Quick start

This document is intended for new users of the service. It gives information on such basics as how to log in, how to run application and where to get help.

Computation Workflow on the HPC

A typical Computation workflow will be:

1. Connect to the HPC
2. Transfer your files to the HPC
3. Compile your code and test it
4. Create a job script
5. Submit your job
6. Wait while
    - your job gets into the queue
    - your job gets executed
    - your job finishes
7. Move your results

Connecting to clusters

Once your account is created you need to connect to the cluster to change your password.

Before starting using the cluster, remember that mesocentre is composed of one cluster and one shared multi-core machine:

1. mesologin1.univ-fcomte.fr dedicated for sequential, array tasks, MPI and openMP programs (only with SGE)
2. mesoshared.univ-fcomte.fr a single SMP machine dedicated for interactive jobs: matlab, visualization, post-processing, ...

You can show the characteristic of each cluster here.

Example, connect and change password:

```
$ ssh user@mesologin1.univ-fcomte.fr
```

And then:

```
$ passwd
```

Connection mode

The easiest way to connect to the cluster is by using SSH protocol. Windows users can use **PuTTY**.

However you can use X2GO client, a powerful graphical user interface to run interactive applications like matlab on `mesoshared.univ-fcomte.fr` machine.

To transfer data from/to cluster you can use `scp`, `sftp` or other graphical applications (based on ssh) like : **winscp**, **filezella**, ...

User workspace

Once connected each user will have a home directory organized as follow :

- `HOME`: this is the home directory, total allowed size: **10GB**
- `WORK`: Computation and software must be **Started** from here, total allowed size: **1TB**

Read more here

Installed software

See the list of installed software and how to use them Logiciels installés

Several software are installed on the clusters. You can show them by :

```
$ module avail
```

This will show the list of software installed and their version.

the output looks like this :

```
$
```

```
------------------------------------------------------------------------------
-------------- /Softs/lumiere-modules ----------------------------------------
------------------------------------------------------------
bio/bamtools/2.3.0           bio/t-coffee/default
intel/14.0.2(default)        lang/java/1.8
lib/libreadline/6.2          numlib/armadillo/icc/6.300.2 tools/gama/1.6.1
bio/bcftools/1.1             bio/velvet/1.2.09            intel/15.0.3
lang/ocaml/4.00.0            lib/libxml2/2.8.0
numlib/fftw/gcc/3.3.4        tools/gate/6.2
bio/beagle/1.1.0             castem/2015
intel/icc/2013_sp1.2.144     lang/perl/5.12.3             lib/libxml2/2.9.1
numlib/fftw/icc/3.3.4        tools/git/2.1.2
bio/bwa/0.7.12               comsol/3.5a
intel/icc/2015.3.187         lang/pgi64/15.7              lib/libxml2/2.9.2
numlib/gsl/1.15              tools/h5utils/1.12.1
bio/cdhit/4.6.4              comsol/4.0
intel/ifort/2013_sp1.2.144   lang/pgi64/2015              lib/ncurses/5.9
numlib/openblas/0.2.15       tools/mercurial/2.5.2
bio/emboss/6.6.0             comsol/4.1
intel/ifort/2015.3.187       lang/python/2.6.5            lib/pteros/2.0
numlib/petsc/3.6.3           tools/mercurial/2.7.2
bio/exabayes-mpi/1.2.1       comsol/4.2
[root@mesocomte0 ~]#
```

You view the software installed in category, for example for languages (lang):

```
$ module avail lang
```

Another way to know if a software is installed or not :

```
$ module avail softwarename
```

For example, let's see if matlab is installed:

```
$ module avail matlab
------------------------------------------------------------------------------
------------
matlab/r2010b matlab/r2011b matlab/r2012a matlab/r2012b
```

Say we need to use matlab, we first need to load **matlab/r2012b** in our environment, this is done by:

```
$ module load matlab/r2012b
```

and then launch matlab:

```
$ matlab
```

to unload module

```
$ module rm softwareName
```

Submitting Jobs with SGE

Jobs are submitted to the compute nodes via the Sun Grid Engine (SGE) batch submission system. Basic SGE batch scripts should conform to the following template:

```
#!/bin/bash

#$ -M toto@univ-xxx.fr   # Email address for job notification

#$ -m abe      # Send mail when job begins, ends and aborts

#$ -pe put_here_pe 12    # Specify parallel environment and legal core size

#$ -q put_here_queue     # Specify queue

#$ -N job_name             # Specify job name

module load xyz          # Required modules

./foo           # run Application
```

Submitting a Job

```
qsub jobscript.sge
```

Where `jobscript.sh` is the script that you have prepared.

These files will be stored in the directory you submitted the jobs from if you use the -cwd option, and in your home directory otherwise. The names are constructed from the name of the job script, `.o` for output or `.e` for errors, and the job number. So the command above might create files such as 'jobscript.sh.o12345' and 'jobscript.sh.e12345' where 12345 is the number that is assigned to your job when you submit it.

JOB must bhe launched from WORK directory.

Cancelling a Job

To cancel a job, use the command:

```
qdel 12345
```

where 12345 is the job number assigned to your job.

Job status

To query the status of your submitted jobs, use the command:

`qstat` or for further information

SGE example SCRIPT

Sequential script

```
#!/bin/bash


#$ -N test_sge
#$ -o $JOB_NAME.$JOB_ID.out
#$ -e $JOB_NAME.$JOB_ID.err
#$ -q all.q


./my program myParam1 myParam2
```

Array tasks script

When you want to run a number of mostly identical jobs with the only difference being input parameters or data sets you should submit an Array Job

```
#!/bin/bash

#$ -N test_sge
#$ -o $JOB_NAME.$JOB_ID.out
#$ -e $JOB_NAME.$JOB_ID.err
#$ -q all.q
#$ -t 1-32     ## 32 tasks
./appli input$SGE_TASK_ID output$SGE_TASK_ID
```

OpenMP script

To use **OpenMP**, you need to specify that you wish to use that Parallel Environment (-pe openmp) , and declare the number of slots

```
#!/bin/bash

#$ -N test_sge

#$ -o $JOB_NAME.$JOB_ID.out
#$ -e $JOB_NAME.$JOB_ID.err
#$ -q all.q
#$ -pe openmp 8  ## we request 8 cores

export OMP_NUM_THREAD=$NSLOTS

## lancement de l'application

./appli input
```

MPI script

To use MPI, you need to specify that you wish to use that Parallel Environment (-pe mpi), and declare the number of slots

```
#!/bin/bash

#$ -N test_sge
#$ -o $JOB_NAME.$JOB_ID.out
#$ -e $JOB_NAME.$JOB_ID.err
#$ -q parallel.q
#$ -pe mpi 64

### load MPI to use
module load mpi/openmpi/icc/1.7.5

## lancement de l'application
```

```
mpirun -np $NSLOTS ./appli_mpi  myparams
```

More about SGE

For further information about SGE visit this link

Getting Help

More information is available in this site (in french) or the FAQ section or you may wish to contact admins: svpmeso@univ-fcomte.fr