Using the general purpose computing clusters at EPFL

scitas.epfl.ch

September 12, 2017
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

- Writing and compiling codes
- Parallelising code
What is a cluster?
What is a cluster?

Compute nodes (back-end)

User

User
What is a cluster?
What is a cluster?
What is a cluster?

Scheduler

Login nodes (frontend)

Compute nodes (backend)

User
What is a cluster?
What is a cluster?
What is a cluster?
## Our clusters

<table>
<thead>
<tr>
<th>Login hostname</th>
<th>Hosts #</th>
<th>Cores # x GHz</th>
<th>RAM GB</th>
<th>Network Gbit/s</th>
<th>Storage TB</th>
</tr>
</thead>
<tbody>
<tr>
<td>castor.epfl.ch</td>
<td>50</td>
<td>16x2.6</td>
<td>64</td>
<td>10 (Eth)</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deneb{1,2}.epfl.ch</td>
<td>376</td>
<td>16x2.6</td>
<td>64</td>
<td>40 (IB)</td>
<td>350</td>
</tr>
<tr>
<td></td>
<td>144</td>
<td>24x2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16x2.6</td>
<td></td>
<td>+ 4x NVidia K40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>16x2.6</td>
<td></td>
<td></td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>32x2.6</td>
<td></td>
<td></td>
<td>512</td>
</tr>
<tr>
<td>fidis.epfl.ch</td>
<td>335</td>
<td>28x2.6</td>
<td>128</td>
<td>56 (IB)</td>
<td>375</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is each cluster optimized for?

Distinctive characteristics

- Network: Fast vs. Slow, Low latency vs. High latency
- Storage: Standard vs. High performance parallel
- Accelerators: GPUs, Xeon Phi, etc...

Parallel multi-node workloads

- deneb and fidis have a fast low latency network interconnect (Infiniband QDR/FDR), and a fast parallel storage system (GPFS)

Single node and serial workloads

- castor has a standard network interconnect (10 Gbit/s) and a slower shared storage system
Shared Storage (cluster)

/scratch

- high performance temporary space
- is not backed up
- low redundancy, built for performance
- local to each cluster
- automatically cleanup procedure deletes files without warning
- for disposable files: intermediary results, temporary files
Shared Storage (global)

/home
- filesystem has per user quotas of 100GB
- backed up to a remote site
- shared, available on all clusters
- for important files: source code, final results, theses

/work
- filesystem has per group quotas
- 50GB for free
- 300CHF/TB for 3 years
- shared, available on all clusters

Getting access to /work
If your group would like to purchase space then get in touch with SCITAS.
Batch Systems

Batch
Goal: to take a list of jobs and execute them according to a priority when appropriate resources become available
Interactive use is possible but it not the principal way of running jobs!

SLURM
We use SLURM on all our clusters. It’s widely used in the HPC world and open source.
http://slurm.schedmd.com

Slurm
Workflow of a job

1. Create job script
2. submit job
3. dispatch job
4. get job output
Connecting to a cluster [Hands-on!]

```
ssh -X username@deneb2.epfl.ch
```

- **Linux**: simply connect using `ssh`
- **Windows**: install and start X server (XMing/Xwin32), connect using PuTTY (with X11 Forwarding enabled)
- **OSX**: install and start X server (XQuartz), connect using `ssh`

*X Forwarding is not strictly needed, but it is used in one of the examples further ahead.*

**Basic shell commands: moving around:**

- `id`
- `pwd`
- `ls /scratch/<username>`
- `cd /scratch/<username>`
The fundamental command is `sbatch` which submits jobs to the batch system.

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.
Exercise 1: sbatch [Hands-on!]

**Prerequisites**

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

ex1.sh

#!/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 scitas-courses
#SBATCH --reservation scitas-courses

sleep 10
echo "hello from $(hostname)"
sleep 10
Exercise 1: SLURM directives

```bash
#SBATCH: directive to the batch system

--nodes 1
# the number of nodes to use - on Castor this is limited to 1

--ntasks 1
# the number of tasks (in an MPI sense) to run per job

--cpu-per-task 1
# the number of cores per aforementioned task

--mem 4096
# the memory required per node in MB

--time 12:00:00
--time 2-6
# the time required (12 hours and 2 days 6 hours respectively)
```
Exercise 1: submit ex1.sh to the batch system [Hands-on!]

No time was specified so it defaults to 15 minutes

```bash
$ sbatch ex1.sh
Submitted batch job 1509281

$ cat /scratch/<username>/slurm-1509281.out
hello from c38
```

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.
Exercise 1: What went on? [Hands-on!]

Sjob job-id

$ Sjob 1451763

<table>
<thead>
<tr>
<th>JobID</th>
<th>JobName</th>
<th>Cluster</th>
<th>Account</th>
<th>Partition</th>
<th>Timelimit</th>
<th>User</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1451763</td>
<td>ex1.sh</td>
<td>deneb</td>
<td>scitas-ge</td>
<td>debug</td>
<td>00:15:00</td>
<td>rmsilva</td>
<td>scitas-ge</td>
</tr>
<tr>
<td>1451763.bat+</td>
<td>batch</td>
<td>deneb</td>
<td>scitas-ge</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Submit</th>
<th>Eligible</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Elapsed</th>
<th>ExitCode</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00:20</td>
<td>0:0</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>00:00:20</td>
<td>0:0</td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NCPUS</th>
<th>NTasks</th>
<th>NodeList</th>
<th>UserCPU</th>
<th>SystemCPU</th>
<th>AveCPU</th>
<th>MaxVMSSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1</td>
<td>r02-node02</td>
<td>00:00.002</td>
<td>00:00.025</td>
<td>00:00:00</td>
<td>145216K</td>
</tr>
</tbody>
</table>

15 / 35
Cancelling jobs

**scancel**

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!]

59GB of memory required

```
#!/bin/bash
#SBATCH --workdir /scratch/<username>
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 8
#SBATCH --mem 59G
#SBATCH --time 00:30:00
#SBATCH --account scitas-courses
#SBATCH --reservation scitas-courses
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<email_address>

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 a custom squeue that shows only your jobs with more useful information.
Exercise 2: what’s **really** going on? [Hands-on!]

### scontrol

```bash
$ scontrol -d show job <jobid>

$ scontrol -d show job 1451756
JobId=1451756 JobName=ex2.sh
   UserId=rmsilva(141830) GroupId=scitas-ge(11902) MCS_label=N/A
   Priority=101877 Nice=0 Account=scitas-ge QOS=scitas
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   DerivedExitCode=0:0
   RunTime=00:02:47 TimeLimit=00:30:00 TimeMin=N/A
   SubmitTime=2017-09-12T11:25:56 EligibleTime=2017-09-12T11:25:56
   StartTime=2017-09-12T11:25:57 EndTime=2017-09-12T11:55:57 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=debug AllocNode:Sid=deneb2:31084
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=r02-node02
   BatchHost=r02-node02
   NumNodes=1 NumCPUs=16 NumTasks=1 CPUs/Task=16 ReqB:S:C:T=0:0:*:*:
   TRES=cpu=16,mem=59G,node=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
      Nodes=r02-node02 CPU_IDS=0-15 Mem=60416 GRESIDX=
   MinCPUsNode=16 MinMemoryNode=59G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   Gres=(null) Reservation=(null)
   OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/rmsilva/tmp/ex2.sh
   WorkDir=/scratch/rmsilva
   StdErr=/scratch/rmsilva/slurm-1451756.out
   StdIn=/dev/null
   StdOut=/scratch/rmsilva/slurm-1451756.out
   Power=
```
Why is a module system needed

- 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.

Lmod

- **Lmod** is a utility that allows multiple, often incompatible, tools and libraries to co-exist on a system.
- It’s a backwards-compatible evolution of the older GNU Modules.
How software is organised under Lmod

- Packages are organized hierarchically: Compiler / MPI / blas
- Lmod is designed to maintain the environment consistent
- Lmod does everything possible to automatically reload any software when one of the hierarchy layers is changed

Basic commands

- module avail
- module load / unload <module-name>
- module spider <name>
- module save / restore <mnemonic-name>
- module purge
Exercise 3: modules [Hands-on!]

ex3.sh: using module files

#!/bin/bash
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 1
#SBATCH --nodes 1
#SBATCH --mem 4G
#SBATCH --time 00:05:00
#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)
MPI parallel jobs

srun to run parallel jobs

#!/bin/bash

#SBATCH --nodes 2
#SBATCH --ntasks 32
#SBATCH --cpus-per-task 1

module purge
module load intel
module load intel-mpi
module load intel-mkl
module load espresso

srun pw.x < input.in
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:
- **Sinteract** - custom tool to access a node
- **salloc** - standard tool for an interactive allocation

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

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)

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 ~]$]
```
Storage locations

The different storage locations are accessible via environment variables.

- **$SCRATCH** - your scratch folder on the current cluster /scratch filesystem
- **$TMPDIR** - a **temporary** folder in a **local** filesystem (generally /tmp)
- **$WORK** - your groups’ folder in the global /work filesystem

Try it!

```
[rmsilva@deneb2 ~]$ Sinteract -p debug
[rmsilva@r02-node02 ~]$ echo $SCRATCH
/scratch/rmsilva
[rmsilva@r02-node02 ~]$ echo $TMPDIR
/tmp/1451790
[rmsilva@r02-node02 ~]$ echo $WORK
/work/scitas-ge
```
salloc [Hands-on!]

salloc then srun for MPI tasks

```
[<user>@deneb2 ]$ salloc -N 1 -n 2 --mem 2048 -p debug
salloc: Granted job allocation 1451788
salloc: Waiting for resource configuration
salloc: Nodes r02-node02 are ready for job

[<user>@deneb2 ]$ hostname
deneb2

[<user>@deneb2 ]$ srun hostname
r02-node02
r02-node02

[<user>@deneb2 ]$ exit
salloc: Relinquishing job allocation 1451788
```
The debug partition

```bash
--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:

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

The number of nodes in the partition and the limits vary by cluster.
Not everyone is equal

A group’s priority on the clusters is related 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
## Current usage and priority

$ Sshare

<table>
<thead>
<tr>
<th>Account</th>
<th>User</th>
<th>RawShares</th>
<th>NormShares</th>
<th>RawUsage</th>
<th>NormUsage</th>
<th>EffectvUsage</th>
<th>FairShare</th>
</tr>
</thead>
<tbody>
<tr>
<td>scitas-ge</td>
<td>aubort</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.195841</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>clemenco</td>
<td>1</td>
<td>0.062500</td>
<td>12486</td>
<td>0.000027</td>
<td>0.138814</td>
<td>0.171577</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>culpo</td>
<td>1</td>
<td>0.062500</td>
<td>186</td>
<td>0.000000</td>
<td>0.002074</td>
<td>0.173310</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>ddossant</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.195841</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>degiorgi</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.195841</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>desbioll</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.183709</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>eroche</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.195841</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>foureste</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.181976</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>guglielm</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.195841</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>nvarini</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.195841</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>query</td>
<td>1</td>
<td>0.062500</td>
<td>1</td>
<td>0.000000</td>
<td>0.000012</td>
<td>0.178510</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>rezzonic</td>
<td>1</td>
<td>0.062500</td>
<td>80</td>
<td>0.000000</td>
<td>0.000897</td>
<td>0.175043</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>richart</td>
<td>1</td>
<td>0.062500</td>
<td>1</td>
<td>0.000000</td>
<td>0.000019</td>
<td>0.176776</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>rmsilva</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000004</td>
<td>0.180243</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>scitasbui+</td>
<td>1</td>
<td>0.062500</td>
<td>77195</td>
<td>0.000170</td>
<td>0.858180</td>
<td>0.169844</td>
</tr>
<tr>
<td>scitas-ge</td>
<td>vkeller</td>
<td>1</td>
<td>0.062500</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.195841</td>
</tr>
</tbody>
</table>
Helping yourself

**man pages are your friends!**

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

1234epfl.ch

- send a mail to 1234epfl.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
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
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.epfl.ch/kb

SLURM man pages:

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