PBS Cheatsheet

cesnet

Information & registration

DOCUMENTATION https://docs.metacentrum.cz

WEB https://metacentrum.cz

Access the frontend

```
ssh <USERNAME>@<FRONTEND>.metacentrum.cz
```

All frontends (e.g., *nympha*, *skirit*, *tarkil* or *zenith*) are listed in the documentation. Every frontend can access all data storages.

Resource request

Each job requires a reservation of the necessary hardware resources, and may also require the specification of additional parameters.

- Always specify the desired **walltime** (in [*hh:mm:ss*] format), number of **CPUs**, and the scratch **size** and **type** (*scratch_local*, *scratch_ssd*, *scratch_shared* or *scratch_shm*).

```
skirit$ qsub \
 -l select=1:ncpus=2:mem=10gb:scratch_local=2gb \
 -l walltime=12:00:00 myscript.sh
```

Examples of modifications

- Maximum job duration (5 days)
  `-l walltime=120:00:00`
- Require Matlab licence
  `-l matlab=1`
- Request 1 node (chunk) with 2 CPUs
  `-l select=1:ncpus=2`
- Request 10 GB of memory
  `-l select=1:mem=10gb`
- Request 2 nodes (chunks) on the same cluster
  `-l select=2:ncpus=1 -l place=group=cluster`
- Interactive job (upper-case i)
  `-I -l select=1`
- Request 1 GPU card
  `-l select=1:ngpus=1`
- Request a GPU card with 10 GB of memory
  `-l select=1:ngpus=1:gpu_mem=10gb`

Storage

Each frontend has its native storage (your */home*). After logging in, use `pwd` to see which storage you're on. You can copy smaller local files to storage via frontend:

```
scp <FILE> <USER>@<FRONTEND>.metacentrum.cz:~
```

For larger files prefer copying directly to the storage:

```
scp <FILE> <USER>@storage-<CITY>.metacentrum.cz:~
```

You can access any other storage from current frontend:

```
cd /storage/<CITY>/home/<USER>/
```

Scratch

The variable `$SCRATCHDIR` represents a per-job temporary storage directory. Copy the job-related inputs there and clean up when finished. You need to specify the scratch type with *qsub*:

```
scratch_local=10gb  # scratch on the HDD or SSD
scratch_ssd=1gb     # fast, exclusively on SSD
scratch_shared=10gb # if nodes access the same data
scratch_shm=true    # ultra fast, in the RAM
```

Applications

The list of currently available software is maintained in the documentation.

```
module add <MODULE-NAME>  # module activation
module avail intel        # list modules starting with "intel"
module avail *intel*      # list modules containing "intel"
module list               # lists already loaded modules
```

Monitoring jobs

- Jobs finished within 24 hours:

```
qstat -u <LOGIN>  # lists user's running/queued jobs
qstat -xu <LOGIN> # lists finished and moved jobs
qstat -f <JOB-ID> # details of running/queued job
qstat -xf <JOB-ID># details of finished job
```

  Call `man qstat` to list all *qstat* options.

- Jobs older then 24 hours:

```
pbs-get-job-history <JOB-ID>
```

```
qdel <JOB-ID>  # kills the job
```

Job Statuses

Q ... queued	E ... exiting after having run
H ... held	F ... finished
R ... running	X ... finished subjob
S ... suspended	W ... waiting

Job outputs

The standard output and error output files are located in the directory from which you submitted the job (where you ran *qsub*).

```
<JOB-NAME>.o<JOB-ID> ... standard output
<JOB-NAME>.e<JOB-ID> ... standard error output
```

These can be merged into a single file by adding parameter `qsub -j oe`, the output will be:

```
<JOB-NAME>.o<JOB-ID> ... merged output
```

You can specify different paths by setting

```
qsub -o /custom-path/output-file-name
qsub -e /custom-path/error-file-name
```

PBS Quick Start

cesnet

1. Input data

Copy a small input file from your local computer to your *home* directory via the chosen frontend (e.g., *tarkil*).

```
$ scp myfile.txt myuser@tarkil.metacentrum.cz:~
> myuser@tarkil.metacentrum.cz's password:
> myfile.txt        100%   10     2.9KB/s   00:00
```

2. Log in

Log in to the frontend server via *ssh*.

```
$ ssh myuser@tarkil.metacentrum.cz
> myuser@tarkil.metacentrum.cz's password:
```

3. Prepare a Python script

Prepare a simple Python script that reads the input file and writes the number of lines to the output file. Either create it using vim or nano directly in your *home*, or copy it from your local computer.

counter.py

```
import sys

infile = sys.argv[1]
outfile = sys.argv[2]

with open(infile, "r", encoding="utf-8") as f:
    line_count = sum(1 for _ in f)
with open(outfile, "w", encoding="utf-8") as f:
    f.write(str(line_count))
```

Decide whether you want to run your job as a batch job or in interactive mode.

4.A - Run a batch job

In your *home* prepare the shell script, which specifies the required resources for PBS, names the job *my_counter*, loads Python module, calls Python script and passes the input file to it, copies the output from scratch to *home* directory and cleans up the scratch.

test.sh

```
#!/bin/bash
#PBS -l walltime=0:30:00
#PBS -l select=1:ncpus=1:mem=1gb:scratch_local=1gb
#PBS -N my_counter

module load python  # load the python module
# copy the files to the scratch directory
cp ${PBS_O_WORKDIR}/counter.py $SCRATCHDIR
cp ${PBS_O_WORKDIR}/myfile.txt $SCRATCHDIR
cd ${SCRATCHDIR}    # go to the scratchdir

python counter.py myfile.txt out_count.txt

# copy the output back to where qsub was called
cp out_count.txt ${PBS_O_WORKDIR}/
# apply a scratch automatic cleanup utility
clean_scratch
```

Submit the job to PBS and check its status (*R=running*).

```
$ qsub test.sh
> 18411451.pbs-m1.metacentrum.cz
$ qstat -xu myuser
> Job ID          Name      User    S
  ----------  ----------  ------  -
  18411451...  my_counter  myuser  R
```

When the job finishes, you can inspect the standard output and error files and the output file, which was created by the Python script.

```
$ ls
> counter.py    my_counter.e18411451
  my_counter.o18411451  out_count.txt   test.sh
$ cat out_count.txt
> 11
```

4.B - Run an interactive job

Request resources for the interactive job.

```
$ qsub -I -l walltime=0:30:00 \
   -l select=1:ncpus=1:mem=1gb:scratch_local=1gb
> qsub: waiting for job 18523820.pbs... to start
> qsub: job 18523820.pbs-m1.metacentrum.cz ready
```

Wait for the job to start and run each command as specified in the batch job.

```
$ module load python
$ cp ${PBS_O_WORKDIR}/counter.py $SCRATCHDIR
$ cp ${PBS_O_WORKDIR}/myfile.txt $SCRATCHDIR
$ cd ${SCRATCHDIR}
```

Call the Python script with the input file and let it count the lines.

```
$ python counter.py myfile.txt out_count.txt
$ cat out_count.txt
> 11
```

Copy the output file back to where *qsub* was run (your *home*) and clean the scratch.

```
$ cp out_count.txt ${PBS_O_WORKDIR}/
$ clean_scratch
```

You can follow this tutorial here, where the code snippets are copyable: