



Introduction to HPC-UGent

Nov 10th 2017

<http://users.ugent.be/~kehoste/hpcugent-intro-20171110.pdf>

hpc@ugent.be

<http://ugent.be/hpc>

About this training – purpose

- Inform you of HPC-UGent services and infrastructure
- Learn what the benefit can be for your research
- Get you started on the central HPC infrastructure at UGent
 - Successfully connect to the HPC infrastructure
 - Successfully launch your first job
 - Figure out how to leverage it for *your* research
- Answer your questions

About this training – VSC manual

- A manual is available, applicable for all VSC infrastructure
- Download it here: <http://www.ugent.be/hpc/en/support/hpctutorial>
- *This is work in progress. If you find errors, do let us know.*
- We will specifically use information from these chapters:
 - 1/ Introduction to HPC
 - 2/ Getting an HPC account
 - 3/ Connecting to the HPC
 - 4/ Running batch jobs
 - 6/ Running jobs with input/output data
 - 8/ Fine-tuning job specifications

What is High Performance Computing?

“*High Performance Computing*” (HPC) is computing on a “*supercomputer*”, a system at the frontline of contemporary processing capacity – particularly in terms of size, supported degree of *parallelism*, network interconnect and (total) available memory & disk space.

A computer *cluster* consists of a set of loosely or tightly connected computers that work together so that in many respects they can be viewed as a single system.

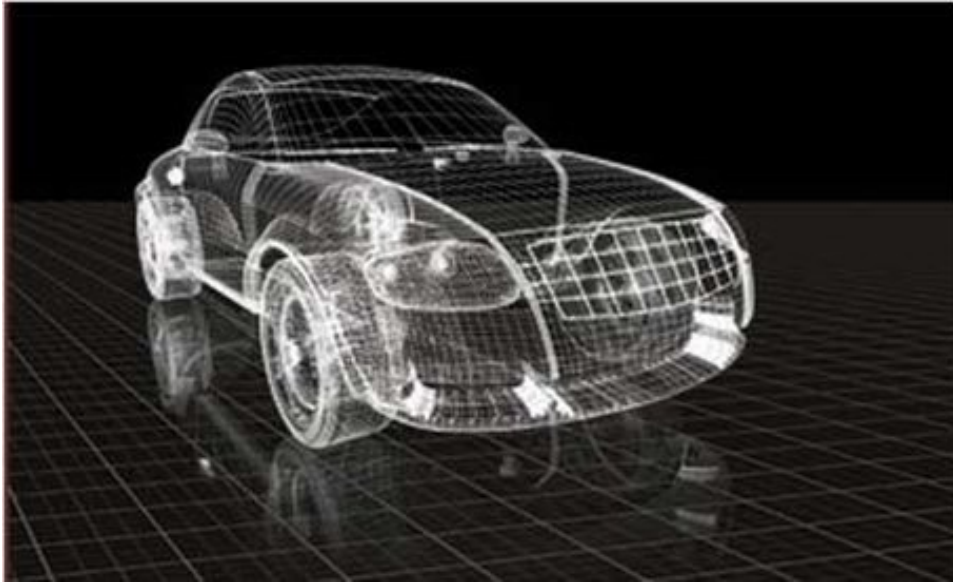
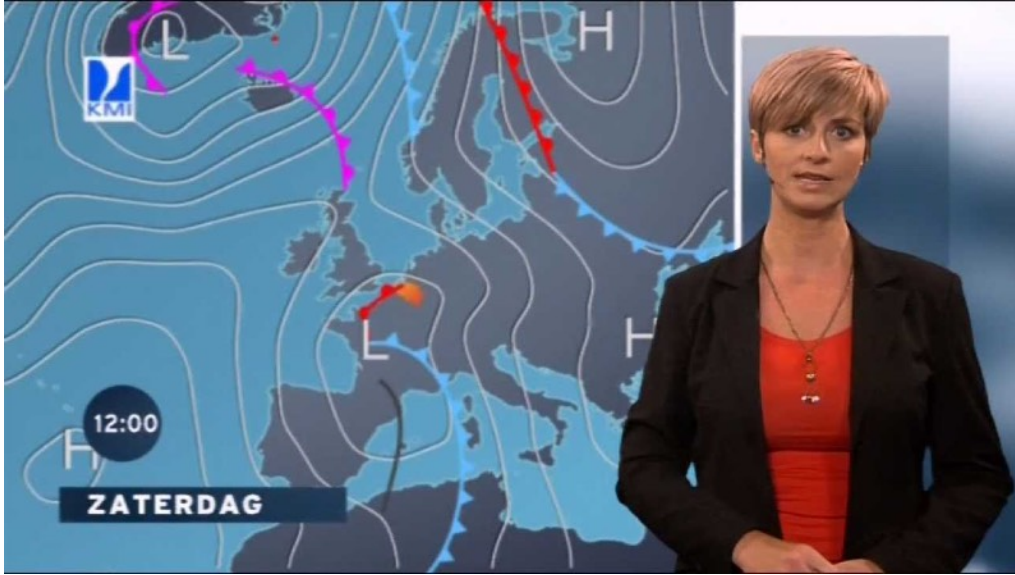
(a.k.a. “supercomputing”)

What is High Performance Computing?

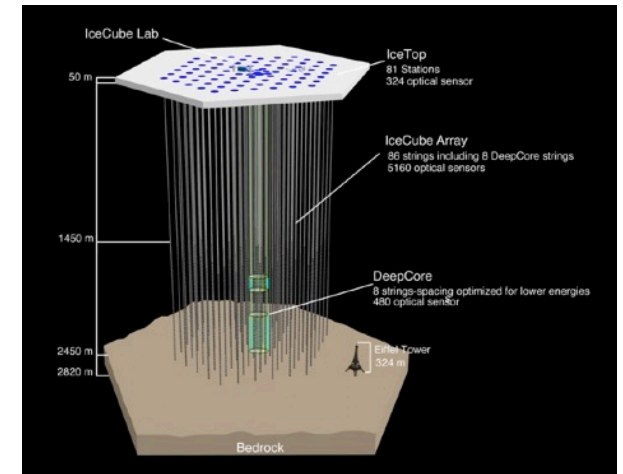
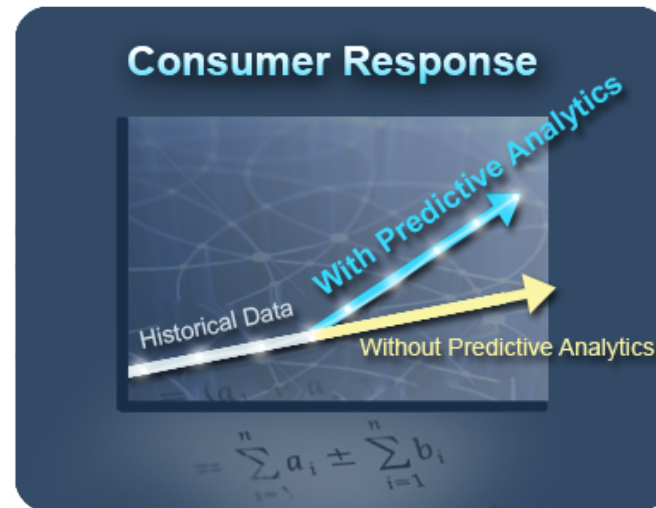
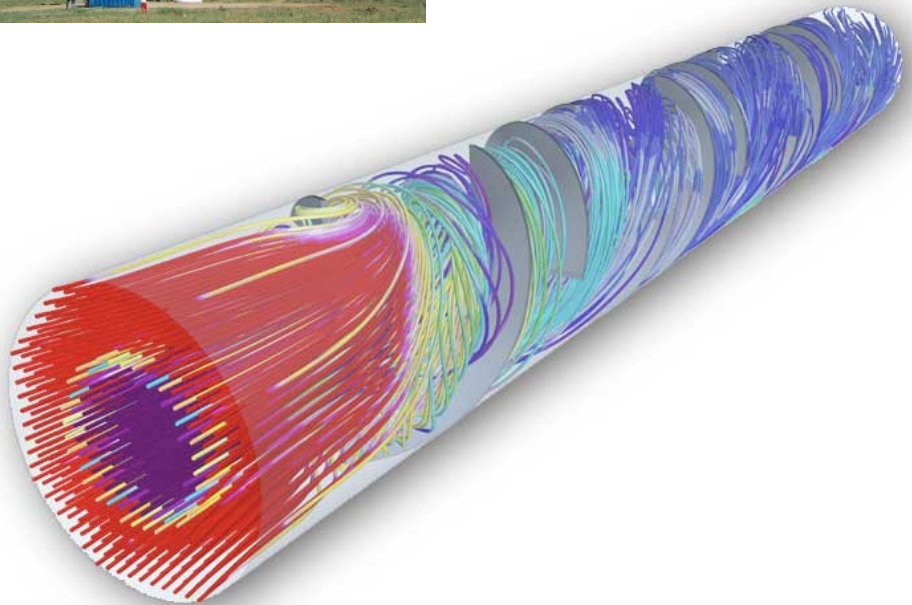
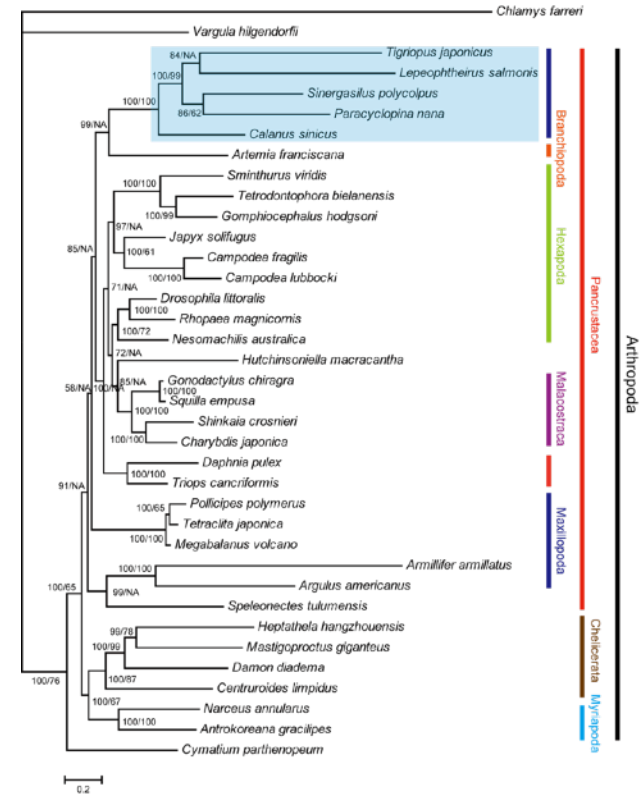
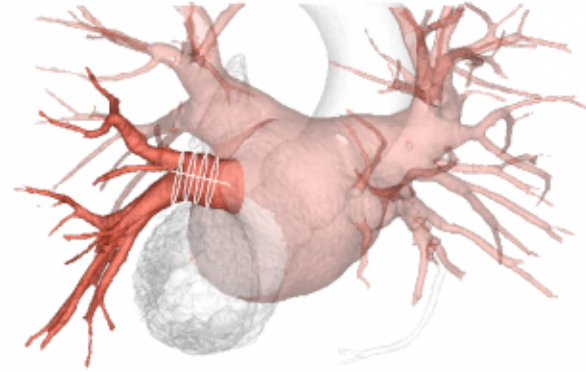
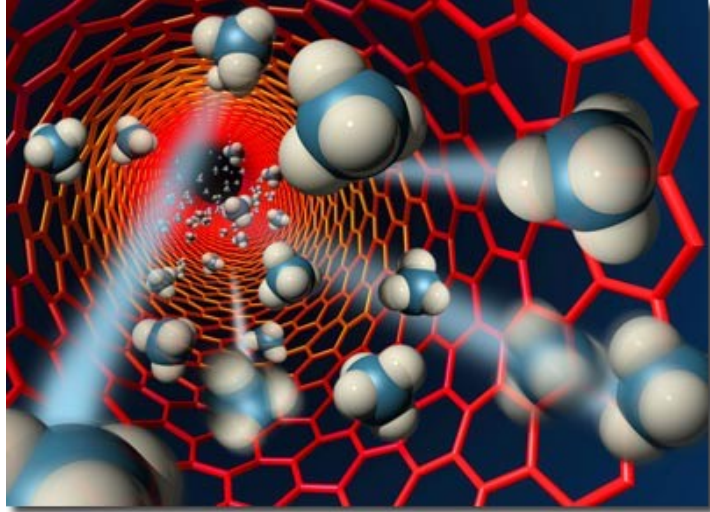
harness power of multiple interconnected cores/nodes/processing units



Everyday applications of supercomputing



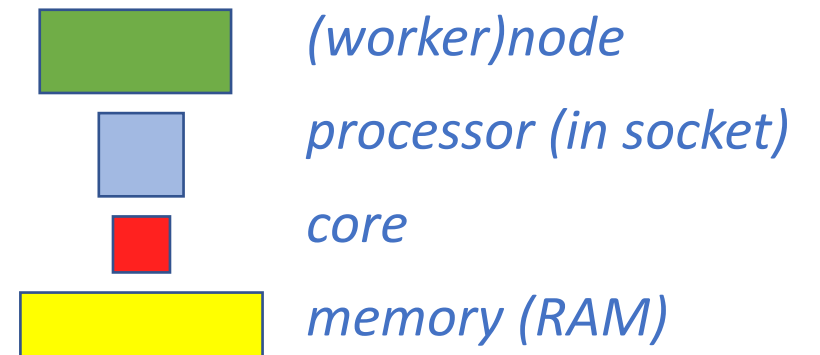
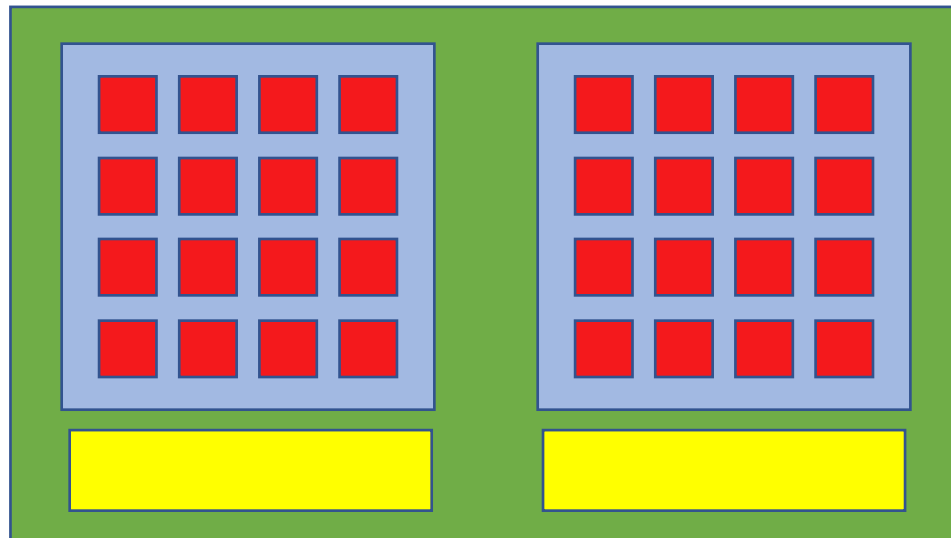
Scientific applications of supercomputing



Cores vs processors vs sockets vs nodes

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

*example:
node with
two 16-core
processors*



Parallel vs sequential software

In **parallel** software, *many* calculations are carried out *simultaneously*.

They are based on the principle that large problems can often be divided into smaller ones, which are then solved concurrently (“in parallel”).

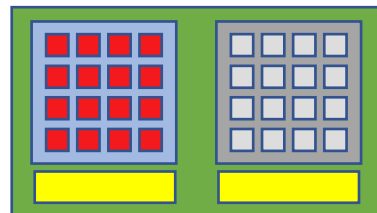
e.g., OpenFOAM can easily use 160 cores at the same time to solve a CFD problem

Parallel programming paradigms:

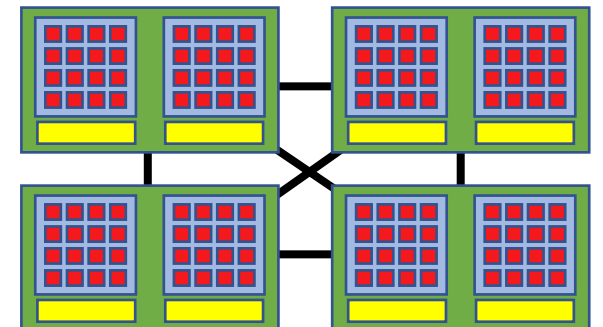
OpenMP for shared memory systems (*multithreading*) -> on cores of a *single* node

MPI for distributed memory systems (*multiprocessing*) -> on *multiple* nodes

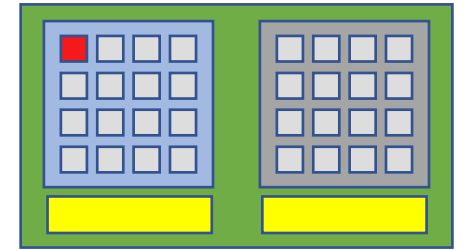
*OpenMP software
can use multiple or
all cores in a node*



*MPI software
can use (all) cores
in multiple nodes*



Parallel vs sequential programs

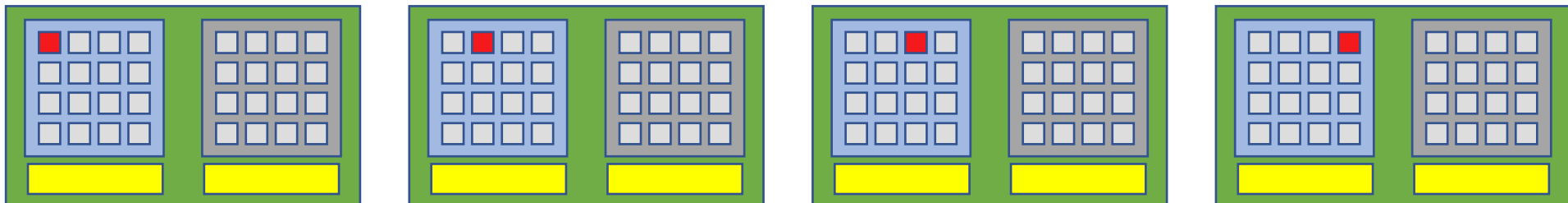


Sequential (a.k.a. serial) software does not do calculations in parallel, i.e. it only uses *one single core* of a single workernode.

(Sequential) software does not become faster by just throwing cores at it...

But, you can run *multiple instances* at the same time on a supercomputer.

e.g., you can easily run a Python script 1000 times at once to quickly analyse 1000 datasets



HPC-UGent

hpc@ugent.be

Part of ICT Department of Ghent University

Our mission

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

Our core values

Empowerment - Centralisation - Automation - Collaboration

HPC-UGent: staff



Stijn De Weirdt
technical lead



Ewald Pauwels
team lead



Kenneth Hoste
user support & training



Wouter Depypere
sysadmin, hardware



Jens Timmerman
sysadmin, security



Kenneth Waegeman
sysadmin, storage



Andy Georges
sysadmin, tools & testing

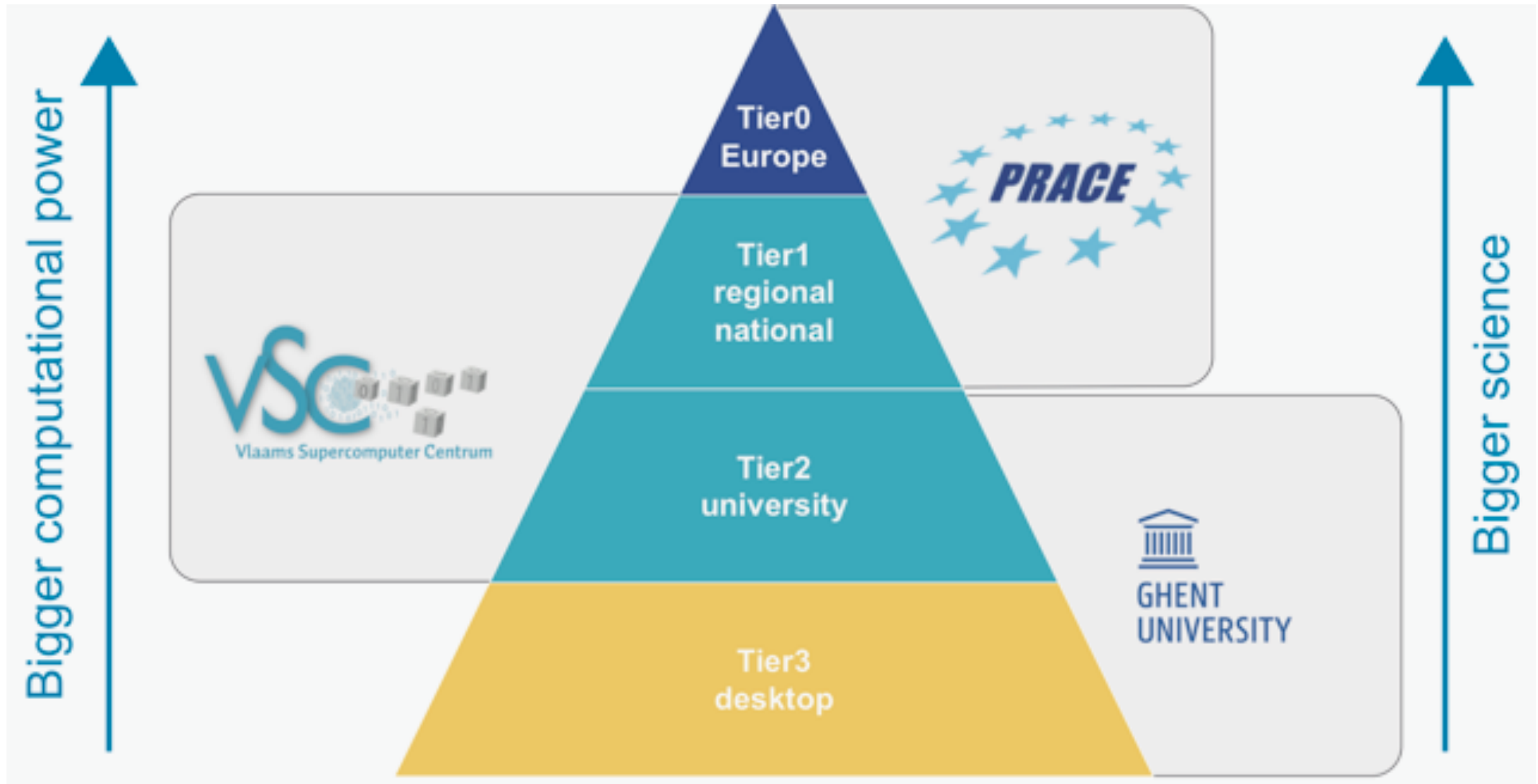


Alvaro Simon Garcia
cloud, user support

Centralised hardware



Centralised hardware



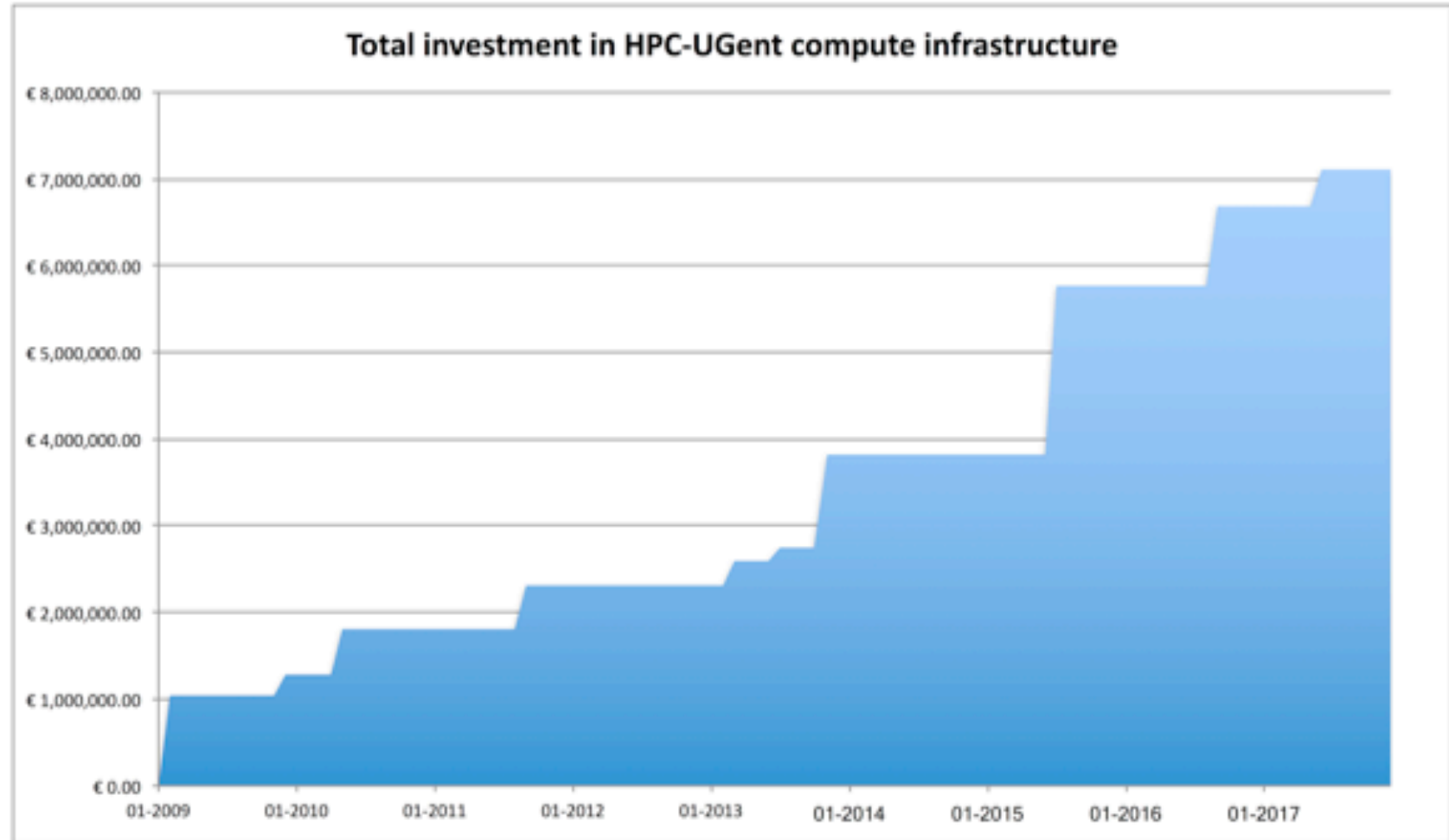
HPC-UGent Tier2 (STEVIN): central investments



1548 - 1620

°Bruges

STEVIN
HPC
infrastructure




HPC-UGent Tier2 (STEVIN)

<https://www.vscentrum.be/infrastructure/hardware/hardware-ugent>

Compute clusters

5 Tier2 clusters; in total 500+ workernodes, 11k+ cores

	#nodes	CPU	Mem/node	Diskspace/node	Network	
	Raichu	64	2 x 8-core Intel E5-2670 (Sandy Bridge @ 2.6 GHz)	32 GB	400 GB	GbE
	Delcatty	160	2 x 8-core Intel E5-2670 (Sandy Bridge @ 2.6 GHz)	64 GB	400 GB	FDR InfiniBand
	Phanpy	16	2 x 12-core Intel E5-2680v3 (Haswell-EP @ 2.5 GHz)	512 GB	3x 400 GB (SSD, striped)	FDR InfiniBand
	Golett	200	2 x 12-core Intel E5-2680v3 (Haswell-EP @ 2.5 GHz)	64 GB	500 GB	FDR-10 InfiniBand
	Swalot	128	2 x 10-core Intel E5-2660v3 (Haswell-EP @ 2.6 GHz)	128 GB	1 TB	FDR InfiniBand

HPC-UGent Tier2 (STEVIN)

Network connections between nodes



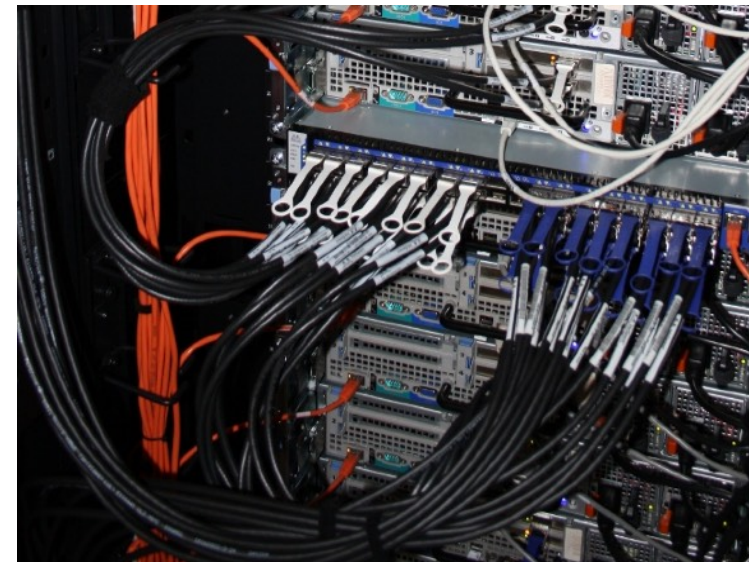
Ethernet: 1 - 10 Gbit/s



€

for single core/node jobs

Infiniband: 24 - 54 Gbit/s



€€(€)

required for MPI jobs



VSC Tier2

Vlaams Supercomputer Centrum
(Flemish Supercomputer Center)



<https://www.vscentrum.be/en/access-and-infrastructure/tier-2>

(GPGPU systems @ KUL: <http://hpc.ugent.be/userwiki/index.php/Tips:Software:GPGPU>)

Antwerp University association

Brussels University association
+ Grid specialization

Ghent University association
+ Big Data specialization

KU Leuven association
Limburg association University-Colleges
+ Shared memory, accelerator specialization



VSC Tier1 – muk (@ HPC-UGent)

For up to date information, see:

<https://www.vscentrum.be/en/access-and-infrastructure/tier-1>



Hardware

retired on Jan 1st 2017

- 528 computing nodes
 - Two 8-core Intel Xeon processors (Sandy Bridge, E5-2670, 2.6 GHz)
 - 64 GiB RAM
- FDR InfiniBand interconnect with a fat tree topology
 - High bandwidth (6.5 GB/s per direction, per link)
 - Low latency
- Storage system
 - Capacity of 400 TB
 - Peak bandwidth of 9.5 GB/s



VSC Tier1 – BrENIAC (@ KU Leuven)

For up to date information, see:

<https://www.vscentrum.be/en/access-and-infrastructure/tier-1>



Hardware

- 580 computing nodes (16,240 cores in total)
 - Two 14-core Intel Xeon processors (Broadwell, E5-2680v4)
 - 128 GiB RAM (435 nodes) or 256 GiB (145 nodes)
- EDR InfiniBand interconnect
 - High bandwidth (11.75 GB/s per direction, per link)
 - Slightly improved latency over FDR
- Storage system
 - Capacity of 634 TB
 - Peak bandwidth of 20 GB/s

VSC Tier1

For academics (all Flemish research centers):

- *Free of charge*
- Starting Grant (100 node days)
 - <https://www.vscentrum.be/en/access-and-infrastructure/tier1-starting-grant>
 - Fill in application form, send it to hpc@ugent.be
- Project access (500-5000 nodedays)
 - 3 evaluation moments per year
 - Application form and more info
 - <https://www.vscentrum.be/en/access-and-infrastructure/project-access-tier1>
- Don't hesitate to contact hpc@ugent.be for help!



VSC Tier1

For industry:

- Exploratory access (100 node days)
 - *Free of charge*
 - Contact hpc@ugent.be
- Contract access
 - FWO/UGent/company contract
 - Payed usage (~13 euro / *node* / day)
 - Contact hpc@ugent.be



Getting a VSC account

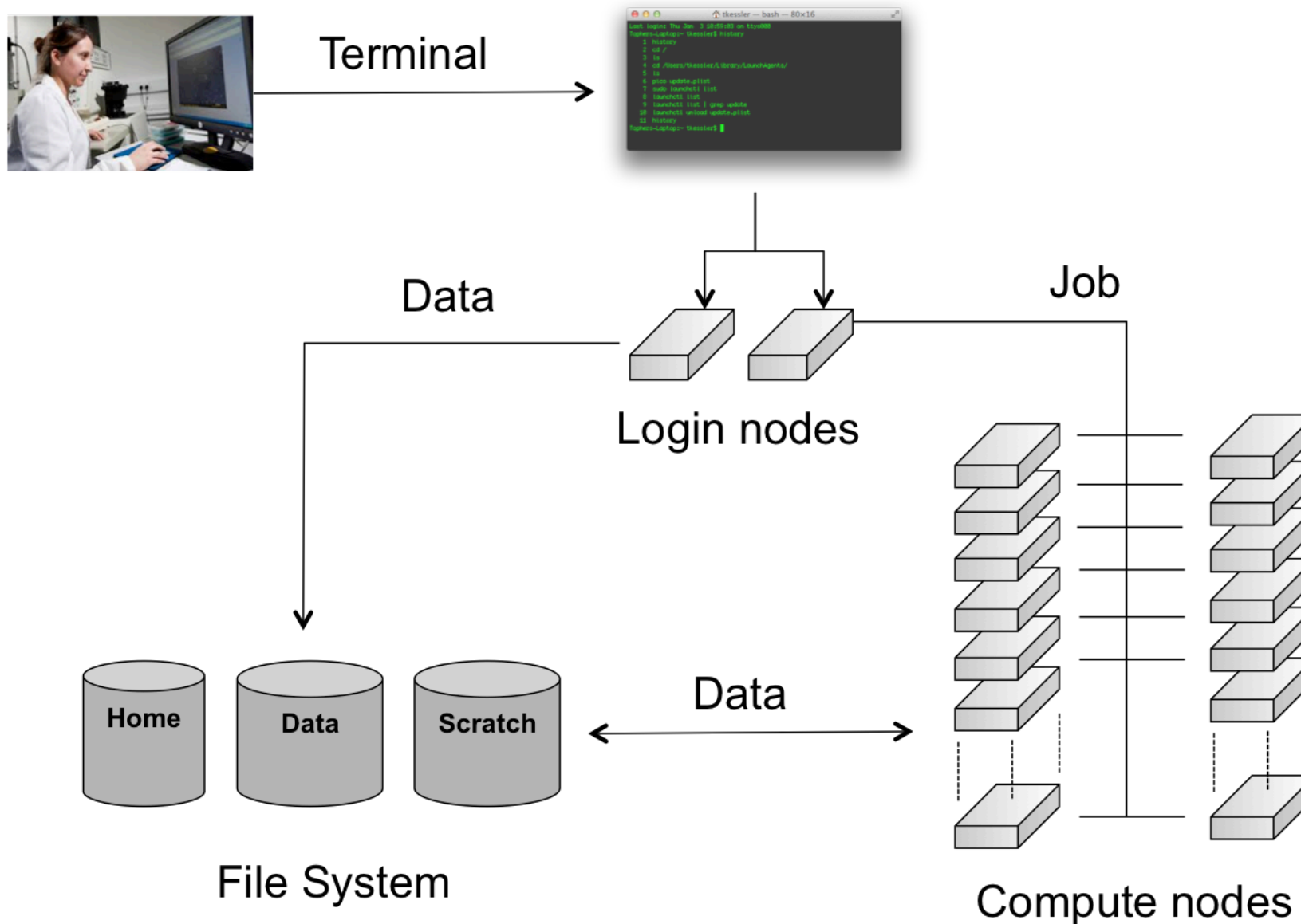


- See Chapter 2 in HPC-UGent intro course notes
- <https://www.vscenrum.be/en/access-and-infrastructure/requesting-access>
- All users of AUGent can request an account
 - Researchers
 - Master/Bachelor students (after motivation of ZAP)
 - Staff
- Subscribed to hpc-announce and hpc-users mailing lists
- Beware of using HPC for teaching/exam purposes!
 - No guarantee on HPC availability (power outage/maintenance)
 - Have a backup plan at hand
 - Advisable teaching/exam formula: project work

Workflow on HPC infrastructure

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

High-level overview of HPC-UGent infrastructure



Workflow on HPC infrastructure

- 1. Connect to login nodes**
- 2. Transfer your files**
3. (Compile your code and test it)

See Chapter 3 in course notes

- Users interact with the infrastructure via the login nodes
- No direct access to the workernodes
- Except when a job is running on it
 - Your job gets executed
 - Your job finishes

- 7. Move your results**

Workflow on HPC infrastructure

1. Connect to login nodes
2. Transfer your files
3. (Compile your code and test it)
- 4. Create a job script**
5. Submit your job

- Choose correct PBS directives (Chapter 4, 8)
- Load software modules (Chapter 3)
- Useful environment variables (Chapter 4)
- Select correct data volume (Chapter 6)

7. Move your results

Job scripts: PBS directives

```
#!/bin/bash
#PBS -N solving_42          ## job name
#PBS -l nodes=1:ppn=all    ## single-node job, all available cores
#PBS -l walltime=10:00:00  ## max. 10h of wall time
#PBS -l vmem=50gb          ## max. 50GB virtual memory
<rest of job script>
```

- required resources can be specified via `#PBS` lines in job script (or via `qsub`)
- **maximum walltime: 72 hours**
- for longer jobs, use *checkpointing*
 - preferable internal/application checkpointing
 - external checkpointing
 - see <http://hpc.ugent.be/userwiki/index.php/User:Checkpointing>

Job scripts: software modules

- All user-end software is made available via *modules*
- Modules prepare the environment for using the software
- Module naming scheme: `<name>/<version>-<toolchain>[-<suffix>]`

Load a module to use the software:

```
$ module load Python/2.7.12-intel-2016b or $ ml Python/...
```

See currently loaded modules using:

```
$ module list or $ ml
```

Get overview of available modules using:

```
$ module avail or $ ml av
```

- Only mix modules built with the same compiler toolchain.
e.g., `intel` (Intel compilers, Intel MPI, Intel MKL (BLAS, LAPACK))
- See also <https://www.vscentrum.be/cluster-doc/software/modules/lmod>

Job scripts: useful environment variables

- **\$PBS_O_WORKDIR**
 - directory from which job was submitted on login node
 - common to use 'cd \$PBS_O_WORKDIR' at beginning of job script
- **\$PBS_JOBID**
 - job id of running job
- **\$PBS_ARRAYID**
 - array id of running job
 - only relevant when submitting array jobs (qsub -t)
- **\$TMPDIR**
 - Local directory specific to running job
 - **Cleaned up automatically when job is done!**
- **\$EBROOTFOO, \$EBVERSIONFOO**
 - root directory/version for software package Foo
 - only available when module is loaded

Job scripts: input data & filesystems

- See Section 6.2 in course notes
- Think about I/O:
 - How will you *stage in* your data and input files?
 - How will you *stage out* your output files?
- Manually (on login nodes) vs automatically (as a part of job script)
- **Home filesystem:** only for limited number of small files & scripts
- **Data filesystem (\$VSC_DATA*):** 'long-term' storage, large files
- **Scratch filesystems (\$VSC_SCRATCH*):** for 'live' input/output data in jobs

Storage quota

- home directory (`$VSC_HOME`): 3GB (fixed)
- personal data directory (`$VSC_DATA`): 25GB (fixed)
- personal scratch directory (`$VSC_SCRATCH`): 25GB (fixed)
- current quota usage can be consulted on VSC accountpage
<https://account.vscentrum.be>
- **more storage quota (GBs, TBs) available for members of virtual organisations (VOs)**
- see <http://hpc.ugent.be/userwiki/index.php/User:VSCVos>
- additional quota can be requested via <https://account.vscentrum.be/django/vo/edit>
- shared with VO: `$VSC_DATA_VO`, `$VSC_SCRATCH_VO`
- personal VO subdirectories: `$VSC_DATA_VO_USER`, `$VSC_SCRATCH_VO_USER`

Job scripts: full example (single-core job)

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

module load Python/2.7.13-intel-2017a
# copy input data from location where job was submitted from
cp $PBS_O_WORKDIR/input.txt $TMPDIR
# go to temporary working directory (on local disk) & run
cd $TMPDIR
python -c "print(len(open('input.txt').read()))" > output.txt
# copy back output data, ensure unique filename using $PBS_JOBID
cp output.txt $VSC_DATA/output_${PBS_JOBID}.txt
```

Job scripts: full example (multi-node job)

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

module load intel/2016b
module load vsc-mypirun

# go to working directory, compile and run MPI hello world
cd $PBS_O_WORKDIR
mpicc mpi_hello.c -o mpi_hello
mypirun ./mpi_hello
```

Workflow on HPC infrastructure

1. Connect to login nodes
 2. Check the status of the system
 3. Check the status of your job
 4. Create a job script
 5. Submit your job
 6. Be patient
 - Your job gets into the queue
 - Your job gets executed
 - Your job finishes
 7. Move your results
- Chapter 4 in course notes
 - Demo: qsub, qstat, qdel
 - Job scheduling

Demo: qsub, qstat, qdel

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

```
$ module swap cluster/golett
$ qsub example.sh
12345.master19.golett.gent.vsc
```

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

```
$ qstat
Job id          Name          User          Time Use      S  Queue
-----
12345.master19  example      vsc40000      07:39:30     R  long
```

- To remove a job that is no longer necessary, use **qdel**:

```
$ qdel 12345
```

Job scheduling

- All our clusters use a *fair-share* scheduling policy.
- No guarantees on when job will start, so **plan ahead!**
- Job priority is determined by:
 - *historical usage*
 - aim is to balance usage over users
 - infrequent/frequent users => higher/lower priority
 - *requested resources* (# nodes/cores, walltime, memory, ...)
 - large resource request => lower priority
 - *time waiting in queue*
 - queued jobs get higher priority over time
 - *user limits*
 - avoid that a single user fills up an entire cluster

Embarrassingly parallel jobs

- Use case: lots of ((very) short) single-core tasks
- Submitting lots of tiny jobs (minutes of walltime) is not a good idea
 - overhead for each jobs (node health checks), lots of bookkeeping (job scripts, failed jobs, output files)
- Better approach:
 - Array jobs (http://hpc.ugent.be/userwiki/index.php/User:VscScripts#Array_Example)
 - Single job script, but still lots of submitted jobs
 - Each job is assigned a unique id (\$PBS_ARRAYID); can be used to select input file, parameters, ...
 - GNU parallel (https://www.gnu.org/software/parallel/parallel_tutorial.html)
 - General-purpose tool to easily running shell commands in parallel with different inputs
 - Use 'parallel' command in your job script
 - **Worker (<https://www.vscentrum.be/cluster-doc/running-jobs/worker-framework>)**
 - One single job that processes a bunch of tasks (multi-core or even multi-node)
 - Job script is parameterized, submit with 'wsub' rather than 'qsub'

Software installations

Request for new software installations: hpc@ugent.be

Always include:

- software name and website
- location to download source files
 - or make install files available in your account
- build instructions (if you have them)
- a simple test case with expected output
 - including instructions on how to run it

Requests may take a while to process; make the request sooner rather than later!

Documentation & training

- Documentation is available at:
 - <https://www.vscentrum.be/en/user-portal>
 - (<http://hpc.ugent.be/userwiki>, being phased out)
- HPC tutorial: <http://www.ugent.be/hpc/en/support/hpctutorial>
- Basic Linux: http://hpc.ugent.be/userwiki/index.php/Tips:Introduction_to_Linux
- **Training sessions** - <https://www.vscentrum.be/en/education-and-trainings>
 - next training sessions being planned in Spring 2018
 - Introduction to OpenMP, Introduction to MPI, Linux hands-on introduction, ...

Getting help

Contact HPC-UGent support: hpc@ugent.be

Always include:

- clear description of problem (or question)
- location of job script and output/error files in your account
 - don't send them in attachment, we prefer to look at it 'in context'
- job IDs, which cluster
- VSC login id

Preferably use your UGent email address

Alternatives:

- short meeting (for complex problems, big projects)
- hpc-users mailing list



Introduction to HPC-UGent

Nov 10th 2017

<http://users.ugent.be/~kehoste/hpcugent-intro-20171110.pdf>

hpc@ugent.be

<http://ugent.be/hpc>