Batch system on the hpc-bw cluster

20.02.2009

Contents

1 Preparations 2

2 The batch system 2
   2.1 Basic interaction with the batch system 2
   2.2 Submitting basic/serial jobs 3
   2.3 Common job submission parameters and batch variables 4
      2.3.1 Job submission parameters 4
      2.3.2 Batch job variables 4
      2.3.3 Advanced resource parameters 5
      2.3.4 Advanced serial jobs 5
   2.4 Parallel Jobs 6
      2.4.1 The PBS nodefile 6
   2.5 Querying the status of a batch job 7
1 Preparations

If you want to start a parallel job or use some scientific software from /opt/bwgrid, you have to use the module system (man module). Modules allow you to change your shell environment without any hassle.

Before you can use the module system, there may be a few preparative steps you have to take, if you have an older account or did use the system before.

First check, if you have already /bin/bash as a login shell. If your login shell is bash and have a new account or haven’t used your account in the past, everything should work already.

If you still have /bin/tcsh as a login shell, then you can change your login shell at the following URL to /bin/bash:

https://www.zdv.uni-tuebingen.de/util/chsh.php

After changing your login shell (echo $SHELL) you should log out and in again. Additionally you have to copy bash_profile and bashrc from /zdv-system/customer/ to ~/.bash_profile and ~/.bashrc rsp.

If the files are missing then copy the templates from /etc/skel to your home directory

Now you are ready to test the module system. Log out of the system and log in again.

You should be able to use module avail to show a list of available modules and load modules with module load like module load openmpi/1.2.8.

With module list you’ll get a listing of all loaded modules and with module rm you can remove modules from your environment, like module rm openmpi/1.2.8.

Finally, if you want to load some modules automatically every time you log in, you can use the commands module initadd, module initlist, module initrm, which can manipulate the module load line in your .bashrc for you. For example module initadd intel adds intel to the module load line in your .bashrc so that the intel compiler can be used

For more further information about the module system, you can consult the module man page (man module).

2 The batch system

2.1 Basic interaction with the batch system

The batch system running on the hpc-bw cluster is torque/maui. Torque is a PBS implementation and understands the usual PBS syntax. man pbs gives an overview of the available PBS commands. The most common commands are listed below:

qsub : submits a batch job
qstat : shows status of a batch job
qalter : alters job’s attributes
qdel : deletes a job
qsig : sends a signal to a job
For additional commands, please consult the PBS man page (man pbs).

2.2 Submitting basic/serial jobs

Jobs can be submitted with the qsub command. To submit a job it is possible to either use qsub jobscript or to pipe a simple script or command into the qsub binary like:

```
    echo "/bin/bash ~/my_script.sh | qsub
```

If everything works right, qsub prints out a job identifier and exits. After that the status of the job can be queried with the qstat command.

While it is possible to submit a job as easy as qsub jobscript, in most cases it is necessary to specify some parameters, like the resources your job needs or the email address which should get the notification, when your job completes.

There are two ways of specifying parameters with the qsub command. You can either use the command line like qsub -l nodes=1:ppn=8 or you can put the qsub parameters into the script. The Syntax for in-script parameters is basically the same as on the command line but the parameters have to be prefixed by #PBS. The previous example would then look like: #PBS -l nodes=1:ppn=8. Listing 1 shows a simple jobscript that requests a node with 8 CPUs for 10 minutes and starts a program.

```
#!/bin/bash

#PBS -l nodes=1:ppn=8
#PBS -l walltime=0:10:0

cd $PBS_O_WORKDIR
./myprog my_param
```

Listing 1: simple script
2.3 Common job submission parameters and batch variables

2.3.1 Job submission parameters

- **-d path**: Selects the working directory. If not selected, the default working directory is your current working directory.

- **-q destination**: The destination queue. Normally you should not specify this parameter, because the batch system automatically selects the execution queue that matches your resource selection.

- **-I**: Requests an interactive job. This is basically the same as to ssh to a random free node.

- **-l resource_list**: This is the most important parameter. It specifies the resources your job needs, like nodes, CPUs, walltime, ...

- **-M list**: With this parameter you can set a list of email addresses that receive mail from PBS.

- **-m mail_options**: This specifies, in which case an email should be send. Whereas a=abort, b=begin and e=end. So if you want an email when the job starts, ends and gets killed from the batch system, then use -m abe.

- **-N**: Sets a job name. Normally the job is named after your script.

- **-v variable_list**: Here you can specify variables that the batch system sets in your target environment. For example -v var1="test.sh",var2=23.5,var3 sets var1 and var2 to the specified values and sets var3 with the value of the current environment. Of course this implies that var3 already exists in the current environment

- **-V**: Makes all variables from the current environment available in the batch environment.

For a complete description of the qsub parameters, please refer to the qsub man page.

2.3.2 Batch job variables

Besides the usual variables like HOME, LANG, LOGNAME, SHELL, .... PBS sets a few additional variables inside your environment, which you might need for your scripts:

- **PBS_O_HOST**: Contains the name of the host, where qsub was started.
- **PBS_O_WORKDIR**: The working directory, from which qsub was started.
- **PBS_O_QUEUE**: Name of the submit queue.
- **PBS_JOBID**: The jobid
- **PBS_JOBNAME**: The jobname. You can set the jobname with the -N parameter. Otherwise it is the name of the submit script.
- **PBS_NODEFILE**: The nodefile. This file contains a list of all nodes the job has allocated, with an entry for every CPU. You will find a description of the file format in 2.4.1.
- **PBS_QUEUE**: Name of the execution queue

You’ll find a complete list in the qsub man page. You can also get a complete list of the variables when you execute echo set | qsub which gives you a list of all variables defined inside the batch environment in the output file. You can also define variables or copy the content of already defined variables with the -v Parameter of qsub and with the -V parameter you can copy your whole shell environment.
2.3.3 Advanced resource parameters

It makes a lot sense to specify the resources a job needs as accurate as possible. In some cases the batch system has to know about some required resources, like memory or disk space, because they can’t be provided on every node. Also if the resources are properly specified, the batch system can make a better choice of execution queues. For example if you want your job to run more than 8 hours, you have to specify a walltime, like walltime=32:0:0 for 32 hours walltime.

Therefore you should consider a few things before you start:

- You should only reserve the resources you need. If you need four cores then reserve only 4 cores (-l nodes=1:ppn=4).
- If you reserve only a partial node, you should tell the batch system how much memory you want.
- If you reserve resources from the batch system, do not overcommit. You shouldn’t start more processes, than you have reserved.

Following are a few examples how to use the resource system:

- `nodes=1:ppn=8+1:ppn=4` : Requests 12 cores and 1GB RAM per process. One node with 8 cores and one node with 4 cores.
- `nodes=2:ppn=8+1:ppn=4` : Requests 20 cores on 3 nodes with a 8,8,4 split.

For further information about available PBS resources you can consult the `pbs_resources` (man pbs_resources) man page.

2.3.4 Advanced serial jobs

PBS executes the job script only on the first node. If you want to use more than one node you have to execute your jobs on the other nodes manually. While you can do this with `rsh` or `ssh` and the nodefile (see 2.4.1), there is a more elegant solution:

For multi node serial jobs, `torque` already bundles a program called `pbsdsh`. With `pbsdsh` the batch system can execute your program on your allocated nodes. If you execute a script with `pbsdsh` the PBS_NODENUM and PBS_VNODENUM variables inside the environment of the executed script contain the number of the node (counted from 0) and the number of the process (counted from 0).

```
#!/bin/bash
echo PROCESSES=$(cat $PBS_NODEFILE | wc -l)
/usr/local/bin/pbsdsh $PBS_O_WORKDIR/ser_job.sh
```

Listing 2: adv_serial_job.sh

```
#!/bin/bash
echo HOSTNAME=$(hostname) NODENUM=$PBS_NODENUM VNODENUM=$PBS_VNODENUM
```

Listing 3: ser_job.sh

```
$ qsub -l nodes=2:ppn=3 adv_serial_job.sh
```

19342.icmu03
For different workloads, `pbsdsh` has a few parameters that might be interesting, like `pbsdsh -u` starts only one process per node. For the rest of the `pbsdsh` parameters you should consult the `pbsdsh` man page (man pbsdsh).

Although the `pbsdsh` approach is very simple and flexible, there is one thing to consider. You shouldn’t start node spanning serial job, if it isn’t absolutely necessary. If you want to start completely serial independent programs on multiple nodes you should just reserve one node (`-l nodes=1:ppn8`) and submit multiple jobs instead.

### 2.4 Parallel Jobs

For parallel jobs (MPI jobs), the base installation already provides `openmpi`, `mvapich`, and `mvapich2`. While these options are present, not all of them are completely tested. If you have the liberty to choose, you should use `openmpi`. Listing 5 shows a very basic mpi script.

```bash
#!/bin/bash

#PBS -l nodes=4:ppn=8, walltime=1:30:00
#PBS -M my_email_address
#PBS -m abe

module load openmpi/1.2.8

#openmpi 1.2.8 uses the TM interface to interface with torque. Therefore it is not necessary to provide a hostfile, and due to a bug in openmpi you will get an error if you try.

mpirun myprog arg1 arg2 arg3
```

Listing 5: simple mpi script

#### 2.4.1 The PBS nodefile

While `openmpi/1.2.8` supports the torques TM interface, and therefore a nodefile is not necessary, you might still need a nodefile for other MPI implementations. Torque generates a nodefile which contains a line with the node name for every allocated CPU on every target node.
An example nodefile for 2 nodes with 3 CPUs (qsub -l nodes=2:ppn=3) would therefore look like:

```
  n020101
  n020101
  n020101
  n030102
  n030102
  n030102
```

Listing 6: Sample nodefile

If you need another format you have to parse the file and produce a new one. Of course openmpi is perfectly happy with the standard nodefile and due to a bug in current implementations, it doesn’t even accept a nodefile anyhow, if it detects the PBS environment.

### 2.5 Querying the status of a batch job

To query the status of your job there are a few possibilities.

First `qstat` gives you an overview over your jobs and prints their status. With `qstat -f jobid`, you’ll get an overview over all the parameters of your job, inclusive the execution hosts when your job is currently running.

Additional information about your job, the job can be queried from maui. While torque is the resource manager, the actual scheduling is the job of maui. Torque only starts and controls your jobs, but the decision, if and when to run your jobs, lies in the responsibility of maui.

With `module load system/maui` you can load the maui module which makes the maui specific commands available.

- `showq`: lists all queued and running jobs.
- `showstart`: tells you when a job may start.
- `showres`: tells you how long the reservation for a job lasts.
- `checkjob`: shows you the status of a job from the maui side and may tell you, why a job hasn’t been started yet.