# Quick Start guide

This may seem to be an obvious consideration in cluster usage but, in order to use cluster's resources, you need to connect into afferent master via SSH client:
$ ssh <hades|lucifer|baal>

The first and main thing to do, after having logged on and to start using the computing nodes cluster, is to load the single required environment module namely **torque**. I cannot advise strongly enough to add the command into your own .bash_profile located into your home directory.

```
$ echo "module load torque" >> ~/.bash_profile
```

You can learn more about Environment-Modules here.

The easiest way to run jobs on the computing resources is to use the following script. This way, your jobs will be in compliance with the cluster usage policies and rules.

run.pbs

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -N <job-name>
#PBS -o <job-name>.out
#PBS -e <job-name>.err

#PBS -q <queue name>
#PBS -l nodes=<number of nodes>:ppn=<number of cores per nodes>
#PBS -l walltime=<walltime in format hh:mm:ss>
#PBS -A <credit account>

#PBS -m abe
#PBS -M <your email>

#PBS -l epilogue=/shared/scripts/ADMIN__epilogue-qsub.example

### FOR EVERYTHING BELOW, I ADVISE YOU TO MODIFY THE USER-part ONLY ###
WORKDIR="/"
NUM_NODES=$(cat $PBS_NODEFILE|uniq|wc -l)
if [ ! -n "$PBS_O_HOME" ] || [ ! -n "$PBS_JOBID" ]; then
        echo "At least one variable is needed but not defined. Please
touch your manager about."
        exit 1
else
        if [ $NUM_NODES -le 1 ]; then
                WORKDIR+="scratch/"
                export WORKDIR+=$(echo $PBS_O_HOME |sed
's#.*/\(home\|workdir\)/\(.*_team\)*.*#\2#g')"/$PBS_JOBID/"
                mkdir $WORKDIR
                rsync -ap $PBS_O_WORKDIR/ $WORKDIR/
        else
```

```
                # WORKDIR+="scratch-dfs/"
                export WORKDIR=$PBS_O_WORKDIR
        fi

        # if you need to check your job output during execution
(example: each hour) you can uncomment the following line
        # /shared/scripts/ADMIN__auto-rsync.example 3600 &
fi

echo "your current dir is: $PBS_O_WORKDIR"
echo "your workdir is: $WORKDIR"
echo "number of nodes: $NUM_NODES"
echo "number of cores: "$(cat $PBS_NODEFILE|wc -l)
echo "your execution environment: "$(cat $PBS_NODEFILE|uniq|while read
line; do printf "%s" "$line "; done)

cd $WORKDIR

# If you're using only one node, it's counterproductive to use IB
network for your MPI process communications
if [ $NUM_NODES -eq 1 ]; then
        export PSM_DEVICES=self,shm
        export OMPI_MCA_mtl=^psm
        export OMPI_MCA_btl=shm,self
else
# Since we are using a single IB card per node which can initiate only
up to a maximum of 16 PSM contexts
# we have to share PSM contexts between processes
# CIN is here the number of cores in node
        CIN=$(cat /proc/cpuinfo | grep -i processor | wc -l)
        if [ $(($CIN/16)) -ge 2 ]; then
                PPN=$(grep $HOSTNAME $PBS_NODEFILE|wc -l)
                if [ $CIN -eq 40 ]; then
                        export PSM_SHAREDCONTEXTS_MAX=$(($PPN/4))
                elif [ $CIN -eq 32 ]; then
                        export PSM_SHAREDCONTEXTS_MAX=$(($PPN/2))
                else
                        echo "This computing node is not supported by
this script"
                fi
                echo "PSM_SHAREDCONTEXTS_MAX defined to
$PSM_SHAREDCONTEXTS_MAX"
        else
            echo "no PSM_SHAREDCONTEXTS_MAX to define"
        fi
fi

function get_gpu-ids() {
    if [ $PBS_NUM_PPN -eq $(cat /proc/cpuinfo | grep -cE
"^processor.*:") ]; then
```

```
        echo "0,1" && return
    fi

    if [ -e /dev/cpuset/torque/$PBS_JOBID/cpus ]; then
        FILE="/dev/cpuset/torque/$PBS_JOBID/cpus"
    elif [ -e /dev/cpuset/torque/$PBS_JOBID/cpuset.cpus ]; then
        FILE="/dev/cpuset/torque/$PBS_JOBID/cpuset.cpus"
    else
        FILE=""
    fi

    if [ -e $FILE ]; then
        if [ $(cat $FILE | sed -r 's/^([0-9]).*$/\1/') -eq 0 ]; then
            echo "0" && return
        else
            echo "1" && return
        fi
    else
        echo "0,1" && return
    fi
}

gpus=$(get_gpu-ids)

## END-DO


##
## USER part
##

## Environment settings (environment module loadings, etc.)
# example: module load openmpi/gnu/4.1.1 torque

## your app calls
# example: mpirun simulation.x

## To well chain your jobs, with afterok directive to be sure the
current job will complete and OK before running the new one
# qsub -d `/shared/scripts/getWorkdir.sh` -W depend=afterok:$PBS_JOBID
<jobscript>

##
## END-USER part
##


# At the term of your job, you need to get back all produced data
synchronizing workdir folder with you starting job folder
# and delete the temporary one (workdir)
# A good practice is to reduce the file list you need to get back with
```

```
rsync
if [ $NUM_NODES -le 1 ]; then
    cd $PBS_O_WORKDIR
    rsync -ap $WORKDIR/ $PBS_O_WORKDIR/
    rm -rf $WORKDIR
fi
## END-DO
```

You just need to customize headers (job name, walltime, account name, number of nodes and cores-per-node, etc.) and add your own job calls into the USER-part block.

| **PBS parameter** | **Description** |
|---|---|
| -S <interpreter> | shell environment |
| -N <jobname> | job name as displayed in the queue |
| -o <filename> | redirect standard output to filename |
| -e <filename> | redirect standard error to filename |
| -l walltime=<hh:mm:ss> | walltime |
| -l nodes=<nodes>:ppn<cores_per_node>[:label] | requested resources |
| -A <project> | project account name |
| -l epilogue=<script.sh> | name of a script to be run at the end of the job |
| -m abe | send emails when job aborts (a), begins (b), ends (e) |
| -M <email> | email address |
| -q <queue> | queue name to manually specify the destination of the job |

As we have multiple graphics card models, if you want to run something on a specific card, you can do so by the label information of the *qsub -l* option. For example, to run something on an RTX-A6000 card, you can access it directly by typing *qsub -l nodes=1:ppn=40:rtxa6000*.

Once done, you just have to submit your job entering in terminal:

```
$ qsub run.pbs
12345.torque1.cluster.lbt
```

Your job has been submitted and its job-ID (here 12345.torque1.cluster.lbt) is returned by qsub.

Concerning Baal cluster, if you want to submit your job into **monop** or **test** queue, dont forget to add “-q <queue>” parameter to your qsub command (or in the PBS section of your script).

It's not a good idea to use OpenMPI library for communications on single node processes. That said, if you are using a binary that has been compiled with, you should disable PSM communication to improve the performance and stability, and by this way to avoid any network issue.

To do this, you just need to add these following lines before your MPI parallel job launcher command in your submission script:

```
[...]
export PSM_DEVICES="self,shm"
export OMPI_MCA_mtl=^psm
export OMPI_MCA_btl=shm,self
[...]
```

To go more away with performance improvement, you can read my performance tips.