如何使用 pbs 提交作业

1 串行任务提交

用户通过qsub命令来向系统提交任务，有两种方式提交：脚本方式和命令行方式。（一般情况下，不允许root用户使用qsub命令提交作业）

1.1 脚本方式提交

用户将需要执行的程序或命令写入脚本中，再加入一些必要或者可选的语句，就可以通过脚本方式提交。脚本提交比较方便，用户可以用最简单的文字编辑器（例如 vi）编写一个脚本，然后使用 qsub 命令提交该脚本，pbs 会按照脚本内容执行相应的任务。脚本提交的不足之处就是需要用户执行任务前编写脚本。

脚本方式提交任务的一般格式为：**qsub [script_name]**，script_name 是任务脚本的名称。

下面介绍 pbs 任务脚本的编写方法。

pbs 任务脚本包括三部分：

- shell 说明语句：用来说明用户使用的是哪种 shell，例如 ***#!/bin/sh*** 说明用户需要使用 Bourne shell，如果用户没有指定，则默认为 Bourne shell
- pbs 指示语句：pbs 指示语句是用户用来请求任务运行时所需的资源或设置任务的一些属性的。以***#PBS*** 开头，如***#PBS –N taskname*** 是用来设置任务名称的。pbs 指示语句是可选的，用户可以不用关心它如何写，系统会自动设置。
  - pbs 指示句的一个主要作用是请求任务执行时所需要的系统资源，如 cpu 数目，存储容量，运行时间，运行优先级等等。以脚本方式提交的任务的资源请求是通过 pbs 指示语句，一般格式为 **#PBS –l [选项=] [选项对应的值]**，如 **#PBS –l ncpus=5** 表示请求 5 个 cpu 为之服务。系统资源如下表所示

资源	描述	举例
arch	所需要的系统结构，只用在资源块中	-l arch=linux
cput	任务的所有进程拥有的最大 cpu 执行时间	-l cput=1:00:00
file	任务能够创建的文件的大小	-l file=45mb
host	指定执行主机的名称	-l nodes=X:host 分配 X 个主机名称中含有 host 的执行节点
mem	任务的所有进程能够分配到的最大物理内存数	-l mem=100mb
ncpus	请求的 cpu 数	-l ncups=5
nice	任务运行时的 nice 优先级值	-l nice=3
pcput	任务的任何一个进程拥有的最大 cpu 执行时间	-l pcput=1:00:00
pmem	任务的任何一个进程能够分配到的最大物理内存数	-l pmem=45mb
pvmem	任务的任何一个进程能够使用的虚拟内存的最大数	-l pvmem=100mb
vmem	任务的所有并发进程能够使用的最大虚存数	-l vmem=100mb
walltime	任务可以处于运行态的最大 wall-clock 时间	-l walltime=1:00:00
custom resources	用户自定义资源	

注：表中最常用的几种资源是：walltime, ncpus, mem, host。其它可以不用太关心。

- 资源请求的两种方式：

  a、资源块方式：资源块是将任务所需的资源作为一个整体，这个整体中说明了所需要的各种资源的数目。其格式为：**-l select=[N:]chunk[+[N:]chunk…]**，如**qsub –l select=2:ncpus=3:mem=4gb:arch=linux** ，select=2表示需要2个这样的资源块，一个资源块包括3个cpu，4gb的内存，系统结构要求是linux，即总共需要6个cpu，8gb的内存。再如：

  -l select=2:ncpus=1:mem=10GB+3:ncpus=2:mem=8GB:arch=solaris

  注意中间的+号，是两个资源块的分隔符

  b、请求全任务(job-wide)资源：格式为 **–l keyword=value[,keyword=value …]**

  如：**qsub -l ncpus=4,mem=123mb,arch=linux**

➢ 任务（程序或命令）：可以是用户程序（如C程序），也可以是系统命令

下面是一个完整的任务脚本例子，脚本名为 mytask：

```
1)  #!/bin/sh                                  //指明所用的 shell
2)  #PBS –N mytask                             //设置任务名称
3)  #PBS -l walltime=1:00:00                   //请求任务执行时间
4)  #PBS -l select=ncpus=4:mem=400mb           //请求任务执行所需资源
5)  #PBS -j oe                                 //设置相关属性(文件合并)
6)  date /t                                    //系统命令（打印日期时间）
7)  .\my_application                           //所要执行的任务(当前目录名为
                                               // my_application的任务)
8)  date /t                                    // 系统命令
```

第 1 行说明所用的 shell；第 2 至 5 行是 pbs 指示语句，设置了任务的一些属性，并请求了资源；第 6 至 8 行是要执行的命令及任务。

编写完脚本后，使用 qsub 命令提交脚本，在 shell 下输入：

qsub mytask

后敲回车。

系统会输出一个**任务标识符：sequence-number.servername**，如 220.cnode01 。sequence-number 是任务编号，后面需要用这个号查看任务的执行状态，servername 是 pbs server 的名称。

1.2 命令行方式提交

命令行方式提交不用写脚本，用户可以直接从命令行输入。输入的内容基本上和在脚本中输入的相同。其基本格式如下：

```
qsub <ret>          //输入qsub命令后回车
[directives]        //pbs指示语句（以#PBS为前缀）
[tasks]             //任务或命令
ctrl-D              //结束输入，提交任务
```

对于上面用脚本方式提交的任务，用命令行方式提交的格式如下：

qsub

```
#PBS –N mytask
#PBS –l walltime=1:00:00
#PBS –l select=ncpus=4:mem=400mb
#PBS -j oe
date /t
./my_application
date /t
```

qsub 命令的常用选项有：

选项	取值	功能
-a *date_time*	年月日时分秒	指定任务可以开始执行的时间（精确到秒）
-e *path*	路径名	指定错误报告文件的输出路径
-h	无	使任务暂时阻塞，推迟执行
-J *X-Y*	X,Y 是向量作业下标	执行一个作业向量
-j *join*	oe:错误报告文件与输出文件合并成一个输出文件 eo:错误报告文件和输出文件合并成一个错误报告文件	将错误报告文件与输出文件合并
-l *resources_list*	资源列表	资源请求列表
-N *name*	任务名称	指定任务名称
-o *path*	路径名	指定输出文件路径
-p *priority*	-1024 到+1023 之间的一个值	指定任务优先级（依赖于操作系统的调度策略）
-q *destination*	队列名称或执行节点名称	指定任务执行的队列或者执行节点
-r *value*	布尔值 n:不可重新执行 y:可重新执行	说明任务是否可以重新执行

注：以命令行方式提交任务时，这些命令选项都要以 pbs 指示语句的格式给出，即选项之前要加前缀#PBS。

2 任务状态查看

任务提交后，用户如果要知道任务的当前运行状态，可以通过 qstat 命令查询。qstat 命令的常用选项有：

- 无选项：当 qstat 命令不带任何选项时，以默认方式显示任务信息，例如

```
[yaliang@cnode03 mpi]$ qstat
Job id              Name             User            Time Use S Queue
----------------    ----------------  ----------------  ---------- -  -----
569.cnode01         test             yaliang          0        R  small
```

其中，Job id 是任务的标识符，Name 是任务名称，User 是任务所有者，Time Use 是 CPU 使用时间，S 是任务当前状态（本例中 R 表示正在运行），Queue 表示任务所在

队列。任务的状态列表如下：

状态(S)	描述
B	只用于任务向量，表示任务向量已经开始执行
E	任务在运行后退出
H	任务被服务器或用户或者管理员阻塞
Q	任务正在排队中，等待被调度运行
R	任务正在运行
S	任务被服务器挂起，由于一个更高优先级的任务需要当前任务的资源
T	任务被转移到其它执行节点了
U	由于服务器繁忙，任务被挂起
W	任务在等待它所请求的执行时间的到来(qsub -a)
X	只用于子任务，表示子任务完成

再举一例，使用**-a**选项指定任务开始执行时间：

[yaliang@cnode03 mpi]$ qsub -a 2102 test

571.cnode01

[yaliang@cnode03 mpi]$ qstat

Job id Name User Time Use S Queue

---------- ---------- ---------- ---------- --- --------

570.cnode01 test yaliang 0 W default

此时任务状态(S)显示 W，表示任务正在等待执行时间的到来

- **-a：qstat –a** 列举出当前执行节点上所有任务的情况，比无选项时列举的项目更多，例如：

```
[yaliang@cnode03 mpi]$ qstat -a

cnode01:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
572.cnode01     yaliang  default  test         --    1  --    --  01:00 W   --
```

--其中常用的几项是：NDS 表示请求的节点数目，Req’d Memory 表示请求的内存大小，Req’d Time 表示请求的 cpu 时间，S 表示任务的当前状态，Elap Time 表示任务已经运行的时间。由于本例中没有请求节点数目、内存大小以及 cpu 时间，所以均没有显示，而任务当前处于等待状态，所以 Elap Time 也没有显示。

- **-f：qstat –f sequence-number** 可以查询任务标识号为 sequence-number 的任务状态，这个任务标识号是在任务提交时系统自动赋予的。例如：

[yaliang@cnode03 mpi]$ qstat -f 572

Job Id: 572.cnode01

Job_Name = test

Job_Owner = yaliang@cnode03

job_state = W

queue = default

server = cnode01

Checkpoint = u

ctime = Fri Dec 5 21:06:34 2008
Error_Path = cnode03:/home/yaliang/mpi/test.e572
Execution_Time = Sat Dec 6 21:02:00 2008
Hold_Types = n
Join_Path = n
Keep_Files = n
Mail_Points = bae
Mail_Users = ltq.yaliang@stu.xjtu.edu.cn
mtime = Fri Dec 5 21:06:54 2008
Output_Path = cnode03:/home/yaliang/mpi/test.o572
Priority = 0
qtime = Fri Dec 5 21:06:34 2008
Rerunable = True
Resource_List.cput = 01:00:00
Resource_List.nodect = 1
Resource_List.nodes = 1
Variable_List = PBS_O_HOME=/home/yaliang,PBS_O_LANG=en_US.UTF-8,
PBS_O_LOGNAME=yaliang,
PBS_O_PATH=/usr/local/Fluent.Inc/bin:/usr/local/matlab704/bin:/opt/intel/fce/10.0.023/bin:/opt/intel/idbe/10.0.023/bin:/opt/intel/cce/10.0.023/bin:/usr/local/mpich2-1.0.7/bin:/usr/java/jdk1.5.0_03/bin:/usr/java/jdk1.5.0_03/jre/bin:/usr/local/TSCMSS:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:.:/home/yaliang/bin,
PBS_O_MAIL=/var/spool/mail/yaliang,PBS_O_SHELL=/bin/bash,
PBS_O_HOST=cnode03,PBS_O_WORKDIR=/home/yaliang/mpi,PBS_O_QUEUE=default
comment = Not Running: Queue not an execution queue.

- **-u：qstat –u user1** 可以查询用户 user1 提交的所有任务的状态。例如：

```
[yaliang@cnode03 mpi]$ qstat -u yaliang

cnode01:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
572.cnode01     yaliang  default  test         --    1  --    --  01:00 W   --
```

注：需要注意的一点是，当任务已经完成后，qstat 命令就不能再查询任务的状态了。这时可以使用“**tracejob sequence-number**”命令来查看任务的历史信息。
任务的错误报告以及输出结果都保存到指定的路径了，如果没有指定，则保存在用户根目录下面，即**/home/user1/**。
错误报告文件的文件名格式为：**taskname.esequence-number**，例如：mytask.e239
输出文件的文件名格式为：**taskname.osequence-number**，例如：mytask.o239
在这两个文件中可以查看你的程序的执行结果或是错误执行结果（如果有错误的话）。

3：任务的删除

可以删除一个正在执行或是处于等待队列中的任务/作业。删除任务使用 qdel 命令。
格式如下：

qdel sequence-number　　// sequence-number 是任务编号，可以使用 qstat 查询

使用 pbs 来进行作业提交的详细过程描述

示例一：使用 pbs 来提交 mpi 并行程序

比如我要提交一个计算 π 的 mpi 程序。程序的源文件为 mpi.c
我写一个 pbs 脚本 test 如下：

```
#!/bin/sh
#PBS -N test
#PBS -M ltq.yaliang@stu.xjtu.edu.cn
#PBS -m bae
cd /home/yaliang/mpi      //程序所在的路径
mpdboot -n 10 -f host
mpicc -o mpi mpi.c
mpiexec -machinefile host -n 2 ./mpi
```

其中 host 文件内容如下：（对于经常进行 mpi 并行计算的专业人员来说，这里的介绍是多余的，你们比我对 mpi 的了解要深刻的多^_^）

```
cnode01
cnode02
cnode03
cnode04
cnode05
cnode06
cnode07
cnode08
cnode09
cnode10
```

脚本写完之后就可以提交了：

```
qsub test
```

任务执行完后，就可以到相应的目录下查看作业的执行结果。test.osequence-number 为执行结果文件，test.esequence-number 为错误输出文件。

示例二：使用 pbs 来提交需要图形界面进行交互的作业，以图形显示 fluent 界面为例：（此时需要在你的 windows 下打开 exceed 客户端）

你需要写一个脚本 test 如下

```
export DISPLAY=your ip:0
fluent
```

注：你必须使用一个外部 ip，即不是 192.168.x.x 的 ip，命令中的 your ip 是你的 ip 地址，注意 ip 后面的:0 一定要有，export 后面有一个空格。
然后 qsub test 提交，就可以看到 fluent 图形操作界面了。
注意：所有非图形界面的程序都可以使用 pbs 进行作业的提交。
目前 matlab 还不能使用 pbs 调出图形界面！

希望大家能够积极配合我们的工作，有问题及时向反馈，谢谢！