LSF作业调度系统的使用

河南师范大学 高性能计算中心

2015-12-5

1. LSF作业管理系统简介
2. 查看队列情况：bqueues
3. 高性能计算系统现有队列
4. 查看计算节点信息：lsload、bhosts
5. 查看用户信息：busers
6. 提交作业：bsub
7. 查看作业情况：bjobs、bpeek
8. 查看作业历史统计信息：bacct
9. 管理作业：bkill、bstop、bresume、btop、bbot、bmod
10. 联系信息

作业调度系统的用途

- 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

LSF作业管理系统的简介

- 当前河南师大高性能计算中心的高性能计算系统采用IBM Platform LSF V9.1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
- 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
- 应提交到合适的队列
- 提交后可利用相关命令查询作业状态等

注意：

- 登录节点（节点名：login）主要用于日常操作，如提交作业、查看作业运行情况、编辑、编译、压缩/解压缩等
- *不要在登录节点不通过作业调度系统直接运行作业*，以免影响其余用户的正常使用

查看队列情况：bqueues I

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
nc	30	Open:Active	–	–	–	–	0	0	0	0
chem	30	Open:Active	–	–	–	–	224	0	224	0
math	30	Open:Active	–	–	–	–	272	80	192	0
other	30	Open:Active	–	–	–	–	0	0	0	0
phy_long	30	Open:Active	–	–	–	–	720	224	496	0
phy_7days	30	Open:Active	–	–	–	–	1056	240	816	0
phy_1day	30	Open:Active	–	–	–	–	0	0	0	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

查看队列情况：bqueues II

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
chem	30	Open:Active	–	–	–	–	224	0	224	0

- MAX：队列对应的最大作业槽数（Job Slot，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核数
- 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

查看队列详细情况：bqueues -l l

队列也许会调整，利用*bqueues␣-l␣[队列名]*查看各队列的详细情况

```
QUEUE: phy_1day
  -- For phy_1day priority jobs .

PARAMETERS/STATISTICS
PRIO NICE STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN SSUSP USUSP  RSV
 30    0  Open:Active       -    -    -    -     0     0     0     0     0     0
Interval for a host to accept two jobs is 0 seconds

 RUNLIMIT
 1440.0 min of manager
 PROCLIMIT
 2 4 8

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

. . .

USERS: ugroupPHY/
HOSTS: groupPHY_1day/
```

查看队列详细情况： bqueues -l II

- 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，1440.0 min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

现有队列

- nc：所有用户可使用
- chem：ugroupCHEM组内用户可使用[1]，运行在groupCHEM节点上[2]
- math：ugroupMATH组内用户可使用，运行在groupMATH节点上
- other：ugroupMATH组内用户可使用，运行在groupOTHER节点上
- phy_long：ugroupPHY_long组内用户可使用，运行在groupPHY_long节点上
- phy_7days：ugroupPHY_7days组内用户可使用，运行在groupPHY_7days节点上
- phy_1day：ugroupPHY_1day组内用户可使用，运行在groupPHY_1day节点上

建议以16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

[1]bugroup ugroupCHEM可以查看组内成员

[2]bhosts groupCHEM可以查看节点名

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
c01n10     ok      0.0   0.0  0.0   0%  3.5  0  2050  9032M  4000M  16G
c01n11     locku   0.0   0.0  0.0   0%  3.5  0  2050  9032M  4000M  16G
```

  - 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、-ok：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）
- 查看c01n02节点：*lsload␣c01n02*

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
c01n01	closed	–	4	2	2	0	0	0
c02n03	ok	–	2	2	1	0	0	0

  - 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

- 查看c01n02节点：*bhosts␣c01n02*
- 查看groupCHEM节点组：*bhosts␣groupCHEM*
- *bhosts␣–l*会显示节点详细信息，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）

查看用户信息：busers

- 利用*busers*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

  - 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

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]

- command之*前*的options：设置队列、CPU核数等LSF的选项
- command之*后*的arguments：设置作业的可执行程序本身所需要的参数
- 作业提交后，应经常检查一下作业的CPU、内存等利用率，判断实际运行效率
  - 可以*ssh*到对应运行作业的节点运行*top*命令
  - 查看Ganglia系统监控：http://59.70.88.121/ganglia
- 请不要ssh到节点后直接运行作业，以免影响作业调度系统分配到此节点的作业

提交到特定队列： bsub -q queue

- 利用*−q*选项可以指定提交到哪个队列
- 如不加*−q*，那么将提交到系统设置的默认队列[3]
- 提交到other队列运行串行程序executable1：
  bsub␣−q␣other␣executable1
  - 如提交成功，将显示类似下面的输出：

```
Job <79722> is submitted to queue <other>.
```

  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

[3]除非了解哪个是默认队列，且默认队列适合此作业，否则不要这么做

指明所需要的CPU核数： bsub -n min_proc[,max_proc]

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），如：

- 采用16 CPU核运行： *bsub␣−n␣16*
- 最少采用16最大采用64 CPU核数运行： *bsub␣−n␣16,64*
  注： 当作业调度尝试开始运行时
  - 如空闲资源满足64 CPU核，那么将采用64 CPU核运行
  - 如空闲资源不满足64 CPU核，但满足16 CPU核，则用16 CPU核运行
- 为了用户作业间不相互干扰，申请的核数最好为系统节点内CPU核数的整数倍，以便同一个作业占据整个节点
- 以上仅仅是建议，具体申请核数应考虑作业实际情况[4]

[4]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

提交到特定节点：bsub -m ”host_name”

- 利用*–m*选项可以指定作业在特定节点上运行，如：*bsub␣–m␣”c01n01␣c02n03”*
- 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

MPI作业的提交：bsub mpijob

- MPI作业一般需要用提交时使用*mpijob*来执行MPI程序
- 指定利用16 CPU核（由*-n␣16*指定）运行MPI程序：
  bsub␣-q␣other␣-n␣16␣mpijob␣executable␣-mpi1

注：*mpijob*命令：

- 适合不通过作业调度系统时直接以*mpirun*或*mpiexec*方式运行的作业
- 不是*mpirun*或*mpiexec*方式运行的作业不能直接这么使用
- 如您了解所使用的MPI环境与LSF的配合，也可不用*mpijob*提交，如直接用*mpirun*等命令运行

给作业起个名字：bsub -P project_name

提交时可以利用*–P*选项给作业起个名字方便查看，如：

- *bsub␣–P␣VASPJOB*

OpenMP等共享内存作业提交： bsub OMP_NUM_THREADS

- 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[5]
- 指定利用8 CPU核运行OpenMP程序：

  bsub␣−q␣other␣−n␣8␣−R␣"span[hosts=1]"␣OMP_NUM_THREADS=8␣exec

  注：*−R␣"span[hosts=1]"*保证在同一个节点内

[5]有些程序可以在输入文件中设置，那么可以不填加OMP_NUM_THREADS

指明需要某种资源作业提交：bsub -R

-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，如：

- *-R␣"span[hosts=1]"*：指定需要在同一个节点内运行
- *-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

串行作业的提交：bsub -n 1

- 运行串行作业，请指明*–n␣1*：
  bsub␣–q␣other␣–n␣1␣executable–serial

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␣other␣–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CPU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：
  - 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

指明输出目录提交：bsub -outdir output_directory

- 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
- 如：*bsub␣−outdir␣"%U/%J_%I"␣myprog*
- 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

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*、*–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

满足依赖关系运行作业：bsub -w

- 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：>、>=、<、<=、==、!=
- 如：*bsub␣–w␣”done(1456)”*

指定时间运行： bsub -b time

- 利用*–b␣[[year:][month:]day:]hour:minute*可以使得新提交的作业在特定时间运行
- 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

指定运行时长：bsub -W time

- 利用*–W␣[hour:]minute*可以使得提交的作业在超过设定时长后终止

在运行前执行特定命令： bsub -E pre_command

- 利用*–E␣”pre_exec_command␣[arguments␣...]”*可以使得作业在运行前，在所分配的节点上运行特定命令
- 如利用此修改程序输入参数： *bsub␣–E␣./modinput.sh␣10␣exec1*

在运行后执行特定命令： bsub -Ep post_command

- 利用*–Ep␣"post_exec_command␣[arguments␣...]"*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
- 如利用此删除core文件：*bsub␣–Ep␣/bin/rm␣–f␣core.*␣exec1*

LSF作业脚本

- 以在LSF脚本中设置队列等参数方式提交，如*my_script.lsf*

```
#!/bin/sh
#BSUB␣-q␣other
#BSUB␣-o␣%J.log␣-e␣%J.err
#BSUB␣-n␣64
mpijob␣./mympi-prog1
cd␣somedir
mpijob␣./mympi-prog2
```

- 不得以直接*./my_script.lsf*等常规脚本运行方式运行
- 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。

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*：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

LSF官方资料：http://scc.ustc.edu.cn/zlsc/lsf/

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   other      login      2*c01n02   *executab1  Mar 12 19:20
                                         1*c01n04
79727  hmli  PEND  phy_1day   login                 *executab2  Mar 12 19:20
```

显示：

- 作业79726分别在c01n02和c01n04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 <other> , Command <executab2>
Sun Mar 12 14:15:07: Submitted from host <login>,
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␣JobID*，可以连续查看作业的连续屏幕输出
- 如在运行中用*–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

终止作业：bkill

- 利用 *bkill* 命令可以终止某个运行中或排队中的作业，如：
  bkill 」79722

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

- 请及时终止有问题或无需再运行的程序，空出计算资源，降低费用

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other队列，并且所需要CPU核数为12：

  bmod␣-Z␣executable2␣-q␣other␣-n␣12␣79727

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

查看作业历史统计信息：bacct

利用*bacct*可以查看已经结束的作业的历史统计信息，如：
bacct

```
Accounting information about jobs that are:
  - submitted by users hmli,
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------
SUMMARY:      ( time unit: second )
 Total number of done jobs:       73      Total number of exited jobs:    159
 Total CPU time consumed:    14649779.0     Average CPU time consumed: 63145.6
 Maximum CPU time of a job: 4266155.5      Minimum CPU time of a job:      0.0
 Total wait time in queues: 1403570.0
 Average wait time in queue: 6049.9
 Maximum wait time in queue:904361.0       Minimum wait time in queue:     2.0
 Average turnaround time:      11671 (seconds/job)
 Maximum turnaround time:     904480       Minimum turnaround time:          2
 Average hog factor of a job:  7.12 ( cpu time / turnaround time )
 Maximum hog factor of a job:  157.84      Minimum hog factor of a job:  0.00
 Total throughput:              0.02 (jobs/hour)  during10055.28 hours
 Beginning time:       Jan 30 14:40       Ending time:          Mar 25 13:57
```

查看某个作业历史统计信息：bacct -l

bacct␣-l␣13624

```
------------------------------------------------------------------------------
Job <13624>, User <hmli>, Project <default>, Status <DONE>, Queue <other>, Comma
                     nd <mpijob /shared/bin/vasp-2013.12.27>
Mon Mar 24 16:43:24: Submitted from host <login>, CWD <$HOME/test>, Output
                     File <%J.log>, Error File <%J.err>;
Tue Mar 25 13:55:15: Dispatched to 32 Hosts/Processors <16*c01n02> <16*c02n03>;
Tue Mar 25 13:57:07: Completed <done>.
Accounting information about this job:
     CPU_T     WAIT      TURNAROUND   STATUS     HOG_FACTOR    MEM      SWAP
    984.17    76311           76423     done         0.0129     30G       39G
------------------------------------------------------------------------------
SUMMARY:      ( time unit: second )
 Total number of done jobs:       1      Total number of exited jobs:     0
 Total CPU time consumed:       984.2    Average CPU time consumed:   984.2
 Maximum CPU time of a job:     984.2    Minimum CPU time of a job:   984.2
 Total wait time in queues: 76311.0
 Average wait time in queue:76311.0
 Maximum wait time in queue:76311.0      Minimum wait time in queue:76311.0
 Average turnaround time:       76423 (seconds/job)
 Maximum turnaround time:       76423    Minimum turnaround time:      76423
 Average hog factor of a job:    0.01 ( cpu time / turnaround time )
 Maximum hog factor of a job:    0.01    Minimum hog factor of a job:    0.01
------------------------------------------------------------------------------
```

查看某时间段内作业历史统计信息：bacct -C -D -S

在2014/03/01,2014/04/01时间段内：

- C：完成的：*bacct␣-l␣-C␣2014/03/01,2014/04/01*
- D：开始运行的：*bacct␣-l␣-D␣2014/03/01,2014/04/01*
- S：提交的：*bacct␣-l␣-S␣2014/03/01,2014/04/01*

联系信息

河南师范大学高性能计算中心：

- 电话：0373-3326142
- 信箱：cxq@htu.cn
- QQ群：171803133
- 主页：http://www.htu.cn/hpcc/
- 办公室：河南师大琢玉楼3楼