



Vlaanderen
is supercomputing

Bioinformatics Analysis at VSC

Alexander Vapirev, ICTS, KU Leuven

with thanks to:

Mag Selwa, Ingrid Barcena, Jan Ooghe, Ehsan Moravveji (ICTS HPC)

Alvaro Cortes (UZ Leuven), Kris Davie (VIB Leuven)

VLAAMS
SUPERCOMPUTER
CENTRUM

*Innovative Computing
for A Smarter Flanders*

vscentrum.be

In this talk...

Part I: The Flemish Supercomputing Center

- What is the VSC?
- VSC infrastructure
- Software environment
- How to get started
- Training and events

Part II: Introduction to Linux

- The Linux environment
- Key concepts
- Basic commands and tools



Vlaanderen
is supercomputing

Part I: Vlaams Supercomputer Centrum

HPC OPTIONS

In-house



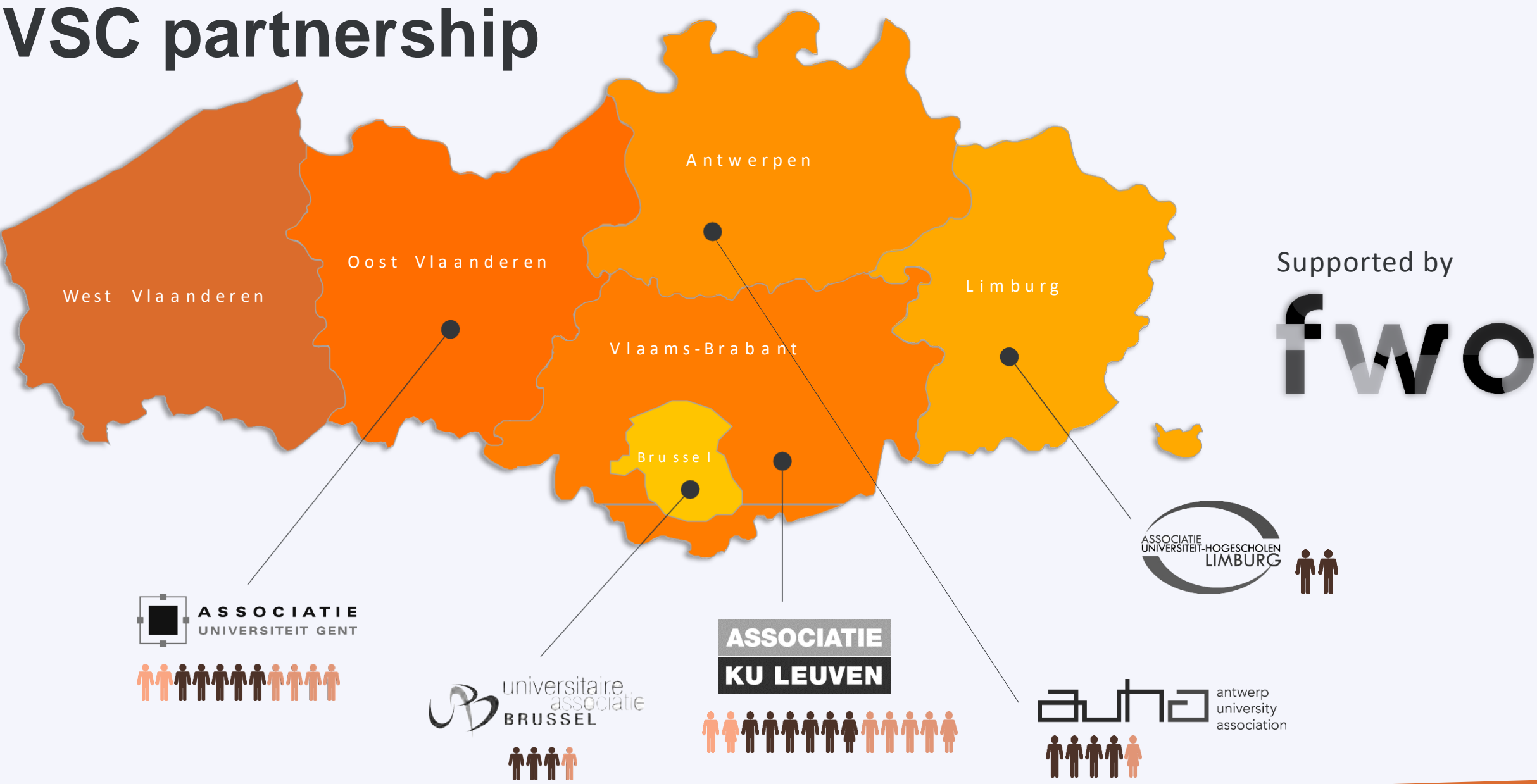
Cloud



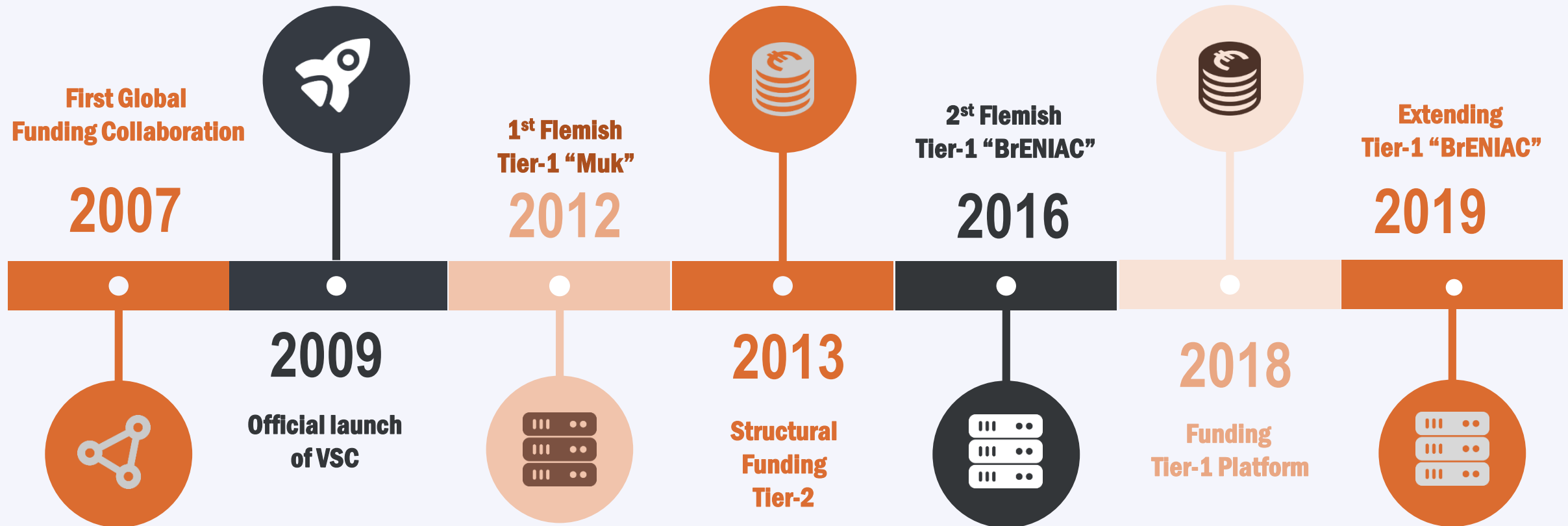
VSC



VSC partnership

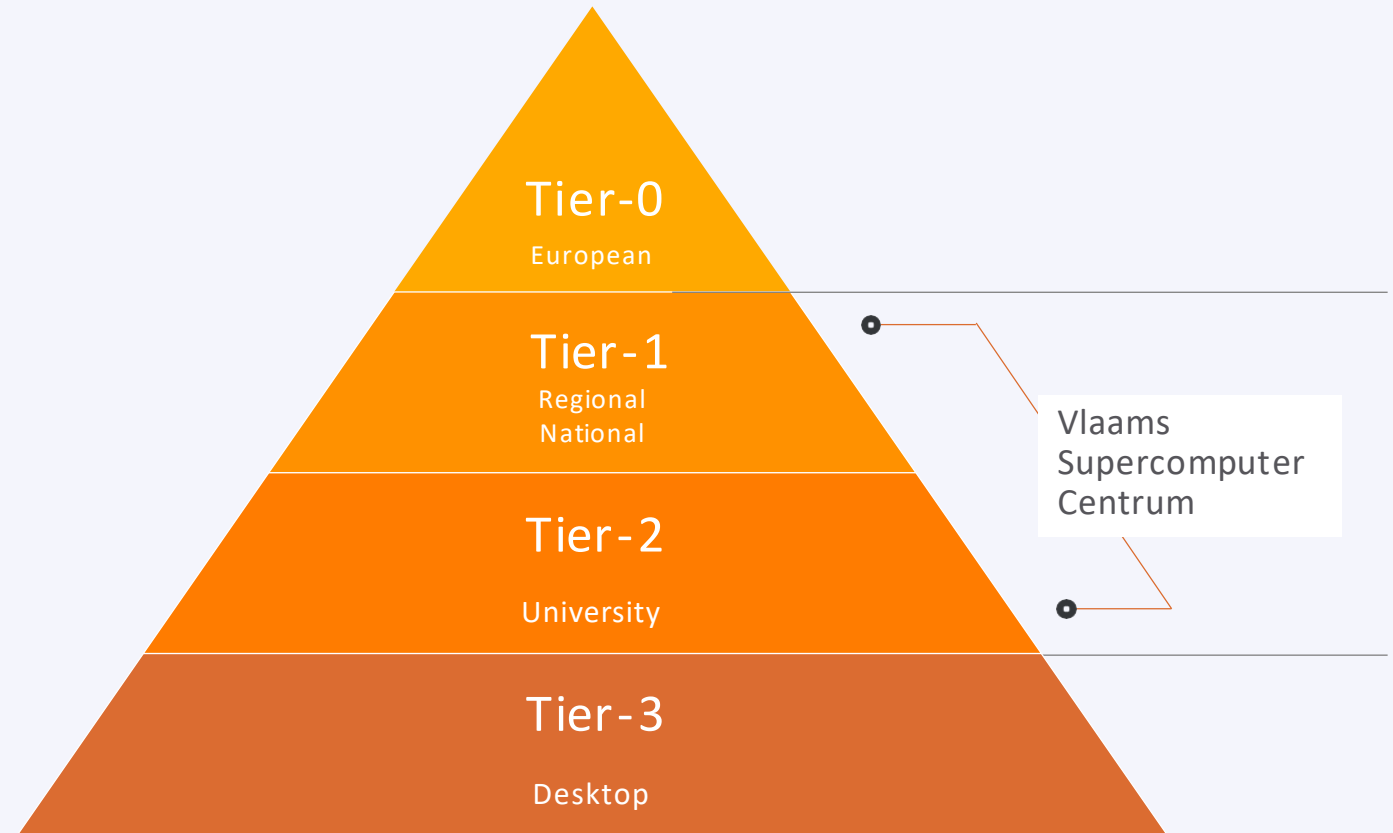


VSC history



HPC tiering model

In the European model for High Performance Computing (HPC), a distinction is made between three levels: the processing capacity which research institutes have at their disposal (Tier-2), the processing capacity which goes beyond the needs and costs of an individual institution and is provided at regional or country level (Tier-1), and the super powerful processing infrastructure (Tier-0).



VSC HPC environment

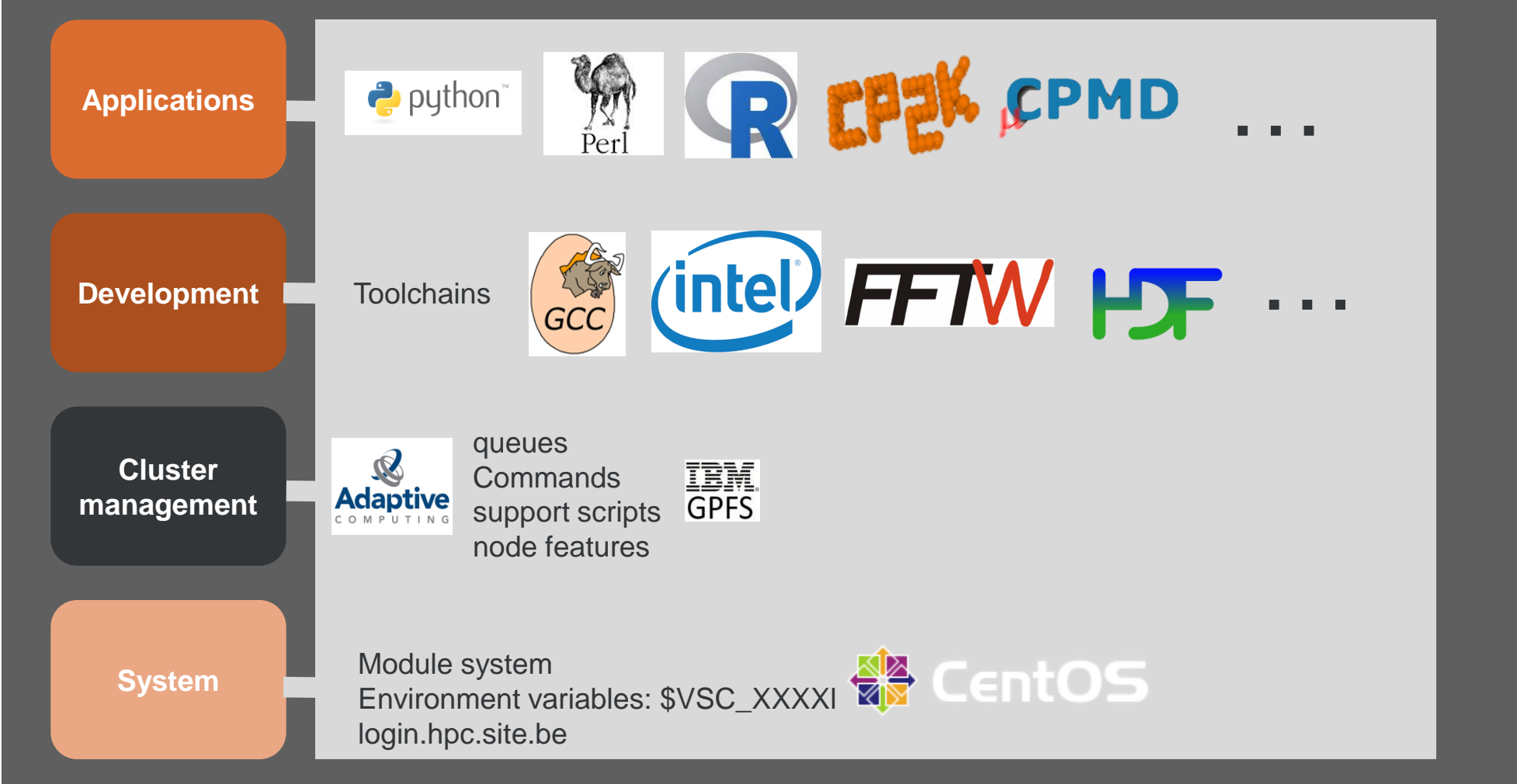


Tier-2 vs Tier-1

	Tier 2	Tier 1
Access	Project Credits	Panel Review ^[1,2]
Free Introduction Credits	2,000 credits ^[4]	500 node days ^[2]
Project Credits	Pay as you GO ^[5,6]	Free (if approved) ^[3]
Computation Limit	None	500 - 5,000 node days
Application deadline	None	6 February 5 June 2 October
Max. Project Duration	Unlimited	8 months
Available walltime	1h, 24h, 72h, 7d, 21d	Max. 3 days

1. The TAB reviewers (international team) are appointed by FWO.
 2. Application form: <https://www.vscentrum.be/en/access-and-infrastructure/tier1-starting-grant>
 3. We support your application (technicalities, scale up, configuration etc) if you send us your application form **max 1 week before** the cut-off deadline.
 4. Intro credits: <https://admin.kuleuven.be/icts/onderzoek/hpc/request-introduction-credits>
 5. Project credits: https://icts.kuleuven.be/sc/forms/Aanvraagformulier_HPC_Credits
 6. 1,000 credits = 3.5 EURO
- For all services check the ICTS service catalog: <https://icts.kuleuven.be/sc>

Software stack



VSC Services

Basic support

- Monitoring and reporting
- Helpdesk

Application support

- Installation and porting
- Optimisation and debugging
- Benchmarking
- Workflows and best practices

Training

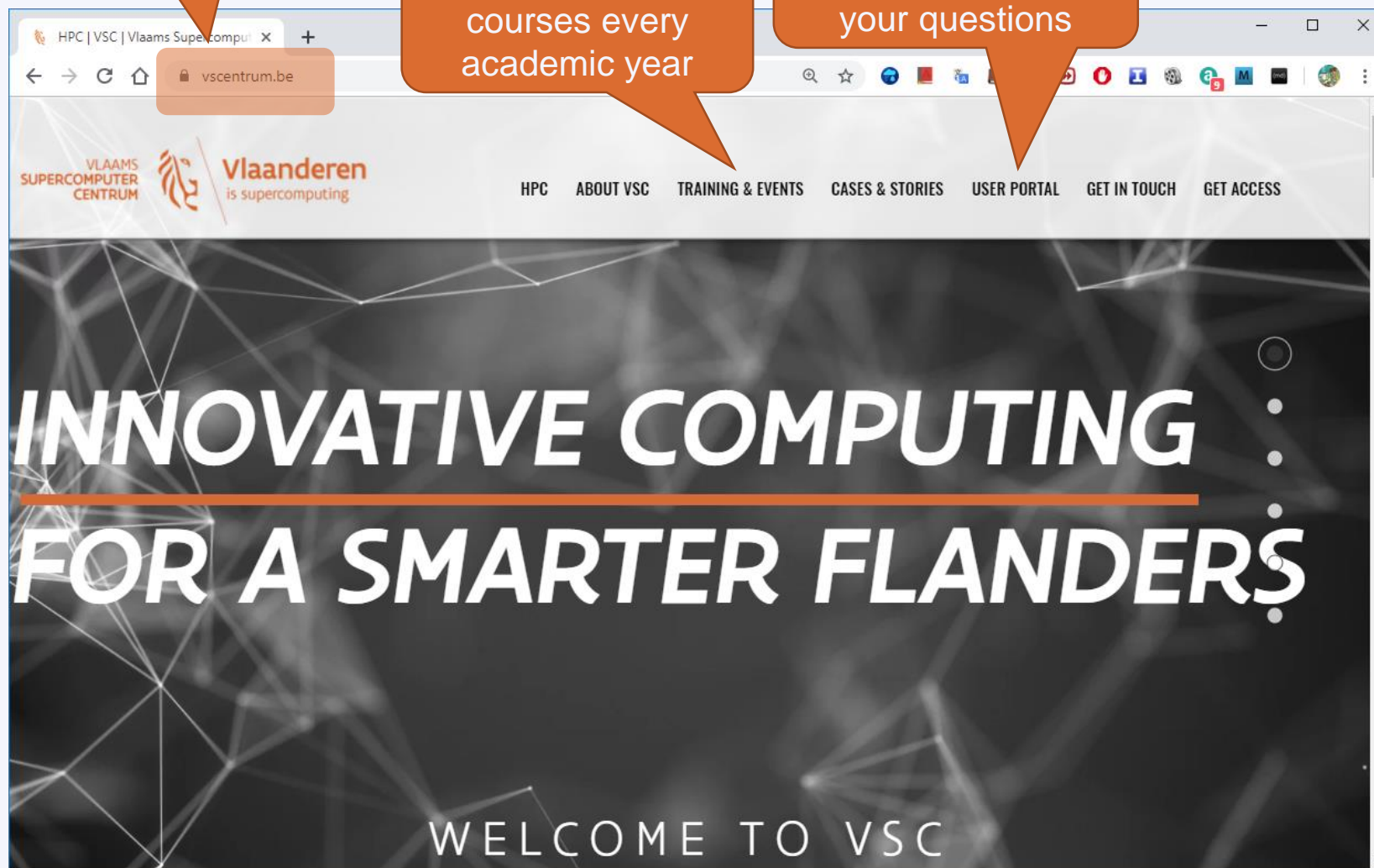
- Documentation and tutorials
- Scheduled trainings / workshops
- On request workshops
- One-to-one sessions

WWW

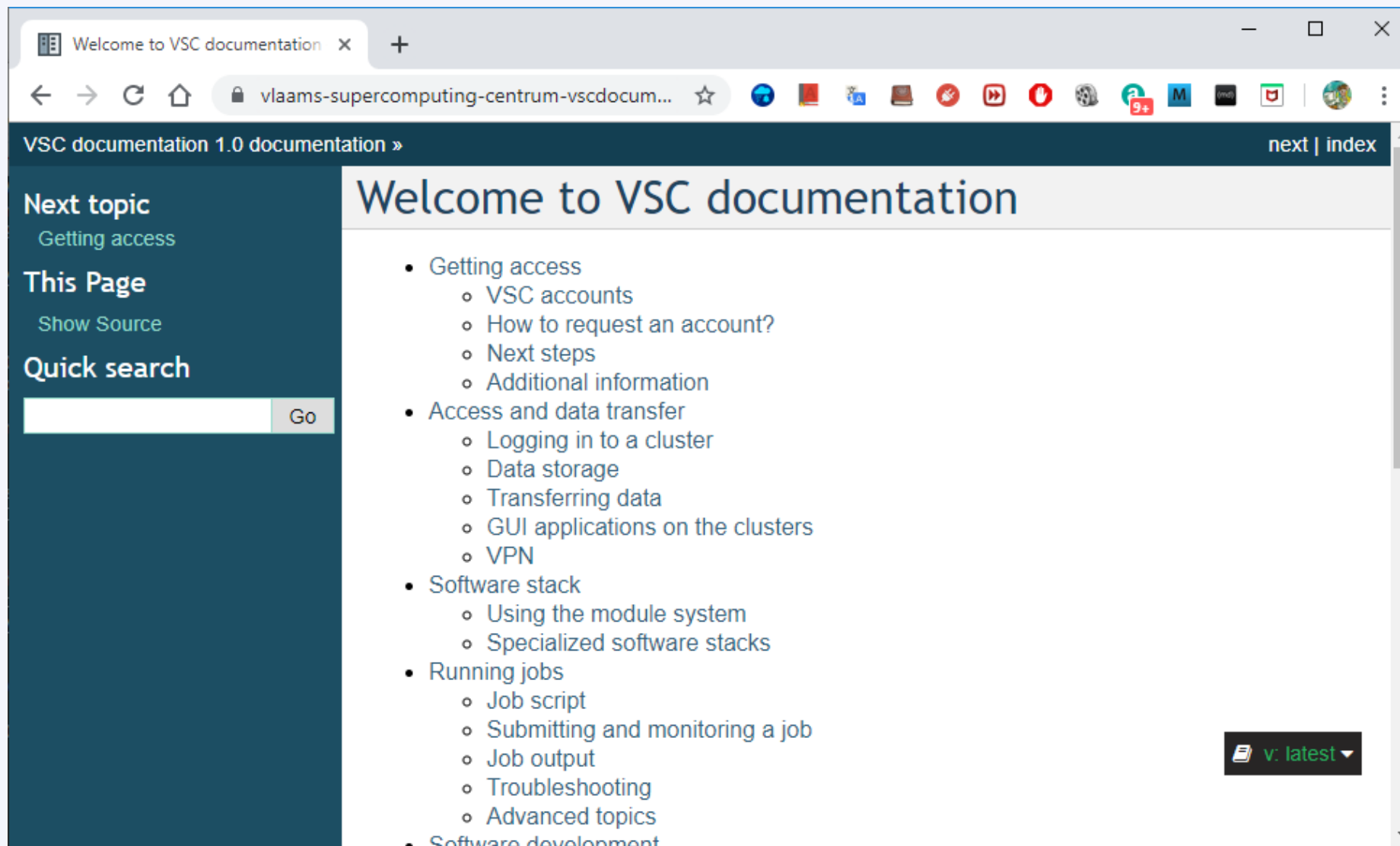
www.vscentrum.be

Very rich suite of
courses every
academic year

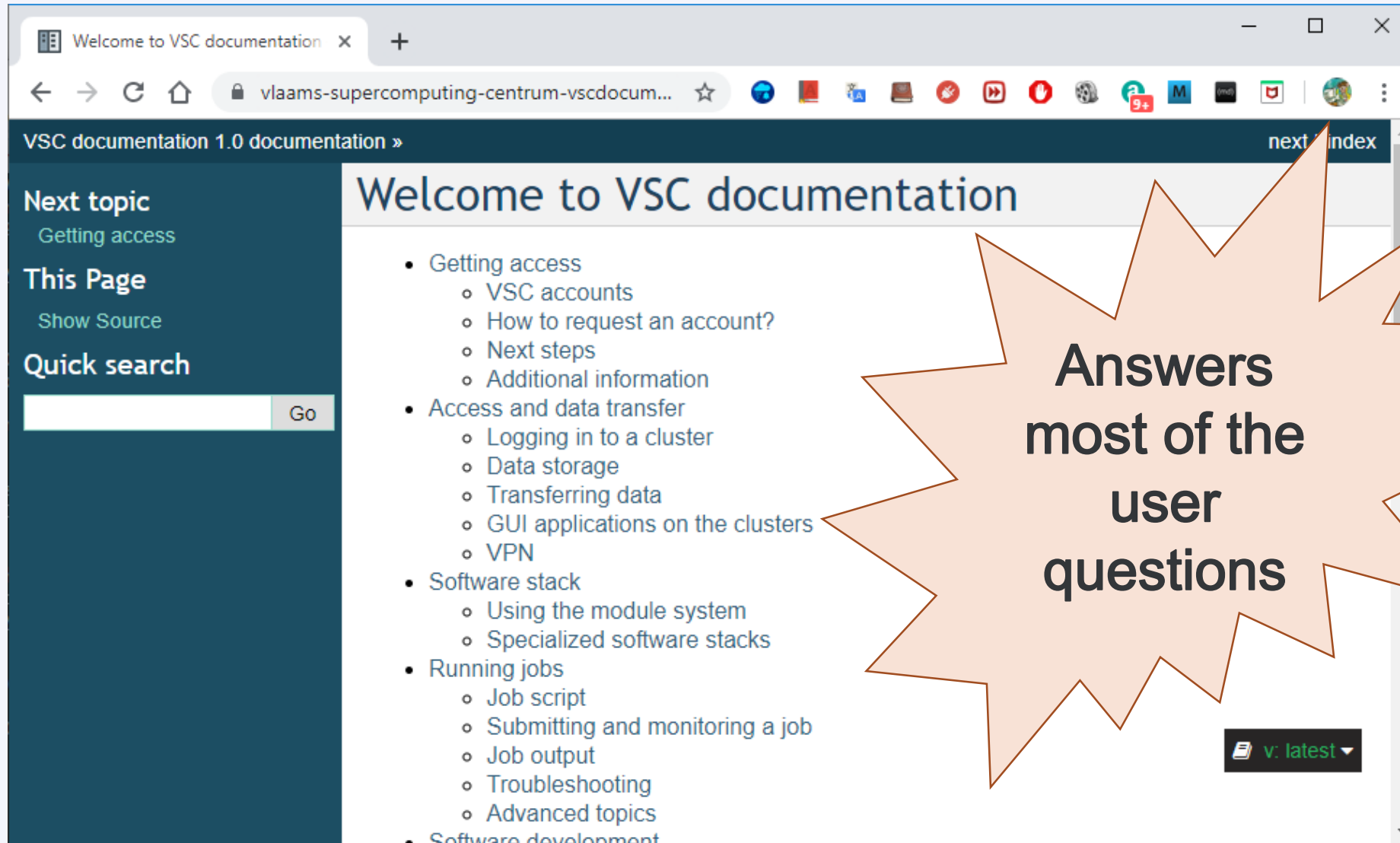
Documentation!
Answers >80% of
your questions



User documentation



User documentation



The screenshot shows a web browser window displaying the 'Welcome to VSC documentation' page. The browser's address bar shows the URL 'vlaams-supercomputing-centrum-vscdocum...'. The page has a dark blue sidebar on the left with links for 'Next topic' (Getting access), 'This Page' (Show Source), and a 'Quick search' bar. The main content area lists several topics: 'Getting access' (VSC accounts, How to request an account?, Next steps, Additional information), 'Access and data transfer' (Logging in to a cluster, Data storage, Transferring data, GUI applications on the clusters, VPN), 'Software stack' (Using the module system, Specialized software stacks), 'Running jobs' (Job script, Submitting and monitoring a job, Job output, Troubleshooting, Advanced topics), and 'Software development'. A large, orange, starburst-shaped graphic is overlaid on the right side of the page, containing the text 'Answers most of the user questions'. In the bottom right corner of the page, there is a small black button with a document icon and the text 'v: latest'.

Welcome to VSC documentation

- Getting access
 - VSC accounts
 - How to request an account?
 - Next steps
 - Additional information
- Access and data transfer
 - Logging in to a cluster
 - Data storage
 - Transferring data
 - GUI applications on the clusters
 - VPN
- Software stack
 - Using the module system
 - Specialized software stacks
- Running jobs
 - Job script
 - Submitting and monitoring a job
 - Job output
 - Troubleshooting
 - Advanced topics
- Software development

Answers most of the user questions

v: latest

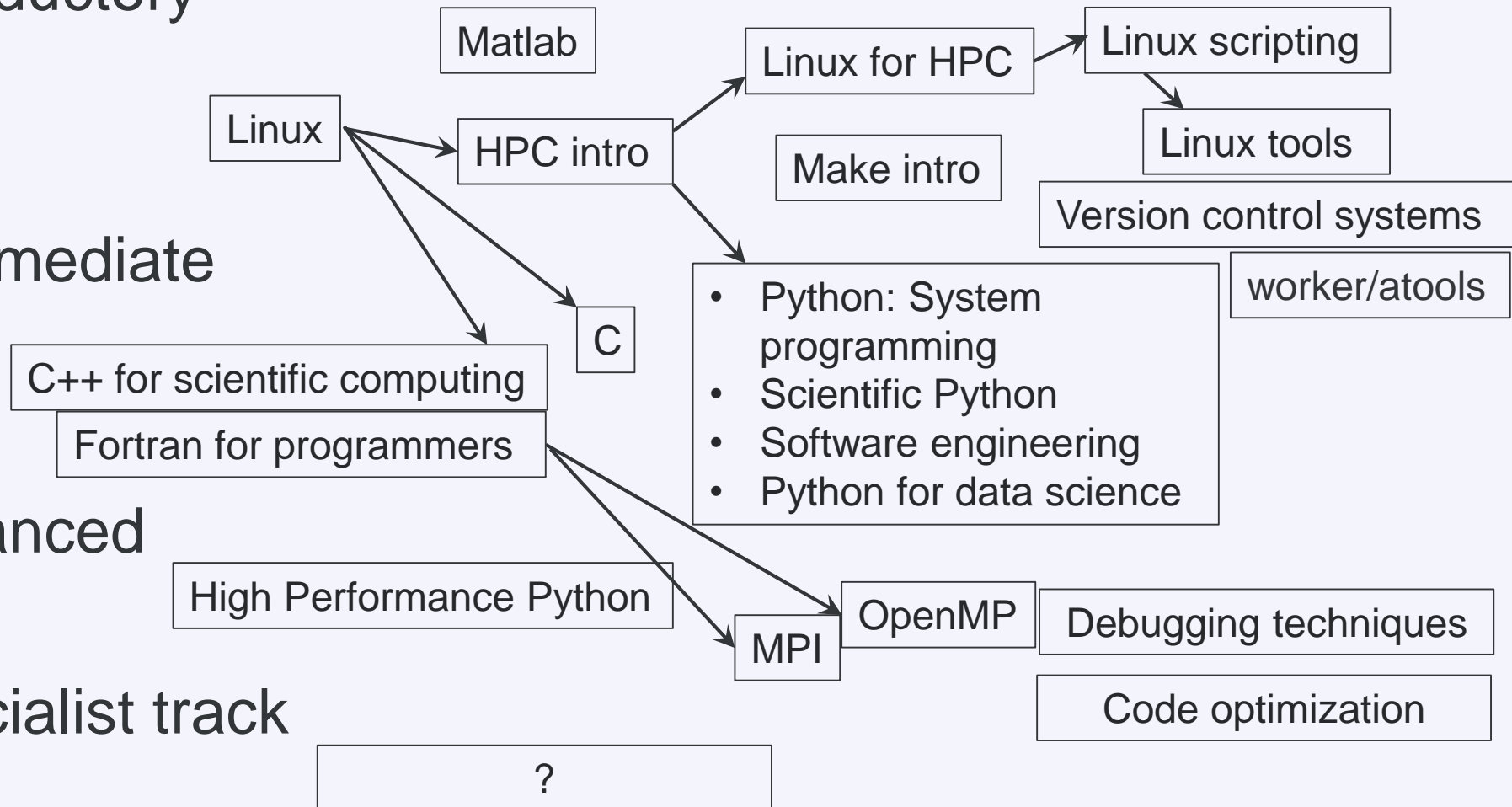
Training and courses at VSC

- Introductory

- Intermediate

- Advanced

- Specialist track



Lunchbox sessions:

- Containers
- Notebooks

Stay up-to-date <https://www.vscenrum.be/en/education-and-trainings>



Vlaanderen
is supercomputing

Tier-2 Status 2019

01/03/2019

KU Leuven Tier-2 environment

Accelerators



20*4 P100

ThinKing



230 Tflop/s
352 nodes / 7.616
cores
FDR

Cerebro



12 Tflop/s
640 cores
14 TB RAM
Numalink

2018 Genius



307 Tflop/s
116 nodes / 4.176 cores
EDR

Login nodes



2+4 login nodes
2 NX nodes
2 visualisation nodes

Storage



60 TB NFS
snapshots

High Perf Storage



1 PB GPFS
/scratch => individual
/staging => group

Storage - Archi



1 PB GPFS
600 TB Archive

KU Leuven Tier-2 environment

Accelerators



20*4 P100

ThinKing

Ivybridge Haswell
208 nodes 144 nodes

Cerebro



12 Tflop/s
640 cores
14 TB RAM
Numalink

2018 Genius



307 Tflop/s
116 nodes / 4.176 cores
EDR

2019 Superdome



8 CPUs/ 112 cores
6TB RAM

Login nodes



2+4 login nodes
2 NX nodes
2 visualisation nodes

Storage



60 TB NFS
snapshots

High Perf Storage



1 PB GPFS
/scratch => individual
/staging => group

Storage - Archi



1 PB GPFS
600 TB Archive

KU Leuven Tier-2 environment

Accelerators



20*4 P100

ThinKing

Ivybridge
208 nodes

Haswell
144 nodes

Cerebro



12 Tflop/s
640 cores
14 TB RAM
Numalink

2018 Genius



307 Tflop/s
116 nodes / 4.176 cores
EDR

2019 Superdome



8 CPUs/ 112 cores
6TB RAM

Login nodes



2+4 login nodes
2 NX nodes
2 visualisation nodes

Storage



60 TB NFS
snapshots

High Perf Storage



1 PB GPFS
/scratch => individual
/staging => group

Storage - Archi



1 PB GPFS
600 TB Archive

KU Leuven Tier-2 environment

Accelerators



20*4 P100

ThinKing

EOL

Haswell
144 nodes

Cerebro



EOL

2018 Genius



307 Tflop/s
116 nodes / 4.176 cores
EDR

2019 Superdome



8 CPUs/ 112 cores
6TB RAM

Login nodes



2+4 login nodes
2 NX nodes
2 visualisation nodes

Storage



60 TB NFS
snapshots

High Perf Storage



1 PB GPFS
/scratch => individual
/staging => group

Storage - Archi



EOL



Parallel computing

Basic concepts

Bird's eye view of a cluster

Accessed via **login nodes**

- For job submission, debugging, pre/post processing
- *Shared resources*: everyone works on the same (set of) login nodes

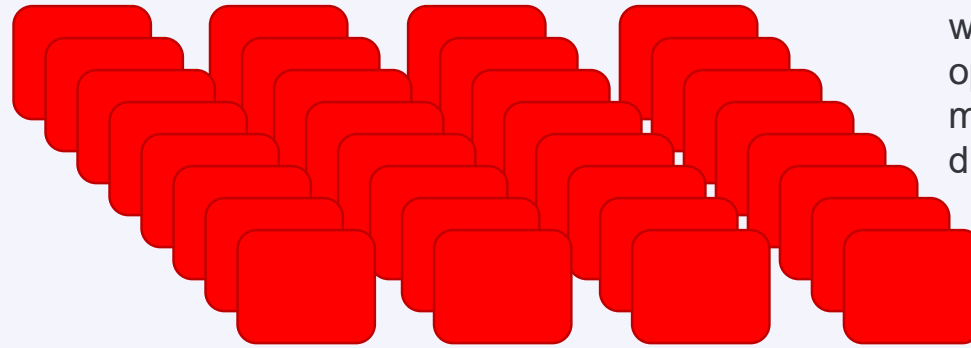


Researchers



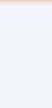
Cluster

Compute nodes

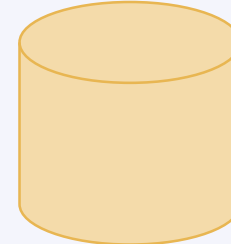


Many independent computers, each with its own operating system, memory, hard disk,...

Infiniband interconnect



Login nodes



Storage



Service nodes

Administration - queue system, job scheduler, user management,...

Parallel computing

- **Serial:**
 - one program, on one core
- **'Embarrassingly parallel' problems:**
 - lots of runs of one program, with different parameters
 - Hadoop, Spark
- **Problems that require 'real' parallel algorithms**
 - OpenMP
 - MPI : Message Passing Interface
 - GPU/Accelerator parallelism interface

Parallel computing

- **Serial:**

- one program, on one core

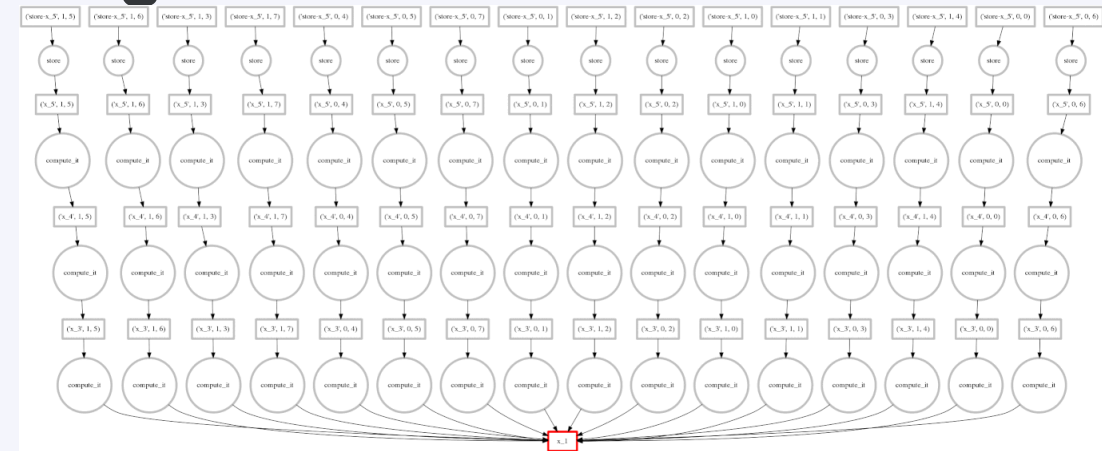
- **'Embarrassingly parallel' problems:**

- lots of runs of one program, with different parameters
- Hadoop, Spark

- **Problems that require 'real' parallel algorithms**

- OpenMP
- MPI : Message Passing Interface

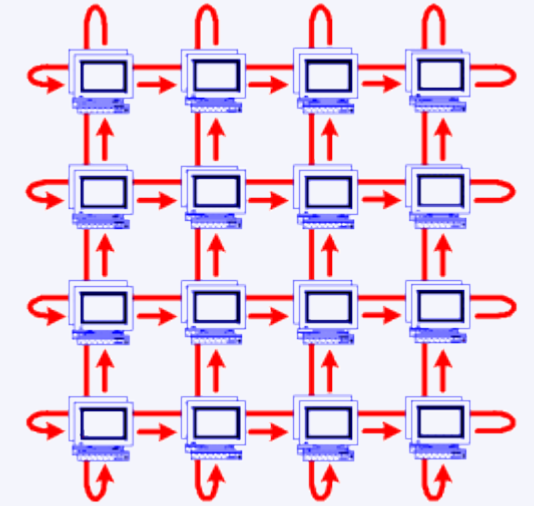
Python
R
Java



Credit: <http://dask.pydata.org/en/latest/shared.html>

Parallel computing

- **Serial:**
 - one program, on one core
- **'Embarrassingly parallel' problems:**
 - lots of runs of one program, with different parameters
 - Hadoop, Spark
- **Problems that require 'real' parallel algorithms**
 - OpenMP
 - MPI : Message Passing Interface



C/C++
Fortran

Credit: <https://decisionstats.com/2010/09/24/parallel-programming-using-r-in-windows/>

What a cluster is *not*...

- An all-powerful and super smart computer that will ***automatically***:
 - Run your application much faster
 - Run your application in parallel

For that you will need to do some extra work

Transition to HPC

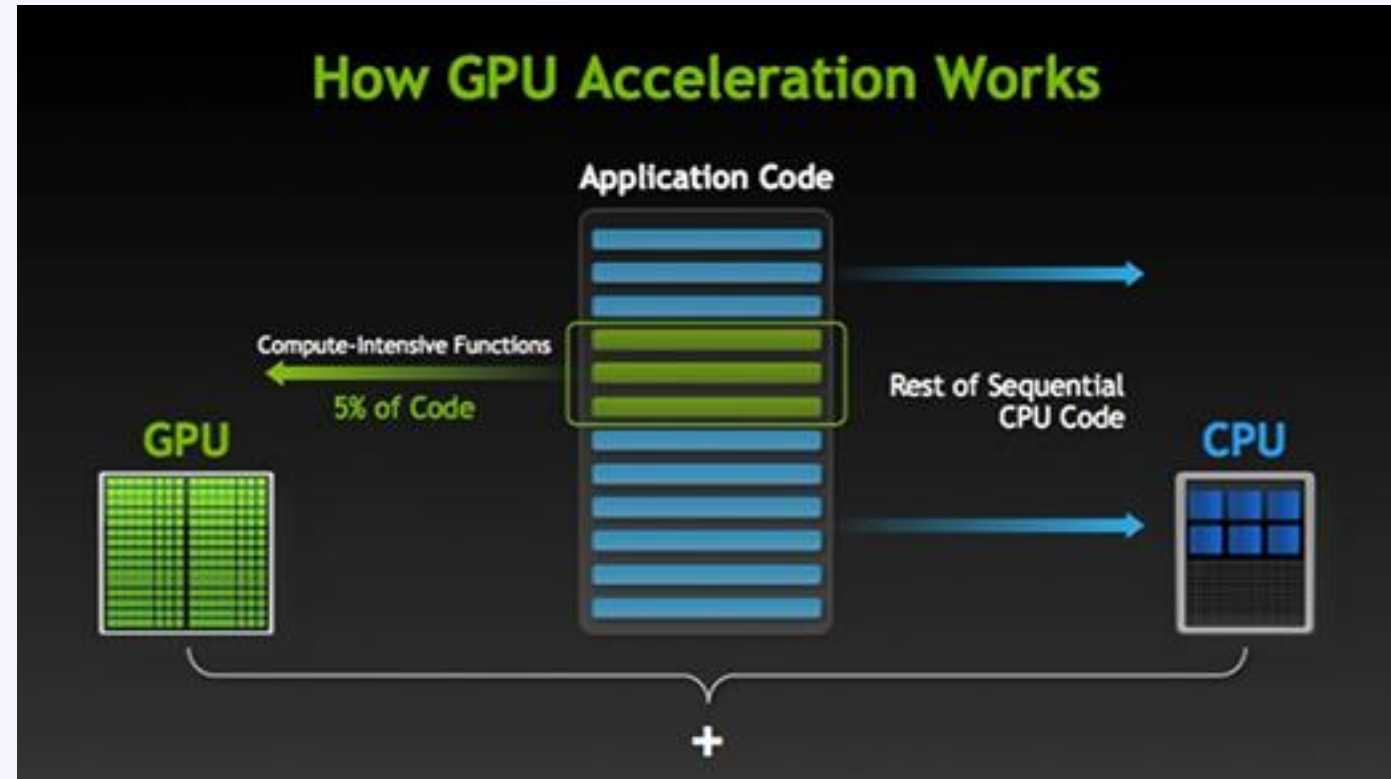
- Moving data to the HPC infrastructure
- Software
 - Existing tools are often not HPC-compatible
 - Often do not scale (algorithm productivity is not straightforward proportional to the processing power)
 - General “big data” algorithms often have to be specifically adapted for NGS compute and analysis
- Post-processing and analysis of the results
 - Analysis of massive datasets is not trivial
 - Query of the results
 - Generating reports



Accelerators (GPUs)

What is GPU accelerated computing

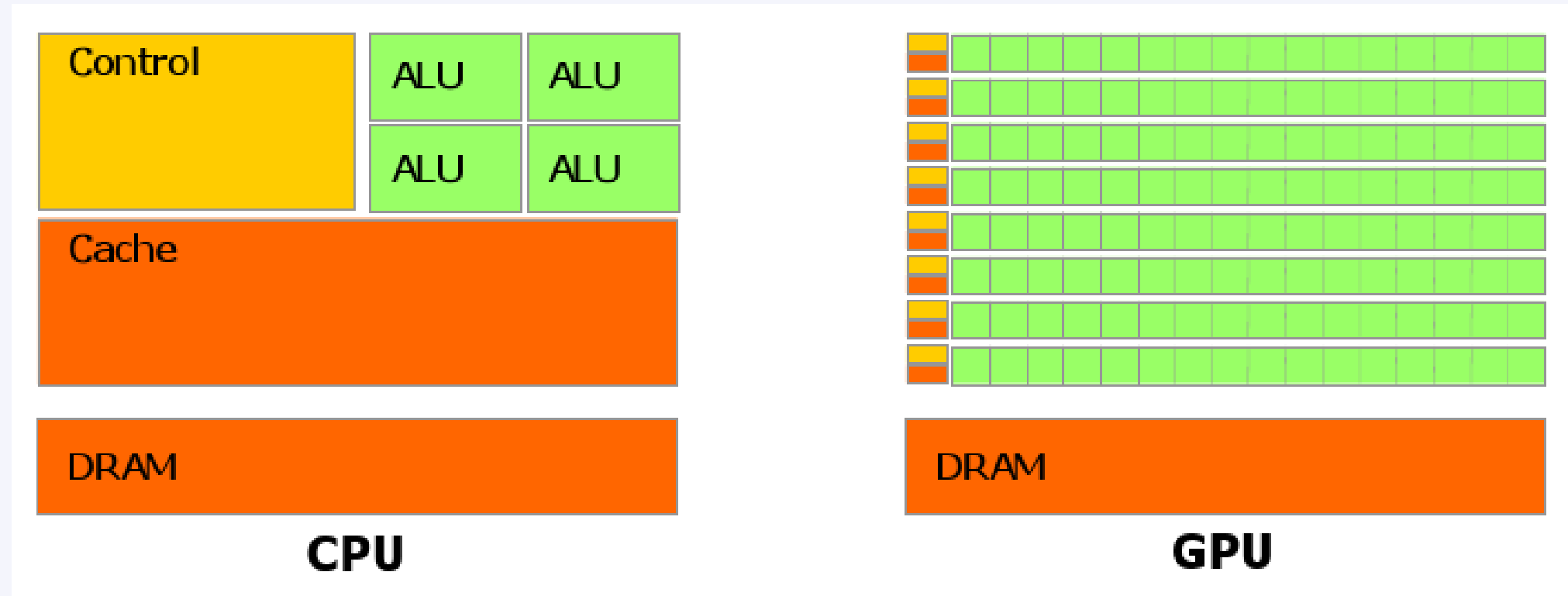
- GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate applications.
- GPU-accelerated computing offers better performance by offloading compute-intensive portions of the application to the GPU, while the remainder of the code still runs on the CPU. From a user's perspective, applications simply run significantly faster.



CPU vs GPU

- **CPU** design goal : optimize architecture for sequential code performance : **minimize latency experienced by 1 thread**
 - **sophisticated** (i.e. large chip area) **control logic** for instruction-level parallelism (branch prediction, out-of-order instruction, etc...)
 - **CPU have large cache memory** to reduce the instruction and data access latency
- **GPU** design goal : **maximize throughput of all threads**
 - # threads in flight limited by resources => lots of resources (registers, bandwidth, etc.)
 - multithreading can **hide latency** => skip the big caches
 - **share control logic** across many threads

CPU vs GPU



GPU Devotes More Transistors to Data Processing (is designed such that about 80% of transistors are devoted to data processing rather than data caching and flow control - the same function is executed on each element of data with high arithmetic intensity).



Data storage

Overview

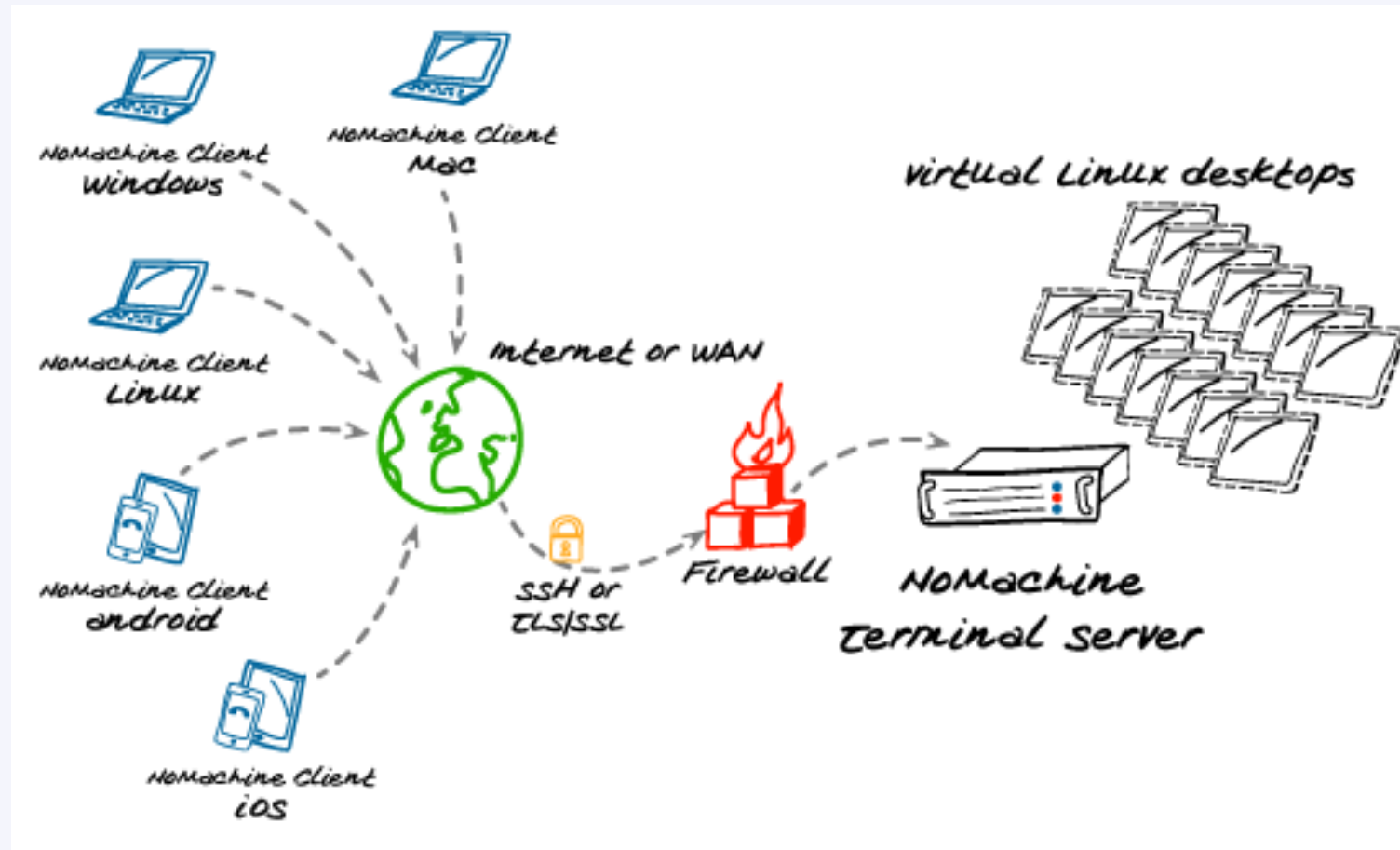
HPC cluster storage at KU Leuven consists of 3 different storage types, optimized for different usage, with different characteristics

- **VSC_HOME (3GB)** and **VSC_DATA (75GB)**
 - NAS storage, fully backed-up with snapshots
- **VSC_SCRATCH** storage – starts at 100GB, can be extended
 - fast parallel filesystem
- **Staging** storage
 - fast parallel filesystem
- **Archive** storage
 - to store large amounts of data for a long time



The NX linux virtual desktop

What is NX

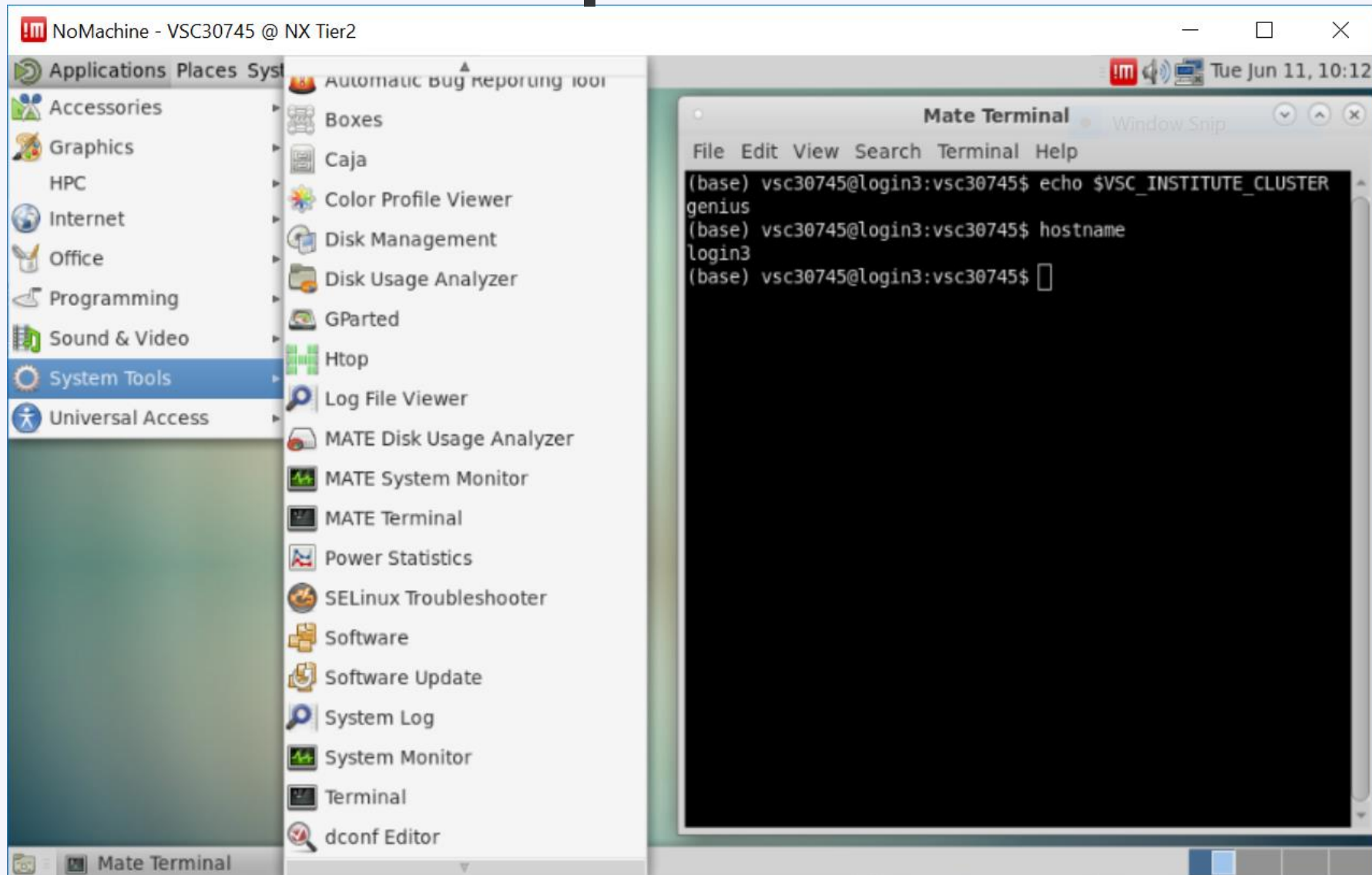


NX works by creating an nx-user on the server machine whose shell is executed any time a remote NX user connects to SSH using NX Client.

Main advantages of NX

- NX allows to suspend and resume sessions and keeps session open (disconnected up to 30 days),
- During suspension, the processes invoked inside the session continue to run,
- Alternative for people using screen,
- More interactive jobs,
- Easy in use for editing, file management, developing software,
- Different limits of CPU (regular login node 36 min, NX node extended to 2 hrs).

The NX virtual desktop





Login nodes

Login nodes and hostnames

To login (with PuTTY or SSH client), you need VSC number and a hostname

```
$> ssh -X vscXXXXX@<hostname>
```

Cluster / Partition	<hostname>	Remark(s)
ThinKing	login-thinking.hpc.kuleuven.be	Recommended
	Login5-tier2.hpc.kuleuven.be	Oldest partition: IvyBridge
	login6-tier2.hpc.kuleuven.be	
	login7-tier2.hpc.kuleuven.be login8-tier2.hpc.kuleuven.be	Haswell partition
Genius	login.hpc.kuleuven.be login-genius.hpc.kuleuven.be	Recommended
	login{1,2}-tier2.hpc.kuleuven.be	No GPU
	login{3,4}-tier2.hpc.kuleuven.be	Nvidia Quadro P6000
Superdome	Any Genius login node (above)	module load superdome



Jobs and job scheduler

Job scheduler: Moab

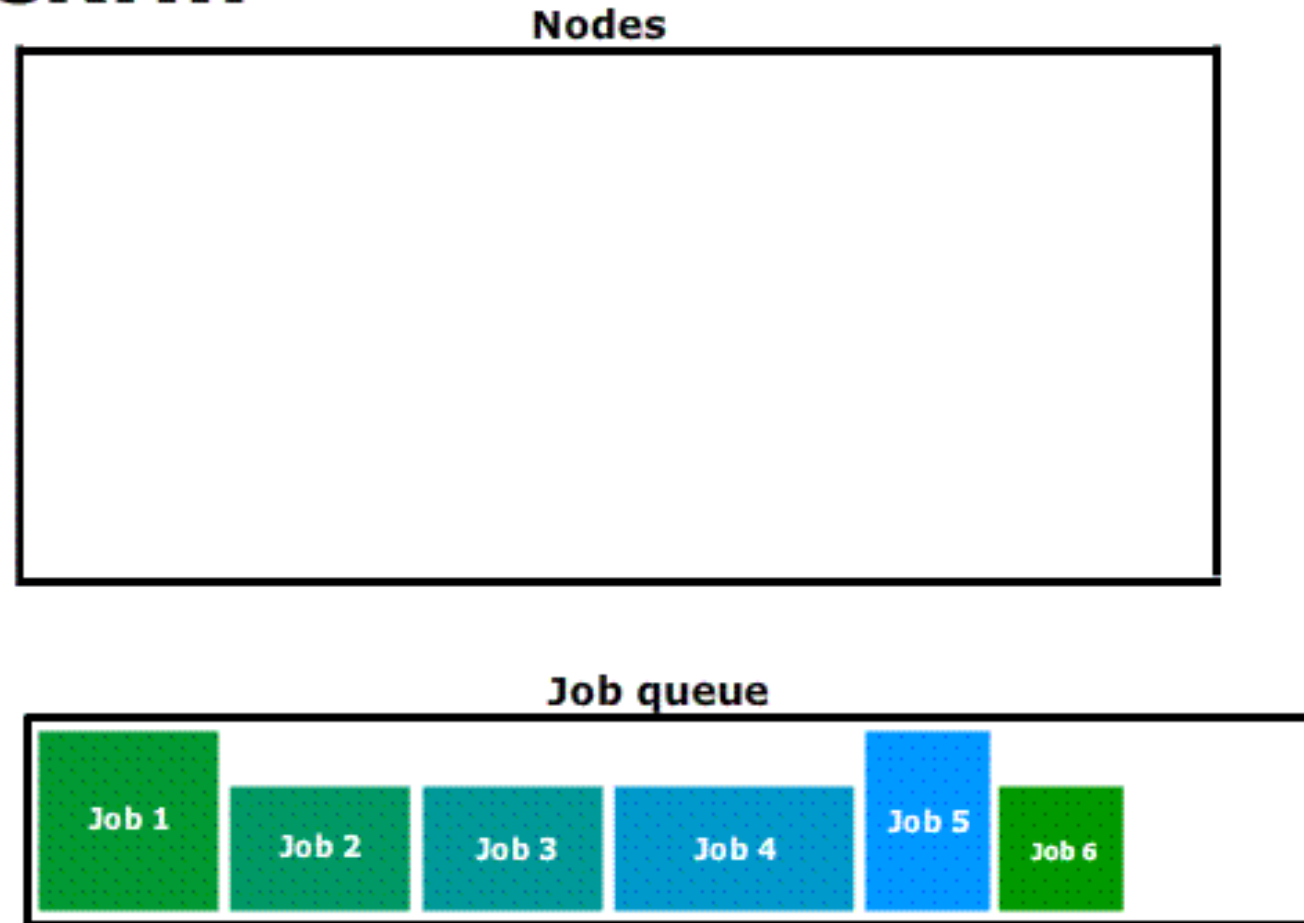
- The job scheduler decides when and where to run jobs using a priority queue
 - Queries resource manager for runnable jobs
 - Tries to optimize resource usage and
 - favors parallel jobs,
 - fair use of resources for all users (taking into account computation during past 7 days),
 - ensures liveness, regardless of starting priority, job will eventually have highest priority
 - Orders resource manager to start (or stop ...) jobs

Job scheduler: Moab

- the whole procedure is redone in every iteration (every minute or so)
- a job can lose its reservation, if a higher priority job gets submitted in the meantime
- a job can start earlier, if a job ends earlier than specified
- the “showstart” command gives a starttime estimate obtained in the last iteration

Job scheduler: Moab

Backfill



Types of jobs

- **Batch jobs:** `qsub mypbs.sh`
 - allow for the most efficient use of the infrastructure
 - a bash shell script that is executed on a compute node
 - are placed in the queue, and the user can forget about it until it is finished.
- **Interactive jobs:** `qsub -I mypbs.sh`
 - are intended to work on one or more compute nodes interactively
 - useful for software development, debugging, application profiling
 - basically, one gets a linux shell on one of the compute nodes.

Job submission

- From command line

```
$ qsub -l walltime=10:00:00 -l <other params> myjob.pbs
```

- In a PBS job script/file

```
#!/bin/bash -l
#PBS -l walltime=10:00:00
#PBS -l nodes=2:ppn=1
#PBS -l pmem=3gb
#PBS -N myjob2
#PBS -A default project
#PBS -m abe
#PBS -M my.name@icts.kuleuven.be

<user's workflow here>
```

If not specified:
walltime=01:00:00

If not specified:
nodes=1:ppn=1

If not specified:
pmem=2400/5000mb

If not specified: job will not start

- default_project = intro credits
- lp_my_project_ = project credits

PBS environment variables

When your PBS script runs, some environment variables are automatically available

- `PBS_O_WORKDIR=/vsc-hard-mounts/leuven-user/304/vsc30468` directory from where job was submitted
- `PBS_JOBID=20590968.moab.tier2.leuven.vsc` job ID of current job
- `PBS_JOBNAME=STDIN` user specified jobname
- `PBS_WALLTIME=3600` requested walltime
- `PBS_NUM_NODES=2` number of nodes allocated to the job
- `PBS_NP=40` number of execution cores for the job
- `PBS_NUM_PPN=20` number of procs per node allocated to the job
- `PBS_NODENUM=0` unique number 0 upwards for each different node
- `PBS_VNODENUM=0` the number of the physical core in your allocation. The numbering continues across the nodes in the allocation
- `PBS_NODEFILE=/var/spool/torque/aux//20590968.moab.tier2.leuven.vsc` file containing line delimited list on nodes allocated to the job
- `PBS_O_PATH` path variable used to locate executables within job script
- `PBS_TASKNUM=1` number of the task

Compute credits

How to check available credits?

```
$ module load accounting; mam-balance
```

Credits card concept:


- **Preauthorization**: holding the balance as unavailable until the merchant clears the transaction
- Balance to be held as unavailable: based on requested resourced (walltime, nodes)
- **Actual charge** based on what was really used: used walltime (you pay only what you use, e.g. when job crashes)

Compute credits

Project credits valid for all Tier-2 clusters:

- ThinKing
- GPU
- Genius
- Superdome

$$\#credits = (0.000278 \cdot \text{walltime} \cdot \#nodes) \cdot f_{type}$$

 $\rightarrow 1/3600$

$$f_{type} = \begin{cases} 4.76 & \text{ThinKing IvyBridge} \\ 6.68 & \text{ThinKing Haswell} \\ 2.86 & \text{ThinKing GPU} \\ 0 & \text{Superdome} \\ 10 & \text{Genius Thin node} \\ 12 & \text{Genius large memory} \\ 20 & \text{Genius GPU (full node 4xP100)} \\ 5 & \text{Genius GPU 1/4 (1xP100)} \end{cases}$$

Example: `-l walltime=1:00:00 -l nodes=1:ppn=1`

$$\#credits = (0.000278 \cdot 3600 \cdot 1) \cdot 10 = 10.01$$

`-l walltime=1:00:00 -l nodes=1:ppn=36`

$$\#credits = (0.000278 \cdot 3600 \cdot 1) \cdot 10 = 10.01$$

Managing and monitoring of jobs

Command	Purpose
<code>\$> qsub ...</code>	Submit a job (batch/interactive)
<code>\$> qdel <JobID></code>	Delete a specific job
<code>\$> checkjob -v -v -v <JobID></code>	Very detailed job info (very useful to diagnose issues)
<code>\$> qstat -n</code>	Status of all recent jobs
<code>\$> qstat -Q -f</code>	Info about available queues
<code>\$> showstart <JobID></code>	Give a <i>rough</i> estimate of start time
<code>\$> showq</code> <code>\$> showq -p gpu</code>	Show minimal info about a queue or partition (-p)
<code>\$> pbstop</code>	Overview of the cluster
<code>\$> mam-balance</code>	Overview of different credit projects that you can use (<code>qsub -A <Project></code>)
<code>\$> mam-list-allocations</code>	Detailed overview of your credit projects



Vlaanderen
is supercomputing

Software

toolchains, modules

Toolchains on VSC

	intel tool chain	foss tool chain
Name	intel	foss
version	2018a	2018a
Compilers	Intel compilers (v 2018.1.163) icc, icpc, ifort	GNU compilers (v 6.4.0-2.28) gcc, g++, gfortran
MPI Library	Intel MPI	OpenMPI
Math libraries	Intel MKL	OpenBLAS, LAPACK FFTW, ScaLAPACK

Never mix different toolchains!

Modules

- `$ module available` or `module available Py`
 - Lists all installed software packages
- `$ module av |& grep -i python`
 - To show only the modules that have the string 'python' in their name, regardless of the case
- `$ module load Python/2.7.14-foss-2018a`
 - Adds the 'matlab' command in your PATH
- `$ module load GCC`
 - 'Load' the (default) GCC version – not recommended, not reproducible
- `$ module list`
 - Lists all 'loaded' modules in current session
- `$ module unload Python/2.7.14-foss-2018a`
 - Removes all only the selected module, other loaded modules – dependencies are still loaded
- `$ module purge`
 - Removes all loaded modules from your environment

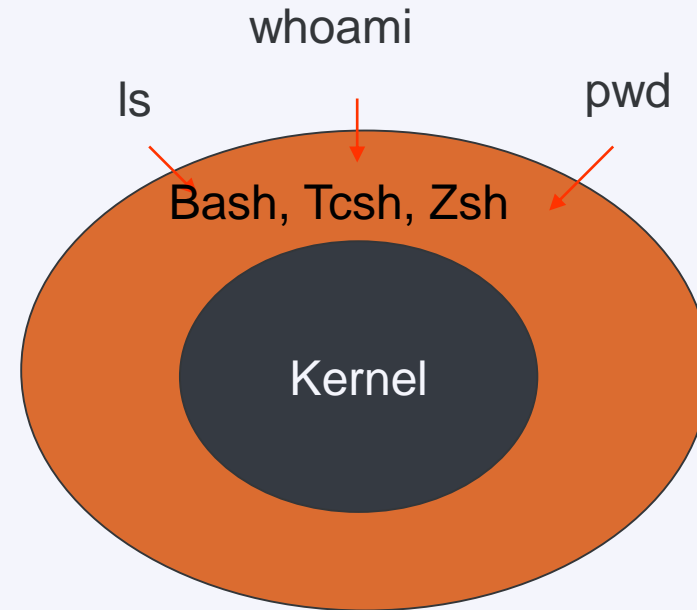


Vlaanderen
is supercomputing

Part II: Intro to Linux

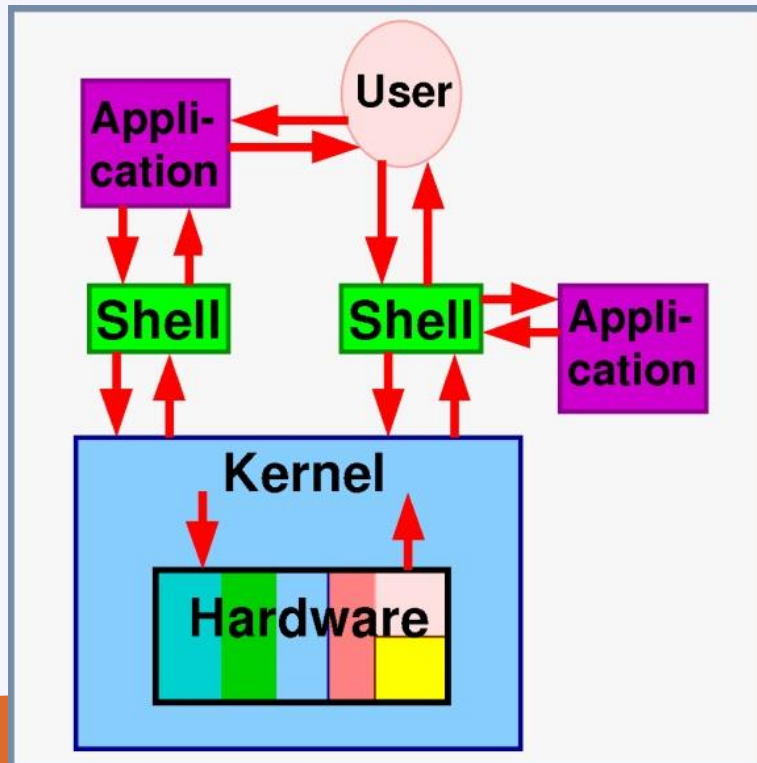
The Linux shell

- Shell interprets the command and request service from kernel
- Similar to DOS but DOS has only one set of interface while Linux can select different shell
- Bourne Again shell (Bash), TC shell (Tcsh), Z shell (Zsh)
- Different shell has similar but different functionality
- Bash is the default for Linux
- Graphical user interface (GUI) in Linux is in fact an application program working in the shell



The Linux shell

- Not just an interface to the computer, also a scripting language – allows automation of tasks
- Shells can be scripted: provide all the resources to write complex programs (variables, conditionals, iterations...)

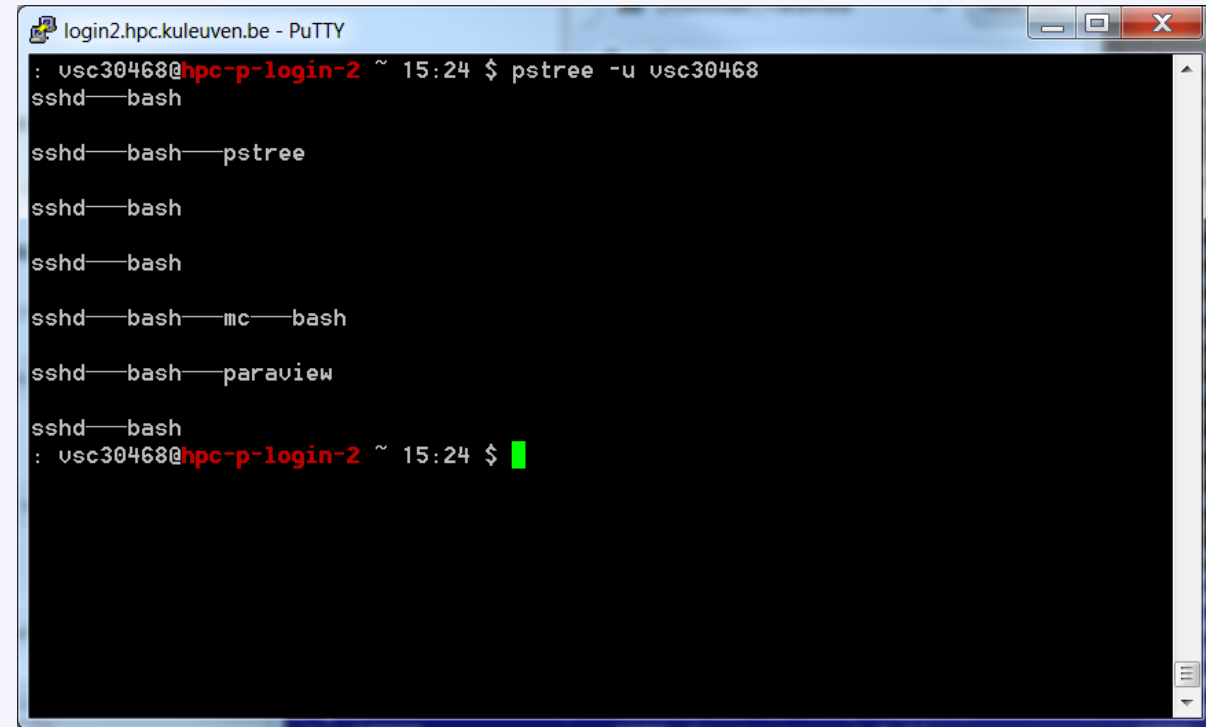


- Upon startup, shell executes commands found in the `~/.*rc` file, allowing users to customize their shell.

Source: opensuse.org

The Linux shell

- When logging into the cluster or starting a job – a new shell is opened
- Processes originating from each shell can be checked with `ps tree`
- `ps tree` shows only processes on the current node (login1 or login2 or NX or compute node)



```
login2.hpc.kuleuven.be - PuTTY
: usc30468@hpc-p-login-2 ~ 15:24 $ ps tree -u usc30468
sshd--bash
sshd--bash--pstree
sshd--bash
sshd--bash
sshd--bash--mc--bash
sshd--bash--paraview
sshd--bash
: usc30468@hpc-p-login-2 ~ 15:24 $ █
```


Key concepts

- Shell commands are **CASE SENSITIVE!**
- Upon executing, every command returns an integer to its parent called a return value.
- The shell variable “\$?” expands to the return value of previously executed command (e.g. 0 when success).



```
login2.hpc.kuleuven.be - PuTTY
: usc30468@hpc-p-login-2 ~ 15:32 $ pwd
/user/leuven/304/usc30468
: usc30468@hpc-p-login-2 ~ 15:32 $ echo $?
0
: usc30468@hpc-p-login-2 ~ 15:32 $ pwwd
-bash: pwwd: command not found
: usc30468@hpc-p-login-2 ~ 15:32 $ echo $?
127
: usc30468@hpc-p-login-2 ~ 15:32 $
```

Popular shells

- There are several types of shells for Linux.
- Check which one are you in with
\$ echo \$SHELL

Shell	Prompt	Name	Note
sh	\$	Bourne Shell	Default on some Unix systems
bash	\$	Bourne Again Shell	Enhanced replacement for the Bourne shell Default on most Linux and Mac OS X systems
csch	%	C Shell	Default on many BSD systems
tcsh	>	TC Shell	Enhanced replacement for the C shell
ksh	\$	Korn Shell	Default on AIX systems

Environment variables

- Shells let the user define ***variables***. They can be reused in shell commands. By convention, shell variables have lower case names.
- You can also define ***environment variables***: variables that are also visible within scripts or executables called from the shell. By convention, environment variables have UPPER CASE names.
- **env**
Lists all defined environment variables and their value.
- `export MY_VAR="/location-to/file"` (or some value)

Standard environment variables

Used by lots of applications!

LD_LIBRARY_PATH

Shared library search path

DISPLAY

Screen id to display X
(graphical) applications on.

EDITOR

Default editor (vi, emacs...)

HOME

Current user home directory

HOSTNAME

Name of the local machine

MANPATH

Manual page search path

PATH

Command search path

PRINTER

Default printer name

SHELL

Current shell name

TERM

Current terminal type

USER

Current user name

~/.bashrc (\$HOME/.bashrc)

~/.bashrc

Shell script read each time a bash shell is started

You can use this file to define

- Your default environment variables (PATH, EDITOR...).
- Your aliases.
- Your prompt (see the [bash](#) manual for details).
- A greeting message.

Find the right command

- Executing the right command can be vital for your system. However in Linux there are so many different command lines that they are often hard to remember. So how do you search for the right command you need? The answer is **apropos**:

- `apropos ownership`

<code>chgrp</code>	(1)	- change group ownership
<code>chgrp</code>	(1p)	- change the file group ownership
<code>chown</code>	(1p)	- change the file ownership
<code>chown</code>	(2)	- change ownership of a file
<code>fchownat</code>	(2)	- change ownership of a file relative to a directory file descriptor

Monitoring the system

- `free`
 - Displays the status of RAM and VRAM
 - Mem: refers to RAM
 - Swap: refers to virtual RAM (the swap file)
 - Too little RAM will cause 'thrashing' (constantly moving information from RAM to VRAM)
- `top`
 - Displays all the tasks, but also available CPU and memory in the top bar
- `/proc/cpuinfo` (`/proc/meminfo`)
 - contains info about processor/memory, no CPU usage

scp

The scp command allows you to copy files over ssh connections.

Copy a file locally

```
$ ssh vsc30468@login1-tier2.hpc.kuleuven.be  
$ touch test.txt  
$ scp $VSC_HOME/test.txt $VSC_DATA/
```

Or copy to a remote host

```
$ scp test.txt username@host-address.be:/some-location/
```

(local environment variables are not defined on the remote host)

Auto-completion

- Have the shell automatically complete commands or file paths.
- Activated using the **<TAB>** key on most systems
- examples
 - `$ whe<TAB>`
 - `$ whereis`
 - `$ ls -l /etc/en<TAB>`
 - `$ ls -l /etc/environment`
- When more than one match is found, the shell will display all matching results (use **<TAB>** twice)
 - `$ ls -l /etc/host<TAB>`

Quotes

- Double (") quotes can be used:
 - to prevent the shell from interpreting spaces as argument separators,
 - to prevent file name pattern expansion.

```
$echo "Hello World"
Hello World
$echo "You are logged as $USER"
You are logged as vsc30468
$echo *.log
$echo "*.log"
*.log
```

Quotes

- Single quotes bring a similar functionality, but what is between quotes is never substituted

```
$echo 'You are logged as $USER'  
You are logged as $USER
```

- Back quotes (`) can be used to call a command within another

```
$cd /lib/modules/`uname -r`; pwd  
/lib/modules/2.6.9-1.6_FC2
```

Back quotes can be used within double quotes

```
$echo "You are using Linux `uname -r`"  
You are using Linux 2.6.9-1.6_FC2
```

&

- `&` is a command line operator that instructs the shell to start the specified program in the background.
- This allows you to have more than one program running at the same time without having to start multiple terminal sessions.
- Starting a process in background: add `&` at the end of your line:
`gedit &`
check with `ps` if the processes is running

Command history: UP arrow and CTRL+R

- Previously executed commands can be recalled by using the **Up Arrow** key on the keyboard.
- Most Linux distributions remember the last five hundred commands by default.
- Display commands that have recently been executed
 - The `history` command displays a user's command line history.
 - You can execute a previous command using `! [NUM]` where NUM is the line number in history you want to recall.
 - The **history** command itself comes at the end of the list. From the command line, the **UP** and **DOWN** arrow keys will quickly traverse this list up and down, while the **LEFT** and **RIGHT** arrow keys will move the cursor to allow the user to edit a given command.
- The shell saves and keeps the command history (*.bash_history*)

Useful keyboard shell commands

- **ALT+B** – Move before the cursor.
- **ALT+F** – Move forward.
- **CTRL+A** – Quickly move to the beginning of line.
- **CTRL+E** – Move to the end of line.
- **CTRL+K** – Delete all characters after the cursor.
- **CTRL+W** – Delete the words before the cursor.
- **CTRL+Y** – Retrieves last item that you deleted or cut.
- **!!** – Repeats the last command.
- **CTRL+L** – Clears the screen and redisplay the line.
- **CTRL+Z** – Stops the current command.
- **CTRL+C** – Stop the currently running command.

Redirect output of a command/program

- Often we want to save output (stdout) from a program to a file. This can be done with the 'redirection' operator.
 - **myprogram** > *myfile* – using the '>' operator we redirect the output from **myprogram** to file *myfile*
- Similarly, we can append the output to a file instead of rewriting it with a double '>>'
 - **myprogram** >> *myfile* – using the '>' operator we append the output from **myprogram** to file *myfile*

Pipes

- Using a pipe operator '|' commands can be linked together. The pipe will link the standard output from one command to the standard input of another.
- Very helpful for searching files, e.g. when we want to list the files, but only the ones that contain test in their name:

```
ls -la | grep -R test9 $VSC_HOME/
```


More commands

- `ls`
- `mv`
move/rename files, e.g., `mv file1 some-directory/file2`
- `cat`
shows the contents of a file on the screen (non-editable)
- `less`
- `more`
- `head -n 10 filename.txt`
- `tail -n 10 filename.txt`
- `find $VSC_HOME -name "test9*"`

Even more commands...

- `alias`
`alias ll='ls -la'`
- `which`
`which grep`
- Multiple commands can be separated with a “;”, e.g.,
`cd $VSC_HOME; pwd`
- “&&” and “||” conditionally separate multiple commands. When commands are conditionally joined, the first will always execute. The second command may execute or not, depending on the return value of the first command. For example, a user may want to create a directory, and then move a new file into that directory. If the creation of the directory fails, then there is no reason to move the file. The two commands can be coupled as follows:
 - `$ echo „one two three four five” > numbers.txt;`
 - `$ mkdir /tmp/my-dir && mv numbers.txt /tmp/my-dir`
- `du -hs $VSC_HOME`

Info about users and processes

- `who`
Lists all the users logged on the system.
- `groups`
Tells which groups I belong to.
- `top`
Displays all processes (change with `<` or `>` for different parameters).
- `ps tree`
Displays process tree (`ps tree -u vsc30468`).
- `tree`
Lists contents of directories in a tree-like format (`tree $VSC_HOME`).

Questions

- Now
- Helpdesk:
 - hpcinfo@kuleuven.be
 - https://admin.kuleuven.be/icts/HPCCinfo_form/HPCC-info-formulier
- VSC web site: <http://www.vscentrum.be/>
 - VSC documentation
 - VSC agenda: training sessions, events
- Systems status page:
<http://status.kuleuven.be/hpc>



Thank you