Batch jobs execution on the DTU HPC cluster

1 General Introduction

HPC is a multi-user environment. In order to ensure an effective usage of the available resources some sort of management of the workload is necessary. This task is accomplished by a *Resource Manager* (for brevity RM), a program (or sometimes a set of programs that work together) that assigns the resources to the users according to the system current and expected load, the user needs, and a predefined assignment policy. This means that the user does not run his application directly, but instead "asks" the system to run his application, usually by means of a *job script*. The Resource Manager parses the script, and tries to optimize the usage of the available resources, scheduling the execution of the applications (jobs) at different times, and on different nodes/cores in the cluster.
There are many Resource Manager softwares available. On the HPC, a combination of the MOAB (scheduler) and Torque (resource manager) is used. However, most of the description in this guide is general. You can find some specific information at http://www.hpc.dtu.dk/?page_id=516.

1.1 The concept of a job script

The Resource Manager takes care of assigning to the user jobs the requested resources so that many jobs can be run simultaneously without interfering with each other, and scheduling the execution of the different requests. The RM needs some user-provided information to be able to do a good job, that's why the user is required to provide a job script (sometimes called also batch file) with the specification of:

- the resources requested, and the job constraints;
- specific queue specification (see later);
- all what makes a working execution environment.

By ***resources*** we mean the number of processors/cores/nodes, the amount of memory needed (total and/or per process), and eventually some specific features (special hardware, for example), and so on. Job constraints are typically time constraints, i.e. for how much time you are reserving the resources. Notice that there are limitations, both for the maximum number of cores, jobs, and time. These limitations change quite frequently, refer to http://www.hpc.dtu.dk/?page_id=384 for the current values.

The concept of ***queue*** is quite intuitive: since not all the applications/jobs are run immediately, they are ordered in a queue for a delayed execution. However, many different queues can be managed by the same scheduler. At DTU, for example, the RM manages different clusters, some of which are for all users and some only for selected groups, and there are also different queue for different kind of applications (serial vs parallel, short vs long, and others), with different restrictions (for example the **app** and **hpc** queue, see http://www.cc.dtu.dk/?page_id=384).

Then you have to provide a functioning ***execution environment***: remember that your application will be run when scheduled by the RM, so the application must be able to run without user intervention (this is called somewhere unattended execution). This means that the correct location (absolute or relative path) of the executables must be provided, together with all the necessary input files, and a correct specification of the environment (for example libraries, modules...).

Notice that the RM reserves the resources for your job (and run it when there are enough, avoiding conflicts with other users' jobs) **based on your request**. It is therefore important that the resources requested correspond to what really is needed by your application.

All these information must be written in a text file, following a simple syntax, that will be shown later. Once you have this text file (the job script, let us call it submit.sh), you must submit it by typing the command

```
qsub submit.sh
```

in a terminal.

NOTE: at present, there is no way to submit jobs other than from the command line

Then you can check the status of your submission issuing the command

```
qstat
```

The meaning of this command will be explained in section 3.2.

2. Preparing a job script

2.1 Preliminary step

The first thing when preparing a batch job is to take care that the application can run unattended, without your supervision. So you have to make sure that the executable are in the right place, that all the input files are specified, and that the output gets written where expected. Let us assume that everything is set up properly, and that you would run your application from the command line as follows:

```
myapplication.x < input.in > output.out
```

that means that you run the program `myapplication.x` reading the input from the file `input.in` and writing the output in the file `output.out`, and that these files are in the directory where you are when you issue the command.

NOTES:

- An important part of the execution environment is the location of files. You have to take care that the working directory, the directory where you want your files to be read and write for example, is correctly specified (see PBS_WORKDIR in the examples).
- It could be necessary to specify the full path of your application, to be sure that everything goes as expected.
- If your application needs some special environment (e.g. some special libraries), you have to make sure that they are loaded before the execution.

2.2 Learning through examples: A “not so basic” job script

A job script can be a simple bash script, without any specific request for the Resource Manager, like

```
#!/bin/sh
sleep 2
```

However, in this way the default values for all the values are used. It is therefore not necessary to specify a lot of options, and a job script can be very simple. However we present here a more complete job script. Most of the options can be omitted, but it is better to know that they exist.
A job script for submitting the same program to a queue, could look like that:

```
#!/bin/sh
# embedded options to qsub - start with #PBS
# -- Name of the job ---
#PBS -N My_Application
# -- specify queue --
#PBS -q hpc
# -- estimated wall clock time (execution time) --
#PBS -l walltime=1:00
# -- number of processors/cores/nodes --
#PBS -l nodes=1:ppn=4
# -- user email address --
#PBS -M xxx@fysik.dtu.dk
# -- mail notification --
#PBS -m abe
# -- run in the current working (submission) directory --
cd $PBS_O_WORKDIR

# here follow the commands you want to execute
myapplication.x < input.in > output.out
```

The file starts with the line

```
#!/bin/sh
```

that tells the system to use the `sh` shell for interpreting the subsequent lines.
The lines that start with **#** are interpreted as comments in the shell environment, and so the rest of the line is not executed.
The lines starting with **`#PBS`** are interpreted by the RM as lines that contain options for the resource manager.

Note: *all the requests for the Resource Manager, that is all the lines starting with #PBS,* ***MUST*** *appear* ***before the first shell command*** *(i.e. the first non-blank line not starting with #). All the lines starting with #PBS that appear after that are simply ignored by the RM, but no warning or error message will signal that.*

After the section with the options for the Resource Manager, you have to insert the command(s) for running your programs.

Some of the most important option are shown in the script

```
#PBS -N My_Application
```

Specify the name of your job, that is useful to easily check the status of your job.
Default: the name of the job-script file

```
#PBS -q hpc
```
Specify the queue you want your job to be run in. Notice that different queues have different defaults, and access to specific queue can be restricted to specific groups of users.
Default: the hpc queue

```
#PBS -l walltime=1:00
```
Specifies that you want your job to run AT MOST 1:00 hours.
Default: 48 hours

```
#PBS -l nodes=1:ppn=4
```
Ask to reserve 1 node and 4 core per each node. For example if you want to ask for a certain number of cores irrespective of the fact that they are on the same node or not, you can simply specify
```
#PBS -l procs=4
```
Default: one single node

```
#PBS -l vmem=8gb
```
Specifies the maximum amount of virtual memory used by all concurrent processes in the job.

```
#PBS -l pvmem=512mb
```
Specifies the maximum amount of virtual memory used by a single process in the job.
Default: 2gb

```
#PBS -M xxx@dtu.dk
```
User email address, so that the user can receive notifications at certain specific events. This can be any valid email address. Specifying an address is not enough to receive a notification. It just stores the email address in memory.
Default: none

```
#PBS -m abe
```
Specifies to write an email to the user address when the job begins(b), end (e) and abort (a). It is not necessary to set all of them.
Default: do not send

Then we encounter the first command that is executed by the shell environment:

```
cd $PBS_O_WORKDIR
```

This change the current directory (`cd` command) to the one set in the `PBS_O_WORKDIR` environment variable.
In fact, the execution of the `qsub` command will also set the value of a certain set of environment variable to be referred to during the execution.

`PBS_O_WORKDIR` is set to the absolute path of the directory from where you executed the `qsub` command. The command change the current directory to the working directory, so that your application can find all the needed files.

After this first command **no other PBS option will be considered**. You can now write the commands you use for running your application:

```
myapplication.x < input.in > output.out
```

This section can be a full script, or even better you could build an external script with all the instructions you need, for example a bash, a python, a perl script, and call it from the submit script. Notice that the application are managed in a modular approach on the HPC (see http://www.hpc.dtu.dk/?page_id=282). This means that the software packages you need are probably not already available on the node where your application is run. So you have to take care to explicitly load the modules you need, issuing the correct

```
module load <list of the modules the program need>
```

before the command that executes your job.

3. Commands for submitting and checking jobs

When running batch jobs, there is no direct way for the user to control the execution of its job directly, as on a normal machine. On the HPC cluster the users are not even allowed to connect to nodes other then the login node, so there is no possibility of using normal shell commands like top, to check the process status, or kill to stop a process. There are therefore special commands to do all these things, from the login node and or from one application node.

3.1 Submitting a job script

The most important command is the one for submitting your job script to the queue:

```
qsub your_job_script
```

`qsub` accepts also a lot of options, so that you could in principle avoid to use the script and run a single long command with all the option you need. This is possible, but not recommended.
For checking the available options, just read the man pages:

```
man qsub
```

Some of the options are implementation dependent, therefore may not all be available in the DTU installation. Once you submit the job, you can still track it. The job is given a job-id, that is shown as output when you submit the job.

3.2 Checking the job status

The command for checking the job status is `qstat`

```
qstat

Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
3597252.hpc-fe1            ...-COOETC_R9-16 s012345          3401:00: R hpc
3640759.hpc-fe1            xterm-linux      s012345                0 R app
```

You can see all your running jobs (User column), with their job-id (first column), the name you assigned them (-N option of PBS), the current runtime, the status (column S), and the name of the queue.
Some common status letters are :

Q job is queued, eligible to run or routed
R job is running
C Job is completed after having run
H Job is held

`qstat` can be used to extract a lot more information about your job. For example too show also the nodes and cores where the job is running:
`qstat -n`

```
hpc-fe1:
                                                                  Req'd    Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory   Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
4242173.hpc-fe1         andbor      hpc      LDermo_UHF_1      16039     1      8    --   72:00:00 R  00:41:11
   n-62-13-2/7+n-62-13-2/6+n-62-13-2/5+n-62-13-2/4+n-62-13-2/3+n-62-13-2/2
   +n-62-13-2/1+n-62-13-2/0
```

For help and additional explanation:
`man qstat`

3.3 Waiting time estimate

If your job is queued (Q), and you want to have an idea about when it will run, pick its job-id from the qstat output, and type

```
showstart <your-job-id>

 showstart 4242173
job 4242173 requires 8 procs for 3:00:00:00

Estimated Rsv based start in              -00:47:16 on Fri Apr 18 11:50:00
Estimated Rsv based completion in        2:23:12:44 on Mon Apr 21 11:50:00

Best Partition: hpc-fe1
```

This will show the estimated start, and stop of your job execution. It is just an estimate, based on the requested resources of your job, and the current schedule of the queue. It also shows you additional information (and work also for already started jobs :)).

3.4 Getting more information on the running job

To get some more information about your jobs, use the command `checkjob` followed by the job-id, eventually with the option -v, if you want a verbose output:

```
checkjob <your-job-id>
```

```
 checkjob 4242173

job 4242173

AName: LDermo_UHF_1
State: Running
Creds:  user:andbor  group:fys  class:hpc  qos:hpc_longhours
WallTime:   00:50:33 of 3:00:00:00
SubmitTime: Fri Apr 18 11:49:59
  (Time Queued  Total: 00:00:01  Eligible: 00:00:01)

StartTime: Fri Apr 18 11:50:00
Total Requested Tasks: 8

Req[0]  TaskCount: 8  Partition: hpc-fe1
Dedicated Resources Per Task: PROCS: 1  MEM: 2048M
NodeSet=ONEOF:FEATURE:hpc_node:fullhpc_node:ibm_fullhpc_node

Allocated Nodes:
[n-62-13-2:8]


Notification Events: JobStart,JobEnd  Notification Address:
andbor@dtu.dk

IWD:            /zhome/10/6/81684/Dermo/LDermo
StartCount:     1
Flags:          RESTARTABLE
Attr:           checkpoint
StartPriority:  120
Reservation '4242173' (-00:50:55 -> 2:23:09:05  Duration: 3:00:00:00)
```

```
checkjob -v 4242173
job 4242173 (RM job '4242173.hpc-fe1')

AName: LDermo_UHF_1
State: Running
```

```
Creds:  user:andbor  group:fys  class:hpc  qos:hpc_longhours
WallTime:   00:51:43 of 3:00:00:00
SubmitTime: Fri Apr 18 11:49:59
  (Time Queued  Total: 00:00:01  Eligible: 00:00:01)

StartTime: Fri Apr 18 11:50:00
Total Requested Tasks: 8

Req[0]  TaskCount: 8  Partition: hpc-fe1
Dedicated Resources Per Task: PROCS: 1  MEM: 2048M
Utilized Resources Per Task:  PROCS: 0.99  MEM: 43M  SWAP: 4224M
Avg Util Resources Per Task:  PROCS: 0.99
Max Util Resources Per Task:  PROCS: 0.99  MEM: 43M  SWAP: 4224M
Average Utilized Memory: 40.96 MB
Average Utilized Procs: 7.76
NodeSet=ONEOF:FEATURE:hpc_node:fullhpc_node:ibm_fullhpc_node
TasksPerNode: 8  NodeCount:  1

Allocated Nodes:
[n-62-13-2:8]


Notification Events: JobStart,JobEnd  Notification Address:
andbor@dtu.dk
Task Distribution: n-62-13-2,n-62-13-2,n-62-13-2,n-62-13-2,n-62-13-
2,n-62-13-2,n-62-13-2,n-62-13-2

IWD:            /zhome/10/6/81684/Dermo/LDermo
UMask:          0000
OutputFile:     n-62-12-
1:/zhome/10/6/81684/Dermo/LDermo/LDermo_UHF_1.o4242173
ErrorFile:      n-62-12-
1:/zhome/10/6/81684/Dermo/LDermo/LDermo_UHF_1.e4242173
StartCount:     1
Partition List: [ALL]
SrcRM:          hpc-fe1  DstRM: hpc-fe1  DstRMJID: 4242173.hpc-fe1
Submit Args:    LDermo_Gas_HF_1.job
Flags:          RESTARTABLE
Attr:           checkpoint
StartPriority:  120
PE:             8.00
Reservation '4242173' (-00:52:29 -> 2:23:07:31  Duration: 3:00:00:00)
```

You can see for example the **walltime** of your job, and how much time has passed;
How many processors you have actually asked for (**Total Requested Tasks**);
The resources used (**Averaged Utilized Menory** and **Procs**).
These information can be useful to check whether your jobs script was correct.

3.5 Deleting a job

It is sometimes necessary to **remove a job from a queue**. This can be done in any stage, i.e. when the program is still waiting to be run (state Q), or during the run (state R). Just get the job id with `qstat`, and type

```
qdel <your-job-id>
```

There are other commands and option that can be useful, but these are more than enough to start using the HPC batch system.

3.6 Checking the status of the queue

It can be useful sometimes to have an overview of the status of a whole queue. This can be done with the `showq` command:

```
showq

active jobs------------------------
JOBID              USERNAME      STATE   PROCS   REMAINING            STARTTIME

4243341               s012345    Running     1    00:10:12  Sat Apr 19 13:53:53
4203911               s012345    Running    96     2:45:20  Sat Apr 19 10:44:01
4243245               s012346    Running     1    14:14:05  Sat Apr 19 12:12:46
4243246               s012346    Running     1    14:14:10  Sat Apr 19 12:12:51
4243247               s012355    Running     1    14:14:14  Sat Apr 19 12:12:55
4243248               s012355    Running     1    14:14:19  Sat Apr 19 12:13:00
. . .
```

It can
Or restricted to specific queue (also called class):

```
showq -w class=compute

active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME

4243343              s012345    Running     1    00:08:52  Sat Apr 19 13:59:30
4243245              s012345    Running     1    14:12:08  Sat Apr 19 12:12:46
4243246              s012355    Running     1    14:12:13  Sat Apr 19 12:12:51
4243247              s012355    Running     1    14:12:17  Sat Apr 19 12:12:55
. . .
```

Or user:

```
showq -w user=<userid>
active jobs------------------------
JOBID              USERNAME      STATE PROCS   REMAINING            STARTTIME

4242173              userid    Running     8  1:21:47:54  Fri Apr 18 11:50:00
4242200              userid    Running     8  1:21:55:58  Fri Apr 18 11:58:04

2 active jobs           16 of 2840 processors in use by local jobs (0.56%)
                        128 of 231 nodes active      (55.41%)
```

```
eligible jobs----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT           QUEUETIME

0 eligible jobs

blocked jobs-----------------------
JOBID              USERNAME      STATE PROCS     WCLIMIT           QUEUETIME

0 blocked jobs

Total jobs:  2
```

3.7 Checking the status of the cluster/nodes

There are also a couple of commands that can be used to check the status of the cluster, of a specific queue/class. The command is `classstat` (three “s”):

```
classstat

queue                 total  used avail
----------------------------------------
cmp_interactive          16     2    14
compile                  16    16     0
compute                 416   212   204
computebigmem            96    38    58
dyna                    144    64    80
fotonano                636   391   245
hpc                    1324   716   608
hpc_intact               64    14    50
ibmtest                  80     0    80
k40_interactive          12     1    11
mek                     120    74    46
mic_interactive          32     0    32
topopt                  300   256    44
visual                   48     4    44
~
```

For each queue the total number of cores used and available are shown. You can always see how busy is the cluster before you submit.

Note: This command is available ONLY on the front-end node.

More details on the status of the single nodes, can be obtained through `nodestat` followed by the name of the queue

```
 nodestat compute

Name                     State   Procs    Load
n-62-18-20                Idle   16:16    0.24
n-62-18-21                Idle   16:16    0.45
n-62-18-22                Idle   16:16    0.38
```

```
n-62-18-23                Idle   16:16    0.40
n-62-18-24                Idle   16:16    0.34
n-62-18-25                Idle   16:16    0.28
n-62-18-26                Idle   16:16    0.41
n-62-18-27                Idle   16:16    0.42
n-62-18-28                Idle   16:16    0.41
n-62-18-29                Idle   16:16    0.45
n-62-18-30                Idle   14:16    2.14
n-62-18-31                Idle    9:16    7.29
n-62-18-32             Running    7:16    8.90
n-62-18-33             Running    6:16   10.31
n-62-18-34             Running    2:16   14.41
n-62-18-36                Busy    0:16   12.45
n-62-18-37                Busy    0:16   12.78
n-62-18-38                Busy    0:16   13.18
n-62-18-39                Busy    0:16   13.52
n-62-18-40                Busy    0:16   12.88
n-62-18-41             Running    1:16   15.19
n-62-18-42                Busy    0:16   16.38
~
```

The first column reports the node-name, and then the status, the number of free cores and the total number of cores on the specific node, separated by a “:”, and the node load.

These are the basic instructions and tools to use the HPC clusters at DTU. You can always get support writing an email to
support@cc.dtu.dk
and find other useful information on the website http://www.hpc.dtu.dk.