Using the general purpose computing clusters at EPFL

scitas.epfl.ch

November 14, 2018
Welcome

What you will learn

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

What you will not learn

- Running MPI jobs
- Writing and (compiling) codes
- Parallelising code
What is a cluster?
What is a cluster?
What is a cluster?

Distinctive characteristics

- Network: Low latency and high bandwidth
- Storage: High performance parallel
- Accelerators: GPUs, Xeon Phi, etc...
Our clusters


Aries
Bellatrix
Castor
Deneb
Eltanin
Fidis
Gacruz
Helvetios
What is each cluster optimised for?

Parallel workloads: deneb+eltanin and fidis+gacrux

- Allocation: whole nodes

Serial workloads: deneb+eltanin serial partition

- Allocation: individual cores and memory
Shared Storage (cluster)

/scratch
- high performance temporary space
- not backed up
- low redundancy, built for performance
- local to each cluster
- automatically cleanup procedure deletes files (older than 2 weeks) without warning (when occupancy reaches a threshold)
- for disposable files: intermediary results, temporary files
## Shared Storage (global)

<table>
<thead>
<tr>
<th>/home</th>
</tr>
</thead>
<tbody>
<tr>
<td>- per user quotas of 100GB</td>
</tr>
<tr>
<td>- backed up to a remote site</td>
</tr>
<tr>
<td>- available on all clusters</td>
</tr>
<tr>
<td>- for important files: source code, final results, theses</td>
</tr>
</tbody>
</table>
Shared Storage (global)

/work

- per group quotas
- 50GB for free
- for more: 300CHF/TB for 3 years (+ 100CHF/TB for backup)
- available on all clusters
- for common files: software, datasets
Connecting to a cluster [Hands-on!]

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

**Interactive**
You sit in front of your computer, open MATLAB and work

**Batch**
You script the work to be done and put it in a queue. Your jobs will be run when appropriate resources become available.
Batch systems and schedulers

**Batch systems**

Take jobs and put them in a queue

**Schedulers**

Decide when and where your job will run depending on the requested resources and your priority.
SLURM
The fundamental command is `sbatch` which submits jobs to the batch system.

**Workflow**

A typical workflow to get your computation done is:

- create a short job-script
- submit it to the batch system
- *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!]

Prerequisites

Copy the examples to your working directory:

```bash
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
- gedit
- ...
Exercise 1: sbatch [Hands-on!]

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

sleep 10
echo "hello from $(hostname)"
sleep 10
```
#SBATCH –something
This is how you tell SLURM what resources you need
The number of nodes per job

--nodes 1
--nodes 64

If not specified then the default is 1
The number of MPI tasks per job

`--ntasks 1`

`--ntasks 64`

If not specified then the default is 1
The number of CPUs per task for multithreaded applications

--cpu-per-task 1

--cpu-per-task 28

If not specified then the default is 1. Note that this value cannot be more than the number of cores/cpus in a compute node!
Socket, Core, Threads vs CPU

**Socket**
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).

**Core**
A complete private set of registers, execution units, and retirement queues needed to execute programs.

**Threads**
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.
The required memory per node

--mem 4096M

--mem 120G

If not specified then the default is 4096MB per CPU. Please don’t ask for 128G.
How long will your job run for?

--time 06:00:00

--time 2-23

If not specified then the default is 15 minutes.
Which partition should my job be sent to?

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

If not specified then the default is parallel on Fidis 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 1509281

$ cat /scratch/<username>/slurm-1509281.out
hello from r02-node12

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.
To cancel a specific job:
```
scancel <JOB_ID>
```

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

```bash
#!/bin/bash
#SBATCH --workdir /scratch/<username>
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 28
#SBATCH --mem 120G
#SBATCH --time 00:30:00
#SBATCH --account <your account>
#SBATCH --reservation intro2clusters

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.
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 the HPC 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 is a utility that allows multiple, often incompatible, tools and libraries to co-exist on a system.

It’s the usual tool for organising software on HPC clusters but each site uses it in subtly different ways.
How software modules are organised

- Packages are organized hierarchically: Compiler / MPI / blas
- **Modules** hides things until you load the dependencies
- **Modules** is designed to maintain the environment consistent
- **Modules** does everything possible to automatically reload any software when one of the hierarchy layers is changed
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!]

- 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

```bash
#!/bin/bash
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 4
#SBATCH --nodes 1
#SBATCH --mem 16G
#SBATCH --time 00:15:00
#SBATCH --account <your account>
#SBATCH --reservation using

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 --help

usage: Sinteract [-c cores] [-n tasks] [-t time] [-m memory] 
[-p partition] [-a account] [-q qos] [-g resource] [-r reservation] 
options:
   -c cores     cores per task (default: 1)
   -n tasks     number of tasks (default: 1)
   -t time      as hh:mm:ss (default: 00:30:00)
   -m memory    as #[K|M|G] (default: 4G)
   -p partition (default: parallel)
   -a account   (default: scitas-ge)
   -q qos       as [normal|gpu|gpu_free|mic|...] (default: )
   -g resource  as [gpu|mic][:count] (default is empty)
   -r reservation reservation name (default is empty)
   -s constraints list of required features (default is empty)

examples:
   /usr/bin/Sinteract -c 16 -p serial
   /usr/bin/Sinteract -p gpu -q gpu_free -g gpu:1
### Sinteract

```
[<user>@deneb2 ~]$ Sinteract -p debug
Cores: 1
Tasks: 1
Time: 00:30:00
Memory: 4G
Partition: debug
Account: scitas-ge
Jobname: interact
Resource:
QOS: scitas
Reservation:

salloc: Granted job allocation 1451785
salloc: Waiting for resource configuration
salloc: Nodes r02-node02 are ready for job
[<user>@r02-node02 ~]$  
```
The S tools

Custom tools to make your life easier

- Sinteract - custom tool to access a node
- Sshare - show fairshare information
- Squeue - show your pending and running jobs
- Sjob - show information about a job
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

[rmsilva@r02-node02 ~]$ echo $TMPDIR
/tmp/1451790
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

```bash
--partition=build
```

All the clusters have a few nodes that are configured for compiling software - please don’t use the login nodes for this!

- Sinteract -p build
Fairshare

Not everyone is equal

A group’s priority on the clusters is related to to 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
man pages are your friends!

- man sbatch
- man sacct
- man gcc
- module load intel; man ifort
Helping yourself

code example

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

See the Modules directory for Comsol and Ansys examples.
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

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