新一代超级计算机“元”
用户快速使用指南

中国科学院计算机网络信息中心

超级计算中心

2015 年 2 月 3 日

目录

注意事项

1. 《新一代超级计算机“元”用户快速使用指南》、《新一代超级计算机“元”远程访问指南》及各应用软件快速使用指南最新版可登陆元系统后到/soft/doc 及/soft/doc/soft 目录下载。

2. 深腾 7000 HOME 数据（2014 年 11 月 15 日之前的数据）已经迁移至元/home/.7000 下，用户可自行将所需数据移至元系统的相应目录下，两个目录下的数据同样占用磁盘份额，无用数据请尽快清理。

3. 程序编译和链接须在 login1-login4 上进行，MIC 程序的编译和链接请在节点 m3101 和 m3102 上运行。

4. 请用 bqueues 命令查看可用队列,用 bqueues －l〈队列名〉查看队列具体设置。对队列资源有特殊要求的用户，可与客服部联系开通专用队列。

5. 如有其它问题可发邮件至 support@sccas.cn 邮箱，邮件中务必提供账号名、出错作业号、输出文件、错误文件或出错信息，关于登陆问题最好能提供截图。提供的信息越明确，越有助于问题快速解决。

6. 对于不通过作业系统提交的非法程序，资源利用率低（占多用少）、影响其他用户作业运行、影响系统正常运行的作业，我们将随时在不通知用户的情况下予以清除。

7. 用于用 bsub 命令提交作业时，ptile 指定数值越小，需要申请的节点就越多，作业越不容易得到调度， ptile 过小会造成资源和机时费的浪费。

1. 基本环境简介

1.1. 硬件环境

新一代超级计算机“元”一期计算系统总体计算能力 303.4Tflops,其中 CPU 峰值 152.32Tflops，MIC 和 GPU 协处理器峰值 151.08Tflops，具体分布如下：

- 刀片计算系统共有 270 台曙光 CB60-G16 双路刀片，CPU 整体峰值性能达到 120.96Tflops。每台刀片计算节点配置 2 颗 Intel E5-2680 V2（Ivy Bridge | 10C | 2.8GHz）处理器，64 GB DDR3 ECC 1866MHz 内存。
- GPGPU 计算系统共有 30 台曙光 I620-G15，GPU 双精度浮点峰值达到 83.64Tflops，其中 CPU 峰值 13.44Tflops，GPU 协处理器峰值 70.2Tflops。每台 GPGPU 计算节点配置 2 块 Nvidia Tesla K20 GPGPU 卡，2 颗 Intel E5-2680 V2（Ivy Bridge | 10C | 2.8GHz）处理器，64 GB DDR3 ECC 1866MHz 内存。支持 CUDA、OpenACC、OpenCL，支持 GPU Direct。
- MIC 计算系统共有 40 台曙光 I620-G15，MIC 双精度浮点峰值达到 98.8 Tflops，其中 CPU 峰值 17.92Tflops，MIC 协处理器峰值 80.88Tflops。每台 MIC 计算节点配置 2 块 Intel Xeon Phi 5110P（8GB 内存）卡，2 颗 Intel E5-2680 V2（Ivy Bridge | 10C | 2.8GHz）处理器，64 GB DDR3 ECC 1866MHz 内存。支持对 Xeon Phi 的 Offload 卸载、Symmetric、Native 原生模式调用。
- 系统配置 1 套采用 56Gb FDR InfiniBand，全线速互连。
- 系统采用 Stornext 并行文件系统做用户 HOME 和公共软件区存储，可靠性高，用户可用容量 165TB；采用曙光 Parastor200 做高性能工作区存储系统，I/O 带宽高，用户可用容量为 1.3PB。
  - 用户 HOME 目录：/home 采用 SNFS 文件系统；
  - 软件安装目录：/soft 采用 SNFS 文件系统；
  - 用户工作目录：/work1 采用曙光 Parastor200 并行存储系统。

- 系统配置 4 台登陆节点，通过均衡负载实现单 IP 登录。
- SGI UV2000 计算系统共有 32 颗 Intel Xeon E5-4620 八核处理器，主频 2.60GHz，系统共享内存 4TB，采用 NUMA 结构，单一系统映像，系统峰值约 5Tflops。

1.2. 软件环境

目前，除操作系统自带软件之外，各类软件都安装/soft 下，大致安装路径是：

主要分类	路径
MPI	/soft/mpi
数学库	/soft/mathlib
工具软件	/soft/tools
CUDA	/soft/cuda
应用软件	/soft/apps
付费应用软件	/soft/apps/Paid

其中应用软件在本系统上的使用指南放在/soft/doc/soft 下，系统的部分基础工具软件安装情况及环境变量设置方法如下：

名称	版本	安装目录	设置环境变量
GNU Compiler	4.4.7	/usr/bin	自动添加
Intel Compiler	2013_sp1.0.080	/soft/compiler/intel/composer_xe_2013_sp1.0.080	module load compiler/intel/composer_xe_2013_sp1.0.080
Intel MIC Compiler	2013_sp1.0.080	/soft/compiler/intel/composer_xe_2013_sp1.0.080	module load mic/compiler/intel/composer_xe_2013_sp1.0.080
PGI Compiler	2014 (14.10)	/soft/compiler/pgi	module load compiler/pgi/2014
Cuda	6.0.37	/soft/cuda/6.0.37	module load cuda/6.0.37
openmpi-intel	1.6.5	/soft/mpi/openmpi/1.6.5/intel	module load mpi/openmpi/1.6.5/intel
mvapich2-intel	1.9	/soft/mpi/mvapich2/1.9/intel	module load mpi/mvapich2/1.9/intel
Intelmpi	4.1.3.049	/soft/mpi/impi/4.1.3.049	module load

			mpi/impi/4.1.3.049
MKL	11.1	/soft/compiler/intel/composer_xe_2013_sp1.0.080	module load mathlib/mkl/11.1
ACML	5.3.1	/soft/mathlib/acml/5.3.1/ifort	module load mathlib/acml/5.3.1/ifort
fftw2-float	2.1.5	/soft/mathlib/fftw/2.1.5/float	module load mathlib/fftw/2.1.5/float
fftw2-double	2.1.5	/soft/mathlib/fftw/2.1.5/double	module load mathlib/fftw/2.1.5/double
fftw3-float	3.3.4	/soft/mathlib/fftw/3.3.4/float	module load mathlib/fftw/3.3.4/float
fftw3-double	3.3.4	/soft/mathlib/fftw/3.3.4/double	module load mathlib/fftw/3.3.4/double
ATLAS	3.11.24	/soft/mathlib/atlas/3.11.24	module load mathlib/atlas/3.11.24
LAPACK	3.5.0	/soft/mathlib/lapack/3.5.0/gnu	module load mathlib/lapack/3.5.0/gnu
MAGMA	1.5.0	/soft/mathlib/magma/1.5.0	module load mathlib/magma/1.5.0
BOOST	1.55.0	/soft/mathlib/boost/1.55.0gnu	module load mathlib/boost/1.55.0/gnu
CMAKE 2	2.8.12.2	/soft/tools/cmake/2.8.12.2	module load tools/cmake/2.8.12.2
CMAKE 3	3.0.2	/soft/tools/cmake/3.0.2	module load tools/cmake/3.0.2
QT 4	4.8.6	/soft/tools/qt/4.8.6	module load tools/qt/4.8.6
QT 5	5.3.2	/soft/tools/qt/5.3.2	module load tools/qt/5.3.2

注：若提交的作业需要用到以上工具软件，请在.bashrc 中设置相应的环境变量。

2. 上机使用

2.1. 系统登录和数据传输

关于新一代超级计算机“元”的远程登陆，数据的上传、下载请参见《新一代超级计算机“元”远程访问指南》。

2.2. 环境变量设置

“元”系统使用 module 程序管理环境变量，module 是环境变量模块化管理工具，可以自动处理编译器及函数库的依赖。具体使用方式如下：

```
module avail                # 查看可用环境变量
module load XXX             # XXX 加载某环境变量
module list                 # 查看已加载环境变量
module unload XXX           # XXX 卸载某环境变量
module purge                # 清除所有环境变量
```

建议用户将平常使用的编译器和 MPI 环境变量写在自己的~/.bashrc 文件里。例如：

- 设置 Intel 编译器的环境变量

  module load compiler/intel/composer_xe_2013_sp1.0.080
- 设置 openmpi1.6.5 的环境变量

  module load mpi/openmpi/1.6.5/intel
- 设置 intelmpi4.1.3.049 的环境变量

  module load mpi/impi/4.1.3.049
- 设置 mvapich2 1.9 的环境变量

  module load mpi/mvapich2/1.9/intel

用户可以用 module avail 查看所需软件环境变量的设置路径，然后用 module load 进行设置。

MPT(Message Passing Toolkit)是 SGI 系统上用于进程间数据交换的软件包，支持 MPI 和 SHMEM 编程。在 SGI UV2000 上使用 mpt2.10，请按如下方式设置环境变量。

ssh sgi

module load mpt-2.10

2.3.程序编译

“元”系统提供 Intel 公司的 C/C++/Fortran 编译器、GNU 编译器，建议用户使用 Intel 编译器进行编译。

注：程序编译和链接须在 login1-login4 上进行，MIC 程序的编译和链接请在节点 m3101 和 m3102 上运行，MPT 程序的编译、链接和运行请在 SGI UV2000 节点上操作。

2.3.1. 串行程序编译

设置 Intel 编译器的环境变量，使用 icc/ifort 进行编译。如：

```
$ module load compiler/intel/composer_xe_2013_sp1.0.080
$ icc -o test test.c
$ ifort -o hello hello.f90
```

编译器各个参数的用法参见 icc –help /ifort –help。

2.3.2. OpenMP 程序编译

Intel 编译器支持 OpenMP 并行，设置 Intel 编译器的环境变量，使用 icc/ifort 进行编译。如：

```
$ module load compiler/intel/composer_xe_2013_sp1.0.080
$ icc -openmp -o test test.c
$ ifort -openmp -o hello hello.f90
```

编译器各个参数的用法参见 icc –help /ifort –help。

2.3.3. MPI 程序编译

目前，“元”系统提供对 openmpi1.6.5、intelmpi4.1.3.049、mvapich2 1.9 的支持，各个版本 MPI 编译情况如下：

- 编译 openmpi 程序

  openmpi 提供了 C/C++，Fortran 等语言的 MPI 编译器，如下表所示：

语言类型	MPI 编译器
C	mpicc
C++	mpicxx
Fortran77	mpif77
Fortran90	mpif90

MPI 编译器是对底层编译器的一层包装，通过-show 参数可以查看实际使用的编译器。比如：

```
$ mpicc -show
icc -I/soft/mpi/openmpi/1.6.5/intel/include -pthread
-L/soft/mpi/openmpi/1.6.5/intel/lib -lmpi -ldl -lm
-lnuma -Wl,--export-dynamic -lrt -lnsl -lutil
```

编译程序示例：

```
$ module load mpi/openmpi/1.6.5/intel
$ mpicc -o hello hello.c
$ mpif90 -o hello hello.f90
```

- 编译 intelmpi 程序

intelmpi 提供了非常完整的 MPI 编译器，如下表所示：

MPI 编译器	编译器说明
mpicc，mpigcc	使用 gcc 编译 C 语言
mpicxx，mpigxx	使用 g++编译 C++语言
mpif77	使用 g77 编译 Fortran77 语言
mpif90	使用 gfortran 编译 Fortran90 语言
Mpifc	使用 gfortran 编译 Fortran77/90 语言
Mpiicc	使用 icc 编译 C 语言，推荐使用
Mpiicpc	使用 icpc 编译 C++语言，推荐使用
Mpiifort	使用 ifort 编译 Fortran 语言，推荐使用

可通过 mpiicc -show 来查看具体的编译器信息：

```
$ mpiicc -show
```

```
icc -I/soft/mpi/impi/4.1.3.049/intel64/include
-L/soft/mpi/impi/4.1.3.049/intel64/lib -Xlinker
--enable-new-dtags -Xlinker -rpath -Xlinker
/soft/mpi/impi/4.1.3.049/intel64/lib -Xlinker -rpath
-Xlinker /opt/intel/mpi-rt/4.1 -lmpigf -lmpi -lmpigi
-ldl -lrt -lpthread
```

编译示例：

```
$ module load mpi/impi/4.1.3.049
$ mpiicc -o hello hello.c
$ mpiifort -o hello hello.f90
```

- 编译 mvapich2 程序

mvapich2 提供了 C/C++，Fortran 等语言的 MPI 编译器，如下表所示：

语言类型	MPI 编译器
C	mpicc
C++	mpicxx
Fortran77	mpif77
Fortran90	mpif90

可通过 mpicc -show 来查看具体的编译器信息：

```
$ mpicc -show
icc -L/lib -Wl,-rpath,/lib -L/lib -Wl,-rpath,/lib
-I/soft/mpi/mvapich2/1.9/intel/include
-L/soft/mpi/mvapich2/1.9/intel/lib -lmpich -lopa
-lmpl -lrt -libumad -libverbs -lpthread
```

编译示例：

```
$ module load mpi/mvapich2/1.9/intel
$ mpicc -o hello hello.c
$ mpif90 -o hello hello.f90
```

2.3.4. MPI+OpenMP 程序编译

MPI+OpenMP 混合程序的编译与 MPI 程序相同，设置相应 MPI 环境变量，用对应的 mpicc/mpif90 或 mpiifort 进行编译，编译时增加-openmp 参数。如：

```
$mpicc -openmp -o mpi_openmp_hello mpi_openmp_hello.c
$mpif90 -openmp -o mpi_openmp_hello mpi_openmp_hello.f90
```

2.3.5. UV2000 节点程序编译

UV2000 上程序编译，首先`$ ssh sgi`登陆 SGI uv2000 节点，设置 Intel 编译器的环境变量，使用 icc/ifort 进行编译。如：

```
$ module load compiler/intel/composer_xe_2013_sp1.0.080
```

- 编译 OpenMP 程序

```
$ icc -openmp -o test test.c
$ ifort -openmp -o hello hello.f90
```

- 编译 MPI 程序

```
$ module load mpt-2.10
$ icc -o test.sgi test.c -lmpi
```

```
$ ifort -o hello.sgi hello.f90 -lmpi
```

也可以使用 mpiicc、mpif90 进行编译。如：

```
$ mpiicc -o test.sgi test.c
$ mpif90 -o hello.sgi hello.f90
```

2.4. 资源管理和作业提交

2.4.1. 队列设置

用 bqueues 命令查看可用队列,用 bqueues –l <队列名〉查看队列具体设置。目前可用队列如下：

注意：bqueues –l 命令显示内容里 RUNLIMIT 表示作业运行最长时间限制，PROCLIMIT 表示作业运行最大核数限制，NJOBS 表示当前队列中占用的 CPU 核数而非作业数，其中 PEND 表示排队作业占用的 CPU 核数，RUN 表示运行作业占用的 CPU 核数。其它更具体的参数含义可以通过$man bqueues 命令查看。

cpu_dbg 是短时间 CPU 通用调试队列，资源可以被高优先级的作业抢占，如发现作业资源被抢占，请用户重新提交作业。

作业队列	计算资源	节点描述	队列说明
cpu	CPU 计算节点	2*Intel E5-2680 V2，20 CPU 核，64GB 内存	CPU 通用队列
gpu	GPGPU 计算节点	2*Intel E5-2680 V2，20 CPU 核，64GB 内存，2*Nvidia Tesla K20 GPGPU 卡	GPU 通用队列
mic	MIC 计算节点	2*Intel E5-2680 V2，20 CPU 核，64GB 内存，2*Intel Xeon Phi 5110P 卡	MIC 通用队列
uv2k	SGI UV2000	32*Intel E5-4620 V2 ,256CPU 核 ,共享内存 4TB	大内存节点队列
cpu_dbg	CPU 计算节点	2*Intel E5-2680 V2，20 CPU 核，64GB 内存	CPU 通用调试队列

2.4.2. bsub 命令提交作业

注意：“元”系统上用户使用资源统计按照作业占用节点统计。

1. MPI 作业提交

“元”系统上用 bsub 命令提交作业用法如下：

```
bsub -W [hour:]minute -n Z -R "span[ptile=Y]" -q QUEUENAME
-o OUTPUTFILE -e ERRFILE  mpijob.MPITYPE  PROGRAM
```

其中：

- 必须用-W 指定作业运行时间（用户最好根据实际情况进行估算）；
- Z 代表作业需要使用的 CPU 核心总数，Y 指定了作业在单个节点上使用的 CPU 核心个数（1≤Y≤20），如果不使用“-R "span[ptile=Y]"”选项，则 cpu 队列默认为每个节点使用 20 个 CPU 核心。**为减少资源浪费，作业及时得到调度，建议 cpu 队列 10≤Y≤20，Y≤5 的作业将予以清除。**
- 通过指定 QUEUENAME 可以将作业提交到不同的作业队列；
- OUTPUTFILE 是标准输出文件；
- ERRFILE 是错误输出文件；
- MPITYPE 指定 MPI 版本，目前支持 openmpi、intelmpi、mvapich2，各版本 MPI 加载脚本分别是 mpijob.openmpi 、 mpijob.intelmpi 、mpijob.mvapich2。
- PROGRAM 是带路径的运行程序名。

例如：

```
$ bsub -W 10 -n 40  -R "span[ptile=20]" -q cpu -o %J.out
-e %J.err mpijob.openmpi ./test.openmpi
$ bsub -W 10 -n 40  -R "span[ptile=20]" -q cpu -o %J.out
-e %J.err mpijob.intelmpi ./test.impi
$ bsub -W 10 -n 40  -R "span[ptile=20]" -q cpu -o %J.out
-e %J.err mpijob.mvapich2 ./test.mvapich2
```

2. MPI+OpenMP 作业提交：

在用户主目录下的.bashrc 文件中添加：

export OMP_NUM_THREADS=<每个进程派生的 OpenMP 线程数〉

```
bsub -W [hour:]minute -n Z -R "span[ptile=Y]" -q QUEUENAME
-o OUTPUTFILE -e ERRFILE  mpijob.MPITYPE  PROGRAM
```

注意：

- 必须用-W 指定作业运行时间（用户最好根据实际情况进行估算）；
- Z 代表作业使用的 CPU 核心总数，这里 Z=<MPI 进程数>*<每个进程派生的 OpenMP 线程数>；
- Y 指定了作业在单个节点上使用的 CPU 核心个数，这里 Y=<每个进程派生的 OpenMP 线程数>。
- 通过指定 QUEUENAME 可以将作业提交到不同的作业队列；
- OUTPUTFILE 是标准输出文件；
- ERRFILE 是错误输出文件；
- 这里 PROGRAM 是带路径的程序名。

例如：

```
$ cat ~/.bashrc
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
       . /etc/bashrc
fi
# User specific aliases and functions
export OMP_NUM_THREADS=20
```

```
$ bsub -n 160 -q cpu -R "span[ptile=20]" -o %J.out -e %J.err
mpijob.intelmpi ./mpi_openmp_hello.intelmpi
$ bsub -n 160 -q cpu -R "span[ptile=20]" -o %J.out -e %J.err
mpijob.openmpi ./mpi_openmp_hello.openmpi
$ bsub -n 160 -q cpu -R "span[ptile=20]" -o %J.out -e %J.err
mpijob.mvapich2 ./mpi_openmp_hello.mvapich2
```

3. OpenMP 作业提交：

```
bsub -W [hour:]minute  -n Z -q QUEUENAME  -o OUTPUTFILE -e
ERRFILE PROGRAM.sh
```

- 必须用-W 指定作业运行时间（用户最好根据实际情况进行估算）；
- Z 代表作业需要使用的 CPU 核心总数， 这里 Z= <OpenMP 线程数>；
- 通过指定 QUEUENAME 可以将作业提交到不同的作业队列；
- OUTPUTFILE 是标准输出文件；
- ERRFILE 是错误输出文件；

- 这里 PROGRAM.sh 通常是一个脚本文件,内容如下：

```
export OMP_NUM_THREADS=< OpenMP 线程数〉
<带路径的程序名〉
```

chmod +x PROGRAM.sh 增加 PROGRAM.sh 的可执行属性。

例如：

```
$cat openmp_hello.sh
export OMP_NUM_THREADS=40
./openmp_hello
$ chmod a+x openmp_hello.sh
$ bsub -W 5 -n 40 -q cpu -o %J.out
-e %J.err ./openmp_hello.sh
```

4. **串行作业提交：**

```
bsub -W [hour:]minute -n 1 -q QUEUENAME -o OUTPUTFILE -e
ERRFILE PROGRAM
```

- 必须用-W 指定作业运行时间（用户最好根据实际情况进行估算）；
- 通过指定 QUEUENAME 可以将作业提交到不同的作业队列；
- OUTPUTFILE 是标准输出文件；
- ERRFILE 是错误输出文件；
- 这里 PROGRAM 是带路径的程序名。

例如：

```
$ bsub -W 5 -n 1 -q cpu -o %J.out -e %J.err ./test
```

5. **UV2000 节点作业提交**

在 SGI UV2000 节点上（`$ ssh sgi`）提交作业需要提交到 uv2k 队列，格式如下：

- MPI 作业提交：

```
bsub -W [hour:]minute -n Z  -q uv2k -o OUTPUTFILE -e ERRFILE
mpijob.sgi  PROGRAM
```

其中：

- 用-W 指定作业运行时间（用户最好根据实际情况进行估算）；
- Z 代表作业需要使用的 CPU 核心总数；
- OUTPUTFILE 是标准输出文件；
- ERRFILE 是错误输出文件；

- PROGRAM 是带路径的运行程序名。

例如：

```
$ bsub -W 5 -n 128 -q uv2k -o %J.out -e %J.err
mpijob.sgi ./test.sgi
```

◆ OpenMP 作业提交：

```
bsub -W [hour:]minute  -n Z -q uv2k  -o OUTPUTFILE -e
ERRFILE PROGRAM.sh
```

- 用-W 指定作业运行时间（用户最好根据实际情况进行估算）；
- Z 代表作业需要使用的 CPU 核心总数， 这里 Z= <OpenMP 线程数>；
- OUTPUTFILE 是标准输出文件；
- ERRFILE 是错误输出文件；
- 这里 PROGRAM.sh 通常是一个脚本文件,内容如下：

```
export OMP_NUM_THREADS=< OpenMP 线程数〉
<带路径的程序名〉
```

chmod +x PROGRAM.sh 增加 PROGRAM.sh 的可执行属性。

例如：

```
$cat openmp_hello.sh
export OMP_NUM_THREADS=128
./openmp_hello.sgi
$ chmod a+x openmp_hello.sh
$ bsub -W 5 -n 128 -q uv2k -o %J.out
-e %J.err ./openmp_hello.sh
```

2.4.3. 使用 bsub 脚本多次提交具有相同参数的作业

bsub 命令可以使用输入脚本多次提交具有相同参数的作业，其格式为：

```
#BSUB -W [hour:]minute
#BSUB -n Z
#BSUB -R "span[ptile=Y]"
#BSUB -q QUEUENAME
#BSUB -o OUTPUTFILE
#BSUB -e ERRFILE
mpijob.MPITYPE PROGRAM
```

该脚本中的参数与命令行下：

```
bsub -W [hour:]minute -n Z -R "span[ptile=Y]" -q QUEUENAME
-o OUTPUTFILE -e ERRFILE  mpijob.MPITYPE  PROGRAM
```

命令的参数含义相同。

提交作业时，仍使用 bsub 命令，格式为：

```
$ bsub < bsub 脚本名
```

推荐用户使用脚本模式提交作业。使用脚本模式提交时，在输出文件中包含提交作业的脚本信息，便于用户分析作业的运行情况并避免多次命令行输入的误操作。

例如：

```
$ cat test.bsub
#BSUB -W 10
#BSUB -n 40
#BSUB -R "span[ptile=20]"
#BSUB -q "cpu"
#BSUB -o %J.out
#BSUB -e %J.err
./mpijob.openmpi ./test.openmpi
$ bsub < test.bsub
```

2.4.4. bsub 命令执行结果

当您执行 bsub 命令成功提交一个作业之后，系统会返回一条类似于

“Job <1655> is submitted to queue <cpu>.”

的信息，这条信息显示了您所提交作业的作业号（第一个尖括号里面的内容）以及您的作业提交到的队列（第二个尖括号中的内容）。建议您每次提交作业后将对应的作业名及作业号记录下来，因为您在提交作业之后对您的作业进行操作或是在作业退出之后查看作业历史和作业输出信息时，都必须用到这个作业号。

2.5. 查看作业运行情况

bjobs 的功能是查看系统中作业的情况。

直接执行“bjobs”命令会得到当前用户正在排队和正在运行的作业列表。bjobs 命令的执行结果很直观地依次列出了作业的作业号、用户名、作业状态、作业所

在队列、提交作业的结点、作业运行所占用的结点、作业名以及作业提交的时间。bjobs 命令的常用参数如下：

- -a:在不加任何参数的情况下，看到的只是自己提交的并且尚未结束的作业。如果您使用了“-a”参数，除了未完成的作业之外，还能看到一些刚结束不久的作业的信息。
- -u:如果需要查看系统中别的用户的作业情况的话，您只需加上“-u”参数，比如想查看用户“user1”的作业情况，那么执行“bjobs -u user1”即可。如果执行了“bjobs -u all”的话，您将会看到所有用户的作业信息。
- -l：加上-l 参数可以查看查看某个作业的详细信息，具体格式是“bjobs -l JOBID”。

下表对常见的作业状态解释：

状态	含义
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

2.6. 查看运行中作业的标准（屏幕）输出信息

在作业运行的过程中，可以使用 bpeek 命令随时查看作业的标准输出信息以确定作业是否在正常的运行。命令格式是“bpeek JOBID”。

2.7. 挂起和释放作业

2.7.1. 挂起作业

作业提交之后，在排队过程中，可能需要暂时不让这个作业被调度执行，或是作业已经在运行的时候，希望它暂时停下来，那么可以通过挂起作业来达到目

的。系统中挂起作业的命令是“bstop”，只需执行“bstop JOBID”即可，如“bstop 5060”。如果作业已经运行，那么它的状态将变为“USUSP”，如果作业还尚未运行，那么它的状态则变为“PSUSP”。

2.7.2. 释放作业

与作业挂起操作相对应的是释放操作，该操作能够让被挂起的作业重新被激活，允许系统对其进行调度并执行。释放作业的命令是“bresume”，命令的格式和挂起操作是一样的，即“bresume JOBID”，对应上面的例子，现在我们需要将 5060 号重新释放，那么执行“bresume 5060”即可。

2.8. 删除作业

如果要删除某个作业，可以很简单的通过“bkill JOBID”来杀掉作业，不管该作业是在排队或是已经被执行。

2.9. 查看作业输出信息

提交作业时以“-o”参数指定了输出文件，作业的标准（屏幕）输出将会保存在这个输出文件中。

2.10. 查看历史作业信息

作业运行完毕之后，作业的相关信息会被保存在系统中，可以用“bhist”命令随时来查看。如果作业结束已经很久了，需要用“bhist -a”才能看到它们了。同 bjobs 一样，默认情况下，系统给出的是当前用户提交的作业的基本信息的列表，如果需要查看别人的作业信息，或是了解作业的详细情况，您还需要加上“-u”、“-l”等参数，其功能与 bjobs 的对应参数相似。

2.11. 其它命令简介

bqueues 命令

bqueues 命令用于查看队列信息，默认情况下，bqueues 命令列出作业系统中定义的全部队列信息，包括队列名、优先级、状态信息、最大可用资源数、排队作业数、运行作业数等信息。

brequeue 命令

brequeue 命令用于作业重新排队，用户可以使用命令 brequeue JOBID 终止指定的、隶属于自己的、并正在运行的作业，该作业将以原有的作业号重新进行排队，重新获得调度、运行。

btop/bbot 命令

btop/bbot 命令用于改变处于“PEND”状态的作业获得调度的次序，用户只能改变自己处于同一队列内的作业的相对次序，btop 使指定作业在同一队列内的，所有同优先级的作业中最先获得调度。bbot 则相反。

bhosts 命令

bhosts 命令用于查看结点状态，处于“ok”状态的结点表示该节点可以接收用户作业。结点上已经有作业运行或者负载过高都会导致“closed”状态。