Create SSH keys

- `ssh-keygen -t rsa -b 4096`
- Use a web browser to log in onto account.vscentrum.be with your UAntwerpen account to create your account and upload the public key (file with .pub)

- Further documentation:
  - www.vscentrum.be/user-portal
  - Follow the "Getting access" instructions
  - Tutorial text for Windows, macOS and Linux: hpc.uantwerpen.be/support

VLAAMS SUPERCOMPUTER CENTRUM

1

Vlaanderen is supercomputing

HPC@UAntwerp introduction

Stefan Becuwe, Franky Backeljauw, Bert Tijskens, Kurt Lust, Carl Mensch, Robin Verschoren, Michele Pugno

Version Fall 2021

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

vscentrum.be

3

Introduction to the VSC

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

VLAAMS SUPERCOMPUTER CENTRUM

6

VSC – Flemish Supercomputer Center

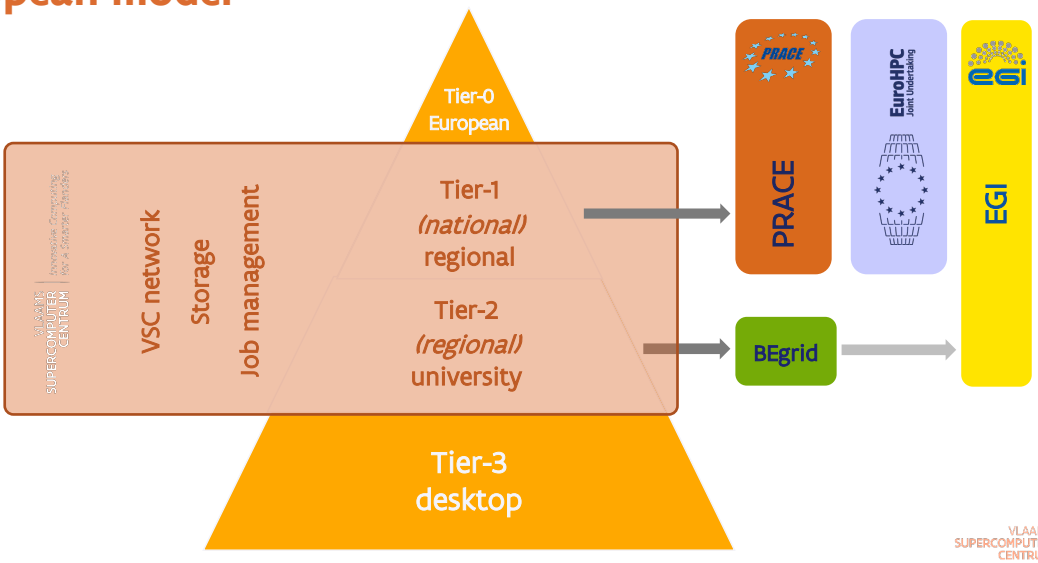
- Vlaams Supercomputer Centrum (VSC)
- Established in December 2007
- Partnership between 5 University associations:
  Antwerp, Brussels, Ghent, Hasselt, Leuven
- Funded by Flemish Government through the FWO (Research Fund – Flanders)
- Goal: make HPC available to all researchers in Flanders (academic and industrial)
- Local infrastructure (Tier-2) in Antwerp:

HPC core facility CalcUA

VLAAMS SUPERCOMPUTER CENTRUM

7

VSC: An integrated infrastructure according to the European model



VLAAMS SUPERCOMPUTER CENTRUM

8

UAntwerp hardware

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 GPU node with quad NVIDIA A100 • 2 GPU nodes with two AMD MI100 each
• 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 (HW 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

10

Tier-1 infrastructure

BrENIAC (KU Leuven)

- inaugurated October 2016
- 580 compute nodes with 2 14-core Broadwell processors
  128 (435 nodes) or 256 (145 nodes) GB memory, EDR-IB network
  - nodes are essentially the same as on Leibniz, but the interconnect is a half generation older
- 408 compute nodes with 2 14-core Skylake processors (621/841 Tflops theoretical peak performance base/turbo)
- Approximately 1.2 PB net storage capacity, peak bandwidth 40 GB/s
- Broadwell nodes will likely be decommissioned at the end of 2021.

196 in Top 500 on June 2016

VLAAMS SUPERCOMPUTER CENTRUM

16

Tier-1 infrastructure

BrENIAC (KU Leuven)

VLAAMS SUPERCOMPUTER CENTRUM

17

Tier-1 infrastructure

BrENIAC (KU Leuven)

VLAAMS SUPERCOMPUTER CENTRUM

18

Tier-1 infrastructure

Hortense (UGent)

- under construction
- 336 compute nodes with 2 64-core AMD Epyc processors
  256 (294 nodes) or 512 (42 nodes) GB memory, HDR100-IB network
- 20 GPU nodes with 2 24-core AMD Epyc processors and 4 NVIDIA Ampère NVLink 3 (40 GB)
- Approximately 3 PB net storage capacity

VLAAMS SUPERCOMPUTER CENTRUM

19

Tier-1 infrastructure

Hortense (UGent)

VLAAMS SUPERCOMPUTER CENTRUM

20

Characteristics of a HPC cluster

- Shared infrastructure, used by multiple users simultaneously
  - So you have to request the amount of resources you need
  - And you may have to wait a little
- Expensive infrastructure
  - So software efficiency matters
  - Can't afford idling for user input
- Therefore mostly for batch applications rather than interactive applications
- Built for parallel jobs. No parallelism = no supercomputing. Not meant for running a single single-core job.
- Remote use model
  - But you're running (a different kind of) remote applications all the time on your phone or web browser…
- Linux-based
  - so no Windows or macOS software,
  - but many standard Linux applications will run.

VLAAMS SUPERCOMPUTER CENTRUM

21

Getting a VSC account

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM
Innovative Computing for A Smarter Flanders

Tutorial Chapter 2

VLAAMS SUPERCOMPUTER CENTRUM

24

SSH keys

- Almost all communication with the cluster happens trough SSH = Secure SHell
  - Protocol to log in to a remote computer
  - Protocol to transfer files (sftp)
  - Protocol to tunnel IP traffic to/from a remote host through firewalls
- We use public/private key pairs rather than passwords to enhance security
  - Private key is never passed over the internet if you do things right
- Keys:
  - The private key is the part of the key pair that stays on your local machine and that you have to keep very secure
    - Protect with a passphrase!
  - The public key is put on the computer you want to access
  - A hacker can do nothing with your public key without the private key
    - And can do nothing with your private key file without the passphrase

VLAAMS SUPERCOMPUTER CENTRUM

25

Required software – Options on Windows

- Windows 10 version 1803 and later come with a Linux-style ssh client in PowerShell
  - Works as the Linux client
  - In combination with other ssh ports, getting the right permissions on .ssh may be a bit tricky
  - Combine with the Windows Terminal (store app)
- PuTTY is a popular GUI SSH client
- MobaXterm combines a SSH/SFTP client, X server and VNC server in one
  - Free edition ("Home Edition") with limited number of simultaneous connections
  - Professional version with unlimited number of connections (€ 60/year)
  - Uses PuTTY keys
- Cygwin Linux emulation libraries
- Windows Subsystem for Linux (WSL)
  - Since Windows 10 version 1709, now a WSL 2 since Windows 10 version 2004
  - Enable in "Additional Windows components"
  - Install your favourite (often free) Linux distribution from the Windows store

VLAAMS SUPERCOMPUTER CENTRUM

26

Required software – macOS and Linux

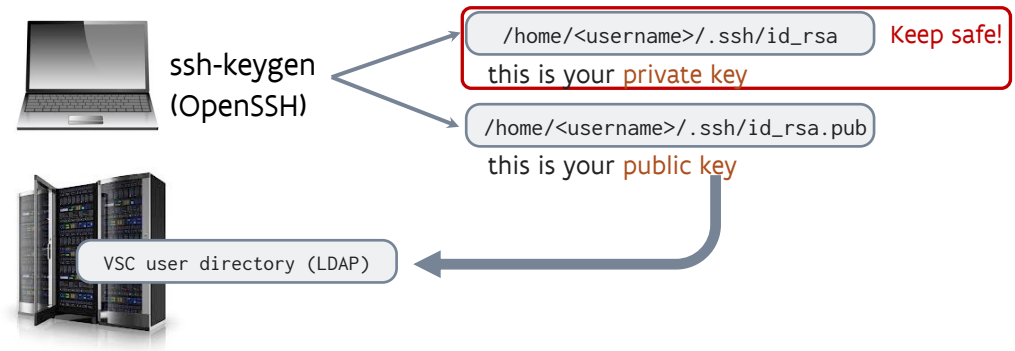
- macOS
  - SSH client included with the OS
  - Terminal package also included with the OS
  - iTerm2 is a free and more sophisticated terminal emulator
  - XQuartz with local xterm is also an option

- Linux
  - SSH client included with the OS (though you may need to install it as a separate package)
  - Choice of terminal emulation packages

VLAAMS SUPERCOMPUTER CENTRUM

27

Create SSH keys



- On Windows: PuTTY with PuTTYgen key generator (use PuTTY key format 2 in latest version):
  - But one file with public+private key (extension .ppk), stays on the PC
  - And a second file with the public key (extension .pub)
  - Can convert to and from OpenSSH key files

VLAAMS SUPERCOMPUTER CENTRUM

28

Create SSH keys
A technical note

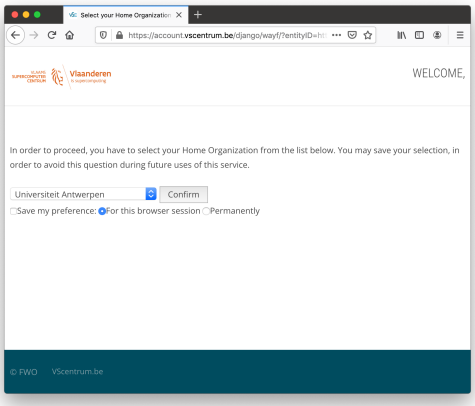
- We currently support two cryptography technologies for keys at the VSC
  - RSA, but it should use at least 4096 bits
    `ssh-keygen -t rsa -b 4096`
  - ed25519 keys
- Our advice is to stick to RSA 4096 bit keys. Some client software does not support ed25519 keys and even longer RSA keys are also often not supported.
- Some SSH implementations use a different file format (PuTTY, SSH2), and those keys may need conversion.

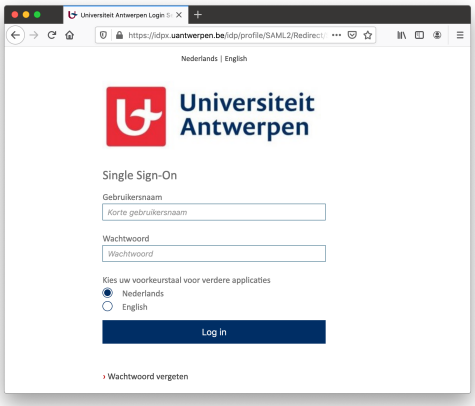
VLAAMS SUPERCOMPUTER CENTRUM

29

Register your account

- Web-based procedure





your VSC username is vsc20xxx

VLAAMS SUPERCOMPUTER CENTRUM

30

Connecting to the HPC infrastructure

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

Tutorial Chapter 3
Tutorial Chapter 6

VLAAMS SUPERCOMPUTER CENTRUM

34

A typical workflow

1. Connect to the cluster
2. Transfer your files to the clusters
3. Select software and build your environment
4. Define and submit your job
5. Wait while
   - your job gets scheduled
   - your job gets executed
   - your job finishes
6. Move your results

VLAAMS SUPERCOMPUTER CENTRUM

35

Connecting to the cluster

- Required software:
  - See slides with SSH implementations
  - When connecting from outside Belgium, you need a VPN client to connect to the UAntwerpen network first (University service, not provided by CalcUA/VSC)
    - Use the Cisco AnyConnect Client on vpn.uantwerpen.be or iOS/Android stores. OS-builtin clients fail to connect from some sites.
- A cluster has
  - login nodes: The part of the cluster to which you connect and where you launch your jobs
  - compute nodes: The part of the cluster where the actual work is done
  - storage and management nodes: As a regular user, you don't get on them
- Generic names of the login nodes:
  - VSC standard: login.hpc.uantwerpen.be ⇒ Leibniz
  - Leibniz: login-leibniz.hpc.uantwerpen.be
  - Vaughan: login-vaughan.hpc.uantwerpen.be

VLAAMS SUPERCOMPUTER CENTRUM

36

Connecting to the cluster

- Windows with PuTTY:
  - Open PuTTY
  - Follow the instructions on the VSC web site to fill in all fields
- OpenSSH-derived clients
  - Open a terminal
    - Windows 10: PowerShell or a bash command prompt in a terminal emulator
    - macOS: Terminal which comes with macOS or iTerm2
    - Ubuntu: Applications > Accessories > Terminal or CTRL+Alt+T
  - Login via secure shell:

```
$ ssh vsc2xxxx@login.hpc.uantwerpen.be
```

(if your private key file has the standard filename, otherwise point to it with –i)

VLAAMS SUPERCOMPUTER CENTRUM

37

The login nodes

- Currently restricted access
  - only accessible from within Belgium
  - external access from elsewhere requires using the UA-VPN service to connect to the Campus network first
- Use for:
  - editing and submitting jobs, also via some GUI applications
  - small compile jobs – submit large compiles as a regular job
  - visualisation (we even have a special login node for that purpose)

Do not use the login nodes for running your jobs

VLAAMS SUPERCOMPUTER CENTRUM

38

The login nodes

- In rare cases you may need specific login names rather than the generic ones:
  - login1-leibniz.hpc.uantwerpen.be (and login2-)
  - viz1-leibniz.hpc.uantwerpen.be (for hardware accelerated visualisation using TurboVNC)
  - login1-vaughan.hpc.uantwerpen.be (and login2-)
- Some cases where you may need those:
  - when using VNC for GUI access
  - when using the screen or tmux commands and you want to reconnect
  - Eclipse or VS Code remote projects
- These are names for external access. Once on the cluster, you can also use internal names:
  - login1.leibniz.antwerpen.vsc (and login2 and ln1 and ln2)
  - viz1.leibniz.antwerpen.vsc
  - login1.vaughan.antwerpen.vsc (and login2 and ln1 and ln2)
- These names can also be used to connect from other VSC clusters

VLAAMS SUPERCOMPUTER CENTRUM

39

A typical workflow

1. Connect to the cluster
2. Transfer your files to the clusters
3. Select software and build your environment
4. Define and submit your job
5. Wait while
   - your job gets scheduled
   - your job gets executed
   - your job finishes
6. Move your results

VLAAMS SUPERCOMPUTER CENTRUM

40

Transfer your files

- sftp protocol – ftp-like protocol based on ssh, using the same keys
- Graphical ssh/sftp file managers for Windows, macOS, Linux
  - Windows: PuTTY, WinSCP (PuTTY keys), MobaXterm (PuTTY keys)
  - Multiplatform: FileZilla (OpenSSH keys)
  - macOS: CyberDuck
- Command line: scp - secure copy
  - copy from the local computer to the cluster

```
$ scp file.ext vsc2xxxx@login.hpc.uantwerpen.be:
```

  - copy from the clusters to the local computer

```
$ scp vsc2xxxx@login.hpc.uantwerpen.be:file.ext .
```

- Consider Globus (globus.org) for big transfers
  - Requires software on the computer from/to where you want to transfer files
  - Supports data and scratch file systems on UAntwerp clusters

VLAAMS SUPERCOMPUTER CENTRUM

41

Directories

- /scratch/antwerpen/2xx/vsc2xxyy
  - fast scratch for temporary files
  - per cluster/site
  - highest performance, especially for bigger files
  - highest capacity (0.6 PB)
  - does not support hard links

$VSC_SCRATCH

- /data/antwerpen/2xx/vsc2xxyy
  - to store long-time data
  - exported to other VSC sites
  - slower and much smaller (60 TB) than /scratch
  - prefer no jobs running in this directory unless explicitly told to do so

$VSC_DATA

- /user/antwerpen/2xx/vsc2xxyy
  - only for account configuration files
  - exported to other VSC sites
  - never run your jobs in this directory!
  - very small but good performance for its purpose

$VSC_HOME

VLAAMS SUPERCOMPUTER CENTRUM

42

Directories (2)

- /scratch uses a parallel filesystem (BeeGFS), hence is more optimized for large files.
- /data and /user use a standard Linux filesystem (xfs) and have better performance for small files (a few kB) which is why we suggest to install packages for Python, perl, R, etc. on /data.
  - The underlying storage technology of /user is very expensive hence it should only be used for what it is meant to be used
- Some fancy software installers still think it is a good idea to use a lot of hard links. These don't work on /scratch
  - So Conda-software should be installed on /data as it will not work on /scratch unless you force conda to use symbolic links instead
- Currently we do take a backup of /user and /data, but in the future we may exclude certain directories if the volume and number of files becomes too large
  - Conda-installed Pythons are an example of directories that may be excluded as every Conda installation tries to install half an OS.

VLAAMS SUPERCOMPUTER CENTRUM

43

Directories: Quota

- Block quota: Limits the amount of data you can store on a file system
- File quota: Limits the number of files you can store on a file system
- Current defaults:

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

- Quota are shown when you log on onto the login nodes or run `myquota`.

VLAAMS SUPERCOMPUTER CENTRUM

44

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 you 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 current license
- Collection to look for in the Globus web app: VSC UAntwerpen Tier2
  - From there access to /data and /scratch
  - Note: VSC is also Vienna Scientific Cluster

VLAAMS SUPERCOMPUTER CENTRUM

46

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
  - Transfer data between scratch volumes of other VSC clusters and your data volume
    - Often more efficient than using Linux cp.
  - Transfer data to/from external sources
- More info on the VSC documentation website

VLAAMS SUPERCOMPUTER CENTRUM

47

Best practices for file storage

- The cluster is not for long-term file storage. You have to move back your results to your laptop or server in your department.
  - There is a backup for the home and data file systems. We can only continue to offer this if these file systems are used in the proper way, i.e., not for very volatile data
  - Old data on scratch can be deleted if scratch fills up (so far this has never happened at UAntwerp, but it is done regularly on some other VSC clusters)
- Remember that the cluster is optimised for parallel access to big files, not for using tons of small files (e.g., one per MPI process).
  - If you run out of block quota, you'll get more without too much motivation (at least on /scratch)
  - If you run out of file quota, you'll have a lot of explaining to do before getting more.
- Text files are good for summary output while your job is running, or data that you'll read into a spreadsheet, but not for storing 1000x1000-matrices. Use binary files for that!

VLAAMS SUPERCOMPUTER CENTRUM

48

Running batch jobs
Building your environment

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

Tutorial Chapter 4

VLAAMS SUPERCOMPUTER CENTRUM

51

A typical workflow

1. Connect to the cluster
2. Transfer your files to the clusters
3. Select software and build your environment
4. Define and submit your job
5. Wait while
   - your job gets scheduled
   - your job gets executed
   - your job finishes
6. Move your results

VLAAMS SUPERCOMPUTER CENTRUM

52

System software

- Operating system : CentOS 7.9 on Leibniz, CentOS 8.4 on Vaughan
  - Red Hat Enterprise Linux (RHEL) 7/8 clone
  - Leibniz and Vaughan will be kept in sync as much as possible. Leibniz will be upgraded to CentOS 8 soon
- Resource management and job scheduling on Leibniz:
  - Torque : resource manager (based on PBS)
  - MOAB : job scheduler and management tools
- Resource management/scheduler on Vaughan: SLURM
  - Leibniz will follow in the near future.

VLAAMS SUPERCOMPUTER CENTRUM

53

Development software

- C/C++/Fortran compilers: Intel Parallel Studio XE Cluster Edition and GCC
  - Intel compiler comes with several tools for performance analysis and assistance in vectorisation/parallelisation of code
  - Fairly up-to-date OpenMP support in both compilers
  - Support for other shared memory parallelisation technologies such as Cilk Plus, other sets of directives, etc. to a varying level in both compilers
  - PGAS Co-Array Fortran support in Intel Compiler (and some in GNU Fortran)
  - Due to some VSC policies, support for the latest versions may be lacking, but you can request them if you need them
- Message passing libraries: Intel MPI (MPICH-based), Open MPI (standard and Mellanox)
- Mathematical libraries: Intel MKL, OpenBLAS, FFTW, MUMPS, GSL, …
- Many other libraries, including HDF5, NetCDF, Metis, …
- Python

VLAAMS SUPERCOMPUTER CENTRUM

54

Application software

- ABINIT, CP2K, OpenMX, QuantumESPRESSO, *Gaussian*
- *VASP*, CHARMM, GAMESS-US, Siesta, *Molpro*, CPMD
- Gromacs, LAMMPS, NAMD2
- Telemac, *FINE-Marine*
- *OMNIS/LB*
- SAMtools, TopHat, Bowtie, BLAST, MaSuRCA
- Gurobi
- *Comsol*, R, MATLAB, …
- Often multiple versions of the same package

Additional software can be installed on demand

VLAAMS SUPERCOMPUTER CENTRUM

55

Application software
The fine print

- Restrictions on commercial software
  - 5 licenses for Intel Parallel Studio
  - MATLAB: Toolboxes limited to those in the campus agreement
  - Other licenses paid for and provided by the users, owner of the license decides who can use the license (see next slide)
- Typical user applications don't come as part of the system software, but
  - Additional software can be installed on demand
  - Take into account the system requirements
  - Provide working building instructions and licenses for a cluster. RPMs rarely work on a cluster.
  - Since we cannot have domain knowledge in all science domains, users must do the testing with us
- Compilation support is limited
  - Best effort, no code fixing
  - An issue nowadays as too many packages are only tested with a single compiler…
- Installed in /apps/antwerpen

VLAAMS SUPERCOMPUTER CENTRUM

56

Using licensed software

- VSC or campus-wide license: You can just use the software
  - Some restrictions may apply if you don't work at UAntwerp but at some of the institutions that have access (ITG, VITO) or at a company
- Restricted licenses
  - Typically licenses paid for by one or more research groups, sometimes just other license restrictions that must be respected
  - Talk to the owner of the license first. If no one in your research group is using a particular package, your group probably doesn't contribute to the license cost.
  - Access controlled via UNIX groups
    - E.g., avasp : VASP users from UAntwerp
    - To get access: Request group membership via account.vscentrum.be, "New/Join group" menu item. The group moderator will then grant or refuse access

VLAAMS SUPERCOMPUTER CENTRUM

57

Building your environment:
The module concept

- If you're not loading modules you're probably doing something wrong…
- Dynamic software management through modules
  - Allows to select the version that you need and avoids version conflicts
  - Loading a module sets some environment variables :
    - $PATH, $LD_LIBRARY_PATH, …
    - Application-specific environment variables
    - VSC-specific variables that you can use to point to the location of a package (typically start with EB)
  - Automatically pulls in required dependencies
- Software on our system is organised in software stacks and toolchains
  - Toolchain: A compiler with compatible MPI library and core mathematical libraries
  - Software stack: Bundling of software installed in the same time frame on the cluster, and with compilers that may have some common components
    - But not all software in a software stack can be loaded at the same time

VLAAMS SUPERCOMPUTER CENTRUM

58

Building your environment:
Toolchains (1)

- Compiler toolchains: Compiler with compatible MPI and libraries
  - Regular toolchains:
    - "intel" toolchain: Intel and GNU compilers, Intel MPI and math libraries.
    - "foss" toolchain: GNU compilers, Open MPI, OpenBLAS, FFTW, …
  - Special ones (no MPI or mathematical libraries)
    - "GCCcore": modules work with both a specific intel and specific foss toolchain
    - System toolchain: Software compiled with the system compilers
  - VSC-wide and refreshed twice per year with more recent compiler versions: 2016b, 2017a, …
    - We sometimes do silent updates within a toolchain at UAntwerp
    - And it is impossible to follow pace since the quality of software is generally going down

VLAAMS SUPERCOMPUTER CENTRUM

59

Building your environment:
Toolchains (2)

- intel/2018a, intel/2018b, intel/2019b and intel/2020a are now our main toolchains and kept synchronous on all systems.
  - We have skipped 2017b at UAntwerp as it offers no advantages over our 2017a toolchain
  - And skipped 2019a because there were too many technical problems
  - Software that did not compile correctly with the 2017a or newer toolchains or showed a large performance regression, is installed in the 2016b toolchain
  - 2020a is the base toolchain for CentOS 8.

VLAAMS SUPERCOMPUTER CENTRUM

60

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 on CentOS 8 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

61

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 toolchain (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

62

Building your environment:
Naming modules

- Full module name is typically of the form: <name of package>/<version>-<toolchain info>-<additional info> with the latter 2 optional, e.g.
  - intel/2020a : The Intel toolchain package, version 2020a
  - Boost/1.73.0-intel-2020a: The Boost libraries, version 1.73.0, built with the intel/2020a toolchain
  - SuiteSparse/5.7.1-intel-2020a-METIS-5.1.0: SuiteSparse library, version 5.7.1, built with the Intel compiler, and using METIS 5.1.0.
    - In <additional info>, we don't add all additional components, but these that matter to distinguish between different available modules for the same package.
- No two modules with the same name can be loaded simultaneously

VLAAMS SUPERCOMPUTER CENTRUM

63

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

64

Building your environment:
Discovering software

```
$ module spider parallel/20200422
…
 You will need to load all module(s) on any one of the lines below before the
"parallel/20200422" module is available to load.

    …
    calcua/2018b
    calcua/2019b
    calcua/2020a
    …
```

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

VLAAMS SUPERCOMPUTER CENTRUM

65

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

66

Modules and reproducibility

- On supercomputers, most software is recompiled at installation to ensure proper optimization for the system.
- Most supercomputer centres use a specialised installation tool to manage the installation process and the module system. The two most popular ones are EasyBuild and Spack.
  - We use EasyBuild
  - Those tools work with “recipes” that provide all installation instructions in a concise way
- What we do:
  - We keep the source files that we used when installing software
  - The recipe is stored with the software for all software installed through EasyBuild and accessible to users (not on all VSC systems).
    - E.g., $EBROOTGROMACS/easybuild contains the recipe and the installation log file which will tell you which version of EasyBuild was used.
  - Even after removal from the system, we keep that information on backup for a number of years.

VLAAMS SUPERCOMPUTER CENTRUM

67

The myth of absolute reproducibility

- Algorithms that rely on random numbers can only reproduce statistics
- Floating point computations are not 100% reproducible
  - Different hardware has a different roundoff error pattern
  - Different order of operations that are mathematically equivalent are not equivalent in floating point computations, and this is an issue with all parallel software
- Not all algorithms are designed to be reproducible in a parallel environment and may not even produce exactly the same result twice on the same hardware
- The operating system may also influence the results that your software generates as all software uses some OS-provided libraries
  - And there is no way that we can stop OS upgrades as this has severe security implications
- Even a cloud environment would not give you complete reproducibility
  - You can reproduce the exact same OS environment (by storing your virtual machine)
  - But even a virtual machine cannot abstract away from processor behaviour unless you would emulate the processor
  - And even then, non-deterministic behaviour in a parallel environment remains an issue.

VLAAMS SUPERCOMPUTER CENTRUM

68

Exercises (block 1)

- Link VSC-tutorials: hpc.uantwerpen.be/support and look for HPC tutorials
- Getting ready (see also §3.1):
  - Log on to the cluster
  - Go to your scratch directory: `cd $VSC_SCRATCH`
  - Copy the examples for the tutorial to that directory: type
    `cp -r /apps/antwerpen/examples .`
- Follow the text from the tutorial in §4.1 “Modules” and try out the module command.
- Load the cluster module calcua/2020a and check the difference between module av and module spider.
  - Try to find modules for the HDF5 library. Did you notice the difference between module av and module spider?
- Which module offers support for VNC? Which command does that module offer?
- Which module(s) offer the CMake tool?
- What is the difference between the HDF5/1.10.6-intel-2020a-MPI HDF5/1.10.6-intel-2020a-noMPI (well, you can guess it from the name but how can you find more information?)

VLAAMS SUPERCOMPUTER CENTRUM

70

Running batch jobs
Defining and submitting jobs

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM
Innovative Computing for A Smarter Flanders

Tutorial Chapter 4

VLAAMS SUPERCOMPUTER CENTRUM

72

A typical workflow

1. Connect to the cluster
2. Transfer your files to the clusters
3. Select software and build your environment
4. Define and submit your job
5. Wait while
   - your job gets scheduled
   - your job gets executed
   - your job finishes
6. Move your results

VLAAMS SUPERCOMPUTER CENTRUM

73

73

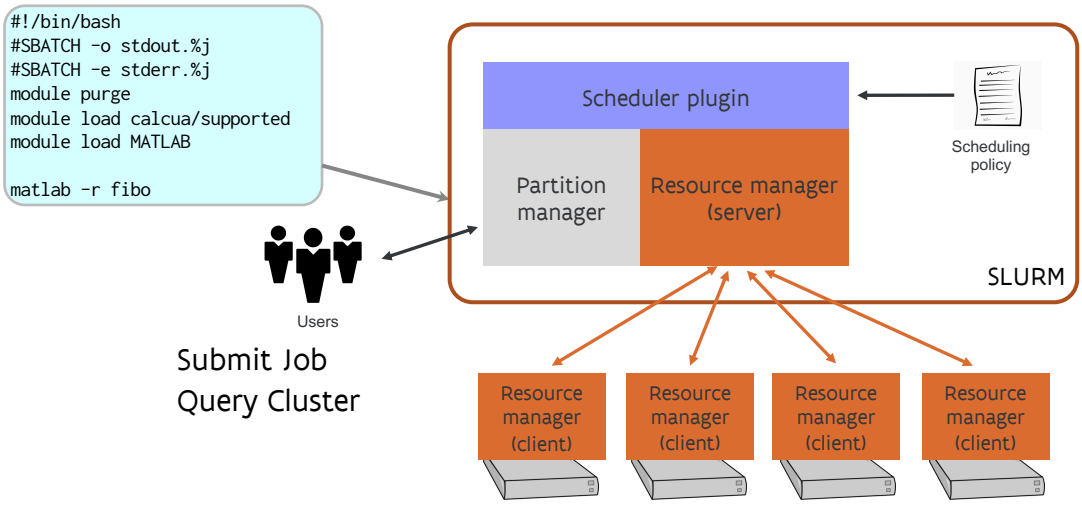
Why batch?

- A cluster is a large and expensive machine
  - So the cluster has to be used as efficiently as possible
  - Which implies that we cannot loose time waiting for input as in an interactive program
- And few programs can use the whole capacity (also depends on the problem to solve)
  - So the cluster is a shared resource, each simultaneous user gets a fraction of the machine depending on his/her requirements
- Moreover there are a lot of users, so one sometimes has to wait a little.
- Hence batch jobs (script with resource specifications) submitted to a queueing system with a scheduler to select the next job in a fair way based on available resources and scheduling policies.

VLAAMS SUPERCOMPUTER CENTRUM

74

Coming up: Job workflow in Slurm
What happens behind the scenes



VLAAMS SUPERCOMPUTER CENTRUM

75

The job script

- Specifies the resources needed for the job and other instructions for the resource manager:
  - Number and type of CPUs
  - Amount of compute time
  - Amount of (RAM) memory
  - Output files (stdout and stderr) – optional
  - Can instruct the resource manager to send mail when a job starts or ends
- Creates the environment
  - Load the modules you need for your job
- Then does the actual work
  - Then specify all the commands you want to execute
  - And these are really all the commands that you would also do interactively or in a regular script to perform the tasks you want to perform

VLAAMS SUPERCOMPUTER CENTRUM

76

The job script (Slurm)

```
#!/bin/bash
#SBATCH --ntasks=1 --cpus-per-task=7
#SBATCH --mem-per-cpu=1g
#SBATCH --time=1:00:00
#SBATCH -o stdout.%j
#SBATCH -e stderr.%j

module purge
module load calcua/supported
module load MATLAB

matlab -r fibo
```

This is a bash script (but could be, e.g., Perl)

Specify the resources for the job (and some other instructions for the resource manager), but look as comments to Bash

Build a suitable environment for your job. The script runs where you launched the batch job!

The command(s) that you want to execute in your job.

VLAAMS SUPERCOMPUTER CENTRUM

77

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

78

Specifying resources in job scripts

example.sh

```
#!/bin/bash                                  ← the shell (language) in which the script is executed

#SBATCH --jobname=name_of_job                ← "name" of the job (optional)
#SBATCH --ntasks=1 --cpus-per-task=1         ← resource specifications (e.g., tasks, processors per task), amount of memory
#SBATCH --mem-per-cpu=1g                       per processor
#SBATCH --time=01:00:00                      ← requested running time (required)

#SBATCH --mail-type=FAIL                     ← mail notification at BEGIN, END, FAIL, ... (optional)
#SBATCH --mail-user=<your e-mail address>      your e-mail address (should always be specified)

#SBATCH -o stdout.file                       ← redirect standard output (–o) and standard error (–e) (optional)
#SBATCH -e stderr.file                         if omitted, a generic filename will be used

module purge                                 ← Make sure you reload the environment as otherwise you may have strange
module load calcua/supported vsc-tutorial      behaviour of MPI programs

./hello_world                                ← the program itself
```

```
$ sbatch example.sh
```

VLAAMS SUPERCOMPUTER CENTRUM

79

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.

VLAAMS SUPERCOMPUTER CENTRUM

80

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

81

Slurm resource requests
Requesting processing resources

- 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

82

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

83

Slurm resource requests
Requesting memory

- Memory in Slurm is specified per CPU and not per task
  - Just a single parameter corresponding to RAM
  - No parameter for virtual memory unlike some other resource managers you may come across
- 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 3g` will allocate 3072 MB = 3 GB per CPU.
  - There is no short option.
- 240 GB available on Vaughan, so `3840m` per core.
  112 GB available on the regular compute nodes of Leibniz so 4g per core

VLAAMS SUPERCOMPUTER CENTRUM

84

And this is enough to submit a job...

- If all resources are specified in the job script, submitting a job is as simple as

```
$ sbatch jobscript.slurm
```

  sbatch returns with a message containing a unique jobid that you'll need for further commands and a cluster name.
- But all parameters specified in the job script in #SBATCH lines can also be specified on the command line (and overwrite those in the job script), e.g.,

```
$ sbatch -n x -c y -mem-per-cpu zg example.slurm
```

  - x: the number of tasks
  - y: the number of CPUs per task
    (maximum: 20 for Hopper, 28 for Leibniz, 64 for Vaughan until we enable hyperthreading)
  - z: the amount of memory needed per CPU

VLAAMS SUPERCOMPUTER CENTRUM

85

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

86

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

87

Slurm resource requests
Partitions

- Slurm does not automatically assign a job to the optimal partition
- The default partition is OK for most jobs on our cluster
- Partitions: `scontrol show partition` or in the output of `sinfo`
  - `vaughan` : Default partition on Vaughan, 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, up to 1 hour. 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

VLAAMS SUPERCOMPUTER CENTRUM

88

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

89

Slurm resource requests
Not using a full node

- What happens if you don't use all memory and cores on a node?
  - Other jobs – but only yours – can use it
    - so that other users cannot accidentally take resources away from your job or crash your job
    - and so that you have a clean environment for benchmarking without unwanted interference
  - You're likely wasting expensive resources! *Don't complain you have to wait too long until your job runs if you don't use the cluster efficiently.*
- If you really require a special setup where no other jobs of you run on the nodes allocated to you, contact user support (hpc@uantwerpen.be).
- If you submit a lot of 1 core jobs that use more than 4 GB each on Leibniz: Please specify the "mem256" feature for the nodes so that they get scheduled on nodes with more memory and also fill up as many cores as possible
  - And a program requiring 16 GB of memory but using just a single thread does not make any sense on today's computers

VLAAMS SUPERCOMPUTER CENTRUM

90

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 fairly good at controlling resources and limiting interference between jobs, 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

91

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

92

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, contrary to Torque where it's only present after the job finished.
- 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

93

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

94

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

95

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 it if we found one.

VLAAMS SUPERCOMPUTER CENTRUM

96

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

97

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 specifications
  - And realise that in parallel computing the optimal set of resources also depends on the size of the problem you are solving!

Share experience

VLAAMS SUPERCOMPUTER CENTRUM

98

Slurm commands

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM
Innovative Computing for A Smarter Flanders

VLAAMS SUPERCOMPUTER CENTRUM

99

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

100

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

101

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

102

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.

VLAAMS SUPERCOMPUTER CENTRUM

103

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

VLAAMS SUPERCOMPUTER CENTRUM

104

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

105

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  00:00.000  00:00.000
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

106

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

107

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

108

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 memory consumption per task is 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`

VLAAMS SUPERCOMPUTER CENTRUM

109

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

VLAAMS SUPERCOMPUTER CENTRUM

110

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

111

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

112

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

VLAAMS SUPERCOMPUTER CENTRUM

113

Exercises (block 2)

- Tutorial §4.3: $VSC_SCRATCH/examples/Slurm/04_Running-batch-jobs contains a short bash script (fibo.slurm) that will print Fibonacci numbers. Extend the script to request 1 core on 1 node with 1 Gb of (virtual) memory and with a wall time of 1 minute in the job script and run the job. Note which files are generated
- Extend the script to give the job a name of your choice. Run and note which files are generated.
- While the job is running, try some of the squeue and sstat commands
  - Extend the walltime to 10 minutes and put a `sleep 300` at the end because otherwise the job will finish so quickly that you don't get the chance to see anything.

VLAAMS SUPERCOMPUTER CENTRUM

115

What we've covered so far...

- Linux (2 lectures) – Some bash scripting skills needed to write job scripts
- Cluster hardware and middleware
  - CPUs have evolved so software should evolve too, and binaries need to be compiled with proper optimizations
  - Overview of middleware used when developing parallel programs and how they may influence starting programs
    - Options for vectorization: parallelism at the core level
    - Options for shared memory: OpenMP, Intel TBB or other libraries, native support in languages, posix threads: Usually need to specify the number of threads
    - Options for distributed memory: MPI, some languages such as Julia and Charm++, PGAS languages and libraries: Usually needs a process starter
    - Hybrid: combine distributed and shared memory parallelism

VLAAMS SUPERCOMPUTER CENTRUM

119

What we've covered so far… (2)

- Logging in to the cluster: ssh and options for various OSes
- Transfering files: sftp, scp and Globus
- Building your environment: Modules
  - Needed as often we need multiple versions of software on the cluster
- Job scripts: have three parts
  - Resource specifications and other instructions for the resource manager and scheduler
  - Create your environment
  - Not yet covered: Start your programs
- Submit and manage your jobs

VLAAMS SUPERCOMPUTER CENTRUM

120

Job types

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

Part B

VLAAMS SUPERCOMPUTER CENTRUM

121

Agenda part B

- Job types
  - Working interactively – Tutorial text Chapter 5 + UAntwerp-specific
  - Parallel jobs – Tutorial text Chapter 7
  - Job workflows
  - Multi-job submission – Not covered in the tutorial text (Chapter 12 discusses a system that we will not be able to support a year from now)
- Monitoring jobs – Various chapters
- Best practices – Chapter 16

VLAAMS SUPERCOMPUTER CENTRUM

122

Working interactively

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

Tutorial Chapter 5 and more

VLAAMS SUPERCOMPUTER CENTRUM

123

GUI support

- The cluster is not your primary GUI working platform, but sometimes running GUI programs is necessary:
  - Licensed program with a GUI for setting parameters of a run that cannot be run locally due to licensing restrictions (e.g., NUMECA FINE-Marine)
  - Visualisations that need to be done while a job runs or of data that is too large to transfer
- Technologies in Linux:
  - X Window System: Rendering on your local computer
    - Sends the drawing commands from the application (X client) to the display device (X server)
    - Software needed on your PC: X server
    - Extensions add features, e.g., GLX for OpenGL support
    - Often too slow due to network latency
  - VNC (Virtual Network Computing): Rendering on the remote computer
    - Rendering on the remote machine by the VNC server, images compressed and sent over the network
    - Needs a VNC client on your PC
    - Works pretty well even over not very fast connections, but hard to set up
  - NoMachine NX / X2Go : sits between those options
    - Not supported at UAntwerp, but used on the clusters at KU Leuven and UGent

VLAAMS SUPERCOMPUTER CENTRUM

124

VNC

- Leibniz has one visualization node with GPU and a VNC & VirtualGL-based setup
  - VNC is the software to create a remote desktop
    - Acts as a X server to programs running locally
  - Simple desktop environment with the xfce desktop/window manager
    - GNOME may be more popular, but does not work well with VNC
  - VirtualGL: To redirect OpenGL calls to the graphics accelerator and send the resulting image to the VNC software. Start OpenGL programs via vglrun.
- Same software stack minus the OpenGL libraries on all login nodes of Leibniz and Vaughan.
  - For 2D applications or applications with built-in 3D emulation (e.g., Matlab)
  - A future version may support software OpenGL emulation in the VNC server through the GLX extension, so all OpenGL applications should run (albeit slower than on the visualization node)
- Need a VNC client on your PC. TurboVNC gives good results with our setup, but others should also work.

VLAAMS SUPERCOMPUTER CENTRUM

125

VNC (2)

- How does VNC work?
  - Basic idea: Don't send graphics commands over the network, but render the image on the cluster and send images.
  - Step 1: Load the vsc-vnc module and start the (Turbo)VNC server with the vnc-xfce command. You can then log out again.
  - Step 2: Create an SSH tunnel to the port given when starting the server.
    - Some clients offer built-in SSH support so step 2 and 3 can be combined
  - Step 3: Connect to the local end of the tunnel with your VNC client
  - Step 4: When you've finished, don't forget to kill the VNC server (with `vncserver -kill`). Disconnecting will not kill the server!
    - We've configured the desktop in such a way that logging out on the desktop will also kill the VNC server, at least if you started through the vnc-* commands shown in the help of the module (vnc-xfce).
- Full instructions: Page "Remote visualization @ UAntwerp" on the VSC documentation site
- Do not use the instructions in some versions of the VSC tutorial on the UAntwerp clusters.

VLAAMS SUPERCOMPUTER CENTRUM

126

GUI programs

- Using OpenGL:
  - The VNC server currently emulates an X-server without the GLX extension, so OpenGL programs will not run right away.
  - On the visualisation node: Start OpenGL programs through the command `vglrun`
    - `vglrun` will redirect OpenGL commands from the application to the GPU on the node
    - and transfer the rendered 3D image to the VNC server
  - E.g.: `vglrun glxgears` will run a sample OpenGL program
  - Even when the VNC server with GLX support will be installed, `vglrun` will still be needed to use the GPU acceleration
- Our default desktop for VNC is xfce.
  GNOME is not supported due to compatibility problems.

VLAAMS SUPERCOMPUTER CENTRUM

127

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
```

VLAAMS SUPERCOMPUTER CENTRUM

128

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

129

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

130

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

VLAAMS SUPERCOMPUTER CENTRUM

131

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_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.

VLAAMS SUPERCOMPUTER CENTRUM

132

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 $SLURM_NODELIST
rXcYYcnZ$ xclock
…
rXcYYcnZ$ exit
login$ exit
```

VLAAMS SUPERCOMPUTER CENTRUM

133

Multi-core jobs

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM
Innovative Computing for A Smarter Flanders

Tutorial Chapter 7 with extensions

VLAAMS SUPERCOMPUTER CENTRUM

137

Why parallel computing?

1. Faster time to solution
   - distributing code over N cores
   - hope for a speedup by a factor of N
2. Larger problem size
   - distributing your code over N nodes
   - increase the available memory by a factor N
   - hope to tackle problems which are N times bigger
3. In practice
   - gain limited due to communication and memory overhead and sequential fractions in the code
   - optimal number of cores/nodes is problem-dependent
   - but no escape possible as computers don't really become faster for serial code… Parallel computing is here to stay.

VLAAMS SUPERCOMPUTER CENTRUM

138

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…

VLAAMS SUPERCOMPUTER CENTRUM

139

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
```

VLAAMS SUPERCOMPUTER CENTRUM

140

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

141

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

142

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 or other tools you may find in old job scripts or VSC web pages
  - 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

143

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

144

Workflows

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

VLAAMS SUPERCOMPUTER CENTRUM

147

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

VLAAMS SUPERCOMPUTER CENTRUM

148

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 to sbatch

VLAAMS SUPERCOMPUTER CENTRUM

149

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.
```

VLAAMS SUPERCOMPUTER CENTRUM

150

Dependent jobs

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

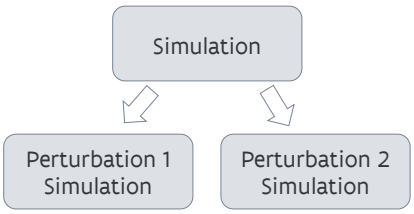
```
sbatch --dependency=afterok:<jobid> jobdepend.slurm
```

  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 sbatch 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

VLAAMS SUPERCOMPUTER CENTRUM

151

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
```

VLAAMS SUPERCOMPUTER CENTRUM

152

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
```

VLAAMS SUPERCOMPUTER CENTRUM

153

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

VLAAMS SUPERCOMPUTER CENTRUM

154

Multi-Job submission

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

Different from Chapter 12 in the tutorial…

VLAAMS SUPERCOMPUTER CENTRUM

155

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 not work with Slurm
  - Slurm srun can be used to launch more tasks than 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

VLAAMS SUPERCOMPUTER CENTRUM

156

Job array

- Starts from a job script for a single job in the array:

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)

VLAAMS SUPERCOMPUTER CENTRUM

157

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.

VLAAMS SUPERCOMPUTER CENTRUM

158

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

VLAAMS SUPERCOMPUTER CENTRUM

159

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 supports
  - 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

160

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

161

Exercises (block 3)

- See the exercise sheet and select those that are most relevant to your work on the cluster
- Do not forget to go to `$VSC_SCRATCH` where your files from the previous session are stored!

VLAAMS SUPERCOMPUTER CENTRUM

162

Job monitoring

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM
Innovative Computing for A Smarter Flanders

VLAAMS SUPERCOMPUTER CENTRUM

166

FAQ

- How do I know how much memory my job used?
- How can I check if a job is running properly?

VLAAMS SUPERCOMPUTER CENTRUM

167

Logging on to the compute node(s)

- While your job is running, you can log on to the compute nodes assigned to the job through ssh from the login nodes
  - Check which compute nodes a job uses: `squeue -j`
- Run `htop` on the compute node and check the use of the cores and memory
  - Quit htop: q
  - Only show your own processes: Press u, select your userid from the list to the left and press the return key
- `htop` also shows the load average, but this may be misleading
  - load = processes that can run
  - If this number is much higher than the number of cores: You are typically running several shared memory applications each using all cores without realising it.
  - But a load > the number of cores does not always mean something is going wrong

VLAAMS SUPERCOMPUTER CENTRUM

170

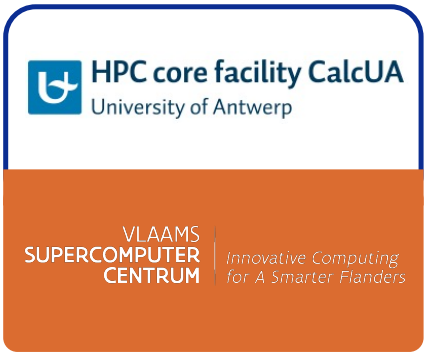
Add monitoring in your job script

- The `monitor` module
  - See Tutorial text §11.2.2
  - Start your (shared memory) application through the monitor command
    `monitor ./eat_mem 10`
    would start `eat_mem` with the command line argument 10
  - Usefull command line arguments:
    - `monitor -d 300 ./eat_mem 10`
      print output every 300 seconds
      - It does not make sense to sample a 3-day job every second
    - `monitor -d 300 -n 12 ./eat_mem 10`
      will only show the last 12 values (so roughly the last hour of this job)
- Note: `eat_mem` is available in the vsc-tutorial module

VLAAMS SUPERCOMPUTER CENTRUM

171

Policies



Tutorial Chapter 9

VLAAMS SUPERCOMPUTER CENTRUM

172

Some site policies

- Our policies on the cluster:
  - Single user per node policy
  - Priority based scheduling (so not first come – first get). We reserve the freedom to change the parameters of the priority computation at will to ensure the cluster is used well
  - Fairshare mechanism – Make sure one user cannot monopolise the cluster
  - Crediting system – Dormant, not enforced
- Implicit user agreement:
  - The cluster is valuable research equipment
  - Do not use it for other purposes than your research for the university
    - No cryptocurrency mining or SETI@home and similar initiatives
    - Not for private use
  - And you have to acknowledge the VSC in your publications (see user documentation on the VSC web site)
- Don't share your account

VLAAMS SUPERCOMPUTER CENTRUM

173

Credits

- At UAntwerp we monitor cluster use and send periodic reports to group leaders.
- On the Tier-1 system at KU Leuven, you get a compute time allocation (number of node days). This is enforced through project credits (Moab Accounting Manager).
  - A free test ride (motivation required), so-called "Starting Grant"
  - Requesting compute time through a proposal
- At KU Leuven Tier-2, you need compute credits which are enforced as on the Tier-1.
  - Credits can be bought via the KU Leuven
- Charge policy (subject to change) based upon :
  - fixed start-up cost
  - used resources (number and type of nodes)
  - actual duration (used wall time)

VLAAMS SUPERCOMPUTER CENTRUM

174

Best practices

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM | Innovative Computing for A Smarter Flanders

Tutorial Chapter 17

VLAAMS SUPERCOMPUTER CENTRUM

177

Debug & test

- Before starting you should always check :
  - are there any errors in the script?
  - are the required modules loaded?
  - Is the correct executable used?
  - Did you use the right process starter (srun) and start in the right directory?
- Check your jobs at runtime
  - login to a compute node and use, e.g., "`top`", "`htop`" , "`vmstat`", …
  - `monitor` command (see User Portal information on running jobs)
  - look at `sstat` and compare `MinCPU` and `AvgCPU`
  - alternatively: run an interactive job for the first run of a set of similar runs
- If you will be doing a lot of runs with a package, try to benchmark the software for scaling issues when using MPI
  - I/O issues
    - If you see that the CPU is idle most of the time that might be the problem

VLAAMS SUPERCOMPUTER CENTRUM

178

Hints & tips

- Job will logon to the first assigned compute node and start executing the commands in the job script
  - don't forget to load the software with module load
- Why is my job not running?
  - use "`squeue`" or look at the fourth line of output of "`scontrol show job <jobid>`"
  - commands might timeout with overloaded scheduler (rather unlikely recently)
- Submit your job and wait (be patient) …
- On Hopper, using `$VSC_SCRATCH_NODE` (/tmp on the compute node) might be beneficial for jobs with many small files (even though the bandwidth is much, much lower than on `$VSC_SCRATCH`).
  - But that scratch space is node-specific and hence will disappear when your job ends
  - So copy what you need to your data directory or to the shared scratch
  - And only available on Hopper, not much space on Leibniz or Vaughan

VLAAMS SUPERCOMPUTER CENTRUM

179

Warnings

- Avoid submitting many small jobs (in number of cores) by grouping them
  - using a job array
  - or using the atools or the Worker framework
- Runtime is limited by the maximum wall time of the queues
  - for longer wall time, use checkpointing
  - Properly written applications have built-in checkpoint-and-restart options
- Requesting many processors could imply long waiting times (though we're still trying to favour parallel jobs)

what a cluster is not

a computer that will automatically run the code of your (PC) application much faster or for much bigger problems

VLAAMS SUPERCOMPUTER CENTRUM

180

User support

HPC core facility CalcUA
University of Antwerp

VLAAMS SUPERCOMPUTER CENTRUM
Innovative Computing for A Smarter Flanders

VLAAMS SUPERCOMPUTER CENTRUM

181

User support

- Try to get a bit self-organized
  - limited resources for user support
  - best effort, no code fixing (AI and bioinformatics codes tend to be a disaster to install…)
  - contact hpc@uantwerpen.be, and be as precise as possible
- VSC documentation on ReadTheDocs: docs.vscentrum.be
- External documentation : slurm.schedmd.com/documentation.html
  - mailing-lists

VSC
www.vscentrum.be

CalcUA
hpc.uantwerpen.be

VLAAMS SUPERCOMPUTER CENTRUM

182

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 : +32 3 265 XXXX with XXXX
    3860 (Stefan), 3855 (Franky), 3852 (Kurt), 3879 (Bert), 8980 (Carl), Robin (9229)
  - 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

183

Evaluation

- Please fill in the questionnaire at tiny.cc/calcua-hpc-intro-survey by November 11

VLAAMS SUPERCOMPUTER CENTRUM

189

Links

- Windows
  - Windows Terminal: docs.microsoft.com/en-us/windows/terminal/
  - Windows Subsystem for Linux: docs.microsoft.com/en-us/windows/wsl/install-win10
  - Cygwin: www.cygwin.com
  - MobaXterm: mobaxterm.mobatek.net
  - PuTTY: www.chiark.greenend.org.uk/~sgtatham/putty
  - WinSCP: winscp.net

VLAAMS SUPERCOMPUTER CENTRUM

190

Links

- macOS:
  - iTerm2: iterm2.com
  - XQuartz: www.xquartz.org
  - CyberDuck: cyberduck.io
- All:
  - FileZilla: filezilla-project.org
  - TurboVNC: www.turbovnc.org

VLAAMS SUPERCOMPUTER CENTRUM

191

Links

- VSC main web site: vscentrum.be
- VSC documentation: docs.vscentrum.be
  - VSC hardware – Tier-2 hardware – UAntwerpen Tier-2 hardware is an important section
- Local UAntwerp CalcUA web site: hpc.uantwerpen.be

VLAAMS SUPERCOMPUTER CENTRUM

192