作业调度系统的使用

李会民

hmli@ustc.edu.cn, lihm@qibebt.ac.cn

中国科学院青岛生物能源与过程研究所 超级计算中心

2009 年 12 月

培训内容

作业管理系统 LSF 的使用

HP Superdome 服务器、HP RX2600集群和联想集群利用 Platform 公司的 LSF 进行资源和作业管理，所有需要运行的作业均必须通过作业提交命令 bsub 提交，提交后可利用相关命令查询作业状态等。为了利用 bsub 提交作业，需在 bsub 中指定各选项和要执行的程序。注意：

- 不要在登录节点直接运行（编译除外）作业，以免影响其余用户的正常使用
- 如果不通过作业调度系统直接在计算节点上运行将会被监护进程直接杀掉

提交作业：bsub

用户需要利用 bsub 提交作业，其基本格式为：

bsub [options] command [arguments]

- options 设置队列、CPU 核数等 LSF 的选项
- arguments 设置作业的可执行程序本身所需要的参数

提交到特定队列：bsub -q

利用 -q 选项可以指定提交到哪个队列
提交到 normal 队列运行串行程序 executable1：
bsub -q normal executable1 或 bsub executable1
如果提交成功，将显示类似下面的输出：

```
Job <79722> is submitted to default queue <normal>.
```

其中 79722 为此作业的作业号，以后可利用此作业号来进行查询及终止等操作。

指明所需要的 CPU 核数：bsub -n

利用 -n 选项指定可所需要的 CPU 核数（一般来说核数和进程数一致）

- 指定利用八个核（由 -n 8 指定）运行 MPI 程序：
  - RX2600 集群：bsub -a mpich_gm -q normal -n 8 executable-mpi1
  - Superdome 服务器：bsub -a hpmpi -q idle -n 8 executable-mpi1
  - 联想集群：bsub -q normal -n 8 mpijob executable-mpi1
- 指定利用两个核（由 -n 2 指定）运行 OpenMP 程序：
  - RX2600 集群：bsub -x -q normal -n 2 executable-mpi1
  - Superdome 服务器：bsub -q idle -n 2 executable-omp1
  - 联想集群：bsub -a openmp -q normal -n 2 executable-omp1

运行串行作业：bsub -q serial

运行串行作业，请使用 serial 队列：

bsub -q serial executable-serial

运行 OpenMP 共享内存作业：bsub -a openmp

集群只能在同一个节点内部运行 OpenMP 共享内存的作业

- HP RX2600 集群可利用 -x 排他性选项保证在同一个节点内部的两颗 CPU 上运行：
  bsub -a openmp -q normal -n 2 executable-omp1
- 联想集群需要添加 -a openmp 选项：
  bsub -a openmp -q normal -n 8 executable-omp1

运行共享内存作业或排他性运行作业：bsub -x

如果需要独占节点运行，此时需要添加 -x 选项：

bsub -x -q normal -n 4 executable-omp1

注意：

- 排他性运行在运行期间，不允许其余的作业提交到运行此作业的节点，并且只有在某节点没有任何其余的作业在运行时才会提交到此节点上运行
- 如果不需要采用排他性运行，请不要使用此选项，否则将导致作业必须等待完全空闲的节点才会运行，也许将增加等待时间
- 另外使用排他性运行时，哪怕只使用某节点内的一个核，也将按照此节点内的所有 CPU 核数进行机时计算

指明输入、输出文件运行：bsub -i -o -e

- 作业的输入文件、正常屏幕输出到的文件和错误屏幕输出的文件可以利用 -i、-o 和 -e 选项来分别指定，运行后可以通过查看指定的这些输出文件来查看运行状态，文件名可利用 %J 与作业号挂钩
- 如指定 executable1 的输入、正常和错误屏幕输出文件分别为 executable1.input、executable1-%J.log 和 executable1-%J.err：
  bsub -i executable1.input -o executable1-%J.log -e executable1-%J.err executable1

交互式运行作业：bsub -I

如需运行交互式的作业（如在运行期间需手动输入参数等），需结合 -I 参数，建议只是在调试期间使用，平常作业还是尽量不要使用此选项，类似选项还有 -Ip 和 -Is：

bsub -I executable1

终止作业：bkill

利用 bkill 命令可以终止某个运行中或者排队中的作业，比如：

bkill 79722

运行成功后，将显示类似下面的输出：

```
Job <79722> is being terminated
```

挂起作业：bstop

利用 bstop 命令可临时挂起某个作业以让别的作业先运行，例如：
bstop 79727
运行成功后，将显示类似下面的输出：

```
Job <79727> is being stopped.
```

- 可以将排在队列前面的作业临时挂起，以让后面的作业先运行
- 虽然也可以作用于运行中的作业，但并不会因为此作业被挂起而允许其余作业占用此作业所占用的 CPU 运行，实际资源不会释放，建议不要随便对运行中的作业进行挂起操作
- 如果运行中的作业不再想继续运行，请用 bkill 终止

继续运行被挂起的作业：bresume

利用 bresume 命令可继续运行某个挂起某个作业，例如：

bresume 79727

运行成功后，将显示类似下面的输出：

```
Job <79727> is being resumed.
```

设置作业最先运行：btop

利用 btop 命令可最先运行排队中的某个作业，例如：

btop 79727

运行成功后，将显示类似下面的输出：

```
Job <79727> has been moved to position 1 from top.
```

设置作业最后运行：bbot

利用 bbot 命令可设定最后运行排队中的某个作业，例如：

bbot 79727

运行成功后，将显示类似下面的输出：

```
Job <79727> has been moved to position 1 from bottom.
```

修改排队中的作业选项：bmod

利用 bmod 命令可修改排队中的某个作业的选项，如想将排队中的作业号为 79727 的作业的执行命令修改为 executable2 并且换到 fat 队列：

bmod -Z executable2 -q fat 79727

```
Parameters of job <79727> are being changed.
```

查看作业的排队和运行情况：bjobs

利用 bjobs 可以查看作业的运行情况，例如：

bjobs

```
JOBID  USER  STAT  QUEUE   FROM_HOST  EXEC_HOST  JOB_NAME    SUBMIT_TIME
79726  hmli  RUN   normal  user       2*node31   *executab1  Mar 12 19:20
                                      1*node4
79727  hmli  PEND  long    user                  *executab2   Mar 12 19:20
```

显示作业 79726 分别在 node31 和 node4 上运行 2、1 个进程；作业 79727 处于排队中尚未运行，查看未运行的原因可以利用 -l 选项：

bjobs -l 79727

```
Job Id <79727>, User <hmli>, Project <default>, Status <PEND>,
Queue <long> , Command <executab2>
Sun Mar 12 14:15:07: Submitted from host <hpc1.ustc.edu.cn>,
CWD <$HOME>, Requested Resources <type==any && swp>35>;
PENDING REASONS:
The user has reached his/her job slot limit;
SCHEDULING PARAMETERS:
            r15s  r1m  r15m   ut  pg    io   ls    it   tmp   swp   mem
loadSched   -     0.7  1.0    -  4.0    -    -     -    -     -     -
loadStop    -     1.5  2.5    -  8.0    -    -     -    -     -     -
```

查看运行中作业的屏幕正常输出：bpeek

利用 bpeek 命令可查看运行中作业的屏幕正常输出，例如：

bpeek 79727

```
<< output from stdout >>
Radius(nm): 300.000
```

如果在运行中用 -o 和 -e 分别指定了正常和错误屏幕输出，也可以通过直接查看指定的文件的内容来查看屏幕输出。

查看各节点的运行情况：lsload

利用 lsload 命令可查看当前各节点的运行情况，例如：

lsload

```
HOST_NAME  status  r15s  r1m  r15m  ut  pg  ls  it    tmp    swp    mem
node10     ok      0.0   0.0  0.0   0%  3.5  0  2050  9032M  4000M  16G
node11     locku   0.0   0.0  0.0   0%  3.5  0  2050  9032M  4000M  16G
```

ut 列表示利用率，status 列中的 locku 表示在进行排他性运行。

查看各节点的空闲情况：bhosts

利用 bhosts 命令可查看当前各节点的空闲情况，例如：

bhosts

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
node12	closed	–	4	2	2	0	0	0
node10	ok	–	2	2	1	0	0	0

STATUS 列中的 ok 表示可以接收新作业，closed 表示已经被占满。

查看队列情况：bqueues

利用 bqueues 可以查看现有队列信息，例如：

bqueues

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
normal	30	Open:Active	–	8	–	–	22	2	20	0
long	30	Open:Active	–	304	–	–	52	12	40	0
fat	30	Open:Active	–	32	–	–	3	0	3	0

主要列的含义为：

- QUEUE_NAME：队列名
- PRIO：优先级，数字越大优先级越高
- STATUS：状态。Open:Active 表示已激活，可使用；Closed:Active 表示已关闭，不可使用
- MAX：队列对应的最大 CPU 核数，- 表示无限，以下类似
- JL/U：单个用户同时可以的 CPU 核数
- NJOBS：排队、运行和被挂起的总作业所占 CPU 核数
- PEND：排队中的作业所需 CPU 核数
- RUN：运行中的作业所占 CPU 核数
- SUSP：被挂起的作业所占 CPU 核数

现有队列

- RX2600 集群现有队列：
  - normal：所需要的 CPU 核数不超过六个
  - long：所需要的 CPU 核数超过六个但不超过 16 个
  - hugemem：所需要内存超过 2 GB，但不超过 12 GB 时，作业将只在 node1 和 node2 上运行
- 联想集群现有的队列：
  - serial：串行作业
  - normal：所需要的 CPU 核数不超过八个
  - mpi：所需要的 CPU 核数超过八个但不超过 40 个
- Superdome 服务器现有的队列：
  - 专有队列：以用户组命名，只有此用户组内的用户可以使用
  - idle：任何用户可以使用，级别很低，必要时运行中的作业将被专有用户的作业抢占

队列也许会调整，请利用 bqueues –l 查看各队列的详细情况

查看用户信息：buser

利用 buser 可以查看用户信息，例如：

busers hmli

USER/GROUP	JL/P	MAX	NJOBS	PEND	RUN	SSUSP	USUSP	RSV
hmli	–	22	40	32	8	0	0	0

作业管理系统 LoadLeveler 简介

- JS22 利用 Tivoli Workload Scheduler LoadLeveler 进行资源和作业管理，所有需要运行的作业均必须通过作业提交命令 llsubmit 提交，提交后可利用相关命令查询作业状态等
- 为了利用 llsubmit 提交作业，用户必须针对此作业创建提交脚本，在脚本里面设定需要运行的作业参数等
- 在这里，我们将分别给出串行和并行的简单脚本，用户请修改此脚本以适用于自己的作业，如需要高级功能等，请参考 Tivoli Workload Scheduler LoadLeveler: Using and Administering

作业命令文件

作业命令文件包含一些 LoadLeveler 的关键词和注释文字，关键词在以 # @ 开始的行中设定，并紧跟 # @，一般来说一行一个关键词。例如：

- 为了指明一个可执行的二进制文件被执行，利用关键词 executable 来指定
- 指明此脚本被执行，可利用 executable 关键词，也可省略此关键词，此时系统假设这个作业的脚本文件本身为所需要执行的作业

作业命令文件包含下面内容：

- LoadLeveler 关键词声明：关键词指的是在一个作业命令文件中的具有特定含义的词，关键词声明是跟随 LoadLeveler 关键词的声明
- 注释声明：用户可以利用注释使得作业命令文件具有可读性，类似普通脚本文件中的用处
- 脚本命令声明：若用户使用脚本作为执行命令，作业命令文件可包含脚本命令
- LoadLeveler 变量：可用于作业脚本中，比如 $(host)、$(jobid)等

作业命令文件的一般规定

- LoadLeveler 关键词以 # @ 开始，在 # 和 @ 之间允许有任意多的空格
- 注释以 # 开始，任何第一个非空格字符为 # ，并且不是 LoadLeveler 关键词的行被认为为注释
- 注释以空格为分割，用户可以在其他分割符之前和之后使用空格来提高可读性
- \ 是续行符，并且要求续行不得以 # @ 开始。如果用户的作业命令文件是需要被执行的脚本，用户必须在续行以 # 开始
- LoadLeveler 关键词忽略大小写，可以使用大写、小写或混合方式

串行作业

对于串行程序，用户可编写命名为 serial_job.cmd（此脚本名可以按照用户喜好命名）的串行作业命令文件，其内容如下：

```
# This job command file lists a job step called 'step1', which input file
# name is 'step1.in', screen output file name is 'step1.log', screen error
# output file name is 'step1.error', the cpu time for this job is 6000s,
# if overtime, the job will be terminated. the class for this job is serial,
# the job's executable file name is executable1.
# @ step_name = step1
# @ input = step1.in
# @ output = step1.log
# @ error = step1.error
# @ wall_clock_limit = 6000
# @ class = serial
# @ executable = executable1
# @ queue
```

串行作业

下面脚本与上面的功能一样，但由于没有用 executable 关键词指定需要运行的作业，此时将此作业命令文件作为作业，即将执行此作业命令文件的内容 executable1（一般为可执行程序名，如 /your/prog/name）。

```
# @ step_name = step1
# @ input = step1.in
# @ output = step1.log
# @ error = step1.error
# @ wall_clock_limit = 6000
# @ class = serial
# @ queue
/your/prog/name
```

作业脚本解释

上述作业命令文件的含义为：

- 提交一个作业名为 job_name 的作业到 serial 队列（与 LSF 不同，在 LoadLeveler 中称为 class）以运行命令 /your/prog/name
- /your/prog/name 输入文件为 step1.in
- 正常屏幕输出到 step1.log
- 错误信息输出 step1.error 中
- 此程序的最长运行时间为 6000 秒

脚本中以 # @ 开头的几行的 = 前的为 LoadLeveler 关键词，其余 # 后面的内容与一般脚本中的一样表示注释。关键词 queue 表示执行由此前信息所设置的作业

作业脚本提交

作业命令文件编写完成后，可以按照下面命令提交作业：
llsubmit ser_job.cmd
如果成功，将有类似下面的输出：

```
llsubmit: The job "js2.74" has been submitted.
```

其中 js2.74 表示作业号，组成形式为 host.jobid，分别对应主机名、作业序号，之后可利用此作业号来进行查询、终止此作业等操作。
对于 32 位程序来说，若程序运行需超过 256MB 内存，需在作业脚本中的执行命令（如 /your/prog/name）前，添加设置环境变量的选项：

- export LDR_CNTRL=MAXDATA=0x40000000 #可用 1 GB 内存
- export LDR_CNTRL=MAXDATA=0x80000000 #可用 2 GB 内存

如需要更大的内存，请在编译时添加 -q64 编译成 64 位的可执行文件

多个作业

LoadLeveler 作业命令文件支持按顺序运行多个作业，既可设置为只有在前一个作业正常完成后才运行下面作业，也可设置为即使前面作业出错下面作业也将运行

```
# This job command file lists two job steps called 'step1' and 'step2'.
# 'step2' only runs if 'step1' completes with exit status = 0. Each job
# step requires a new queue statement.
# @ step_name = step1
# @ executable = executable1
# @ input = step1.in
# @ output = step1.$(jobid).$(stepid).out
# @ error = step2.err
# @ queue
# @ dependency = (step1 == 0)
# @ step_name = step2
# @ executable = executable2
# @ input = step2.in
# @ output = step2.$(jobid).$(stepid).out
# @ error = step2.$(jobid).$(stepid).err
# @ queue
```

脚本解释

- 设置了两个作业，分别为 step1 和 step2，将分别执行 executable1 和 executable2，由于设置了关键词 dependency =(step1 == 0)，step2 只有在 step1 正常完成后才会运行
- 如果在上面的作业命令文件中去掉 dependency 关键词，同时添加关键词 coschedule = true，那么只有在两个作业都获取到所需资源后，作业才开始同时运行，如果没必要要求两个作业必须同时运行，请不要设置此参数，否则会影响作业及时被运行
- 如果几个作业直接无任何依赖关系，请不要添加 dependency 或 coschedule 关键词，以免影响运行
- 上述作业的输出文件名由于引用了作业运行时的 jobid 和 stepid 变量，将会与运行的作业号等相联系，这对提交多个作业来说非常有用，可避免冲突。LoadLeveler 提供了非常多的变量供用户使用

并行作业

对于并行作业，需编写类似下面脚本 par_job.cmd：

```
# An example for parallel job.
# @ job_type = parallel
# set to run parallel job
# @ environment = COPY_ALL
# set to copy all environment variable to node
# @ input = step1.in
# @ output = step1.log
# @ error = step1.error
# @ node = 1
# set to use 1 node to run.
# @ tasks_per_node = 8
# set to fork 8 threads for every node.
# @ wall_clock_limit = 6000
# @ notification = never
# @ class = medium
# @ queue
/usr/bin/poe /your/prog/name
```

脚本解释

与串行程序的作业脚本文件相比：

- 通过关键词 job_type 指明为并行程序
- 利用 environment 设置将所有环境变量复制到运行节点
- 利用 node 设置节点数
- 利用 tasks_per_node 设置每个节点的进程数（每个节点上有四个核，而且 POWER6 支持同时多线程(SMT)，因此最大可设置为 8，请结合自己程序的特点，设置为 4 或 8，以获取最高性能）
- 对于 MPI 程序需采用 poe 的命令格式提交并行可执行程序
- 对于 OpenMP 程序不应该使用 poe，应该通过设置 OMP_NUM_THREADS=8 或 4 来设置进程数
- 系统默认作业完成后，将发送信件给用户，这里设置了关键词 notification = never，表示作业完成后将不发送信件，还可设置为 always、error、start、complete。

与串行作业一样，可使用下面方式提交：

llsubmit par_job.cmd

常用作业管理命令

用户常用的与作业相关的 LoadLeveler 命令主要有：

- llcancel：取消已存在的作业
- llclass：查询队列信息
- llhold：挂起一个作业
- llmodify：修改作业的运行参数
- llstatus：显示节点信息
- llsubmit：提交作业
- llprio：修改作业的优先级
- llq：显示作业状态等详细信息

下面只对最常用的做简单介绍，更多的相关命令及详细用法（利用 -H 参数可以查看命令详细信息），请参考 Tivoli Workload Scheduler LoadLeveler: Using and Administering。

提交作业：llsubmit

作业命令文件编写完成后，可以按照下面命令提交作业：

llsubmit ser_job.cmd

如果成功，将有类似下面的输出：

```
llsubmit: The job "js2.74" has been submitted.
```

- js2.74 表示作业号，组成形式为 host.jobid，分别对应主机名、作业序号，之后可利用此作业号来进行查询、终止此作业等操作
- 利用 llq 等查询时还多出一项对应脚本中的 step 的 stepid，实际作业号形式为 host.jobid.stepid

终止作业：llcancel

llcancel 可终止一个作业，比如下面命令将终止 js2.84.0 作业的运行：

llcancel js2.84.0

查询队列信息：llclass

用户作业需要提交到特定队列（class）才能运行，查看可以使用的队列信息，可以利用 llclass 命令，其输出类似：

```
Name        MaxJobCPU     MaxProcCPU     Free   Max  Description
          d+hh:mm:ss    d+hh:mm:ss   Slots  Slots
--------  ------------  -------------  -----  -----  -----------
serial      undefined     undefined        2      3  low priority serial queue
medium      undefined     undefined      120    128  normal parallel queue
```

上面显示有两种队列 serial 和 medium 可使用，允许运行的最大作业数目（Max Slots）分别为 3 和 128，当前空闲的数目（Free Slots）分别为 2 和 120。

常用参数：

- -c classname：显示某个队列的信息
- -l：显示队列的详细信息

挂起作业和取消挂起作业：llhold

qhold 命令可以挂起作业，被挂起的作业将暂停执行，以让其余作业优先得到资源运行，被挂起的作业在用 llq 命令查询时显示的状态标志为 H

- 挂起作业号为 js2.84.0 的作业：
  llhold js2.84.0
- 释放已被挂起的作业 js2.84.0 重新进入排队：
  llhold -r js2.84.0

修改作业参数：llmodify

利用 llmodify 可以修改作业的队列类型、时间界限等，比如
llmodify -W 30 js2.110.0
将时间限制扩大 30 分钟

修改作业优先级：llprio

- 如作业在提交时没有特别设置优先级，且作业所需的资源一致，那么作业的优先级将相同，按照先提交先运行的原则进行调度
- 如有两个作业 js2.110.0 和 js2.111.0，js2.110.0 先于 js2.111.0 提交，如想让 js2.111.0 先于 js2.110.0 运行，可利用 llprio 降低 js2.110.0 的优先级或升高 js2.111.0 的优先级，比如将 js2.111.0 优先级增加 10：

llprio +10 js2.111.0
运行 llq 将显示：

Id	Owner	Submitted	ST	PRI	Class	Running
js2.111.0	hmli	3/30 19:02	I	60	medium	
js2.110.0	hmli	3/30 19:01	I	50	medium	

查看队列中的作业状态：llq

查看现在作业的运行状态，可以利用 llq，将给出类似下面的输出：

Id	Owner	Submitted	ST	PRI	Class	Running o
js2.83.0	hmli	3/30 15:06	R	50	medium	node14
js2.84.0	hmli	3/30 15:06	H	50	medium	
js2.85.0	hmli	3/30 15:07	I	50	medium	

上面几列的含义分别为：作业号、用户名、提交时间、作业状态、优先级、资源名、运行程序的节点，其中作业状态中的 R、H 和 I 分别表示作业处于运行、被挂起和排队中。

查询某用户的作业：llq -u namelist

比如查询用户 hmli 的作业：llq -u hmli

查看队列中的作业未运行的原因：llq -l

查看 js2.85.0 尚没运行的原因，可利用 llq –l js2.85.0：

```
......
          Unix Group: nic
Negotiator Messages: User = hmli has reached the maximum number jobs
                           allowed running.
       Bulk Transfer: No
......
```

Negotiator Messages 一行显示用户 hmli 已经到达最大可运行的数目

查看队列中的作业未运行的原因：llq -s

利用 llq −s js2.84.0 也可以查看挂起的原因：

```
.....
================== EVALUATIONS FOR JOB STEP js2.84.0 ==================

The status of job step is : User Hold
Since job step status is not Idle, Not Queued, or Deferred, no attempt has
been made to determine why this job step has not been started.
.....
```

Status: User Hold 显示作业没有运行的原因是作业被用户自己挂起。

按照特定格式显示作业信息：llq -f category_list

llq −f category_list 按照 category_list 指定格式显示作业信息，如作业名（%jn）、所有者（%o）、状态（%st）、分配节点数（%nh）等

显示节点状态：llstatus

llstatus 可以显示当前各节点状态，其输出类似：

```
Name      Schedd   InQ   Act  Startd  Run  LdAvg  Idle  Arch    OpSys
node01    Avail     0     0    Idle    0   0.00   9999  R6000   AIX61
......    .....     .     .    ....    .   ....   ....  .....   .....
node15    Avail     0     0    Run     8   7.96   9999  R6000   AIX61

R6000/AIX61     16 machines       3  jobs      20  running tasks
Total Machines  16 machines       3  jobs      20  running tasks

The Central Manager is defined on js2

The BACKFILL scheduler is in use
```

用户比较关心的是 LdAvg 一列，显示的是节点当前的负载，应该与所有用户通过作业调度系统申请的进程数差不多

- 如严重偏小，说明利用率不高，最好查找原因，看看瓶颈在哪里
- 如比指定的进程数大很多，有可能是有用户没通过作业管理系统而是直接到节点上运行作业，可以 rsh 节点名进入此节点，并运行 topas命令（AIX 系统下无 top 命令，对应的是 topas）看看是哪个进程，哪个用户在违规使用，如发现此问题，请联系管理员

TORQUE 和 Maui 简介

- KD-50-I 利用 TORQUE 和 Maui 进行资源和作业管理
- 所有需要运行的作业无论是用于程序调试还是业务计算均必须通过 qsub 命令提交，提交后可以利用 TORQUE 和 Maui 的相关命令查询作业状态等
- 为了利用 qsub 提交作业，用户需针对此作业创建提交脚本，在脚本里面设定需要运行的作业参数等
- 在此分别给出串行和并行的简单脚本，用户可以修改此脚本以适用于自己的作业，如需要更加高级的功能请参考 TORQUE 手册

串行作业脚本

对于串行程序，用户可编写命名为 serial_job.sh（此脚本名可以按照用户喜好命名）的串行作业脚本，其内容如下：

```
#!/bin/sh
#PBS -N job_name
#PBS -o job.log
#PBS -e job.err
#PBS -q dque
cd yourworkdir
echo Running on hosts `hostname`
echo Time is `date`
echo Directory is $PWD
echo This job runs on the following nodes:
cat $PBS_NODEFILE
echo This job has allocated 1 node
./yourprog
```

- TORQUE 建立在 PBS 作业管理系统之上，PBS 的参数需在作业提交脚本中利用 #PBS 设置
- 上述脚本表示进入 yourworkdir 目录后，提交到 dque 队列，其作业名为 job_name，标准输出和错误输出将分别存在此目录下的 job.log 和 job.err 文件中

提交作业

qsub ser_job.sh
如果成功，将有类似下面的输出：

```
37.kd50
```

其中 37.kd50 表示作业号，由两部分组成，37 表示的是作业序号，kd50 表示的是作业管理系统的主机名，也就是登录节点名，之后可以用此作业号来查询作业及终止此作业等。

并行作业脚本

与串行作业类似，并行作业需编写类似下面脚本 par_job.sh：

```
#!/bin/sh
#PBS -N job_name
#PBS -o job.log
#PBS -e job.err
#PBS -q dque
#PBS -l nodes=16
cd yourworkdir
echo Time is `date`
echo Directory is $PWD
echo This job runs on the following nodes:
cat $PBS_NODEFILE
NPROCS=`wc -l<$PBS_NODEFILE`
echo This job has allocated $NPROCS nodes
mpiexec -machinefile $PBS_NODEFILE -np $NPROCS ./yourprog
```

与串行程序的脚本相比，主要不同之处在于在#PBS 开头的 -l 参数后设置：nodes=所需要的进程数，另外请注意需采用 mpiexec 的命令格式提交并行可执行程序。

与串行作业类似，可使用下面方式提交：

qsub par_job.sh

常用作业管理命令

- canceljob：取消已存在的作业
- checkjob：显示作业状态、资源需求、环境、限制、信任、历史、已分配资源和资源利用等
- nqs2pbs：将 nqs 作业脚本转换为 pbs 作业脚本
- pbsnodes：显示节点信息
- printjob：显示指定作业脚本中的作业信息
- qdel：取消指定的作业
- qhold：挂起一个作业
- qmove：将一个作业从一个队列移到另一个队列中
- qnodes：pbsnodes 的别名，显示节点信息
- qorder：交换两个作业的排队顺序
- qrls：将被挂起的作业送入准备运行的队列中
- qselect：显示符合条件的作业的作业号
- qstat：显示队列、服务器和作业的信息
- qsub：提交作业
- showbf：显示有特殊资源需求的资源的可用性
- showq：显示已激活和空闲的作业的优先级细节
- showstart：显示空闲作业的估计开始时间
- tracejob：追踪作业信息

查看队列中的作业状态：qstat

qstat 可以查看作业的运行状态：
输入 qstat 命令后，将给出类似下面的输出：

Job id	Name	User	Time Use	S	Queue
48.kd50	job_name4	user	0	E	dque
49.kd50	job_name1	user	00:00:00	R	dque
50.kd50	job_name2	user	0	H	dque
51.kd50	job_name3	user	0	Q	dque

上面几列的含义分别为：作业号、作业名、用户名、使用的时间、状态、队列名，其中状态中的 E、Q、H 和 R 分别表示作业处于退出、挂起、排队和运行中。

挂起作业：qhold

qhold 命令可以挂起作业，被挂起的作业将不被执行，这样可以让其余作业优先得到资源运行，被挂起的作业在用 qstat 命令查询时显示的状态标志为 H，下面命令将挂起作业号为 50.kd50 的作业：

qhold 50.kd50

取消挂起：qrls

被挂起的作业可以利用 qrls 来取消挂起，重新进入等待运行状态：

qrls 50.kd50

终止作业：qdel 和 canceljob

用户如果想终止一个作业，可以利用 qdel 或 canceljob 来取消：

qdel 50.kd50

canceljob 51.kd50

查看作业状态：checkjob

checkjob 可以查看作业的状态：
checkjob 51.kd50

```
checking job 51

State: Hold
Creds:  user:user  group:user  class:dque  qos:DEFAULT
WallTime: 00:00:00 of 99:23:59:59
SubmitTime: Sun Dec  2 19:22:19
  (Time Queued  Total: 00:46:13  Eligible: 00:24:40)

Total Tasks: 16
Req[0]  TaskCount: 16  Partition: ALL
Network: [NONE]  Memory >= 0  Disk >= 0  Swap >= 0
Opsys: [NONE]  Arch: [NONE]  Features: [NONE]

IWD: [NONE]  Executable:  [NONE]
Bypass: 0  StartCount: 0
PartitionMask: [ALL]
Flags:       RESTARTABLE
PE:  16.00  StartPriority:  24
cannot select job 51 for partition DEFAULT (non-idle state 'Hold')
```

从上面的 State: Hold 可以看出作业已被挂起。

查看作业状态：checkjob

checkjob 49.kd50

```
checking job 49
State: Running
Creds:  user:user  group:user  class:dque  qos:DEFAULT
WallTime: 1:07:14 of 99:23:59:59
SubmitTime: Sun Dec  2 19:02:10
  (Time Queued  Total: 00:00:01  Eligible: 00:00:01)
StartTime: Sun Dec  2 19:02:11
Total Tasks: 8
Req[0]  TaskCount: 8  Partition: DEFAULT
Network: [NONE]  Memory >= 0  Disk >= 0  Swap >= 0
Opsys: [NONE]  Arch: [NONE]  Features: [NONE]
NodeCount: 8
Allocated Nodes:
[node08:1][node07:1][node06:1][node05:1]
[node04:1][node03:1][node02:1][node01:1]

IWD: [NONE]  Executable:  [NONE]
Bypass: 0  StartCount: 1
PartitionMask: [ALL]
Flags:       RESTARTABLE

Reservation '49' (-1:06:52 -> 99:22:53:07  Duration: 99:23:59:59)
PE:    8.00  StartPriority:  1
```

从 State: Running 可看出作业处于运行中，且可看到占用的资源状态。

交换两个作业的排队顺序：qorder

qorder 可以交换两个作业的排队顺序：
当前作业状态：

Job id	Name	User	Time Use	S	Queue
52.kd50	job_name1	user	0	H	dque
53.kd50	job_name2	user	0	Q	dque
54.kd50	job_name3	user	0	Q	dque

qorder 53.kd50 54.kd50
利用 qstat 看执行之后的作业状态：

Job id	Name	User	Time Use	S	Queue
52.kd50	job_name1	user	0	H	dque
54.kd50	job_name3	user	0	Q	dque
53.kd50	job_name2	user	0	Q	dque

作业 53.kd50 和 54.kd50 的排队顺序相互对换了，作业 54.kd50 将优先于 53.kd50 运行。

选择符合特定条件的作业的作业号：qselect

qselect 可以用来显示符合一定条件的作业的作业号，比如选择被挂起的作业，可用下面的命令：

qselect -s H

52.kd50

显示队列中作业的信息：showq

showq 可以显示队列中作业的信息
showq

```
ACTIVE JOBS--------------------
JOBNAME   USERNAME      STATE   PROC   REMAINING            STARTTIME
52              user   Running     16  99:22:44:09   Sun Dec 2 21:04:37
     1 Active Job      16 of    16 Processors Active (100.00%)
IDLE JOBS----------------------
JOBNAME   USERNAME      STATE   PROC      WCLIMIT            QUEUETIME

54              user       Idle     16  99:23:59:59   Sun Dec 2 21:04:45 1
Idle Job

BLOCKED JOBS-------------------
JOBNAME   USERNAME      STATE   PROC      WCLIMIT            QUEUETIME

53              user       Hold     16  99:23:59:59   Sun Dec 2 21:04:37
Total Jobs: 3   Active Jobs: 1    Idle Jobs: 1    Blocked Jobs: 1
```

显示节点信息：pbsnodes 和 qnodes

pbsnodes 和 qnodes（实际是同一个命令的两个名字）可显示系统各个节点的信息，如空闲（free）、当机（down）、离线（offline）。例如：显示所有空闲的节点：

pbsnodes -l free

其输出为：

node0101	free
node0102	free
node0104	free
node0105	free

联系信息

- 中国科大超算中心：
  - 主页：http://scc.ustc.edu.cn
  - 电话：0551-3602248
  - 信箱：sccadmin@ustc.edu.cn
- 青能所超算中心：
  - 当前主页：http://124.16.151.186
  - 将来域名：http://scc.qibebt.cas.cn
  - 电话：0532-80662613
  - 信箱：scc@qibebt.ac.cn
- 李会民：
  - 主页：http://staff.ustc.edu.cn/~hmli/
  - 电话：0532-80662613
  - 信箱：hmli@ustc.edu.cn、lihm@qibebt.ac.cn

欢迎指出错误和改进意见。