LSF作业调度系统使用手册

一、LSF作业调度系统用途

- 资源管理器：管理超算系统的硬件资源及认证信息等
- 队列管理器：管理当前已经提交但还未完成的作业
- 调度器：为作业分配资源
- 主要作用：
  - 根据用户作业提出的需求分配对应的资源给作业，告诉作业给它分配 哪些节点等
  - 避免作业之间无序干扰，尽量让整个系统的负载一致
  - 保证用户占用资源的长期内公平

二、查看队列情况: bqueues

1、查看现有的队列信息

```
[para-test@login03 ~]$ bqueues      #查看所有队列的信息
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
bench            90  Closed:Active     -    -    -    -     0     0     0     0
medium           90  Open:Active       -    -    -    -  4260     0  4260     0
test             90  Open:Active       -    -    -    -     0     0     0     0
asc20            90  Closed:Active     -    -    -    -     0     0     0     0
admin            50  Closed:Active     -    -    -    -     0     0     0     0
short            40  Open:Active       -    -    -    - 10320     0 10320     0
debug            30  Open:Active       -   80    -    -     0     0     0     0
ser              30  Open:Active       -    -    -    -  1226     0  1226     0
gpu              30  Open:Active       -    -    -    -    82     0    82     0
smp              30  Open:Active       -    -    -    -   282     0   282     0
large            30  Open:Active       -    -    -    -   760     0   760     0
[para-test@login03 ~]$ bqueues ser      #查看ser队列的信息
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
ser              30  Open:Active       -    -    -    -  1146     0  1146     0

################################################################################
# - QUEUE_NAME：队列名
# - PRIO：优先级，数字越大优先级越高
# - STATUS：状态
#   - Open：队列开放，可以接受提交新作业
#   - Active：队列已激活，队列中未开始运行的作业可以开始运行
#   - Closed：队列已关闭，不接受提交新作业
#   - Inact：队列未激活，可接受提交新作业，但队列中的等待运行的作业 不会开始运行
# - MAX：队列对应的最大作业槽数（JobSlot，一般与CPU核数一致， 以下通称CPU核数），-表示无限
# - JL/U：单个用户同时可以使用的CPU核数
# - JL/P：每个处理器可以接受的CPU核
# - JL/H：每个节点可以接受的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
# - RSV：为排队作业预留的CPU核数
################################################################################
```

2、查看指定队列详细情况

```
[para-test@login03 ~]$ bqueues -l short      #查看指定队列详细情况
QUEUE: short
  -- short jobs, The maximum run limit for the queue is 72H.  This is the
default queue.

PARAMETERS/STATISTICS
PRIO NICE STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV
PJOBS
 40    0  Open:Active       -    -    -    - 10600    80 10520     0     0    0
    2

 RUNLIMIT
 21600.0 min
```

```
 TASKLIMIT
 40 100 20000

 MEMLIMIT
    170 G

SCHEDULING PARAMETERS
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

SCHEDULING POLICIES:  FAIRSHARE  EXCLUSIVE
USER_SHARES:  [default, 1]

SHARE_INFO_FOR: short/
 USER/GROUP   SHARES  PRIORITY  STARTED  RESERVED  CPU_TIME  RUN_TIME  ADJUST
ess-baoxy        1       0.333      0         0         0.0         0       0.000
phy-xiaoxl       1       0.333      0         0         0.0         0       0.000
ess-wangw        1       0.333      0         0         0.0         0       0.000

#############################################################################
# - QUEUE：队列名，跟着的下一行是描述
# - PRIO：优先值，越大越优先
# - NICE：作业运行时的nice值，即作业运行时的操作系统调度优先值， 从-20到19，越小越优先
# - RUNLIMIT：作业单CPU运行时间限制，以系统中的manager节点为基准， 1440.0min表示可以运行一
天
#            （60*24）
# - *CPULIMIT：单个作业运行机时（核数*墙上时间）限制，以系统中的 manager节点作为参考，目前未
限制
# - 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，则使用默认4核， 目前未限制
# - PROCESSLIMIT：单个作业最大核数限制，目前未限制
# - MEMLIMIT：每个进程能使用的内存数，默认以KB为单位，目前未限制
# - THREADLIMIT：作业最大线程数，目前未限制
#############################################################################
```

三、高性能计算系统现有队列

- “太乙”集群目前共开放以下七个队列:

队列	队列时长限制	应用场景	单作业核心数限制
large	36小时	大规模	600核以上（含600）
medium	72小时	中等规模	200核以上（含200）
short	144小时	小规模	40核以上（含40）
ser	168小时	串行	无限制
smp	168小时	大内存	无限制
debug	20分钟	调试	80核以上（含80）
gpu	168小时	GPU	无限制

四、查看计算节点信息: lsload、bhosts

1、查看各节点的运行情况

- 利用lsload命令可查看当前各节点的运行情况

```
[para-test@login01 build]$ lsload | more     #查询所有节点的运行情况
HOST_NAME       status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
r01n16              ok   0.0   0.1   0.0   0%   0.0   0 63616  212G    0M  165G
r01n20              ok   0.0   0.0   0.0   0%   0.0   0  1e+5  210G  1.4G  167G
r01n25              ok   0.0   0.0   0.0   0%   0.0   0 88512  212G  3.5G  168G
r02n17              ok   0.0   0.0   0.0   0%   0.0   0  3e+5  212G  1.4G  167G
r03n06              ok   0.0   0.0   0.0   0%   0.0   0  3e+5  212G    0M  167G
r03n07              ok   0.0   0.0   0.0   0%   0.0   0 62752  212G  689M  168G
r03n09              ok   0.0   0.0   0.4   0%   0.0   0 16056  212G  3.7G  169G
r03n28              ok   0.0   0.0   0.8   0%   0.0   0  2e+5  212G  1.1G  168G
.....................................................................................
[para-test@login01 ~]$ lsload r04n48    #查询r04n48节点运行情况
HOST_NAME       status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
r04n48              ok  40.8  41.1  41.5 100%   0.0   0  2e+5  212G  531M  151G

#################################################################################
# - ut：列表示利用率
# - status列：
#   - ok：表示可以接受新作业，只有这种状态可以接受新作业
#   - closed：表示系统在运行，但已被调度系统关闭，不接受新作业
#   - locku：表示在进行排他性运行
#   - busy：表示负载超过限定
#   - unavail：作业调度系统服务有问题
# - r15s、r1m、r15m列：分别表示15秒、1分钟、15分钟利用率平均
# - tmp：/tmp目录大小
# - swp：swp虚拟内存大小
# - mem：内存大小
# - io：硬盘读写（-l选项时才出现）
#################################################################################
```

2、查看各节点的空闲情况

- 利用bhosts命令可查看当前各节点的空闲情况

```
[para-test@login01 ~]$ bhosts | more     #查看所有队列节点的空闲情况
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
gpu01              closed          -     40     40     40      0      0      0
gpu02              closed          -     40     40     40      0      0      0
gpu03              closed          -     40     40     40      0      0      0
gpu04              closed          -     40     40     40      0      0      0
lico01             unavail         -      0      0      0      0      0      0
login01            closed          -      0      0      0      0      0      0
login02            closed          -      0      0      0      0      0      0
login03            closed          -      0      0      0      0      0      0
login04            closed          -      0      0      0      0      0      0
lsf01              closed          -     16      0      0      0      0      0
lsf02              closed          -      8      0      0      0      0      0
r01n01             ok              -     40     32     32      0      0      0
r01n02             closed          -     40     40     40      0      0      0
r01n03             ok              -     40      0      0      0      0      0
```

```
.....................................................................................
]$ bhosts  hg_ser   #查看hg_ser节点组里的节点的空闲情况
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
r01n01             ok              -     40      2      2      0      0      0
r01n02             closed          -     40     40     40      0      0      0
r01n03             closed          -     40     40     40      0      0      0
r01n04             ok              -     40     20     20      0      0      0
r01n05             closed          -     40     40     40      0      0      0
.....................................................................................
]$ bhosts  r01n02   #查看r01n02节点的空闲情况
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
r01n02             closed          -     40     40     40      0      0      0
]$ bhosts -l r01n02 #查看r01n02节点的详细信息
HOST  r01n02
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
DISPATCH_WINDOW
closed_Full     60.00     -     40     40     40     0      0        0       -
 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem
slots
 Total           1.0   1.0   1.0  100%   0.0     2    0  7188  212G  154M 165.3G
  0
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M
   -
 LOAD THRESHOLD USED FOR SCHEDULING:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -
 CONFIGURED AFFINITY CPU LIST: all
#####################################################################################
# - STATUS:
#   - ok：表示可以接收新作业，只有这种状态可以接受新作业
#   - closed：表示已被作业占满，不接受新作业
#   - unavail和unreach：系统停机或作业调度系统服务有问题
# - r15s、r1m、r15m列：分别表示15秒、1分钟、15分钟利用率平均（-l选项时才出现）
# - tmp：/tmp目录大小（-l选项时才出现）
# - swp：swp虚拟内存大小（-l选项时才出现）
# - mem：内存大小（-l选项时才出现）
# - io：硬盘读写（-l选项时才出现）
# - bhosts -l：其中slots表示目前最大可以接受作业槽数（默认一般与CPU核数一致）
#####################################################################################
```

3、查看各队列的节点

```
[para-test@login01 ~]$ bmgroup  #查看该集群所有的节点组和节点组里的节点
GROUP_NAME    HOSTS
hg_r01        r01n61 r01n18 r01n62 r01n19 r01n63 r01n64 r01n40 r01n41 r01n42
r01n43
hg_r02        r02n40 r02n41 r02n42 r02n43 r02n44 r02n45 r02n46 r02n47 r02n48
r02n49
hg_r03        r03n44 r03n45 r03n46 r03n47 r03n48 r03n49 r03n20 r03n21 r03n22
r03n23
hg_skl6148    hg_debug/ hg_r13a/ hg_r10/ hg_r11/ hg_r12/ hg_r01/ hg_r02/ hg_r03/
hg_r04/ hg_r05/ hg_r06/ hg_r07/ hg_r08/ hg_r09/
hg_large      hg_r09/
hg_ser        hg_r01_1/
hg_medium     hg_r10_2/ hg_r01_2/ hg_r12/ hg_r03/ hg_r04/
hg_short      hg_r10_1/ hg_r13a/ hg_r11/ hg_r02/ hg_r06_1/ hg_r06_2/ hg_r05/
hg_r07/ hg_r08/
hg_v100       gpu01 gpu02 gpu03 gpu04
hg_skl8160    s001 s002
master_hosts lsf01 lsf02
```

五、查看用户信息：busers

1、查看用户信息

- 利用busers可以查看用户信息

```
[para-test@login03 ~]$ busers para-test
USER/GROUP          JL/P    MAX  NJOBS   PEND    RUN  SSUSP  USUSP    RSV
para-test              -      -      0      0      0      0      0      0
################################################################################
# - 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
################################################################################
```

六、提交作业： bsub

1、格式一（普通）

```
#!/bin/bash
#BSUB -J vasp
#BSUB -q test
#BSUB -n 80
#BSUB -e %J.err
#BSUB -o %J.out
#BSUB -R "span[ptile=40]"
cd $LS_SUBCWD
echo -n "start time  " > time ; date >> time
#-------------intelmpi+ifort-----------------------------------------
source
/share/intel/2017u8/compilers_and_libraries_2017.8.262/linux/bin/compilervars.sh
-arch intel64 -platform linux
```

```
source
/share/intel/2017u8/compilers_and_libraries_2017.8.262/linux/mpi/intel64/bin/mpi
vars.sh intel64
vasp_std="/share/apps/vasp/5.4.4/vasp_std"
vasp_gam="/share/apps/vasp/5.4.4/vasp_gam"
vasp_ncl="/share/apps/vasp/5.4.4/vasp_ncl"
#-----------------------------------------------------------------------
#$COMMAND_std > $LSB_JOBID.log 2>&1
mpiexec.hydra -machinefile $LSB_DJOB_HOSTFILE -np $LSB_DJOB_NUMPROC $vasp_std >
stdout 2>&1
echo -n "end   time  " >> time ; date >> time
```

2、格式二（指定每个节点使用多少核）

```
#!/bin/bash
#BSUB -J test
#BSUB -q test
#BSUB -n 200
#BSUB -e %J.err
#BSUB -o %J.out
#BSUB -R "span[ptile=40]"
#
NP=`expr $LSB_DJOB_NUMPROC / 2`
echo $LSB_HOSTS |sed 's/ /\n/g'|sort|uniq >mycluster
nodes=`paste -d, -s mycluster`
cd $LS_SUBCWD
#
#-------------intelmpi+ifort-------------------------------------------
source /share/intel/2018u4/compilers_and_libraries/linux/bin/compilervars.sh -
arch intel64 -platform linux
source /share/intel/2018u4/impi/2018.4.274/intel64/bin/mpivars.sh
module load vasp/5.4.4
vasp_std="/share/apps/vasp/5.4.4/vasp_std"
vasp_gam="/share/apps/vasp/5.4.4/vasp_gam"
vasp_ncl="/share/apps/vasp/5.4.4/vasp_ncl"
#-----------------------------------------------------------------------
mpirun -np $NP  -ppn 20 -host $nodes  $vasp_std > $LSB_JOBID.log 2>&1
#-------------------------------------------------------------------------------
bsub <  test.lsf

参数解析：
-np $NP ：  实际要运行的总核数
-ppn 20 ：  每个节点需要运行多少核
-host $nodes ：  指定运行在那些节点上，如果不指定将无法实现每个节点运行20核
```

3、提交到特定队列

- 利用-q选项可以指定提交到哪个队列

```
]$ bsub -q ser < mytest.lsf
Job <1436712> is submitted to queue <ser>.
```

- 如不加-q，那么将提交到系统设置的默认队列

```
]$ bsub < mytest.lsf   #不加-q的前提是申请的核数需大于或等40核，因为默认队列是short有核数限制
Job <1436715> is submitted to default queue <short>.
]$ bsub < mytest.lsf   #申请的核数少于40，不指定队列ser无法提交。
Too few tasks requested. Job not submitted.
```

- 提交到ser队列运行串行程序

```
]$ bsub  -q ser  executable1    #如提交成功，将显示类似下面的输出：
Job <79722> is submitted to queue <ser >.
#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。
#如提交不成功，则显示相应提示
```

4、指明所需要的CPU核数

- 利用-n [min_proc,max_proc]选项指定所需要的CPU核数（一般来说核数应和进程数或线程数一致）

```
]$ bsub -n 16 < mytest.lsf     #采用16CPU核运行
]$ bsub -n 16,64 < mytest.lsf  #最少采用16最大采用64CPU核数运行
```

- 注：当作业调度尝试开始运行时
  - 如空闲资源满足64CPU核，那么将采用64CPU核运行
  - 如空闲资源不满足64CPU核，但满足16CPU核，则用16CPU核运行
- 以上仅仅是建议，具体申请核数应考虑作业实际情况
- 即使同一个计算软件，在计算不同条件时，也有可能不一样，请务必仔细研究自己所使用的软件

5、提交到特定节点

- 利用-m选项可以指定作业在特定节点上运行

```
]$ bsub -m "r01n01 r02n03"  > test.lsf
```

如非必要，建议不要添加此选项，以免导致作业无法及时运行

6、给作业起个名字

- 提交时可以利用-J选项给作业起个名字方便查看

```
]$ bsub -J VASPJOB < mytest.lsf
```

7、OpenMP等共享内存作业提交

- 只能共享同一节点内的内存，需要保证在同一个节点内运行， 如Gaussian程序
- 程序启动前利用OMP_NUM_THREADS设定指定的线程数，一般应与申请的核数一致
- 指定利用8CPU核运行OpenMP程序：

```
]$ bsub -q ser -n 8 -R "span[hosts=1]" OMP_NUM_THREADS=8 executable
```

注：-R "span[hosts=1]"保证在同一个节点内

8、指明需要某种资源作业提交

```
]$ bsub -R "res_req" [-R "res_req" ...] #可以使得作业在需要满足某种条件的节点上运行
]$ bsub -R "span[hosts=1]"  #指定需要在同一个节点内运行
]$ bsub -R "1*{mem>5000} + 9*{mem>1000}" #一个CPU核需要至少5GB内存，另外9个每个至少需要1GB内存
```

9、串行作业的提交

- 运行串行作业，请指明-n 1

```
]$ bsub -q ser -n 1 executable-serial
```

10、运行排他性运行作业

- 如果需要独占节点运行，此时需要添加-x选项：
  - 注：太乙集群小于40核的作业暂不支持独占节点

```
]$ bsub -x -q ser -n 4 < mytest.lsf
```

- 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CPU核，也将按照此节点内的所有CPU核数进行机时计算

11、指明输入、输出文件运行

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：
  - exec1 < file1，可用bsub -i file1 exec方式提交
  - exec1 file1或exec1 -i file1等，则不可用bsub -i file1 exec方式提交
- 如指定exec1的屏幕输入、正常和错误屏幕输出文件分别为exec1.input、 exec1-%J.log和exec1-%J.err

```
]$ bsub -i exec1.input -o exec1 -%J.log -e exec1 -%J.err exec1
################################################################################
# -o和-e及变种-oo和-eo：
# -o     如日志原文件存在，正常屏幕输出将追加到原文件后
# -oo    如日志原文件存在，正常屏幕输出将覆盖原文件
# -e     如日志原文件存在，出错时屏幕输出将追加到原文件后
# -eo    如日志原文件存在，出错时屏幕输出将覆盖原文件
################################################################################
```

12、指明输出目录提交

- 默认情况下，屏幕输出等存放在提交作业时的目录下
- 如想将其放到其它目录可以采用bsub −outdir output directory 方式

```
]$ bsub -outdir "%U/%J_%I"  myprog < mytest.lsf
################################################################################
# 常用变量：
# - %J：作业号
# - %JG：作业组
# - %I：作业组中的索引
# - %EJ：执行作业号
# - %EI：作业组中的执行作业索引
# - %P：作业名
# - %U：用户名
# - %G：用户组名
################################################################################
```

13、交互式运行作业

- 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、−Ip和−Is等参数
- 建议只在调试期间使用，一般作业尽量不要使用此选项

```
]$ bsub -I executable1
```

14、满足依赖关系运行作业

- 利用−w选项可以使得新提交的作业在满足一定条件时才运行，比如与其它作业的关联：
  - done(job_ID | "job_name")：作业结束时状态为DONE时运行
  - ended(job_ID | "job_name")：作业结束时状态为DONE或EXIT时运行
  - exit(job_ID| "job_name" [,[operator]exit_code])：作业结束时状态为EXIT，且退出代码满足一定条件时运行
  - external(job_ID| "job_name","status_text")：作业状态变为某状态时运行，如变为SUSP
  - 支持的条件之间的条件表达式：&&（和）、||（或）、!（否）
  - 支持的条件内的条件算子：>、>=、<、<=、==、!=

```
# 当1456号作业计算正常结束后，该作业才会运行
]$ bsub -w "done(1456)" < mytest.lsf
```

15、指定时间运行

- 可以使得新提交的作业在特定时间运行

```
]$ bsub -b  [[year:][month:]day:]hour:minute
```

- 系统会预留资源给此作业，如果在时间到达时所需资源满足则会运行，不满足则继续等待

16、指定运行时长

- 可以使得提交的作业在超过设定时长后终止

```
]$ bsub -W [hour:]minute
```

17、在运行前执行特定命令

- 可以使得作业在运行前，在所分配的节点上运行特定命令

```
]$ bsub -E  pre_command
```

- 如利用此修改程序输入参数

```
]$ bsub -E ./modinput.sh 10 exec1
```

18、在运行后执行特定命令：

- 可以使得作业在运行结束时，在所分配的节点上运行特定命令

```
]$ bsub -Ep post_command
```

- 如利用此删除core文件：

```
]$ bsub -Ep /bin/rm –f  core.* exec1
```

19、LSF作业脚本

- 以在LSF脚本中设置队列等参数方式提交，如mytest.lsf

```
]$ vi mytest.lsf
#!/bin/bash     #声明解释器是bash
#BSUB -J test   #定义作业名称
#BSUB -q ser    #指定使用的队列
#BSUB -n 80     #作业的总核数
#BSUB -R "span[ptile=40]"   #指定每个节点使用四十核，如不指定则自动分配。
#BSUB -W 12:00  #指定作业运行时间
#BSUB -R "select[hname!='r13n18']"  #排除使用那个节点进行计算，如不指定则随机分配
#BSUB -e %J.log #指定错误输出日志
#BSUB -o %J.out     #指定作业完成后输出的日志
#module load intel/2018.4 mpi/intel/2018.4 vasp/5.4.4   #加载环境变量
source /work/para-test/yys/test.sh &> /work/para-test/yys/100.txt   #运行脚本进行计
算
```

- 不得以直接./my_script. lsf等常规脚本运行方式需要用$<$传递给bsub命令运行：

```
]$ bsub  <  my_script. lsf
```

- 如果bsub后面跟-q等LSF参数，将会覆盖掉LSF脚本中的设置
- 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OpenMP结合的作业。

20、LSF作业脚本常见变量

- 主要有以下变量比较常用，在作业运行后，这些变量存储对应的作业信息，具体的请参看LSF官方手册：

```
##############################################################################
```

```
# LS_JOBPID：作业进程号
# LSB_HOSTS：调度系统分配的节点名
# 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
# LSB_DJOB_HOSTFILE：包含节点列表的主机文件，用于运行批处理作业的
# LSB_MCPU_HOSTS：节点列表和指定的或默认的，运行批处理作业的每个节点的ptile值。
# LS_SUBCWD：提交作业时的目录路径
# LSB_DJOB_NUMPROC：申请核数
# LSF官方资料：http://scc.ustc.edu.cn/zlsc/lsf/
################################################################################
```

七、查看作业情况：bjobs、bpeek

1、查看作业的排队和运行情况

- 利用bjobs可以查看作业的运行情况

```
]$ bjobs        #查看所有作业运行的情况（精简）
JOBID   USER    STAT  QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1433531 para-te RUN   ser       login01      r13n44      test       Mar 8 20:24
################################################################################
# 下表对常见的作业状态解释：
# PEND：作业正在队列中排队
# RUN：作业正在被执行
# DONE：作业已经执行完毕，并且正常退出
# EXITED：作业非正常退出
# PSUSP：作业在排队过程中被挂起
# USUSP：作业在运行过程中被人为强制挂起
# SSUSP：作业在运行过程中被系统挂起
################################################################################
]$ bjobs -l     #查看所有作业运行的详细情况
Job <1433531>, Job Name <test>, User <para-test>, Project <default>, Status <PE
                     ND>, Queue <ser>, Command <#BSUB -J test;#BSUB -q ser;#BSU
                     B -n 1;#BSUB -R "span[ptile=40]";#BSUB -e %J.err;#BSUB -o
                     %J.out;#BSUB -m "r01n03";module load intel/2018.4 mpi/inte
                     l/2018.4 vasp/5.4.4;#mpirun -machinefile $LSB_DJOB_HOSTFIL
                     E -np 100 echo -n 1 &> /work/para-test/yys/100.txt;hostnam
                     e>
Mon Mar  8 20:24:11: Submitted from host <login01>, CWD <$HOME/yys>, Output Fil
                     e <1433531.out>, Error File <1433531.err>, Re-runnable, Re
                     quested Resources <span[ptile=40]>, Specified Hosts <r01n0
                     3>;
 RUNLIMIT
 21600.0 min of lsf01
 PENDING REASONS:
 Job slot limit reached: 1 host;
 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut    pg   io   ls    it    tmp    swp    mem
 loadSched  -     -     -     -     -     -    -     -     -      -      -
 loadStop   -     -     -     -     -     -    -     -     -      -      -
 RESOURCE REQUIREMENT DETAILS:
 Combined: select[type == any] order[-slots] span[ptile=40]
```

```
 Effective: -
]$ bjobs -l JOBID   #查看 JOBID 这个作业的详细情况
]$ bpeek -f JOBID   #跟踪查看某任务屏幕输出
]$ bjobs -p 1433531 #查看作业仍在排队等待的原因
JOBID    USER    STAT  QUEUE    FROM_HOST    EXEC_HOST    JOB_NAME    SUBMIT_TIME
1433531 para-te PEND  ser      login01                    test       Mar 8 20:24
 Job slot limit reached: 1 host;
```

八、管理作业：bkill、bstop、bresume、btop、bbot、bmod

```
]$ bkill  JOBID     #终止JOBID 这个作业运行
]$ bkill JOBID1 JOBID2 JOBID3   #终止多个作业运行
]$ bstop JOBID      #临时挂起某个计算作业，为其它计算腾出资源
]$ bresume JOBID    #恢复由 bstop 挂起的作业
]$ lshosts | more   #查看节点的信息
]$ bhosts | more    #查看节点的作业使用信息
]$ bqueues          #查看所有任务队列的状态
]$ bjobs -p         #查看作业仍在排队等待的原因
]$ btop 1433531     #设置作业最先运行
Job <1433531> has been moved to position 1 from top.
]$ bbot 1433531     #设置作业最后运行
Job <1433531> has been moved to position 1 from bottom.
]$ bswitch medium 3048471  #将作业3048471移动到medium队列运行

#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other队 列，并且所需要CPU核数为12：
]$ bmod -z executable2 -q short -n 12 79727

#显示作业的当前估计完成百分比。
[para-test@login01 ~]$ bjobs -WP 2336827
JOBID      USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME
SUBMIT_TIME   %COMPLETE
2336827    chem-pe RUN   short      login03     40*r11n09   1-f6-2     Oct  4
14:15  0.55% L
#显示作业的估计剩余运行时间。
[para-test@login01 ~]$ bjobs -WL 2336827
JOBID      USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME
SUBMIT_TIME   TIME_LEFT
2336827    chem-pe RUN   short      login03     40*r11n09   1-f6-2     Oct  4
14:15  71:35 L
#显示正在运行或挂起的作业的预计完成时间。 对于已完成或已退出的作业，显示实际完成时间。
[para-test@login01 ~]$ bjobs -WF 2336827
JOBID      USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME
SUBMIT_TIME   FINISH_TIME
2336827    chem-pe RUN   short      login03     40*r11n09   1-f6-2     Oct  4
14:15  Oct  7 14:15 L
#显示以下状态的作业槽数：运行 (RUN)、系统挂起 (SSUSP)、用户挂起 (USUSP)、挂起 (PEND)、转发到远程集群和挂起(FWD_PEND) 和未知。
[para-test@login01 ~]$ bjobs -sum
RUN         SSUSP       USUSP       UNKNOWN     PEND        FWD_PEND    PSUSP
0           0           0           0           0           0           0

[para-test@login01 bin]$ bjobs   -noheader -sum 2336889
```

```
200        0          0          0          0          0          0
[para-test@login01 bin]$ bjobs   -noheader  2336889
2336889    mse-wan RUN   medium     login01     40*r03n36   Wu-MAPB    Oct  4
14:50
                                              40*r03n64
                                              40*r12n10
                                              40*r12n14
                                              40*r03n01
```