作业调度系统的使用

李会民
hmli@ustc.edu.cn

中国科学技术大学 超级运算中心

2011年10月

作业管理系统LSF的使用

当前超算中心的超算系统除IBM JS22刀片服务器之外，其它的超算系统采用Platform公司[1]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

[1]最近刚被IBM收购

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
```

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

查看队列详细情况：bqueues -l

队列也许会调整

- 请注意登录系统后的提示
- 请利用*bqueues -l*查看各队列的详细情况

```
QUEUE: normal
  -- For normal priority jobs, allow 2-8 CPU cores, Max Job Slots is 40 This is the default queue.

PARAMETERS/STATISTICS
PRIO  NICE  STATUS          MAX  JL/U  JL/P  JL/H  NJOBS   PEND    RUN  SSUSP  USUSP   RSV
 30    20   Open:Active       -    40     -     -    288     48    240      0      0     0

CPULIMIT                    PROCLIMIT
 345600.0 min of E5410       2 2 8

 PROCESSLIMIT
       8
```

以上面为例：

- CPULIMIT：单个作业运行时间限制，以系统中的某个节点E5410作为参考，运行时间（核数*墙上时间）为345600.0分钟
- PROCLIMIT：单个作业核数限制，2 4 8，表示使用此队列时，最少使用2个核，最多使用8核，如提交时没用-n指定具体核数，那么使用默认4核
- PROCESSLIMIT：单个作业最大核数限制，为8

现有队列：联想1800集群

- serial：串行作业
- normal：所需要的CPU核数2-8个，并且在单节点内运行，MPI程序和OpenMP等共享内存程序都可使用
- mpi：所需要的CPU核数超过8个，多个节点同时运行，MPI程序可以使用，但OpenMP等共享内存的程序无法使用

- 建议以8的倍数申请核数，以尽量独占单个节点，避免作业间相互干扰
- 具体申请核数应考虑作业实际情况，如：选择最佳并行效率等[2]，在满足作业本身要求的情况下尽量以8的倍数

[2]即使同一个计算软件，在计算不通条件时，也是不一样的，请务必仔细研究自己所使用的软件

现有队列：联想7000G GPU集群

- c1060：GPU作业队列，保证分配的节点含有C1060 GPU卡
- gtx295：GPU作业队列，保证分配的节点含有GTX295 GPU卡
- cuda4：GPU作业队列，保证分配的节点含有支持CUDA4
- normal：所需要的CPU核数4-8个，并且在单节点内运行，MPI程序和OpenMP等共享内存程序都可使用
- long：所需要的CPU核数超过八个，在多个节点上运行

建议以8的倍数申请核数，以尽量独占单个节点，避免作业间相互干扰

现有队列：曙光CB-60G刀片集群

- serial：串行作业
- normal：所需要的CPU核数不超过12个，并且在单节点内运行，MPI程序和OpenMP等共享内存程序都可使用
- long：所需要的CPU核数超过12个，多个节点同时运行，MPI程序可以使用，但OpenMP等共享内存的程序无法使用

建议以12的倍数申请核数，以尽量独占单个节点，避免作业间相互干扰

现有队列：浪潮TS850和曙光A950胖节点

只有normal队列，单个作业可以使用1-48核

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

- 作业提交后，可以ssh到对应运行作业的节点运行*top*命令查看一下作业的CPU、内存等利用率，查看实际运行效率
- 但请不要ssh到节点后直接运行作业，将会被监控自动杀掉

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，如果不加-q，那么将提交到系统设置的默认队列
- 提交到serial队列运行串行程序executable1：

  bsub -q serial executable1

  如果提交成功，将显示类似下面的输出：

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

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）
- 为了用户作业间不相互干扰，申请的核数最好为系统节点核数的整数倍，以便同一个作业占据整个节点
  - 比如对每个节点为8核的系统，申请核数为8的整数倍，节点核数为12的系统，申请核数为12的整数倍
    - 联想1800和7000G GPU集群：每个节点8核
    - 曙光刀片集群和联想刀片集群：每个节点12核
  - 以上仅仅是建议，具体申请核数应考虑作业实际情况，如：选择最佳并行效率等[3]，在满足作业本身要求的情况下尽量以8的倍数

[3]即使同一个计算软件，在计算不通条件时，也是不一样的，请务必仔细研究自己所使用的软件

MPI作业的提交

- MPI作业一般需要用提交时使用mpijob来调用MPI程序，可以使用normal、long和mpi等队列
- 指定利用八个核（由-n 8指定）运行MPI程序:

bsub -q normal -n 8 mpijob executable-mpi1

OpenMP等共享内存作业的提交

- OpenMP等共享内存程序，需要保证在同一个节点内运行，只能使用normal队列，不能使用long等队列
- 指定利用八个核（由-n 8指定）运行OpenMP程序：

  bsub -q normal -n 8 executable-omp1

GPU作业的提交

- GPU作业必需使用GPU队列才可以保证运行的作业在GPU节点上运行，当前只有联想7000G GPU集群支持GPU，当前可以使用的队列为c1060、gtx295和cuda4。近期我们新购买的20块c2050 GPU卡到货安装调试后也会推出。
- 因为与MPI结合的GPU程序要求不统一，现在系统尚未提供专有的GPU和MPI结合程序的脚本供大家来调用，如需要，请与我们联系，我们来处理。如果对LSF熟悉，也可以自己写所需要的脚本

串行作业的提交：bsub -q serial

- 运行串行作业，请使用serial队列：
  bsub -q serial executable-serial
- 科大超算中心鼓励并行作业，因此给串行作业的资源少，请尽量用自己的系统运行串行作业，在科大超算平台上运行并行作业

运行排他性运行作业：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

- 作业的屏幕输入文件、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
- 屏幕输入文件指的是存储程序运行时需要手动在屏幕上输入的内容的，其内容可以利用<将此文件中的内容重定向以代替手动屏幕输入传递给可执行程序的，并不是指的程序本身自带的输入文件，如不通过作业调度系统时的提交方式为：
  - executable1 < file1，可以用下述方式提交
    bsub -i file1 executable
  - executable1 file1或executable1 -i file1等，则不可用下述方式提交
    bsub -i file1 executable
- 如指定executable1的屏幕输入、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，如：
  bkill 79722

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

- 请及时终止有问题，或者无需再运行的程序，以便空出计算资源

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

```
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

```
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

```
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long队列，并且所需要核数为12：

  bmod -Z executable2 -q long -n 12 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

查看作业详细信息：bjobs -l

- 查看作业的详细信息-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   -    -     -    -     -     -
```

注意，从PENDING REASONS可以看出为什么还在排队等待中。

查看作业仍在排队等待的原因：bjobs -p

- 查看作业仍在排队等待的原因可以利用-p选项：
  bjobs -p 79727

```
The user has reached his/her job slot limit;
```

上述显示达到了用户自己的作业数等限制。

- 如发现很多节点空闲，自己的作业又没有达到限制，感觉应该运行而没有运行，也许是系统存在问题，请与管理员联系处理。

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
```

- *bpeek -f* 作业号，可以连续查看作业的连续屏幕输出
- 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

- 主要列的含义：MAX最大可以同时运行的核数，NJOBS当前所有运行和待运行作业所需的核数，PEND排队等待运行的作业所需要的核数，RUN已经开始运行的作业占据的核数

LSF作业脚本

- 也可以使用LSF脚本方式提交，在LSF脚本中设置队列等，比如下面*my_script*脚本文件

```
#!/bin/sh
#BSUB -q long
#BSUB -o outfile -e errfile # my default stdout, stderr files
#BSUB -n 16
sim1.exe
```

- 需要传递给*bsub*命令运行：*bsub -q night < my_script*
- 不得以直接*./my_script*等常规脚本运行方式运行
- 一般用户，没必要写此类脚本，直接通过命令行传递LSF参数即可。对于当前设置满足不了作业需求，且用户比较了解LSF中的各规定，对shell脚本编写比较在行，那么用户完全可自己编写脚本提交作业，比如提交特殊需求的MPI与GPU结合的作业。

LSF作业脚本常见变量

主要有以下变量比较常用，具体的请参看LSF官方手册：

- *LS_JOBPID*：作业进程号
- *LSB_HOSTS*：存储系统分配的节点名
- *LSB_JOBFILENAME*：作业脚本文件名
- *LSB_JOBID*：作业号
- *LSB_QUEUE*：作业队列
- *LSB_JOBPGIDS*：作业进程组号组
- *LSB_JOBPIDS*：作业进程号组

作业管理系统LoadLeveler简介

- IBM JS22刀片服务器利用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
- 在此，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，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
- 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

- 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

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

作业脚本提交

- 作业命令文件不得以通常脚本方式运行，如*./ser_job.cmd*，应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1GB内存
  - *export LDR_CNTRL=MAXDATA=0x80000000* #可用2GB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*，则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
```

与串行作业一样提交方式一样：

llsubmit par_job.cmd

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程序应该通过设置OMP_NUM_THREADS来设置进程数，不应该使用poe
- 默认系统作业完成后将发送邮件给用户[4]，如不想收到邮件通知，可以设置关键词*notification = never*，还可设置为always、error、start、complete，分别表示总是、出错时、开始时以及完成时发送邮件通知。

[4]当前系统没有配置外发学校邮箱等功能，只能发送到js2系统上

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

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

- 查看可以使用的队列（class）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

- *llhold*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
- 被挂起的作业可以用*llhold -r*恢复执行
- 被挂起的作业在用*llq*命令查询时显示的状态标志为H

比如：

- 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
- 下面命令将时间限制扩大30分钟

  llmodify -W 30 js2.110.0

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 on
----------------- -------- ------------ -- ---- --------- --------
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 on
------------------------ -------- ------------ --  ---  ---------- ----------
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

利用 *llq -u* 可以查询某用户的作业，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

利用-l参数可以查看队列作业详细信息以及未运行的原因等，如：查看js2.85.0尚没运行的原因

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*也可以查看挂起的原因，如：
llq -s js2.84.0：

```
......
==================== 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*命令可以按照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

联系信息

李会民:

- 办公室：科大东区新科研楼A座网络信息中心二楼205室
- 办公电话：0551-3602248
- 电子信箱：hmli@ustc.edu.cn
- 个人主页：http://hmli.ustc.edu.cn
- 中国科大超算平台：http://scc.ustc.edu.cn