PBS Professional Quick Start Guide for ELIXIR-CZ

Registration and other information:

Access to frontend

$ ssh elmo.elixir-czech.cz
Password:

Resource request

The job is implicitly assigned with one processor and 400MB of memory on a single node, if not specified otherwise.
- Always specify desired walltime (in [hh:mm:ss] format)
- Always specify size, memory and type of scratch

Applications

module add <MODULE-NAME> # module activation
module avail # lists all available modules
module avail mrbay # lists modules starting with "mrbay"
Appication Wiki page:
https://wiki.metacentrum.cz/wiki/Kategorie:Applications

Modifications: Attention, for better comprehensibility, following examples aren't complete. Memory, scratch, walltime ... may not be assigned!
- Maximal duration of job 1 hour / 5 days:
  qsub -l walltime=1:00:00
  qsub -l walltime=12:00:00
- Request 2 CPU cores on 1 node (chunk):
  qsub -l select=1:ncpus=2
- Request 2 nodes (chunk) each with 1 CPU:
  qsub -l select=2:ncpus=1
- Request for specific node:
  qsub -l select=1:ncpus=1:vnode=elmo5-1:mem=1gb ...
- Request for machine with cgroups:
  qsub -l select=1:ncpus=1:cgroups=True ...
- Request for 2 chunks on exclusive node/s:
  qsub -l select=2:ncpus=1 -l place=excl ...
- Using scratch <scratch_local|scratch_ssd|scratch_shared>, there is no default type of scratch!
  qsub -l select=1:ncpus=1:mem=4gb:scratch_ssd=1gb ...
- Interactive job:
  qsub -l -1 select=1:ncpus=4:mem=1gb ...
- GPU computing (GPU cards IDs in CUDA_VISIBLE_DEVICES variable):
  qsub -l select=1:ncpus=1:ngpus=2 -q gpu ...
- Request concrete OS
  qsub -l select=1:ncpus=2:mem=1gb:os=debian9 ...
- Infiniband request:
  -l select=3:ncpus=1 -l place=group=infiniband

qstat -u <login> # lists all user running or waiting jobs
qstat -xu <login> # list all user jobs (also finished)
qstat <jobID> # outputs basic information about the job
qstat -f <jobID> # outputs more detailed information about running or waiting job
qstat -xf <jobID> # outputs more detailed information about running, waiting or finished job
qdel <jobID> # kills the job (if necessary)

See the man qstat page for a description of all options.

Status of the job (most common):
Q ... queued  E ... exiting
R ... running F ... finished

Getting the standard output (stdout) and standard error output (stderr)

Once a job terminates (either regularly or forcibly), there are the following files (representing standard output and standard error output) created in the directory from which you submitted the job (the directory where the qsub command was performed):

<job_name>.o<jobID> ... standard output
<job_name>.e<jobID> ... standard error output

for example, the files "script.o12345" and "script.e12345". Exploring the files can show you job's results, or can inspire you in tracing its error run.

Both outputs can be merged to a single file by adding parameter qsub -j oe.
<job_name>.o<jobID> ... standard and error output

Useful links

Qsub assembler:
https://metavo.metacentrum.cz/pbsmon2/person
PBSMon web application monitoring jobs:
https://metavo.metacentrum.cz/pbsmon2/
Documentation:
https://wiki.metacentrum.cz/wiki/
EXAMPLE: Running the batch job

1. Log on the frontend:

```bash
$ ssh <USERNAME>@elmo.metacentrum.cz
Password:
```

2. Prepare the job's input data: Copy the job's input data to your home (sub)directory (/storage/.../home/<USERNAME>/)

3. Prepare the job's startup script (named, for example, script.sh):

```bash
#!/bin/bash
#PBS -l select=1:ncpus=2:mem=4gb:scratch_local=10gb
#PBS -l walltime=01:30:00
#PBS -N example
# initialize the required application (e.g. Python, version 3.4.1, compiled by gcc)
module python-3.4.1-gcc

# storage is shared via NFSv4
DATADIR="/storage/brno3-cerit/home/$LOGNAME/example"

# clean the SCRATCH when job finishes (and data are successfully copied out) or is killed
trap 'clean_scratch' TERM EXIT

# copy input files from shared storage on disk to scratch
cp $DATADIR/app.py $DATADIR/input.txt $SCRATCHDIR

cd $SCRATCHDIR

# ... the computation ...
python app.py input.txt

# copy resources from scratch directory back on disk
# field, if not successful, scratch is not deleted
cp output.txt $DATADIR || export CLEAN_SCRATCH=false
```

Instruct your script to transfer the job's input data to a subdirectory of the fast /scratch storage volume (before the application run), and subsequently, to transfer the job's output data and to clean the working directory (after the application run):

### Shared storage (home via NFS)

```bash
# secure copy using scp (disks not connected via NFS)
DATADIR="storage-brno3.metacentrum.cz:~/example"

# clean the SCRATCH when job finishes (and data are successfully copied out) or is killed
trap 'clean_scratch' TERM EXIT

# copy input files from shared storage on disk to scratch
scp $DATADIR/app.py $DATADIR/input.txt $SCRATCHDIR

# use "scp -R ..." in case of copying directories

# ... the computation ...
python app.py input.txt

# copy resources from scratch directory back on disk
# field, if not successful, scratch is not deleted
scp output.txt $DATADIR || export CLEAN_SCRATCH=false
```

### Network connected storage

```bash
# copy input files from shared storage on disk to scratch
cp $DATADIR/app.py $DATADIR/input.txt $SCRATCHDIR

# use "scp -R ..." in case of copying directories

cd $SCRATCHDIR

# ... the computation ...
python app.py input.txt

# copy resources from scratch directory back on disk
# field, if not successful, scratch is not deleted
scp output.txt $DATADIR || export CLEAN_SCRATCH=false
```

4. Submit the batch job:

Specify the job’s resource requirements (number of execution nodes, requested processors, memory, scratch size and type, application license, etc.) and the maximum job run-time (walltime).

These resource requirements can be specified:

- inside the script file (see step 3) – in such a case, “qsub script.sh” is sufficient
- as command-line arguments to qsub utility (see below)
- in both places – in such a case, the command-line arguments have higher priority and override the values written inside the script

```bash
elmo$ qsub -q elixircz -l select=1:ncpus=2:mem=10gb:scratch_local=1gb -l walltime=4:00:00 -l matlab=1 script.sh
```