M` is 10 megabyte (default)
  - `10g` or `10G` is 10 gigabyte but `3.5g` is invalid!
- Requesting memory:
  - Long option: `--mem-per-cpu=3072m` or `--mem-per-cpu 3072m` will allocate 3072 MB = 3 GB per task.
  - There is no short option.
- 240 GB available on Vaughan, so `3840m` per core.

VLAAMS SUPERCOMPUTER CENTRUM

42

Slurm resource requests
Constraints to specify features

- Vaughan is currently a homogeneous cluster which is why features have little use.
- Expect to see them re-introduced when Leibniz is transfered to Slurm, for the same reasons as in Torque
  - E.g., to help the scheduler to bundle tasks that use more than the standard amount of memory per core on a single node with 256 GB RAM instead of two nodes with 128 GB.
- Features are specified with `--constraint`
  - `--constraint=mem256` would request a node with the mem256 feature which we may use on Leibniz in the future
  - They can be combined in very powerful ways, e.g., to get all nodes in one rack. But there is a better solution for this…

VLAAMS SUPERCOMPUTER CENTRUM

43

Slurm resource requests
Faster communication

- The communication network of both Vaughan and Leibniz have a tree structure
  - Nodes are grouped on a number of edge switches (24 per switch on Leibniz, up to 44 on Vaughan)
  - These switches are connected with one another through a second level of switches, the top switches
  - Hence traffic between two nodes either passes through just a single switch or through three switches (edge – top – edge)
- Some programs are extremely latency-sensitive and run much better if they only get nodes connected to a single switch
  - Example: GROMACS
- Requesting nodes on a single switch: `--switches=1`
  - But this will increase your waiting time…

VLAAMS SUPERCOMPUTER CENTRUM

44

Slurm resource requests
Requesting resources: Discussion

- Slurm has other ways of requesting resources
  - See the manual page for sbatch (google man sbatch)
  - Be very careful when you experiment with those as they will more easily lead to inefficient use of the nodes
    - E.g., you may be allocating resources in a way that a node may only be used for a single job, even if you have more jobs in the queue.
- If other options are needed, e.g., once we transfer the GPU nodes to Slurm, they will be documented and we will mention that in our mailings
  - Not knowing something because you didn't read our emails is not our fault!

VLAAMS SUPERCOMPUTER CENTRUM

45

Slurm resource requests
Partitions

- Torque automatically assigned a job to the optimal queue on our systems
  Slurm does not automatically assign a job to the optimal partition (Slurm equivalent of Torque queue)
- The default partition is OK for most jobs on our cluster
- Partitions: `scontrol show partition` or in the output of `sinfo`
  - `vaughan` : Default partition, for regular jobs up to 3 days, single user per node
  - `short` : Partition for shorter jobs, up to 6 hours. Priority boost and higher node limit.
  - `debug` : Partition for debugging scripts. Dedicated resources, but not more than 2 nodes, and just a single job in the queue
- Specifying the partition: No need to specify if the default partition is OK
  - Long option: `--partition=short` or `--partition short` submits the job to partition short
  - Short option: `-p short` or `-pshort` submits the job to partition short
- Check the documentation page for the individual cluster for the available partitions (e.g., for vaughan)

VLAAMS SUPERCOMPUTER CENTRUM

46

Slurm resource requests
Single user per node policy

- Policy motivation :
  - Parallel jobs can suffer badly if they don't have exclusive access to the full node as jobs influence each other (L3 cache, memory bandwidth, communication channels, ….)
  - If a node of Vaughan is too large for a single user, Leibniz and the old Hopper nodes are the better alternative
  - Slurm is better at controlling resources and limiting interference between jobs than Torque, but there are still resources that cannot be controlled properly and there may be ways to escape from the control of Slurm
- Remember: the scheduler will (try to) fill up a node with several jobs from the same user
  - But could use some help from time to time on heterogeneous clusters

if you don't have enough work for a single node,
you need a good PC/workstation and not a supercomputer

VLAAMS SUPERCOMPUTER CENTRUM

47

Non-resource-related parameters
Job name

- Just as in Torque it is possible to assign a name to your job
  - Long option: `--jobname=myjob` or `--jobname myjob` assigns the name myjob to the job.
  - Short option: `-J myjob` or `-Jmyjob`

VLAAMS SUPERCOMPUTER CENTRUM

48

Non-resource-related parameters
Redirecting I/O

- By default Slurm redirects stdout and stderr to the file `slurm-<jobid>.out`.
  - And that file is present as soon as the job starts and does output.
- Redirecting all output to a different file:
  - Long option: `--output=myfile.txt` or `--output myfile.txt` will redirect all output to myfile.txt
  - Short option: `-o myfile.txt`
- Separating output to stderr and redirect that output to a different file:
  - Long option: `--error=myerrors.txt` or `--error myerrors.txt` will redirect output to stderr to myerrors.txt
  - Short option: `-e myerrors.txt`

VLAAMS SUPERCOMPUTER CENTRUM

49

Non-resource-related parameters
Redirecting I/O

- Hence
  - No `--output` and no `--error` : stdout and stderr redirected to `slurm-<jobid>.out`
  - `--output` but no `--error` : stdout and stderr redirected to the given file
  - No `--output` but `--error` specified: stdout redirected to `slurm-<jobid>.out`, stderr to the file given with `--error`.
  - Both `--output` and `--error` : stdout redirected to the file pointed to by `--output` and stderr redirected to the file pointed to by `--error`.
- It is possible to insert codes in the file name that will be replaced at runtime with the corresponding Slurm information.
  - Examples are “%J” for the job name or “%j” for job id.
  - See the manual page of sbatch, section “filename pattern”.

VLAAMS SUPERCOMPUTER CENTRUM

50

Non-resource-related parameters
Notifications

- The cluster can notify you by mail when a job starts, ends or is aborted.
- When?
  - `--mail-type=BEGIN,END` : At the start and end of the job
  - `--mail-type=ALL` : At a selection of important events
  - But there are more options, e.g., after certain fractions of the requested wall time have expired.
    - `--mail-type=TIME_LIMIT_90` : when the job has reached 90 percent of its time limit
- To whom?
  - `--mail-user=first.last@uantwerpen.be`
  - The default value is the mail address associated with the VSC-account of the submitting user.

VLAAMS SUPERCOMPUTER CENTRUM

51

Non-resource-related parameters
Job dependencies

- Just as with Torque/Moab it is possible to set up workflows submitting all jobs at once but specifying dependencies to ensure that jobs don't start before their input has been produced.
- Specified with `--dependency`

  Some examples:
  - `--dependency=afterany:job1:job2` : Start when the jobs with jobID job1 and job2 have finished.
  - `--dependency=afterok:job1` : Start when job1 has completed successfully
    - And there is a similar option for individual tasks between two job arrays of the same size
  - More options: See the manual page of the Slurm sbatch command.
- Note that currently if a dependency cannot start, it will not be removed from the queue automatically so that you can see that it did not start when you check your jobs
  - We may change that in the future.

VLAAMS SUPERCOMPUTER CENTRUM

52

Slurm job environment

- A Slurm job inherits the environment from the shell from which the allocation was made
  - This includes loaded modules, which can be a problem as those modules were not loaded in the context of the compute node
  - Hence it is best to clean and rebuild the environment:

```
module --force purge
module load calcua/2020a
module load MyApplication
```

  - We are looking for a more elegant way and announce if we found one.

VLAAMS SUPERCOMPUTER CENTRUM

53

Slurm job environment
Predefined variables

- Slurm also defines a lot of variables when a job is started. Some are not always present.
  - `$SLURM_SUBMIT_DIR` : The directory from which sbatch was invoked
  - `$SLURM_JOB_ID` : The Slurm jobID
  - `$SLURM_JOB_NAME` : The name of the job
  - `$SLURM_NTASKS` : The number of tasks requested/allocated for the job if this was specified in the request, otherwise it depends on how the request was made
  - `$SLURM_CPUS_PER_TASK` : Number of CPUs per task if this was specified in the request
  - `$SLURM_JOB_NODELIST` : List of nodes allocated to the job
  - Additional variables for array jobs, see the example later in the session
- Full list: sbatch manual page, section "OUTPUT ENVIRONMENT VARIABLES".
  - Not all variables are always defined!
- And there are of course the `VSC_*` variables and the various `EB*` variables when modules are loaded.

VLAAMS SUPERCOMPUTER CENTRUM

54

Slurm job environment
Predefined variables

- To check:

print-variables.slurm

```
#!/bin/bash
#SBATCH -n 16 -c 8 --mem-per-cpu=3840m
#SBATCH -t 00:05:00

module --force purge
module load calcua/2020a
module load vsc-tutorial

echo -e "\nSLURM environment variables:\n$(env | grep ^SLURM_)\n"
echo -e "VSC environment variables:\n$(env | grep ^VSC_)\n"
echo -e "EasyBuild-defined environment variables:\n$(env | egrep "EBROOT|EBVERSION" | sort)\n"
```


Slurm resource requests

- Full list of options in the sbatch manual page (google “man sbatch”)
  - We discussed the most common ones.
- Remember that available resources change over time as new hardware is installed and old hardware is decommissioned.
  - Share your scripts
  - But understand what those scripts do (and how they do it)
  - And don’t forget to check the hardware you request/use
  - And realise that in parallel computing the optimal set of resources also depends on the size of the problem you are solving!

Share experience

Slurm commands

VLAAMS SUPERCOMPUTER CENTRUM

57

Slurm commands
Submitting a batch script: sbatch

- Basically the equivalent of qsub in Torque
- `sbatch <sbatch arguments> MyJobScript <arguments of job script>`
  - Exits immediately when the job is submitted, so it does not wait for the job to start or end
  - Can also read the job script from stdin instead
- What it does:
  - Makes a copy of the environment as seen by the command (exported environment variables)
  - Submits the job script to the selected partition
- What Slurm then does after sbatch returns:
  - Creates the allocation when resources become available
  - Creates the batch job step in which the batch script runs, using the environment saved by sbatch and passing the command line arguments of the job script to the job script
- The sbatch command returns the job id but as part of a sentence
  - Return just the jobid for a succesfully submitted script: Use `--parsable` (may work differently in the future)

VLAAMS SUPERCOMPUTER CENTRUM

58

Slurm commands
Starting a new job step: srun

- `srun` can be used in a job script to start parallel tasks
  - Can be used from the login nodes also with the same command line options as `sbatch` and will then request an allocation before running the tasks, but the results may be unexpected, in particular with MPI programs
- In Slurm terminology, `srun` creates a job step that can run one or more parallel tasks
  - And for advanced users it is possible to run multiple job steps simultaneously, each using a part of the allocated resources
- It is the Swiss Army Knife in Slurm to create and sometimes manage tasks within a job
  - The best way of starting MPI programs in Slurm jobs
- Best shown through examples later in this tutorial, but it is impossible to cover all possibilities

VLAAMS SUPERCOMPUTER CENTRUM

59

Slurm commands
Creating only an allocation: salloc

- Dangerous command but very useful for interactive work
  - But you have to realise very well what you're doing and understand environments
- What `salloc` does:
  - Request the resources (specified on the command line or through environment variables) to Slurm and wait until they are allocated
  - Then starts a shell on the node where you executed salloc (usually the login node)
    - And this is the confusing part as most likely the shell prompt will look identical to the one you usually get so you won't realise you're still working in the allocation
  - Frees the resources when you exit the shell or your requested time expires
- From the shell you can then start job steps on the allocated resources using `srun`.

VLAAMS SUPERCOMPUTER CENTRUM

60

Slurm commands
Getting an overview of your jobs in the queue: squeue

- Check the status of your own jobs:

```
$ squeue
     JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
     26170   vaughan     bash vsc20259  R       6:04      1 r1c01cn4
```

  - The column ST shows the state of the job. There are roughly 25 different states. Some popular ones:
    - R for running
    - PD for pending (waiting for resources)
    - F for failed, but this is only visible for a few minutes after finishing
    - CD for succesful completion, but this is only visible for a few minutes after finishing
    - See "JOB STATE CODES" on the squeue manual page.
  - The column NODELIST(REASON) will show the nodes for a running job, or reasons why a job is pending
    - Roughly 30 possibilities, see "JOB REASON CODES" on the squeue manual page.

Slurm commands
Getting an overview of your jobs in the queue: squeue

- Getting more information:
  - `squeue --long` or `squeue -l` reports slightly more information (also the requested wall time)
  - `squeue --steps` or `squeue -s` also shows the currently active job steps for each job
    - You may want to combine with `--jobs` or `-j` to specify which jobs you want to see information for in a comma-separated list.
  - It is possible to specify your own output format with `--format` or `-o`, or `--Format` or `-O`, the latter with longer labels as the Latin alphabet doesn't have enough letters for all options of `--format`.
    - See the squeue manual page, look for the command line option.
- Getting an estimate for the start time of jobs: `squeue --start`
  - But jobs may start later because higher priority jobs enter the queue
  - Or may start sooner because other jobs use far less time than their requested wall clock time

Slurm commands
Deleting a job: scancel

- Equivalent of `qdel` in Torque
- Cancel a single job: `scancel 12345` will kill the job with jobid `12345` and all its steps.
  - For a job array (see the examples later on) it is also possible to just kill some jobs of the array, see our documentation.
- It is possible to kill a specific job step (e.g., if you suspect an MPI program hangs but you still want to execute the remainder of the job script to clean up and move results):
  `scancel 56789.1` will kill job step `1` from job `56789`.
- If you want to remove all pending jobs that you still have in the queue:
  `scancel --state=pending --user=vsc20XXX`
  (and the --user may not be needed).
- scancel manual page

VLAAMS SUPERCOMPUTER CENTRUM

63

Slurm commands
Real-time information about running jobs: sstat

- Show (a lot of) real-time information about a particular job or job step:
  `sstat -j 12345`
  `sstat -j 56789.1`
- It is possible to specify a subset of fields to display using the `-o`, `--format` or `--fields` option.
- Example for an MPI job: Get an idea of the load balancing:

```
$ sstat -a -j 12345 -o JobID,MinCPU,AveCPU
       JobCPU.     MinCPU.     AveCPU
------------ ---------- ----------
12345.extern             213503982+
12345.batch   00:00.000  00:00.000
12345.0        22:54:20   23:03:50
```

shows for each job step the minimum and average amount of consumed CPU time. Step 0 in this case is an MPI job, and we see that the minimum CPU time consumed by a task is close to the average which indicates that the job may be running fairly efficiently and that the load balance is likely OK.

VLAAMS SUPERCOMPUTER CENTRUM

64

Slurm commands
Information about running jobs: sstat

- Checking resident memory:

```
$ sstat -a -j 12345 -o JobID,MaxRSS,MaxRSSTask,MaxRSSNode
       JobID     MaxRSS MaxRSSTask MaxRSSNode
------------ ---------- ---------- ----------
12345.extern
12345.batch       4768K          0   r1c06cn3
12345.0         708492K         16   r1c06cn3
```

  shows that the largest process in the MPI job step is consuming roughly 700MB at the moment and is task 16 and running on r1c06cn3.

- More information: sstat manual page.

VLAAMS SUPERCOMPUTER CENTRUM

65

Slurm commands
Information about (terminated) jobs: sacct

- `sacct` shows information kept in the job accounting database.
  - So for running jobs the information may enter only with a delay
  - The command to check resource use of a finished application
- Default output for a job:

```
$ sacct -j 12345
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
12345        NAMD-S-00+    vaughan antwerpen+         64  COMPLETED      0:0
12345.batch       batch            antwerpen+         64  COMPLETED      0:0
12345.extern     extern            antwerpen+         64  COMPLETED      0:0
12345.0           namd2            antwerpen+         64  COMPLETED      0:0
```

- Select what you want to see:
  - `--brief` : Very little output, just the state and exit code of each step
  - `--long` : A lot of information, even more than `sstat`
  - `-o` or `--format` : Specify the columns you want to see

VLAAMS SUPERCOMPUTER CENTRUM

66

Slurm commands
Information about (terminated) jobs: sacct

- Example: Get resource use of a job:

```
$ sacct -j 12345 --format JobID,AllocCPUS,MinCPU%15,AveCPU%15,MaxRSS,AveRSS --units=M
       JobID  AllocCPUS          MinCPU          AveCPU     MaxRSS     AveRSS
------------ ---------- --------------- --------------- ---------- ----------
12345                64
12345.batch          64        00:00:00        00:00:00      4.62M      4.62M
12345.extern         64        00:00:00        00:00:00          0          0
12345.0              64     11-03:40:46     11-03:40:46  28014.53M  28014.53M
```

  - This was a single node shared memory job which is why the CPU time and memory consumption per task are high.
  - `--units=M` to get output in megabytes rather than kilobyes
  - `%15` in some field names: Use a 15 character wide field rather than the standard width
- List of all fields: `sacct --helpformat` or `sacct -e`

Slurm commands
Information about (terminated) jobs: sacct

- Selecting jobs to show information about:
  - By default: All jobs that have run since midnight
  - `--jobs` or `-j` : give information about a specific job or jobs (when specifying multiple jobids separated by a comma)
  - `--starttime=<time>` or `-S <time>` : Jobs that have been running since the indicated start time, format: HH:MM[:SS] [AM|PM], MMDD[YY] or MM/DD[/YY] or MM.DD[.YY], MM/DD[/YY]-HH:MM[:SS] and YYYY-MM-DD[THH:MM[:SS]] ([] denotes an optional part)
  - `--endtime=<time>` or `-E <time>` : Jobs that have been running before the indicated end time.
- There are way more features to filter jobs, but some of them are mostly useful for system administrators
- More information: sacct manual page

Slurm commands
Getting an overview of the cluster: sinfo

- Show information about partitions (=queues) and nodes in the cluster
- Default behaviour: Per partition

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
vaughan*     up 3-00:00:00    146  alloc r1c01cn[1-4],r1c02cn[1-4], …
vaughan*     up 3-00:00:00      6   idle r1c03cn4,r1c04cn3,r2c02cn2, …
short        up    6:00:00    146  alloc r1c01cn[1-4],r1c02cn[1-4], …
short        up    6:00:00      6   idle r1c03cn4,r1c04cn3,r2c02cn2, …
debug        up    1:00:00    146  alloc r1c01cn[1-4],r1c02cn[1-4], …
debug        up    1:00:00      6   idle r1c03cn4,r1c04cn3,r2c02cn2, …
```

  - Shows that jobs have a wall time limit of 3 days in partition vaughan
  - 146 nodes are running jobs and 6 are idle at the moment
    - Note that this does not mean that your job will start, as you may already be using the maximum allowed for a single user.

VLAAMS SUPERCOMPUTER CENTRUM

69

Slurm commands
Getting an overview of the cluster: sinfo

- Node-oriented overview:

```
$ sinfo -N -l
NODELIST   NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
r1c01cn1       1     short   allocated 64     2:32:1 245760        0      1 r1,r1c01 none
r1c01cn1       1  vaughan*   allocated 64     2:32:1 245760        0      1 r1,r1c01 none
r1c01cn1       1     debug   allocated 64     2:32:1 245760        0      1 r1,r1c01 none
r1c01cn2       1     short   allocated 64     2:32:1 245760        0      1 r1,r1c01 none
r1c01cn2       1  vaughan*   allocated 64     2:32:1 245760        0      1 r1,r1c01 none
r1c01cn2       1     debug   allocated 64     2:32:1 245760        0      1 r1,r1c01 none
```

  Shows to which partitions a node belongs
  - Shows the memory that can be allocated in KB
  - Shows the structure of the node in the S:C:T column: 2:32:1 stands for 2 sockets, 32 cores per socket, 1 HW thread per physical core (CPU in Slurm)
  - `AVAIL_FE` show available features for the node

VLAAMS SUPERCOMPUTER CENTRUM

70

Slurm commands
Advanced job control: scontrol

- The `scontrol` command is mostly for administrators, but some of its features are useful for regular users also, and in particular the `show` subcommand to show all kinds of information about your job.
- Show information about a running job:
  `$ scontrol -d show job 12345`
  will show a lot of information about the job with jobid 12345
- To get a list of node names in a job script that is not in abbreviated notation but can be used to generate node lists for programs that require this (such as NAMD in some situation):
  `$ scontrol show hostnames`
  in the context of a job will show the allocated host names, one per line:

```
r5c09cn3$ echo $SLURM_NODELIST
r5c09cn[3-4]
r5c09cn3$ scontrol show hostnames
r5c09cn3
r5c09cn4
```

VLAAMS SUPERCOMPUTER CENTRUM

71

Slurm examples

VLAAMS SUPERCOMPUTER CENTRUM

72

Starting a shared memory job

- A single task with multiple CPUs per task
- Shared memory programs start like any other program, but you will likely need to tell the program how many threads it can use (unless it can use the whole node).
  - Depends on the program, and the autodetect feature of a program usually only works when the program gets the whole node.
    - e.g., MATLAB: use `maxNumCompThreads(N)`
  - Many OpenMP programs use the environment variable OMP_NUM_THREADS:
    `export OMP_NUM_THREADS=7`
    will tell the program to use 7 threads.
    - Intel OpenMP recognizes Slurm CPU allocations
  - MKL-based code: MKL_NUM_THREADS can overwrite OMP_NUM_THREADS for MKL operations
  - OpenBLAS (FOSS toolchain): OPENBLAS_NUM_THREADS
  - Check the manual of the program you use!
    - NumPy has several options depending on how it was compiled…

Starting a shared memory job

- Example script :

generic-omp.slurm

```
#!/bin/bash
#SBATCH --job-name=OpenMP-demo
#SBATCH --ntasks=1 --cpus-per-task=64      ← 1 task with 64 CPUs (so 64 threads)
#SBATCH --mem=2g                           ← 2 gb per CPU, so 128 GB total memory
#SBATCH --time=00:05:00
module --force purge
module load calcua/2020a                   ← load the software stack module
module load vsc-tutorial                   ← load vsc-tutorial, which loads the Intel
                                             toolchain (for the OpenMP run time)
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK ← Set the number of threads to use
export OMP_PROC_BIND=true                  ← Will cause the threads to be nicely spread
                                             over the cores and stay on the core
omp_hello                                  ← Run the program
```

Starting a MPI job

- Every distributed memory program needs a program starter as it uses more than one process
  - Some packages come with their own command to launch the program that uses the system starter internally
  - Check the manual, it may have an option to explicitly set the program starter which is used internally
- Preferred program starter for Slurm: srun
  - It knows best about how Slurm wants to distribute processes across nodes and CPUs
  - It usually needs no further arguments if you requested resources in the right way (with `--ntasks=` the number of MPI ranks and `--cpus-per-task=1`)
- mpirun will work but it does depend on variables that are set in the intel module
  - So ensure that the module is reloaded in your job script as those variables are not set on the login nodes
  - Don't set I_MPI_HYDRA_BOOTSTRAP or I_MPI_HYDRA_RMK in your job script!

VLAAMS SUPERCOMPUTER CENTRUM

75

Starting an MPI job

- (Intel MPI) example script :

generic-mpi.slurm

```
#!/bin/bash
#SBATCH --job-name mpihello
#SBATCH --ntasks=128 --cpus-per-task=1   ← 128 MPI processes, i.e., 2 nodes with 64
#SBATCH --mem-per-cpu=1g                    processes each on Vaughan
#SBATCH --time=00:05:00
module --force purge
module load calcua/2020a                 ← load the software stack module
module load vsc-tutorial                 ← load vsc-tutorial, which loads the Intel
                                            toolchain (for the MPI libraries)
srun mpi_hello                           ← run the MPI program (mpi_hello)
                                            srun communicates with the resource
                                            manager to acquire the number of nodes,
                                            cores per node, node list etc.
```

VLAAMS SUPERCOMPUTER CENTRUM

76

Starting a hybrid MPI job on Slurm

- Contrary to Torque/Moab, we need no additional tools to start hybrid programs.
  - So no need for torque-tools or vsc-mympirun
  - srun does all the miracle work (or mpirun in Intel MPI provided the environment is set up correctly)
- Example (next slide)
  - 8 MPI processes
  - Each MPI process has 16 threads => Two full nodes on Vaughan
  - In fact, the OMP_NUM_THREADS line isn't needed with most programs that use the Intel OpenMP implementation

VLAAMS SUPERCOMPUTER CENTRUM

77

Starting a hybrid MPI job on Slurm (2)

generic-hybrid.slurm

```
#!/bin/bash
#SBATCH --job-name hybrid_hello
#SBATCH --ntasks=8 --cpus-per-task=16      ← 8 MPI processes with 16 threads each, i.e., 2 nodes with 64 processes each on Vaughan
#SBATCH --mem-per-cpu=1g
#SBATCH --time=00:05:00
module --force purge
module load calcua/2020a                   ← load the software stack module
module load vsc-tutorial                   ← load vsc-tutorial, which also loads the Intel toolchain (for the MPI libraries)
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK ← Set the number of OpenMP threads
export OMP_PROC_BIND=true                  ← Will cause the threads to be nicely spread over the cores and stay on the core
srun mpi_omp_hello                         ← run the MPI program (mpi_omp_hello) srun does all the magic, but mpirun would work too with Intel MPI
```

VLAAMS SUPERCOMPUTER CENTRUM

78

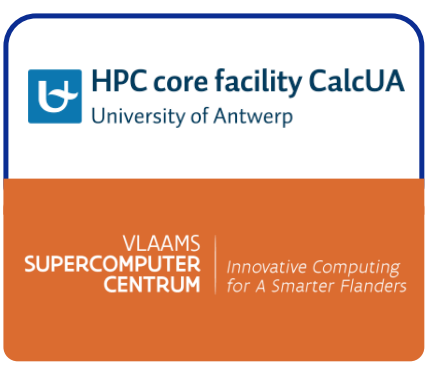
Note

- If you want to experiment, there are sample programs in the vsc-tutorial module to experiment with OpenMP, MPI and hybrid MPI/OpenMP programs
  - Load the module and do “man vsc-tutorial” for more information
- See also /apps/antwerpen/examples
- And the examples are also discussed in our documentation:
  - Shared memory job
  - MPI program
  - Hybrid MPI/OpenMP program

VLAAMS SUPERCOMPUTER CENTRUM

79

Job arrays and parameter exploration



VLAAMS SUPERCOMPUTER CENTRUM

80

Running a large batch of small jobs

- Many users run many serial jobs, e.g., for a parameter exploration
- Problem: submitting many short jobs to the queueing system puts quite a burden on the scheduler, so try to avoid it
- Solutions:
  - Job arrays: Submit a large number of related jobs at once to the scheduler
  - atools, a (VSC-developed) package that makes managing array jobs easier
  - Worker framework does no longer work
  - Slurm srun can be used to launch more tasks then requested in the job request running no more than the indicated number simultaneously
    - Contact us if you have a case where you think you can use that
  - GNU parallel (module name: parallel) can work with Slurm but you'll have to combine with srun with more advanced options than we can cover here.
    - Actually the previous bullet, but using GNU parallel to generate arguments and input for the commands

Job array

- Very similar to Torque/MOAB:

job_array.slurm

```
#!/bin/bash
#SBATCH --ntasks=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=512M
#SBATCH --time 15:00

INPUT_FILE="input_${SLURM_ARRAY_TASK_ID}.dat"
OUTPUT_FILE="output_${SLURM_ARRAY_TASK_ID}.dat"

./test_set -input ${INPUT_FILE} -output ${OUTPUT_FILE}
```

← for every run, there is a separate input file and an associated output file

```
$ sbatch --array 1-100 job_array.slurm
```

- program will be run for all input files (100)

Atools
Example: Parameter exploration

- Assume we have a range of parameter combinations we want to test in a .csv file (easy to make with Excel)
- Help offered by atools:
  - *How many jobs should we submit?* atools has a command that will return the index range based on the .csv file
  - *How to get parameters from the .csv file to the program?* atools offers a command to parse a line from the .csv file and store the values in environment variables.
  - *How to check if the code produced results for a particular parameter combination*? atools provides a logging facility and commands to investigate the logs.

Atools with Slurm
Example: Parameter exploration

weather.slurm

```
#!/bin/bash
#SBATCH --ntasks=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=512m
#SBATCH --time=10:00
module --force purge
ml calcua/2020a atools/slurm

source <(aenv --data data.csv)
./weather -t $temperature -p $pressure -v $volume
```

data.csv

```
temperature, pressure, volume
293.0,       1.0e05,   87
...,         ...,      ...
313,         1.0e05,   75
```

(data in CSV format)

```
login$ module load atools/slurm
login$ sbatch --array $(arange --data data.csv) weather.slurm
```

- weather will be run for all data, until all computations are done
- Can also run across multiple nodes

Atools
Remarks

- Atools has a nice logging feature that helps to see which work items failed or did not complete and to restart those.
- Advanced feature of atools: Limited support for some Map-Reduce scenarios:
  - Preparation phase that splits up the data in manageable chunks needs to be done on the login nodes or on separate nodes
  - Parallel processing of these chunks
  - Atools does offer features to aggregate the results
- Atools is really just a bunch of Python scripts and it does rely on the scheduler to start all work items
  - It supports all types of jobs the scheduler support
  - But it is less efficient than worker for very small jobs as worker does all the job management for the work items (including starting them)
  - A version of worker that can be ported to Slurm is under development

VLAAMS SUPERCOMPUTER CENTRUM

85

Further reading on array jobs and parameter exploration

- atools manual on readthedocs
- Presentation and training materials used in the course @ KULeuven on Worker and atools. This material is based on Torque but still useful.
- Examples in /apps/antwerpen/examples/atools/Slurm, or point your browser to github.com/hpcuantwerpen/cluster-examples to have all documentation nicely formatted.

VLAAMS SUPERCOMPUTER CENTRUM

86

Workflows in Slurm

VLAAMS SUPERCOMPUTER CENTRUM

87

Example workflows

- Some scenarios:
  - Need to run a sequence of simulations, each one using the result of the previous one, but bumping into the 3-day wall time limit so not possible in a single job
    - Or need to split a >3 day simulation in shorter ones that run one after another
  - Need to run simulations using results of the previous one, but with a different optimal number of nodes
    - E.g. in CFD: First a coarse grid computation, then refining the solution on a finer grid
  - Extensive sequential pre- or postprocessing of a parallel job
  - Run a simulation, then apply various perturbations to the solution and run another simulation for each of these perturbations
- Support in Slurm
  - Passing environment variables to job scripts: Can act like arguments to a procedure
    Alternative: Passing command line arguments to job scripts
  - Specifying dependencies between job scripts with additional sbatch arguments
- See also the UAntwerp documentation

VLAAMS SUPERCOMPUTER CENTRUM

88

Passing environment variables to job scripts

- As Slurm passes the whole environment in which the sbatch command is executed to the job, this is trivial.
  - But make sure that the variables that need to be passed are actually exported as otherwise sbatch cannot see them
  - Remember you can also pass environment variables to a command on the command itself, e.g.,

```
multiplier=5 sbatch my_job_script.slurm
```

  will pass the environment variable multiplier with value 5 so sbatch

Passing command line arguments to job scripts

- Any command line argument after the name of the job script is considered to be a command line argument for the job batch script and passed to it as such
  - Create the batch script:

get_parameter.slurm

```
#!/bin/bash
#SBATCH --ntasks=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=500m
#SBATCH --time=5:00

echo "The command line argument is $1."
```

  - Now run:

```
sbatch get_parameter.slurm 5
```

  - The output file contains:

```
The command line argument is 5.
```

Dependent jobs

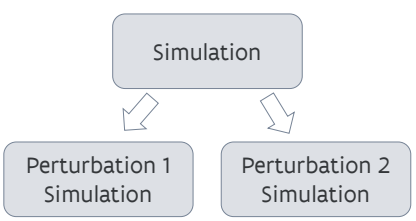
- It is possible to instruct Slurm to only start a job after finishing one or more other jobs:

```
sbatch --dependency=afterok:<jobid> jobdepend.slurm
```

  will only start the job defined by jobdepend.slurm after the job with job ID <jobid> has finished successfully
- Many other possibilities, including
  - Start another job only after a a list of jobs have ended
  - Start another job only after a job has failed
  - And many more, check the Slurm manual page and look for `--dependency`.
- Useful to organize jobs and powerful in combination with passing environment variables or specifying command line arguments to job scripts

91

Dependent jobs
Example



job.slurm

```
#!/bin/bash
#SBATCH --ntasks=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=1g
#SBATCH --time=30:00

echo "10" >outputfile ; sleep 300

multiplier=5
mkdir mult-$multiplier ; pushd mult-$multiplier
number=$(cat ../outputfile)
echo $(($number*$multiplier)) >outputfile; sleep 300
popd

multiplier=10
mkdir mult-$multiplier ; pushd mult-$multiplier
number=$(cat ../outputfile)
echo $(($number*$multiplier)) >outputfile; sleep 300
```


92

Dependent jobs
Example

- A job whose result is used by 2 other jobs

job_first.slurm

```
#!/bin/bash
#SBATCH --ntasks=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=1g
#SBATCH --time=10:00

echo "10" >outputfile ; sleep 300
```

job_depend.slurm

```
!/bin/bash
#SBATCH --ntasks=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=1g
#SBATCH --time=10:00

mkdir mult-$multiplier ; cd mult-$multiplier
number=$(cat ../outputfile)
echo $(($number*$multiplier)) >outputfile; sleep 300
```

job_launch.sh

```
#!/bin/bash
first=$(sbatch --parsable --job-name job_leader job_first.slurm)
multiplier=5  sbatch --job-name job_mult_5  --dependency=afterok:$first job_depend.slurm
multiplier=10 sbatch --job-name job_mult_10 --dependency=afterok:$first job_depend.slurm
```


Dependent jobs
Example

- After start of the first job – squeue

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             24869   vaughan job_mult vsc20259 PD       0:00      1 (Dependency)
             24870   vaughan job_mult vsc20259 PD       0:00      1 (Dependency)
             24868   vaughan job_lead vsc20259  R       0:25      1 r1c01cn1
```

- Some time later:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             24869   vaughan job_mult vsc20259  R       0:01      1 r1c01cn1
             24870   vaughan job_mult vsc20259  R       0:01      1 r1c01cn1
```

- When finished: Output of ls:

```
job_depend.slurm  job_launch.sh  mult-10  outputfile       slurm-24869.out
job_first.slurm   job.slurm      mult-5   slurm-24868.out  slurm-24870.out
```

- cat outputfile → 10
- cat mult-5/outputfile → 50
- cat mult-10/outputfile → 100

Interactive jobs

95

Interactive jobs
Method 1: srun for a non-X11 job

- Use the regular resource request options on the command line of srun and end with `--pty bash`
- Example: An interactive session to run a shared memory application

```
login$ srun -n 1 -c 16 -t 1:00:00 --pty bash
rXcYYcnZ$ module --force purge
rXcYYcnZ$ ml calcua/2020a vsc-tutorial
rXcYYcnZ$ omp_hello
…
rXcYYcnZ$ exit
```

- Example: Starting an MPI program in an interactive session

```
login$ srun -n 64 -c 1 -t 1:00:00 --pty bash
rXcYYcnZ$ module --force purge
rXcYYcnZ$ ml calcua/2020a vsc-tutorial
rXcYYcnZ$ srun mpi_hello
…
rXcYYcnZ$ exit
```


96

Interactive jobs
Method 1: srun for an X11 job

- First make sure that your login session supports X11 programs:
  - Log in to the cluster using `ssh -X` to forward X11 traffic
  - Or work from a terminal window in a VNC session
- Same as for non-X11 jobs but simply add the `--x11` option before `--pty bash`
- Few or no X11 programs support distributed memory computing, so usually you'll only be using one task...

```
login$ srun -n 1 -c 64 -t 1:00:00 --x11 --pty bash
rXcYYcnZ$ module --force purge
rXcYYcnZ$ ml calcua/2020a …
rXcYYcnZ$ xclock
rXcYYcnZ$ exit
```

- You can even start X11 programs directly through srun, e.g.,

```
login$ srun -n 1 -c 1 -t 1:00:00 --x11 xclock
```

but this has the same problems as the next approach...

VLAAMS SUPERCOMPUTER CENTRUM

97

Interactive jobs
Method 2: salloc for a shared memory program

- Here you have to really understand how Linux environments work.
  This method will have to change if the compute nodes and login nodes have a different architecture.
- Example for a shared memory program:

```
login$ salloc -n 1 -c 64 -t 1:00:00
login$ module --force purge
login$ ml calcua/2020a vsc-tutorial
login$ srun omp_hello
…
login$ exit
```

Problem: Software for the login nodes may not work on the compute nodes or vice-versa

VLAAMS SUPERCOMPUTER CENTRUM

98

Interactive jobs
Method 2: salloc for a distributed memory program

- Example for an MPI program

```
login$ salloc -n 64 -c 1 -t 1:00:00
login$ module --force purge
login$ ml calcua/2020a vsc-tutorial
login$ srun mpi_hello
…
login$ exit
```

Problem: Software for the login nodes may not work on the compute nodes or vice-versa

Interactive jobs
Method 2: salloc for running X11 programs

- First make sure that your login session supports X11 programs:
  - Log in to the cluster using `ssh -X` to forward X11 traffic
  - Or work from a terminal window in a VNC session
- Next use salloc to ask for an allocation. It usually doesn't make sense to use more than 1 task when running X11 programs.

```
login$ salloc -n 1 -c 64 -t 1:00:00 --mem-per-cpu=3g
```

- From the login shell in your allocation, log in to the compute node using

```
login$ ssh -X $SLURM_JOB_NODELIST
```

- You are now on the compute node *in your home directory* (because of ssh) and can now load the modules you need and start the programs you want to use.

Interactive jobs
Method 2: salloc for running X11 programs

- First make sure that your login session supports X11 programs:
  - Log in to the cluster using `ssh -X` to forward X11 traffic
  - Or work from a terminal window in a VNC session
- Then:

```
login$ salloc -n 1 -c 64 -t 1:00:00
login$ ssh -X $SLURM_JOB_NODELIST
rXcYYcnZ$ xclock
…
rXcYYcnZ$ exit
login$ exit
```


File transfer with Globus

Globus

- Service to transfer large amounts of data between computers
  - It is possible to initiate a direct transfer between two remote computers from your laptop (no software needed on your laptop except for a recent web browser)
  - It is also possible to initiate a transfer to your laptop
    - Globus software needed on your laptop
    - After disconnecting, the transfer will be resumed automatically
- Via web site: globus.org
  - It is possible to sign in with your UAntwerp account
  - You can also create a Globus account and link that to your UAntwerp account
- You do need a UAntwerp account to access data on our servers
  - Data sharing features not enabled in our license
- Collection to look for in the Globus web app: VSC UAntwerpen Tier2
  - From there access to /data (vsc2* accounts only) and /scratch (all users)
  - Note: VSC is also Vienna Scientific Cluster

VLAAMS SUPERCOMPUTER CENTRUM

103

Globus (2)

- Use for
  - Transfer large amounts of data to/from your laptop/desktop
    - Advantage over sftp: auto-restart
  - Transfer large amounts of data to a server in your department
    - Working on obtaining a campus license so that all departments can install a fully functional version
    - Advantage over sftp: You can manage the tranfer from your laptop/desktop/smartphone
  - Transfer data between different scratch volumes on VSC clusters
    - Often more efficient than using Linux `cp`.
  - Transfer data between the scratch volume of another VSC cluster and your data volume
  - Transfer data to/from external sources
- There is documentation on the VSC documentation web site.

VLAAMS SUPERCOMPUTER CENTRUM

104

User support

VLAAMS SUPERCOMPUTER CENTRUM

105

Documentation

- VSC documentation at docs.vscentrum.be
  - Specific section on UAntwerp infrastructure in “VSC hardware” -> “Tier-2 hardware”
- External documentation
  - SLURM documentation
    - CÉCI Slurm Quick Start Tutorial is excellent and most of it will work here too

VSC
www.vscentrum.be

CalcUA
calcua.uantwerpen.be

VLAAMS SUPERCOMPUTER CENTRUM

106

User support

- mailing-lists for announcements : calcua-announce@sympa.uantwerpen.be
  (for official announcements and communications)
- Every now and then a more formal "HPC newsletter"
  - We do expect that users read the newsletter and mailing list messages!
- contacts :
  - e-mail : hpc@uantwerpen.be
  - phone : 3860 (Stefan), 3855 (Franky), 3852 (Kurt), 3879 (Bert), 8980 (Carl), 9229 (Robin)
  - office : G.309-311 (CMI)

Don't be afraid to ask, but being as precise as possible when specifying your problem definitely helps in getting a quick answer.

VLAAMS SUPERCOMPUTER CENTRUM

107