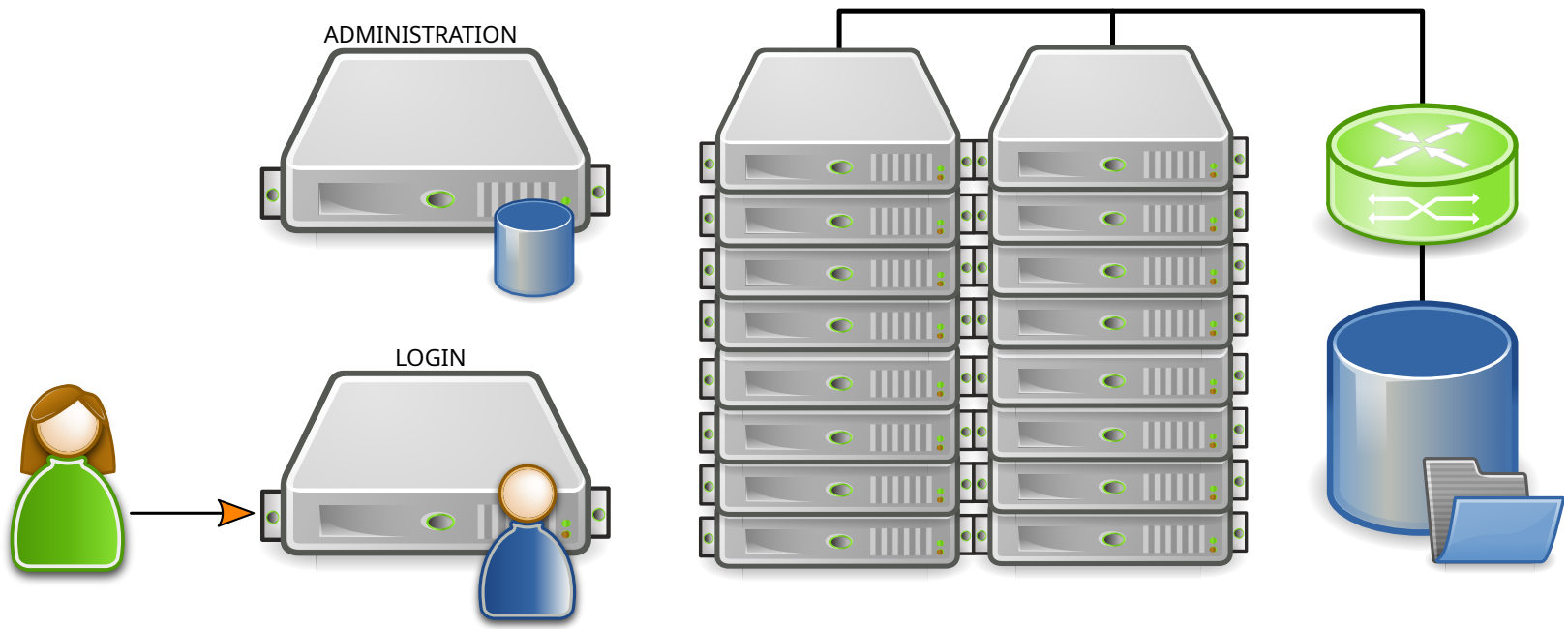
Intro to Izar, the GPU cluster at EPFL

scitas.epfl.ch

May 2, 2024

What is a cluster?



Connecting to Izar

`ssh <username>@izar.epfl.ch`

- Linux: connect using `ssh`
- OS X: connect using `ssh`
- Windows: PuTTY, WSL, install git-bash

Basic shell commands, moving around

- `id`
- `pwd`
- `cd /scratch/izar/<username>`
- `ls /scratch/izar/cs-552_examples`
- `cp -r /scratch/izar/cs-552_examples .`

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 (older than 2 weeks) without warning (when occupancy reaches a threshold)
- **for disposable files: intermediary results, temporary files**

Shared Storage (global)

/home

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

Data transfer

Copying files to/from cluster

- `scp` to transfer a few files or folders
- `rsync` to synchronize a lot of data

Examples

```
# To the cluster
scp my-script.py <username>@izar:./code/
rsync -a ./dataset <username>@izar:/scratch/izar/<username>/

# From the cluster
scp -r <username>@izar:/scratch/izar/<username>/results .
rsync -a <username>@izar:./updated-code/ Documents/updated-code

# To copy the slides to your computer
scp <username>@izar:/scratch/izar/cs-552_examples/slides.pdf .
```

Batch versus Interactive

Interactive

You sit in front of your computer, open Jupyter/MATLAB on the cluster 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

SLURM

sbatch

sbatch

The main 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.

Exercise 1: sbatch

Sinteract

```
#!/bin/bash -l
#SBATCH --chdir /scratch/izar/<put-your-username-here>
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 1
#SBATCH --mem 4G
#SBATCH --time 2:00
#SBATCH --gres gpu:1
#SBATCH --account cs-552
#SBATCH --qos cs-552
#SBATCH --reservation cs-552

# The --reservation line only works during the two 1-week periods
# when 80 GPUs are available. Remove otherwise.
echo "hello from $(hostname)"
sleep 60
date
```

#SBATCH --something

This is how you tell SLURM the resources you need

Note: There is no blank space between # and SBATCH

ntasks

The number of MPI tasks per job

```
--ntasks 1
```

```
--ntasks 40
```

If not specified the default is 1

cpu-per-task

The number of CPUs per task for multithreaded applications

```
--cpu-per-task 1

--cpu-per-task 20
```

If not specified the default is 1. Note that this value cannot be more than the number of cores/cpus in a compute node!

mem

The required memory per node

```
--mem 4096M
```

```
--mem 90G
```

If not specified the default is 4096 MB per core.

Don't be greedy

Don't ask for the exact amount of RAM on a node (192 GB).
The OS also needs some.

time

How long will your job run for?

```
--time 06:00:00

--time 1-20:00:00
--time   44:00:00
```

You can run jobs of up to 3 days of wall time.

Choose the time wisely

Small jobs fit in the gaps between big jobs.

Exercise 1: submit to the batch system

Let's submit our job

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

$ cat /scratch/izar/<username>/slurm-1509281.out
hello from i02
```

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.

Checking the queues

squeue

To check your own jobs:

```
squeue -u <username>
```

To check just your pending (PD) jobs:

```
squeue -u <username> -t PD
```

To check jobs from the course:

```
squeue -A cs-552
```

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
```

Modules

Scientific software

Many scientific codes are not packaged under most Linux distributions.

module

Module 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 slightly different ways.

Modules

How software modules are organised

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

Exercise 2: modules

Basic commands

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

Exercise 2: modules

Loading python

- `module av`
- `module load gcc openmpi`
- `module av`
- `module load python/3.10.4`
- `module load py-torch`
- `module list`
- `module purge`
- `module list`

Exercise 3: Python Virtual Environment

Creating a virtual environment

```
module load gcc python
virtualenv --system-site-packages ~/venvs/course_py-3.10

# --system-site-packages will use modules we compiled
# on our clusters. for some specific use cases you may
# need to create the venv *without* this option

# to activate it
source ~/venvs/course_py-3.10/bin/activate
# upgrade pip the first time you load the environment
pip install --upgrade pip

pip3 install jupyter
# to stop using the virtual environment
deactivate
```

Interactive access

Why interactive?

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

Sinteract

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

Sinteract

Sinteract

```
[<user>@izar ~]$ Sinteract -a cs-552 -q cs-552 -g gpu:1 -r cs-552
Cores:              1
Time:               00:30:00
Memory:             4G
Account:            cs-552
Jobname:            interact
Resource:           gpu:1
QOS:                cs-552
Reservation:        cs-552

salloc: Granted job allocation 1451785
salloc: Waiting for resource configuration
salloc: Nodes i40 are ready for job
Waiting for X11 setup...
[<user>@i40 ~]$
```

Change the default options

Check `Sinteract -h` to see how to change cores, RAM, ...

Interactive jobs - Key concepts

One user at a time

In interactive sessions, cores, RAM, GPUs are reserved for you, whether being used or not:

- You need to close your session once done, to free resources for others
- The debug partition/queue is available to everyone. For the course you can use:

```
Sinteract -a cs-552 -p debug -q debug -g gpu:1
```

Jupyter runs on the nodes

- They can be accessed from your browser even if running remotely
- As in any interactive session, the resources are reserved
- More info here: `https://scitas-doc.epfl.ch/advanced-guide/python/jupyter/`

Helping yourself

man pages are your friends!

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

Getting help

1234@epfl.ch

- send a mail to 1234@epfl.ch
- start the subject with **HPC**
- organize with the TAs! They likely know the answer to basic problems

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

Useful links

links

SCITAS web site:

`http://scitas.epfl.ch`

SCITAS courses (Intro to Linux, Clusters):

`https://www.epfl.ch/research/facilities/scitas/documentation/training/`

(IMPORTANT) SCITAS documentation space:

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

SLURM man pages:

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

Additional LLM Cluster Ops

Checkpoint Management

- The `/scratch/izar/<username>` directory is:
  - **not backed up**
  - **old files automatically purged every 14 days**

- How you should version your checkpoints:
  - Intermediate or cache checkpoints can be kept in /scratch
  - Permanent and **final checkpoints should be kept in the home directory**

    If the checkpoint is large or the home directory is full, upload the models to huggingface-hub (https://huggingface.co/)

Runtime Estimates

- **You should estimate how long your job will take to run to completion**
  - Set an appropriate time in the sbatch script
  - If you set the time too short, your job will be killed before finishing running
    - Save checkpoints periodically!
  - If you set an unreasonable time (ex. 3 days), you'll wait on the queue for a long time

- **Tips for better estimation**
  - Run a few iterations (train & validation), check the average runtime per iteration
  - Check the runtime of loading the dataset and a checkpoint, and of saving checkpoints using simple Python timer tools

Job Management

- **You should not be submitting many jobs simultaneously.**
  - Your job priority might go down if you do.

- During periods with peak demand, we will actively monitor the jobs and preempt repeated submissions.