深腾 7000 使用指南

中国科学院计算机网络信息中心

超级计算中心

2009 年 9 月 2 日

注意事项

1. 《深腾 7000 使用指南》最新版可登陆深腾 7000 后到/home_soft/doc 目录下载。

2. 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。

3. 胖节点部分为 IA64 结构，只能将 IA64 可执行代码提交到胖节点队列；集群部分为 X86_64 结构，只能将 X86 可执行代码提交到集群队列。

4. 由于计算刀片采用无盘架构，不允许用户在/tmp 目录下放大量临时文件。

5. /datastore/userdata 目录用户可以存放非常用的重要数据，该目录下的数据将定期迁移到磁带库，需要的用户可以向超算中心提出申请。常用文件建议在其它目录有拷贝，因为迁移至带库的文件访问效率很低；建议小文件打包后再放在该目录下，便于数据迁移。不要将该目录作为工作目录。

6. /datastore/workspace 目录作为用户工作空间已经对用户开放使用，需要的用户可以向超算中心提出申请。但该目录数据不做备份，重要数据请勿在该目录下长期存放。

7. 厚节点专用文件系统/gpfs1,/gpfs2 已经对用户开放使用，需要的用户可以向超算中心提出申请。但该目录数据不做备份，重要数据请勿在该目录下长期存放。

8. 商业软件的使用请参照各商业软件使用指南。

1. 系统配置简介

深腾 7000 百万亿次机采用混合结构的高性能计算集群架构，由采用 Intel Xeon 处理器的集群部分与采用 Intel Itanium2 处理器的胖节点部分组成，适于不同的应用计算。但要注意的是，这两部分节点的程序二进制代码不兼容。

集群部分包括认证服务器、前端机、管理节点、登陆节点、启动节点、I/O 节点、备份节点和计算节点。其中，计算节点包括刀片和厚节点，配置如下：

- 1140 个普通（刀片）节点，每个刀片配置两颗四核 Xeon 处理器 E5450，主频 3.00GHz，32GB 内存；
- 38 个厚节点，IBM X3950M2 服务器，每个厚节点配置 16 颗四核 Xeon 处理器 X7350，主频 2.93GHz，512GB 内存；
- 12 个可视化节点（采用机架式服务器），2 颗 Intel Xeon E5450，32GB 内存，Nvidia 9800GTS 显卡；
- 其它节点配置与计算刀片相同，包括 2 个认证服务器、2 个前端机、2 个管理节点、8 个登录节点、12 个启动节点、120 个 IO 节点、2 个备份节点；
- 操作系统：Red Hat Enterprise Linux Server release 5.1，Linux kernel 2.6.18-53.el5。

胖节点采用 2 套 SGI Altix4700，使用 Intel Itanium2 处理器，可向下兼容深腾 6800 目标代码(可直接运行深腾 6800 环境中的部分应用程序,MPI 程序)，其配置如下：

- 系统组成
  - 包括 2 个计算节点、1 个头节点、2 个文件服务节点
    - 胖节点（计算节点），节点采用 NUMA 结构，配置 94 个双路刀片模块，共 188 颗处理器，376 个 CPU 核心，共享内存约 2.5TB，主要用于批处理作业计算；
    - 头节点，由 1 台胖节点通过逻辑分区实现，用于用户登录、编译与调试程序、提交作业。头节点配置 4 个双路刀片，共 16 个 CPU 核心，64GB 内存。
    - 文件服务节点，由 1 台 SGI Altix450 通过逻辑分区功能成为 2 台 CXFS 文件系统器使用，用户不可直接访问。
  - 共 384 颗 Intel Itanium2 双核处理器
    - 型号 9140M, 1.66GHz 主频，666MHz 总线，
    - L1 Cache: 16KB（Data)＋16KB(指令)，L2 Cache: 256KB(Data)＋1MB(指令)，L3 Cache: 18MB
    - TDP 104W
  - 系统浮点峰值 5Tflops
  - 内存总量 5TB
- 互连网络
  - 节点内采用 NUMALink 互连（双向带宽 6.4GB/s，延迟 1us)；

- 节点间采用千兆以太网互连，用于管理、NFS 文件系统；
- 计算节点间采用 2 条 20Gbps 的 Infiniband 互连。

- 操作系统
  - 采用 SUSE Linux 10SP2，内核版本为 2.6.16.60-0.21.default；

其它：

- 系统网络：节点间采用千兆以太网与 20Gbps 的 Infiniband 实现互连；
- 作业管理系统，LSF HPC 7.0。
- 全局共享文件系统
  - 用户 $HOME 目录：/home_soft/home 采用 SNFS 文件系统；
  - 软件安装目录：/home_soft/soft 采用 SNFS 文件系统；
  - /work：采用 SNFS 文件系统，是/datastore 目录的链接；
  - /luster：用户工作空间，使用 Lustre 文件系统；
  - /datastore/userdata 目录：用户重要数据存放空间，采用 SNFS 文件系统，数据定期向磁带库迁移。
  - /datastore/workspace 目录：用户临时工作目录，采用 SNFS 文件系统。
- 胖节点专用工作空间(仅胖节点可见)：/workspace 采用 SGI CXFS 共享文件系统。
- 厚节点专用工作空间（仅厚节点可见）：/gpfs1、/gpfs2，采用 IBM GPFS 文件系统。
- 深腾 6800 原有用户数据：
  - 已全部迁移至：/home_soft/Lenovo6800/DeepComp6800
  - 包括用户$HOME 目录、/userdata[1-8]等
- 2008 年 11 月，SGI 胖节点测试期间用户数据：
  - 保存于 /home_soft/home.sgi

2. 用户登录、数据存储、传输与工作空间

- 深腾 7000 公网接入 IP：159.226.49.66。域名：shenteng.sccas.cn
- 深腾 7000 胖节点公网接入 IP：159.226.49.80. 域名：altix.sccas.cn
  - 胖节点与其它节点内部互通，可通过 ssh 命令在登录节点与胖节点接入节点(head)间转变工作环境。
- 深腾 6800 已有用户都已迁至新系统中，账号名与密码不变。
- 支持协议：仅支持 SSH 方式登录，不支持 telnet、FTP 这些非加密方式；可使用支持 SSH 协议的工具登录（Linux 下直接用 ssh，Windows 环境中可采用 putty 等）。
- IP 地址绑定、身份认证方式及认证口令均与深腾 6800 相同，使用原来可以登录深腾 6800 的 IP 地址登陆。即：

(1) 登陆深腾 7000 集群：

a) 打开 IE 或 Firefox 连览器，在地址栏键入 http://shenteng .sccas.cn。按照屏幕提示输入身份认证用户

名和口令。

b) ssh -l username shenteng.sccas.cn 登录进入深腾 7000 集群。

c) 由于采用了新的 RSA 身份认证系统，在用 ssh 登录时需要输入 2 遍密码。（正式启用 RSA 系统后，将使用动态口令卡认证）

(2) 登陆深腾 7000 胖节点：

a) 打开 IE 或 Firefox 连览器，在地址栏键入 http://altix.sccas.cn 按照屏幕提示输入身份认证用户名和口令。

b) ssh -l username altix.sccas.cn 登录进入深腾 7000 胖节点。

- 数据上传、下载：使用支持 SFTP、SCP 方式的工具（Windows 环境中可采用 winscp）。

  注：在向 159.226.49.66 上传数据之前，建议先对 MPI 版本进行设置。即在自己主目录下建立.mpi_type 文件，文件内容是所需使用的 MPI 版本(如：intelmpi、mvapich、openmpi)。如果没有设置 MPI 版本有可能无法通过 scp 和 sftp 上传数据。

- 工作空间
  - $HOME：用户默认工作目录
  - /work、/lustre 全局文件系统，与厚节点的/gpfs、胖节点的/workspace 需要用户联系管理员申请使用。

3. 程序开发调试环境

集群部分工具软件安装情况如下：

名称	版本	安装目录
Intel C/C++ Compiler	11.0.081	/home_soft/soft/x86_64/compiler/intel/11.0
Intel Fortran Compiler	11.0.081	/home_soft/soft/x86_64/compiler/intel/11.0
Intel Idb Debugger	11.0.081	/home_soft/soft/x86_64/compiler/intel/11.0
Intel Vtune	9.1_002	/home_soft/soft/x86_64/tools/vtune
Intel Clustered Math Kernel Library	10.1.1.019	/home_soft/soft/x86_64/lib/mkl
Intel Integrated Performance Primitives	6.0.2-076	/home_soft/soft/x86_64/lib/ipp
Intel Thread Building Blocks	2.1_015	/home_soft/soft/x86_64/lib/tbb
Intel Trace Analyzer & Collector	7.2.011	/home_soft/soft/x86_64/tools/itac
Intel Thread Checker	3.1_012	/home_soft/soft/x86_64/tools/itt
Intel Thread Profiler	3.1_012	/home_soft/soft/x86_64/tools/itt
Intel MPI	3.2.011	/home_soft/soft/x86_64/mpi/impi
OpenMPI	1.3.	/home_soft/soft/x86_64/mpi/openmpi
MVAPICH2	1.2p1	/home_soft/soft/x86_64/mpi/mvapich2/
Gnu	4.1.2	/usr

C/C++/Fortran/Objc/Java		
TotalView Debugger	8.6.2-2	/home_soft/soft/x86_64/tools/toolworks /totalview.8.6.2-2
PGI Workstation Complete	8.0-5	/home_soft/soft/x86_64/compiler/pgi
Java Runtime Environment		/home_soft/soft/x86_64/java
Perl	5.8.8	/usr
Python	2.4.3	/usr

其中：

- Intel C/Fortran 编译器/Cluster OpenMP 可以在 8 个登录节点（即：LB270107- LB270110,LB270207- LB270210）上使用；
- PGI 编译器只能在 LB270107 和 LB270108 上使用,license 也分别在这 2 个登陆节点上；
- Vtune 和 ThreadChecker 只能在 LB270107-LB270110 这 4 个登陆节点上使用；
- ITAC 可以在 8 个登陆节点上使用；
- Totalview 可以在 8 个登陆节点上使用, 限制为 3 个并发用户；
- TBB, IPP, MKL, IMPI 都可以在 8 个登陆节点上使用；
- 目前，Intel C/C++/Fortran/idb、GNU C/C++/Fortran 及 PGI 编译器的环境变量已在系统中设为默认，用户可直接使用，用户若要使用其它工具软件，还需要自行设置环境变量。

在胖节点中安装了 Intel、GNU 系列开发工具、数学库，具体如下：

名称	版本	安装目录
Intel C/C++ Compiler	10.1.008	/opt/intel/cc/10.1.008
Intel Fortran Compiler	10.1.008	/opt/intel/fc/10.1.008
Intel Idb Debugger	10.1.008	/opt/intel/idb/10.1.008
Intel Vtune	9.0	/opt/intel/vtune
Intel Clustered Math Kernel Library	10.0.010	/opt/intel/cmkl/10.0.010
Intel Integrated Performance Primitives	5.3p	/opt/intel/ipp/5.3
Intel Thread Building Blocks	2.0pu	/opt/intel/tbb/2.0
Gnu C/C++/Fortran/Objc/Java	4.1.2	/usr/
Java Runtime Environment	1.4.2.17	/usr/lib/jvm/java-1_4_2_sun-1.4.2.17/jre
Perl	5.8.8	/usr
Python	2.4.2	/usr

其中：

- Intel C/C++/Fortran/CMKL/idb/tbb 及 GNU C/C++/Fortran 等开发工具的环境变量已在系统中设为默认，用户可直接使用。
- 如果要用 Intel Vtune/IPP/，需要自行设置相应环境变量。

4. 并行开发环境

4.1.集群部分

在深腾 7000 上，预装了 Intel 公司的 C/C++/Fortran 编译器、GNU 的 C/C++/Fortran 编译器和 PGI 编译器，建议用户使用 Intel 编译器进行编译。

深腾 7000 上安装的 MPI 版本有 IntelMPI 3.2、MVAPICH2-1.2p1、OpenMPI 1.3.2，用户可在用户主目录下编写.mpi_type 文件，指定想要使用的 MPI 版本（如：intelmpi、mvapich、openmpi），登陆时，系统会自动为用户设置相应 MPI 环境变量，若不指定，缺省使用 IntelMPI。例如：

```
[test@LB270107 ~]$ cat .mpi_type
MPI_TYPE="intelmpi"
[test@LB270107 ~]$
```

注：为保证 MPI 设置正确，在修改.mpi_type 内容后，最好能重新登录。

MPI 作业可以直接使用 mpicc、mpif77、mpif90 编译，对于 IntelMPI 建议使用 mpiicc、mpiifort 编译。

- 串行程序使用 icc/ifort 进行编译，如 `icc -o test test.c;`
- MPI 程序可以使用 MPI 编译器编译，如 `mpicc -o cpi cpi.c;`
- Intel 编译器支持 OpenMP 并行，OpenMP 程序编译如下：
  `icc -openmp -o hello hello.c`
  OpenMP 并行程序运行前需要设置环境变量 OMP_NUM_THREADS=n
  其中，n 是每个进程派生的 OpenMP 线程数。
- MPI+OpenMP 程序编译如下：
  `mpicc -openmp -o mpi_openmp_hello mpi_openmp_hello.c`

注：程序的编译和链接必须在 8 个登陆节点上进行，8 个登陆节点包括 LB270107- LB270110,LB270207- LB270210 。

4.2.SGI 胖节点

胖节点采用 NUMA 结构，节点内并行可采用 SHMEM 或 OpenMP 并行编程方式。胖节点也安装了 MPI 并行通讯环境 SGI MPT 1.2，支持节点内并行及节点间并行。胖节点也支持 MPI＋OpenMP 的混合并行模式。由于节点内的 NUMALink 性能远高于节点间的 Infiniband 网络，不建议用户运行节点间并行程序；如用户需要运行并行规模＞376 的节点间并行程序，请提前联系超算系统管理人员。

- OpenMP 程序编译：Intel Compiler 支持 OpenMP 并行编程，例如：
  `icc -openmp -o hello hello.c`
- MPI 并行程序编译：
  - SGI MPT 直接安装在系统目录/usr 下，mpirun 等可执行程序在

/usr/bin，mpi.h 等头文件在/usr/include，libmpi.so 等库文件在/usr/lib

- MPT 中没有提供 mpicc、mpif77 等命令，用户可直接用 Intel icc/ifort 编译，在链接时加上-lmpi 即可。例如：

```
icc  -o hello hello.c  -lmpi
```

- MPI+OpenMP 并行程序编译：例如：

```
icc -openmp -o mpi_openmp_hello mpi_openmp_hello.c -lmpi
```

- **SGI 胖节点环境中的程序编译环境在 head 节点上。**

5. 作业提交、资源管理

5.1. 队列设置

深腾 7000 高性能计算机与深腾 6800 一样采用 Platform LSF HPC 管理资源、调度作业。

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。目前暂时可用队列如下：

注：胖节点部分为 IA64 结构，只能将 IA64 可执行代码提交到胖节点队列；
集群部分为 X86_64 结构，只能将 X86 可执行代码提交到集群队列。

队列名		节点体系结构	CPU 核心数	执行时间	队列说明
胖节点	altix_dbg	IA64	1-64	≤15 分钟	胖节点调试队列
	altix_s	IA64	1-32	≤6 小时	胖节点小作业队列
	altix_n	IA64	32-128	≤6 小时	胖节点中作业队列
	altix_l	IA64	129-374	≤6 小时	胖节点大作业队列
集群部分	x64_dbg	X86_64	1-256	≤10 分钟	刀片部分调试队列
	x64_blades	X86_64	8-511	≤6 小时	刀片部分中作业队列
	x64_large	X86_64	400-2048	≤6 小时	刀片部分大作业队列
	X64_small	X86_64	1-8	≤6 小时	短时间小作业队列
	x64_3950dbg	X86_64	9-512	≤10 分钟	厚节点调试队列
	x64_3950small	X86_64	1-8	≤144 小时	厚节点长时间小作业队列

	x64_3950	X86_64	9-512	≤6 小时	厚节点中作业队列
商用软件	fluent	X86_64	1-32	≤24 小时	fluent 应用队列
	cfx	X86_64	1-8	≤24 小时	cfx 应用队列
	cfx_sgi	IA64	1-8	≤24 小时	cfx 应用队列
	dyna	X86_64	1-32	≤24 小时	LS_Dyna 应用队列
	ansys	X86_64	1-8	≤24 小时	ansys 应用队列
	msi	X86_64		≤24 小时	materials studio 应用队列
	matlab	X86_64	1-8	≤24 小时	matlab 单节点应用队列
	matlab_dce	X86_64	1-64	≤24 小时	matlab 分布式计算应用队列
	其它				一些专用队列

5.2. 使用 bsub 提交作业

5.2.1. bsub 命令常见用法

深腾 7000 上用 bsub 命令提交作业用法如下：

(1) 集群部分 MPI 作业提交：

bsub -W [hour:]minute **-a MPITYPE** -n Z -R "span[ptile=Y]" -q QUEUENAME -o OUTPUTFILE -e ERRFILE **mpirun.lsf** 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）；
- MPITYPE 指定 MPI 版本，目前支持 intelmpi、mvapich、openmpi；
- 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（在刀片上，1≤Y≤8；在厚节点上，1≤Y≤64），如果不使用“-R "span[ptile=Y]"”选项，则默认为每个节点使用 8 个 CPU 核心。**为减少资源浪费，刀片节点建议 Y≥4，厚节点建议 Y≥8；**
- 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；
- PROGRAM 是带路径的运行程序名。

注：集群部分提交 MPI 作业时，bsub 命令中-a 参数指定的 MPITYPE 一定要

与.mpi_type 文件中指定的 MPI 类型相同。

(2) 对于集群部分 OpenMPI 作业，由于作业提交脚本 bug，使用 129 及以上节点的作业无法正常运行，此时需要用以下命令提交：

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

（对于 OpenMPI 作业，都可采用这种方式提交）

(3) 胖节点上 MPI 作业提交：

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

(4) OpenMP 作业提交：

bsub -W 6:00 -a openmp -n Z -R "span[hosts=1]" -q QUEUENAME -o OUTPUTFILE -e ERRFILE PROGRAM.sh

注意这里 PROGRAM.sh 通常是一个脚本文件,内容如下：

export OMP_NUM_THREADS=<每个进程派生的 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。

(5) 集群部分 MPI+OpenMP 作业提交：

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

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

注意：

- 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>。在刀片上，1≤Y≤8；在厚节点上，1≤Y≤64，**为减少资源浪费，刀片节点建议 Y≥4，厚节点建议 Y≥8**;
- 这里 PROGRAM 是带路径的程序名。

(6) 胖节点 MPI+OpenMP 作业提交：

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

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

注意：

- 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>;
- 这里 PROGRAM 带路径的程序名。

因为目前 altix_s,altix_n,altix_l 队列包含节点仅 1 台胖节点,所以胖节点上 MPI+OpenMP 作业提交时无需-R "span[ptile=Y]"参数，altix_dbg 队列包含 head 和 1 台胖节点，提交时可以用-R "span[ptile=Y]"参数指定每

个节点使用的 CPU 核心个数。

(7) 刀片节点上串行作业提交：

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

例如：

(1) 刀片节点上的串行作业提交：

```
bsub -W 5 -n 1 -q x64_small -o out -e err./mytest
```

(2) 刀片节点上的 Intel MPI 作业提交：

```
bsub -W 1:0 -a intelmpi -n 16 -q x64_blades -o out -e err mpirun.lsf ./test.impi
```

(3) 刀片节点上的 OpenMPI 作业提交：

```
bsub -W 2:0 -a openmpi -n 128 -q x64_blades -o out -e err mpirun.lsf ./test.ompi
```

(4) 刀片节点上的 OpenMPI 大作业（256 节点）提交：

```
bsub -W 2:0 -n 1024 -R “span[ptile=4]” -q x64_blades -o out -e err mpijob.openmpi ./test.ompi
```

(5) 厚节点上的 MPI 作业提交

```
bsub -W 5:0 -a intelmpi -n 128 -R “span[ptile=64]” -q x64_3950 -o out -e err mpirun.lsf ./test.impi
```

(6) 厚节点上的 OpenMP 作业提交

```
bsub -W 5 -a openmp -n 8 -R “span[hosts=1]” -q x64_3950 -o out -e err ./openmp.sh
```

openmp.sh 的内容如下：

```
export OMP_NUM_THREADS=8  #需要使用的处理器核心个数
./hello
```

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

(7) 刀片节点 MPI+OpenMP 作业提交：

```
export OMP_NUM_THREADS=6
bsub -W 15 -x -a intelmpi -n 12 -R “span[ptile=6]” -q x64_dbg -o out -e err mpirun.lsf ./mpi_openmp_hello.impi
```

(8) 厚节点 MPI+OpenMP 作业提交：

```
export OMP_NUM_THREADS=12
bsub -W 15 -x -a intelmpi -n 24 -R “span[ptile=12]” -q x64_3950dbg -o out -e err mpirun.lsf ./mpi_openmp_hello.impi
```

(9) 胖节点上的 MPI 作业提交：

```
bsub -W 6:00 -q altix_n -n 128 -o outfile -e errmsg
mpijob.sgi ./xhpl
```

(10) 胖节点上 OpenMP 作业提交：

```
bsub -W 3:00 -a openmp -n 128 -R “span[hosts=1]” -q altix_n -o
outfile -e errmsg ./openmp.sh
```

openmp.sh 的内容如下：

```
export OMP_NUM_THREADS=128 #需要使用的处理器核心个数
./test
```

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

(11) 胖节点上 MPI+OpenMP 作业提交：

```
export OMP_NUM_THREADS=8
bsub -W 10 -n 64 -q altix_n -o mpi_openmp_hello.out -e
mpi_openmp_hello.err mpijob.sgi ./mpi_openmp_hello.mpt
```

5.2.2. 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 -a MPITYPE
#BSUB -n Z
#BSUB -R "span[ptile=Y]"
#BSUB -q QUEUENAME
#BSUB -o OUTPUTFILE
#BSUB -e ERRFILE
mpirun.lsf PROGRAM
```

该脚本中的参数与命令行下：

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

命令的参数含义相同。

提交作业时，仍使用 bsub 命令，格式为：

```
[user@LB270209 ~]$ 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：

上述在厚节点上提交 MPI+OpenMP 作业，换成脚本提交模式如下：

- 编写提交脚本 bsubmpiopenmp 如下：

```
#BSUB -W 15
```

```
#BSUB -a intelmpi
#BSUB -q x64_3950dbg
#BSUB -n 24
#BSUB -R "span[ptile=12]"
#BSUB -o out
#BSUB -e err
mpirun.lsf ./mpi_openmp_hello.impi
```

- 用 bsub 提交作业：

```
[user@LB270209 ~]$ export OMP_NUM_THREADS=12
[user@LB270209 ~]$ bsub < bsubmpiopenmp
```

5.2.3. 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 <11108> is submitted to queue <normal>.”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。

注：

1. **对于 OpenMP 作业，通过环境变量 OMP_NUM_THREADS 设置的并发线程数一定要小于或等于 bsub 命令“-n”申请的 CPU 核心数。否则，我们不得不停掉您的作业。**

5.3. 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

5.4.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”。

5.5.挂起和释放作业

5.5.1. 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”。

5.5.2. 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。

5.6.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。

5.7.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。

5.8.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。

5.9.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 LSF 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。