HOW TO USE THE CLUSTER AT LNC

What is FRONTEX?

FRONTEX is the name of the LNC's server, which includes 3 main elements:
- a server which is the head of the system called the ***master***
- a server for data storage called ***nvisu*** (also called ***filer*** or ***shared***)
- a cluster for computation.

Master
This is the 'admin' server, connected to both nvisu and the cluster. Jobs executed on the cluster are submitted from the master through SLURM (link). Users have restricted space on the master (200 G) that should be used only to debug and submit jobs. No permanent storage should be done on the master given its fundamental role and its restricted storage capacity.

Nvisu / Filer / Shared
This is the storage server, with a capacity of 80 To. It has multiple features:
- it's a graphic processing unit, allowing visualization of the data
- it's backed-up (physically outside of the ENS building)
- multiple users can have access and work on the same project.

The Cluster
on which it's possible to launch a bunch of jobs using MATLAB or Python.
This cluster is composed of 50 nodes with 4 different generations.
- firstgen from node0 to node20 with on each node 12 CPUs and 44 Gb of RAM
- secondgen from node21 to node36 with on each node 16 CPUs and 60 Gb of RAM
- fastgen for node37 and node38 with on each node 16 CPUs and 60 Gb of RAM
- dellgen for node43 to node50 with on each node 40 CPUs and 120 Gb of RAM

This was the technical part, just remember the number of threads (cpus) by node that will be useful later.

To launch jobs

Ask for an account

The first thing to do is to ask to your IT person an account on FRONTEX. You can send an email to Julie.costil@ens.fr.
It will be necessary to meet her to set your password and it's always fun to talk to an IT guy.

Depending on the size of your dataset and the needed resources (computation only, computation+storage), you will be attributed:
- a home directory on the master
- a project directory on nvisu (for data storage).

You will also be added to the list of users for the cluster.

Connect to the master

You can now connect to the master via SSH using the command:
`$ ssh login@129.199.81.23.`

If you are using Windows I suggest to use MobaXterm (link)

(IMPORTANT : go to "Settings", "Configuration", "X11" and select disabled to "X11 remote access")
If you are using Linux or MAC OS you can use the terminal integrated

When logged in you are automatically connected to your home directory.

Upload your work

To launch your jobs, you have to upload your code/data in your home directory on the master (very small dataset) or in your project directory on nvisu (large dataset).

On the master, you can load your data using the following commands:
$ scp *filetoupload login*@129.199.81.23**:~**
OR
$ scp –r *directorytoupload login*@129.199.81.23:~

On nivsu, you can load your data using the following commands:
$ scp *filetoupload login*@129.199.81.42:/shared/projects/project_*nameoftheproject*/.
OR
$ scp –r *directorytoupload*
login@129.199.81.42:/shared/projects/project_*nameoftheproject*/.

The "-r" flag use recursivity of scp command which permit to upload a directory and not several files one by one.

Create a submission script

A submission script is a bash script. Bash is the "program" you are using to interact with the system (by launching commands for example).
SBATCH is the command to launch jobs to SLURM, here is an example of script:

```
#!/bin/bash
#SBATCH --job-name=test_slurm
#SBATCH --output=test_dwalter.out
#SBATCH --error=test_dwalter.err
#SBATCH --partition=firstgen
#SBATCH --array=[1-30]
#SBATCH --mem-per-cpu=4000
#SBATCH --share

module load matlabR2012a

# Create a local working directory
mkdir -p /tmp/dwalter/$SLURM_JOB_ID

# Kick off matlab
matlab -nodesktop -nojvm -r frontex_randomization_main

# Cleanup local work directory
rm -rf /tmp/dwalter/$SLURM_JOB_ID
```

These are some useful commands to use in your submission script:

#SBATCH --partition = firstgen……	Launch your jobs on a specific generation. You can specify multiple generations, separating the name by a coma.
#SBATCH --mem-per-cpu = 4000……	Specify a limit of memory usage per CPU, in MB.
#SBATCH --job-name = ……	Name your job
#SBATCH –output = ……	Name your output files
#SBATCH –error = ……	Name your errors files
#SBATCH --mail-user = ……	Specify your email address
#SBATCH --mail-type = ……	Choose what type of mail you want: “ALL”, “BEGIN”, “END”, “FAIL”, “REQUEUE”
#SBATCH --workdir = ……	Set the working directory of the batch script
#SBATCH --export = ……	You can choose to export your environment variables to the batch job. The options are “ALL” (the default one), “NONE” or specify the one you want to export (ie #SBATCH --export=PATH)
#SBATCH --requeue = ……	To requeue the job if there is a node failure
#SBATCH -- array = ……	To specify a maximum number of simultaneous jobs running at the same time

!! Pay attention not to use “mem” option and “mem-per-cpu” option in the same submission script !!

!! Remember that the cluster is a shared resource. In order to have everything working smoothly, it's a good idea to limit the amount of resources used by each user.
2 ways to do that:

- **use the minimum amount of memory per cpus using this command line**

#SBATCH --mem-per-cpu= xxx
The minimal amount of memory per CPUs is 4000 for most of the nodes but it all depends on your job. You might need a higher value to have your jobs running. In that case, keep in mind that i twill use more CPUs.

- **limit the number of simultaneous jobs using this command line**

#SBATCH --array=0-XXX
Depending on the number of users using the cluster at the same time as you (see with squeue command), it’s a good idea to keep a reasonable amount of simultaneous jobs running. !!

Launch your jobs

To launch your jobs, use you submission script and execute:
$ sbatch *submission_script.sbatch*

Manage your jobs

Here a few commands provided by SLURM that can be useful:

$ **sbatch** to submit a job script
$ **sinfo** show the state of partitions (queues) and nodes.
$ **squeue** show the state of jobs
$ **scancel** to cancel your pending/running job by using JOBID
All these commands are provided with a lot of options. You can have a look at the SLURM website for documentation (link).

To go further

If you need help

If you need some help to create your submission script or to find other useful options, contact the IT person.
In the same way if you have any information that can be useful to the others please contact the IT person so we can update this procedure

Tips

MODULE LOAD

As you could see in the example of submission script, I used the linux command "module load". You can use this command to launch module which in our case permit to modify environment variables.
You can find all the modules you can launch with the command: $module avail
Don't pay attention to modules in the folder */usr/share/Modules/modulefiles* and */etc/modulefiles*, which your account don't have access to. But I created 3 new modules in */usr/local/etc/modulefiles* :
Anaconda2
matlabR2012a
matlabR2016b
If you code in Python, the "anaconda2" module should be useful and for MATLAB's hardcore developers you can use one or the other to change version. If you need another one it's still possible to install
Use the "module purge" command if you need to use another module

SLURM IDs

In SLURM there is several IDs you can use. If you try to launch an array of job there 3 IDs that are important:
- SLURM_JOB_ID
- SLURM_ARRAY_JOB_ID
- SLURM_ARRAY_TASK_ID

For example if you launch your job with this line in your bash script: #SBATCH --array=1-3
And get this response: *Submitted batch job* ***666***

The IDs you could use are :
- SLURM_JOB_ID=666
- SLURM_ARRAY_JOB_ID = 666

- SLURM_ARRAY_TASK_ID = 1

- SLURM_JOB_ID=667
- SLURM_ARRAY_JOB_ID = 666
- SLURM_ARRAY_TASK_ID = 2

- SLURM_JOB_ID=668
- SLURM_ARRAY_JOB_ID = 666
- SLURM_ARRAY_TASK_ID = 3

See the difference between SLURM_JOB_ID and SLURM_ARRAY_JOB_ID.
Note that 667 and 666_2 are equivalent to identify an element of a job array

In the same way, you can name your stdin/stdout/stderr files with two options:
%A which is the SLURM_ARRAY_JOB_ID
%a which is the SLURM_ARRAY_TASK_ID
For example:
#SBATCH --output=”slurm-%A_%a.out”