Using the general purpose computing clusters at EPFL

`scitas.epfl.ch`

October 8, 2019

Welcome

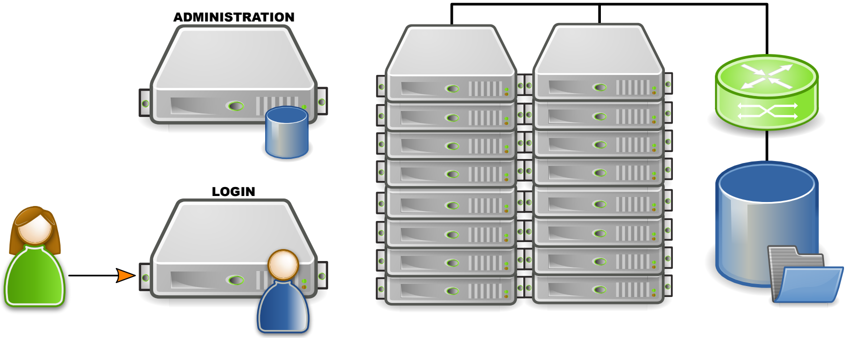
What we will look into

- What is a cluster
- What is a scheduler
- How the environment is organised
- How to run simple jobs on our clusters

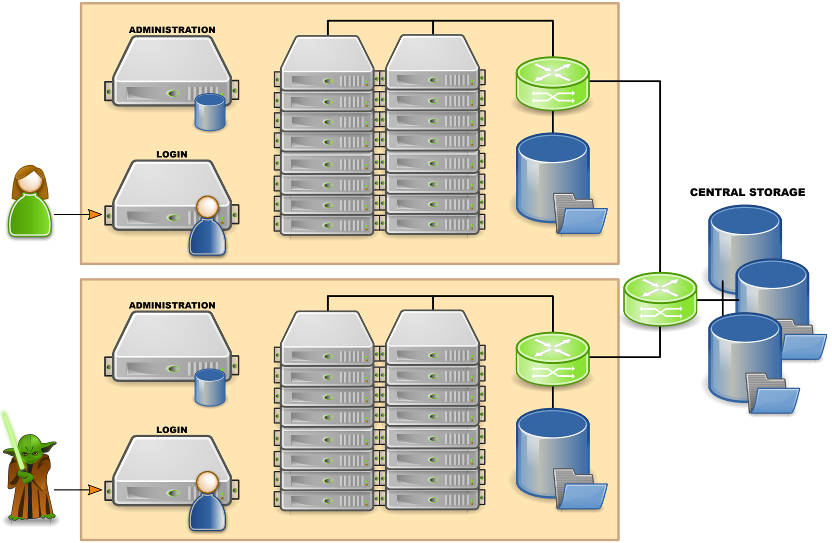
What is not covered

- Running parallel/MPI jobs
- Writing and (compiling) code
- Parallelising code

What is a cluster?



What is a cluster?



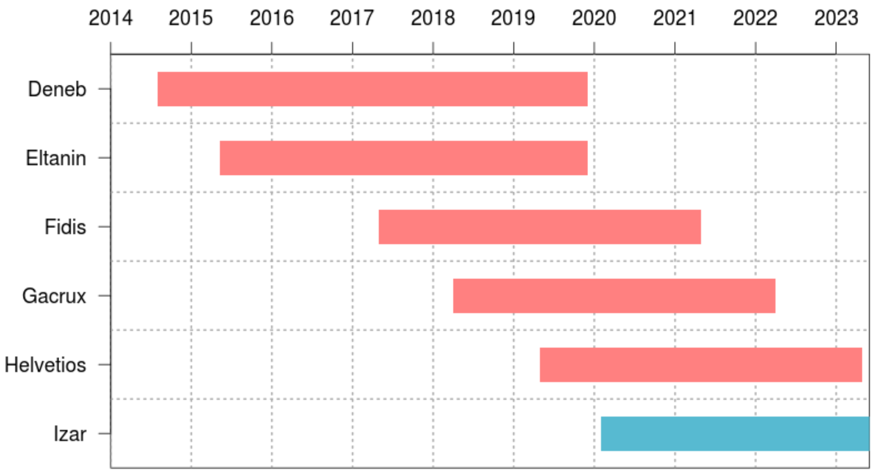
What is a cluster?

Distinctive characteristics

Configuration optimized for performance:

- CPU and Memory: optimal configuration
- Network: low latency and high bandwidth
- Storage: high performance parallel
- Accelerators: GPUs

Our clusters



What is each cluster optimised for?

Multi-node (or 'parallel') workloads

- The whole node is allocated to a single user, nodes are never shared amongst users.
- **parallel** partition, clusters: `deneb`, `fidis`, `helvetios`

Single-node (or 'serial') workloads

- Only the resources requested are allocated (cores and memory), jobs from other users can run on the remaining node resources.
- **serial** partition, clusters: `deneb`

Shared Storage (cluster)

`/scratch`

- high performance temporary space
- not backed up
- low redundancy, built for performance
- local to each cluster
- automatic cleanup procedure deletes files without warning (older than two weeks, when occupancy reaches a threshold)
- **for disposable files: intermediary results, temporary files**

Shared Storage (global)

`/home`

- per user quotas of 100GB
- *off-site* backup (to another building on campus)
- available on all clusters
- **for important files: source code, final results, theses**

Shared Storage (global)

`/work`

- per group quotas
- 50GB for free
  then 300CHF/TB for 3 years ($\times 2$ for *off-site* backup)
- available on all clusters
- **for shared group files: software, datasets**

Connecting to a cluster [Hands-on!]

Connect using SSH

```
ssh username@fidis.epfl.ch
```

- Linux: simply connect using `ssh`
- Windows: install git-bash, connect using `ssh` from git-bash
- OSX: connect using `ssh`

Basic shell commands, moving around

- `id`
- `pwd`
- `ls /scratch/<username>`
- `cd /scratch/<username>`

Batch versus Interactive use

Interactive

You are in front of your computer, just open your application and start working.

Batch

You describe the work to be done (in a script) and put it in a queue. Your jobs will be run when matching resources become available.

Batch system and Scheduler

Batch system

Keeps jobs in a queue until the *scheduler* decides they should be started, and then prepares the nodes and starts the jobs.

Scheduler

Decide when and where your job will run depending on the requested resources and your priority. Its objective is to maximise the amount of work done with the resources available.

We will sometimes use both terms interchangeably.

SLURM

Slurm commands: `sbatch`

`sbatch`

This is the fundamental command used to submit jobs to the batch system. Normally returns immediately, as all it does is read the script and check your requirements.

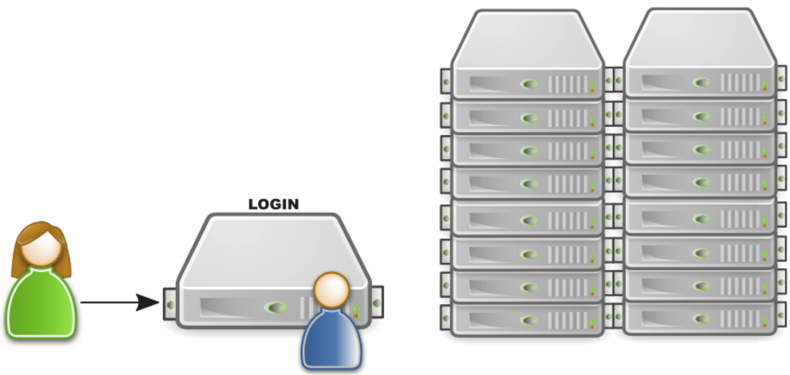
Workflow

A typical workflow looks like this:

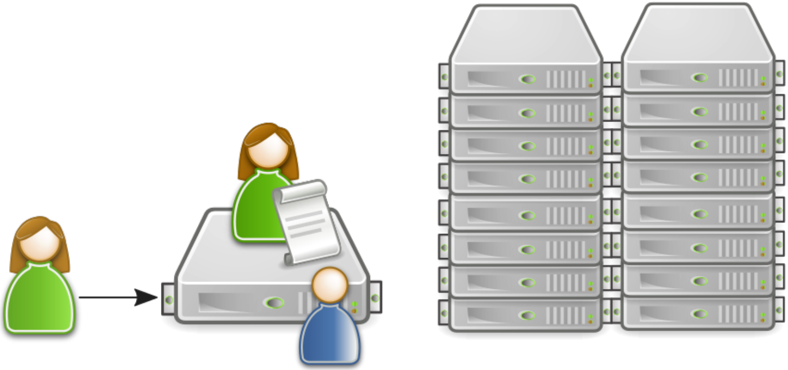
- create a job-script
- submit it to the batch system (with `sbatch`)
- *it will get executed*
- look at the output

The job **will wait in the queue** until resources are available to run it.

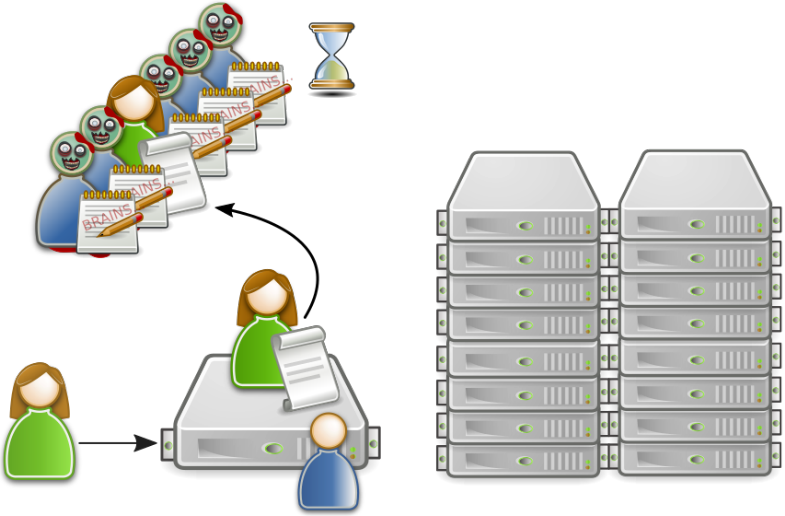
Workflow of a job



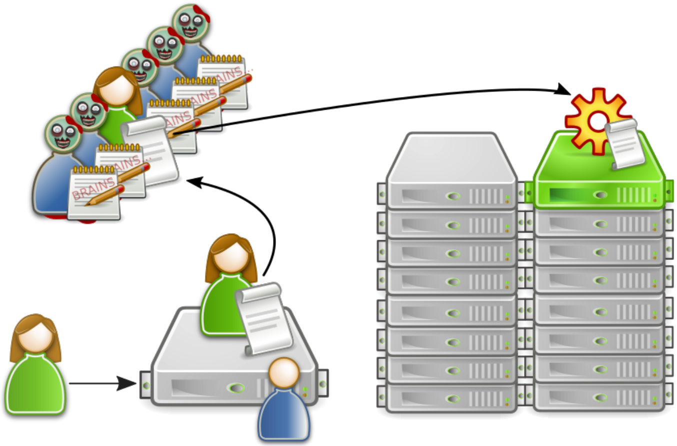
Workflow of a job



Workflow of a job



Workflow of a job



Exercise 1: sbatch [Hands-on!]

Copy the examples to your working directory

```
cp -r /scratch/examples/using-the-clusters .
```

Open and edit the first exercise

Open the file ex1.sh with your editor of choice:

- `nano`
- `emacs`
- `vim`
- ...

Exercise 1: sbatch [Hands-on!]

ex1.sh

```bash
#!/bin/bash
#SBATCH --chdir /scratch/<put-your-username-here>
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 1
#SBATCH --mem 1G
#SBATCH --account scitas-courses
#SBATCH --reservation scitas-courses

sleep 10
echo "hello from $(hostname)"
sleep 10
```

SBATCH directives

`#SBATCH --something`

This is how you tell Slurm what resources the job needs.

SBATCH: nodes

The number of nodes per job

Examples:

```
--nodes 1
--nodes 64
```

If not specified then the default is 1.

SBATCH: ntasks

The number of MPI tasks per job

Examples:

```
    --ntasks 1
    --ntasks 256
```

If not specified then the default is 1.

SBATCH: cpus-per-task

The number of CPUs per task for multithreaded applications

Examples:

```
--cpus-per-task 1
```

```
--cpus-per-task 28
```

If not specified then the default is 1.
Cannot be more than the number of cores/cpus in a compute node!

SBATCH: mem

The required memory per node

Examples:

```
--mem 4096M
--mem 120G
```

If not specified then the default is 4096MB per CPU.

Beware of edge cases!

For example, on `fidis` if you ask for 128G, you are targetting 192G or 256G nodes, as our *128G* nodes only have 125G.

SBATCH: time

How long will your job run for?

Examples:

```
--time 06:00:00

--time 2-23
```

If not specified then the default is 15 minutes.

SBATCH: partition

Which partition should my job be sent to?

Examples:

```
--partition debug
--partition serial
```

If not specified then the default is parallel on Fidis/Helvetios, and for Deneb it depends on what resources you have requested!

Exercise 1: submit ex1.sh to the batch system [Hands-on!]

Let's submit our job

```
$  sbatch ex1.sh
Submitted batch job 12345678

$ cat /scratch/<username>/slurm-12345678.out
hello from f103
```

Remember the Job ID

The number returned by `sbatch` is known as the **Job ID** and is the unique identifier for a task. If you ever need to ask for help you'll need to know this number.

Cancelling jobs

scancel

To cancel a specific job:

```
scancel <jobid>
```

To cancel all your jobs:

```
scancel -u <username>
```

To cancel all your jobs that are not yet running:

```
scancel -u <username> -t PENDING
```

Exercise 2: squeue [Hands-on!]

120GB of memory required

```
#!/bin/bash
#SBATCH --chdir /scratch/<username>
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 28
#SBATCH --mem 120G
#SBATCH --time 00:30:00
#SBATCH --account scitas-courses
#SBATCH --reservation scitas-courses


cd /scratch/examples/linpack/
./runme_xeon64
```

Exercise 2: what's going on? [Hands-on!]

squeue

With no arguments squeue will list all jobs currently in the queue! The output and information shown can be refined somewhat by giving options.

- `squeue -t R -u username`
- `squeue -t PD -u username`
- `squeue -t PD -u username --start`

Squeue

Squeue is an custom squeue that shows only your jobs with more useful information.

Cancelling jobs [Hands-on!]

`scancel`

- Try and cancel your pending jobs.
- Try and cancel the jobs of the person to your left.

Software

What's the problem?

- The OS version is restricted to an older one due to compatibility requirements of storage systems and specialized interconnects.
- The above is often in direct conflict with the needs of our user community, for which newer versions bring performance improvements and support for newer hardware (new CPU features).
- Many scientific codes are not even packaged under most Linux distributions.

Modules to the rescue

modules

`(GNU) modules` is a utility that allows multiple, often incompatible, tools and libraries to co-exist on a system. It's a widely used tool for organising software on HPC clusters but each site uses it in subtly different ways.

Modules

How are software packages organised?

- packages are organized hierarchically: `Compiler / MPI / blas`
- packages are hidden until one loads the dependencies
- **modules** keeps the environment consistent
- **modules** automatically reloads a package when dependencies change

Exercise 3: modules [Hands-on!]

Basic commands

- `module av(ailable)`
- `module load / unload <module-name>`
- `module spider <name>`
- `module purge`

Exercise 3: modules [Hands-on!]

Example: loading python

- `module load intel`
- `module load python`
- `module list`
- `module load python/2.7.14`
- `module load gcc`
- `module list`

Exercise 3: modules [Hands-on!]

ex3.sh: using module files

```
#!/bin/bash -l
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 4
#SBATCH --nodes 1
#SBATCH --mem 16G
#SBATCH --time 00:15:00
#SBATCH --account scitas-courses
#SBATCH --reservation scitas-courses

echo STARTING AT $(date)

module purge
module load matlab

matlab -nodesktop -nojvm -r mymfile

echo FINISHED AT $(date)
```

Interactive access

Why interactive?

For debugging or running applications such as Matlab interactively we don't want to submit a batch job.

Sinteract or salloc

There are two main ways of getting access depending on what you want to achieve:

- `salloc` - standard tool for an interactive allocation for multi-node jobs
- `Sinteract` - custom tool to access a node

Behind the scenes both use the same mechanism as `sbatch` to get access to resources.

Sinteract [Hands-on!]

Sinteract

```
[<user>@fidis ~]$ Sinteract -p debug
Cores:            1
Tasks:            1
Time:             00:30:00
Memory:           4G
Partition:        debug
Account:          scitas-courses
Jobname:          interact
Resource:
QOS:              scitas
Reservation:

salloc: Granted job allocation 12345678
salloc: Waiting for resource configuration
salloc: Nodes f106 are ready for job
[<user>@f106 ~]$
```

The S tools

Wrappers around Slurm commands to make your life (slightly) easier:

- `Sinteract` - get interactive access to a node
- `Sshare` - show fairshare information
- `Squeue` - show user's pending and running jobs
- `Sjob` - show information about a job

Sinteract: storage [Hands-on!]

Storage locations

The different storage locations are accessible via environment variables.

- $TMPDIR - a **temporary** folder in a **local** filesystem (generally /tmp)
- $SCRATCH - your scratch folder at /scratch/username

Try it: Sinteract -p debug

```
[<user>@f107 ~]$ echo $TMPDIR
/tmp/123456789
```

Note: these are only set within a job!

The debug partition

`--partition=debug`

All the clusters have a few nodes that only allow short jobs and are intended to give you quick access to debug jobs:

- `#SBATCH -p debug`
- `Sinteract -p debug`

The number of nodes in the partition and the limits vary by cluster.

The build partition

`--partition=build`

All the clusters have a few nodes that are configured for compiling software.

- `Sinteract -p build -c <n>`

(Using the login nodes will be slower when `<n>` is a number of cores greater than 1.)

Fairshare

Not everyone is equal

A group's priority on the clusters is related to to the type of account and the number of shares (percentage of the cluster) that they have committed to use (paid) and their recent usage relative to their shares. Within a group the relative consumption of the members decides who

has more priority. `http://slurm.schedmd.com/fair_tree.html`

`http://slurm.schedmd.com/priority_multifactor.html`

Helping yourself

man pages are your friends!

- `man sbatch`
- `man sacct`
- `man gcc`
- `module load intel; man ifort`

Helping yourself

get the examples!

```
git clone https://c4science.ch/diffusion/SCEXAMPLES/scitas-examples.git
```

Getting help

1234@epfl.ch

- send a mail to 1234@epfl.ch
- start the subject with **HPC**

We need to know as many of the following as possible

- the Job ID
- the directory location and name of the submission script
- where the “slurm-*.out” file is to be found
- how the “`sbatch`” command was used to submit it
- the output from “`env`” and “`module list`” commands

Going further

SCITAS offers courses in

- Compiling code and using MPI
- MPI, an introduction to parallel programming
- MPI, advanced parallel programming
- Introduction to profiling and software optimisation
- Computing on GPUs
- Data Management: code and large files
- Introduction to Linux

Useful links

links

Change your shell at:

```
https://cadiwww.epfl.ch/cgi-bin/accountprefs/
```

SCITAS web site:

```
http://scitas.epfl.ch
```

(in particular) SCITAS documentation space:

```
http://scitas-data.epfl.ch/kb
```

SLURM man pages:

```
http://slurm.schedmd.com/man_index.html
```