清华大学生物高性能计算平台

用户使用手册

V3.0

**清华大学蛋白质设施实验技术中心**

**计算平台**

**2015年1月16日**

目录

[目录 II](#_Toc409186283)

[版本变更记录 III](#_Toc409186284)

[1 集群硬件环境 1](#_Toc409186285)

[1.1 登录节点 1](#_Toc409186286)

[1.2 计算节点 1](#_Toc409186287)

[1.3 存储节点 1](#_Toc409186288)

[1.4 管理节点 1](#_Toc409186289)

[1.5 网络互连 2](#_Toc409186290)

[2 系统环境及磁盘共享 2](#_Toc409186291)

[2.1 操作系统版本 2](#_Toc409186292)

[2.2 磁盘共享 2](#_Toc409186293)

[3 用户登录 3](#_Toc409186294)

[3.1 远程访问软件 3](#_Toc409186295)

[3.2 登录步骤 3](#_Toc409186296)

[3.3 数据传输 3](#_Toc409186297)

[3.4 使用集群 4](#_Toc409186298)

[4 编译及测试环境 5](#_Toc409186299)

[4.1 访问编译环境 5](#_Toc409186300)

[4.2 软件资源 5](#_Toc409186301)

[4.3 配置用户的环境变量 5](#_Toc409186302)

[4.4 编译及测试 6](#_Toc409186303)

[4.4.1 Intel编译器编译串行程序及Openmp程序 6](#_Toc409186304)

[4.4.2 Intel编译器编译运行mpi并行程序 8](#_Toc409186305)

[4.4.3 其他注意事项 10](#_Toc409186306)

[5 作业管理系统使用说明 10](#_Toc409186307)

[5.1 LSF作业管理系统使用说明 10](#_Toc409186308)

[5.1.1 队列设定 11](#_Toc409186309)

[5.1.2 提交作业(bsub) 11](#_Toc409186310)

[5.1.3 状态查看 14](#_Toc409186311)

[5.1.4 控制作业执行 16](#_Toc409186312)

[5.1.5 作业提交脚本 17](#_Toc409186313)

[5.2 SGE作业管理系统使用说明 18](#_Toc409186314)

[5.2.1 提交作业(qsub) 18](#_Toc409186315)

[5.2.2 状态查看 19](#_Toc409186316)

[5.2.3 控制作业执行 21](#_Toc409186317)

[附录1：Linux基本命令 22](#_Toc409186318)

[附录2：Vi使用 25](#_Toc409186319)

版本变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本号** | **作者** | **修改说明** |
| 2013-11-01 | V1.0 | 葛云峰、甄亚楠 | 文档创建 |
| 2014-06-02 | V2.0 | 林宇 | 调整集群目录结构 |
| 2015-01-16 | V3.0 | 葛云峰、林宇 | 增加SGE作业管理系统使用说明 |
|  |  |  |  |

# 集群硬件环境

IBM-A集群共有60个计算节点，960个处理器核；IBM-B集群共有40个计算节点，640个处理器核；IBM-C集群计算机共有20个计算节点，320个处理器核。所有计算节点的处理器均采用Intel Xeon E5-2650，系统的理论浮点峰值计算性能达到30.72TFlops，存储总容量达1.4PB。另外,系统还配置3个Nvidia Tesla M2090 的GPU节点2个512G内存的Bnode节点。

## 登录节点

|  |  |  |
| --- | --- | --- |
|  | **数据网地址** | **校园网地址** |
| **IBM-A集群** | 10.10.0.48（login01） | 166.111.30.164(mgt)166.111.30.165(login01) |
| **IBM-B集群** | 10.10.0.40 | 　 |
| **IBM-C集群** | 10.10.0.22 | 166.111.156.54 |

## 计算节点

A集群计算节点共计60个，编号形式为node01~node60；B集群计算节点共计40个，编号形式为node01~node40；C集群计算机节点共计20个，编号形式为node-0-1~node-0-20。A、B集群均需要通过LSF作业管理系统提交作业，加载程序；C集群需要通过SGE作业管理系提交作业，加载程序。

单节点配置为：计算节点均采用两个 Intel Xeon E5-2650八核处理器（2.00GHz，20MB Cache）, 500G SATA硬盘。

每个节点都是一个多核SMP服务器，计算节点用于运行串行和并行计算任务，支持MPI、OpenMP及MPI/OpenMP混合并行编程模式。生物高性能计算平台集群作业管理系统以CPU核作为并行作业的资源分配单位，实现并行作业的调度运行。生物高性能计算平台集群每个计算节点为16核的SMP服务器,可以最大支持**120\* 16 =1920**核并行作业的计算。

## 存储节点

A、B集群的IO存储系统各由两个IO存储节点组成，名字相同，分别为ionode01和ionode02。用于提供文件存储和共享服务，受控于管理结点。存储系统采用GPFS并行文件系统进行管理。所有用户目录下/Share目录为全局共享，所有节点/Share目录都有读写权限。

## 管理节点

管理节点为mgt，A集群的IP地址为166.111.30.164；B级群的IP地址为10.10.0.41（禁止普通用户登录）；C集群的IP地址为10.10.0.22（166.111.156.54）。mgt节点负责管理整个高性能计算机群。

## 网络互连

生物高性能计算平台集群由InfiniBand QDR通信网络构成，理论带宽40Gb。所有节点间均可以通过InfiniBand网络实现高速通信。支持MPI并行任务间通信，并实现全局文件系统的数据传输。

生物高性能计算平台A集群通过登录节点login01，C集群通过mgt节点接入校园网，校内外用户通过以太网访问生物高性能计算平台。

# 系统环境及磁盘共享

## 操作系统版本

A集群系统采用CentOS release 5.8 (Final)(内核2.6.18-308)，B、C集群采用CentOS release 6.3(Final)(内核2.6.32-279) 遵循POSIX，LSB等标准，提供了64位程序开发与运行环境。

## 磁盘共享

软件共享目录：基础软件共享目录为/Share/util，应用软件共享目录为/Share/app。

用户目录~：登录节点、所有计算节点通过NFS模式共享用户目录管理节点6T的存储空间。**自家目录下所有文件在登录节点login01.用户只能通过login01修改~/下的数据。**

**用户目录提供稳定的磁盘访问模式，用户的软件、模型数据（输入文件等）建议存放在用户目录下。**

A、B集群用户必须使用lsf作业管理软件C集群用户用sge提交作业，才能使用计算节点。（lsf和sge使用方法见下第5章）用户在撰写提交脚本时，**建议直接使用vi编辑器编写。用户也可以异地编写上传到用户工作目录中，但是一定要注意dos2unix转换一下。**

# 用户登录

计算平台所有集群登录步骤类似，仅输入IP地址不同，以下以A集群为例进行介绍：

## 远程访问软件

用户需要使用支持SSH协议的相关软件访问系统，我们推荐使用SSH Secure Shell、SecureCRT等。

## 登录步骤

输入登录前端机login01，**IP：166.111.30.165**，并键入申请的用户名密码。


## 数据传输

单击SSH Secure Shell工具栏中的File Transfer键

得到如下窗口，将源程序及数据文件拷贝到登录节点上。


## 使用集群

用户可以根据需求，进行程序调试（参见4编译及测试环境）或提交作业（参见5作业管理系统使用说明）进行计算。

**注意：初次使用集群系统的用户，必须在编译环境中调试软件。确保软件正常运行后，MPI并行作业跨节点运行正常，再使用LSF或SGE提交作业，以免作业运行出错导致机器故障（死机或网络阻塞）！！**

# 编译及测试环境

## 访问编译环境

## 软件资源

系统在软件服务器（appserver）预安装intel编译器，并基于Intel编译器安装各类MPI并行库、数学库及各类应用软件。所有节点共享软件服务器软件资源，基础软件共享目录为/Share/util，应用软件共享目录为/Share/app。用户在login01可访问共享目录内容。

集群系统采用Intel X86\_64处理器，**推荐用户优先使用intel编译器及mkl数学库进行软件安装优化，**以提高程序执行效率。集群系统采用infiniband网络，系统安装了基于infiniband网络的3种MPI并行编程环境impi-4.1.0、openmpi-1.6.2，**推荐用户使用以上3种MPI并行编程环境**，获得高速网络通信。

平台主要的软件资源如下表所示：

|  |  |
| --- | --- |
| 目录 | 资源说明 |
| /Share/util | 平台基础软件目录，包括gcc，mpi，java，fftw，glibc等 |
| /Share/app | 平台应用软件目录，包括relion，R，EMAN，EMAN2，Xmipp等 |
| /opt/intel | Intel软件（C/C++/fortran/mkl/impi等） |

表格 4.2‑1软件资源目录分布

## 配置用户的环境变量

用户在编译及运行程序之前，**必须在用户自家的~/.bashrc文件中配置环境变量指定使用的编译器、MPI编译环境、数学库等相关路径，对PATH和LD\_LIBRARY\_PATH进行正确设置。**手册提供了各类编译环境下的环境变量设置方法，请用户恰当选择编译环境，并将对应命令行添加在~/.bashrc文件中，完成环境设置。

A集群环境设置：

**glibc环境设定：**

source /Share/scripts/gcc44.sh

source /Share/scripts/glibc214.sh

 **gcc4.4.7环境设定**

source /Share/scripts/gcc44.sh

* **Intel编译器串行、OpenMP并行程序环境设定：**

source /opt/intel/bin/iccvars.sh intel64

source /opt/intel/bin/ifortvars.sh intel64

* **Intel编译器及Intelmpi环境设定：**

source /opt/intel/bin/iccvars.sh intel64

source /opt/intel/bin/ifortvars.sh intel64

source /opt/intel/impi/4.1.0/bin64/mpivars.sh

* **Intel编译器及openmpi环境设定：**

source /opt/intel/bin/iccvars.sh intel64

source /opt/intel/bin/ifortvars.sh intel64

export LD\_LIBRARY\_PATH=/Share/util/ompi162-gnu/lib:$LD\_LIBRARY\_PATH

export PATH=/Share/util/ompi162-gnu/bin:$PATH

export LD\_LIBRARY\_PATH=/opt/intel/lib/intel64:$LD\_LIBRARY\_PATH

* **MKL数学库环境设定：**

source /opt/intel/composer\_xe\_2013.1.117/mkl/bin/intel64/ mklvars\_intel64.sh

* **其他环境设置模板：**

export LD\_LIBRARY\_PATH=库路径:$LD\_LIBRARY\_PATH

export PATH=可执行文件路径:$PATH

C集群环境设置：

**openmpi编译器环境设定：**

export PATH=/opt/openmpi/bin:$PATH

export LD\_LIBRARY\_PATH=/opt/openmpi/lib:$LD\_LIBRARY\_PATH

## 编译及测试

### Intel编译器编译串行程序及Openmp程序

* icc:编译C程序：

编译：icc –o prog prog.c

运行：./prog

* icpc:编译C++程序：

编译：icpc –o prog prog.cpp

运行：./prog

* ifort:编译fortran程序

编译：ifort –o prog prog.f90

编译：ifort –o prog prog.for

运行：./prog

* 编译Openmp程序

编译：icc –o prog-omp –openmp prog-omp.c

编译：ifort –o prog-omp –openmp prog-omp.f90

运行：export OMP\_NUM\_THREADS=启动线程数 (启动线程数<=12)

./ prog-omp

**常用编译选项：**

（1）优化选项 ：

-O0：禁止优化

-O1：优化代码大小和代码局部性。

-O2（缺省值）：**优化代码速度（推荐使用）**

-O3：-O2+激进的优化（循环、存储访问转换、预取）。**需要注意的是，-O3并不一定适合所有程序。**

-fast：打开-O3、-ipo、-static、-no-prec-div和 –xP

-ipo：过程间优化

（2） 输出和调试选项

-c：只生成目标文件

-S：只生成汇编文件

-g：调试选项

-o <file>：指定生成的输出文件名

（3） 浮点选项

-mp：维持浮点精度（禁止某些优化）

-mp1：改善浮点精度。和-mp相比，-mp1对性能影响较小

（4） 链接选项

-L<dir>：指定链接时搜索的库路径

-l<string>：链接特定库

-static：静态链接

-shared：生成共享库

### Intel编译器编译运行mpi并行程序

系统基于intel编译器安装了多种mpi，安装目录在/opt/intel/impi下。impi openmpi支持infiniband网络，可获得较快的计算速度.

**并行程序编译运行之前，请参看4.3部分，核对环境变量的设置，确认无误后再进行mpi程序的编译。**

* **intel mpi的使用**

**程序安装路径：**

/opt/intel/impi/

**impi程序编译：**

使用mpiicc、mpiicpc、mpiifort来编译c、c++、fortran程序，底层调用的是intel编译器的icc、icpc、ifort进行编译。

编译方法如下:

mpiicc –o prog-mpi prog.c

mpiicpc –o prog-mpi prog.cpp

mpifort –o prog-mpi prog.for

mpifort –o prog-mpi prog.f90

**impi也提供mpicc和mpif90内部命令,其底层调用的gcc和gfortran编译程序。用户在编译及安装软件时请注意这一点。**

**impi程序运行：**

intel mpi与其他mpi编程工具不同，运行之前需要启动MPD守护进程，再通过-machinefile文件指定进程分布。

1. 指定运行作业的节点。建立hosts文件，内容为：

node01:12

node02:12

其中node01为运行节点，12为在node01运行的进程数。

1. 启动impi所需要的后台进程，

使用mpdboot –n 2 –r ssh –f hosts

-n 2：为启动两个节点

-r ssh:使用ssh 协议

-f hosts:使用hosts文件作为进程启动说明文件。

1. 查看mpd进程是否启动

运行命令mpdtrace，可以看到

node01

node02

说明启动成功

1. 运行mpi

mpiexec –machinefile hosts –n 24 ./prog-mpi

-n:启动进程数

1. 关闭后台进程

程序结束后，运行mpdallexit

* **openmpi的使用：**

**程序安装路径：**

/Share/util/ompi162-gnu

**openmpi的编译：**

openmpi使用mpicc、mpicxx、mpif77、mpif90来编译程序c、c++、f77、f90，底层均调用的是intel编译器。

mpicc –o prog-mpi prog.c

mpicxx –o prog-mpi prog.cpp

mpif77 –o prog-mpi prog.for

mpif90 –o prog-mpi prog.f90

**openmpi的运行：**

使用mpiexec或mpirun直接运行命令，无需启动mpd后台进程。

1. 指定运行作业的节点。建立hosts文件，内容为：

node01:12

node02:12

1. 运行mpi

mpiexec –machinefile hosts –n 24 ./prog-mpi

-n:启动进程数

**常见问题：**

1. warning: feupdateenv is not implemented and will always fail

解决：mpicc -o cpi -limf cpi.c

1. orted: error while loading shared libraries: libimf.so

解决：各类库冲突,或者没有查找到。检查intel编译器及openmpi环境变量是否设置正确。在LD\_LIBRARY\_PATH 中添加export /opt/intel/lib/intel64

### 其他注意事项

* 1. 程序运行以后想杀掉程序，直接按ctrl+c，就可以杀掉一个mpirun启动的所有进程。
	2. 平台推荐用户使用intel、及基于Intel编译器的mpi并行编程环境。用户如果需要其他环境配置，可直接和管理员联系，管理员将根据需求安装gcc、pgi编译器及基于相关编译器的mpi编程环境。
	3. 每个用户的自家目录都限制了磁盘限额，**请不要上传和计算无关文件，并及时做好数据备份和清理工作**。
	4. **系统/tmp目录为内存虚拟目录，大小只为100M。如果程序需要有临时文件写入，可将临时文件目录指定为：/scratch。**
	5. **用户在编写程序的时候，应尽力减少文件的产生。在内存足够的情况下，尽量把中间结果存在内存中，从而减少由于磁盘IO访问所带来的计算瓶颈。**

# 作业管理系统使用说明

## LSF作业管理系统使用说明

A、B集群使用LSF作业管理系统进行作业的管理与分配。用户只需用LSF提交命令（bsub）将作业提交到集群，系统就会按照管理员制定的作业分配策略自动进行调度，决定何时以及在哪些计算结点运行程序。作业管理系统不仅方便用户使用，更提高了整个系统使用效率。

### 队列设定

目前系统80nodes A中建立了13个队列,可使用bqueues命令查看：

QUEUE\_NAME PRIO STATUS MAX JL/U JL/P JL/H NJOBS PEND RUN SUSP

SUILAB 60 Open:Active 160 - - - 160 0 160 0

APPION2.2 60 Open:Active 160 - - - 0 0 0 0

APPION2.3 60 Open:Active 160 - - - 0 0 0 0

NICHE 30 Open:Active 480 - - - 0 0 0 0

normal 30 Open:Active 0 - - - 0 0 0 0

GPU 30 Open:Active 10 - - - 0 0 0 0

BNODE 30 Open:Active 32 - - - 0 0 0 0

GIANT 30 Open:Active 480 - - - 0 0 0 0

HUGE 30 Open:Active 480 - - - 208 0 208 0

MID 30 Open:Active 480 - - - 160 0 160 0

SMALL 30 Open:Active 480 - - - 352 160 192 0

TINY 30 Open:Active 80 - - - 0 0 0 0

TEST 30 Open:Active 480 - - - 0 0 0 0

TOTAL 30 Open:Active 480 - - - 0 0 0 0

### 提交作业(bsub)

#### bsub命令基本用法

1. 提交作业:bsub command

$ **bsub sleep 60**

Job <55163> is submitted to default queue <HUGE>.

向LSF提交作业，获得唯一ID55163,作业提交成功。

1. 向某个队列提交作业：bsub –q。

$ **bsub -q TOTAL sleep 60**

Job <55164> is submitted to queue <TOTAL>.

1. 用-o.-e制定标准输出和error文件位置

$ **bsub -o output.%J -e errors.%J ls-l**

Job <55165> is submitted to queue <TOTAL>.

%J 代表作业ID

**注意：用户的可执行程序必须写在-o –e选项后面**

1. 用-i指定输入文件

有些可执行程序运行时采用<方式来输入可执行文件

lsf可使用-i指定输入文件

5. 用-m 指定运行机器

**$ bsub –m “hosta hostb” hostname**

**bsub详细用法可以使用man bsub，参考说明**

#### OpenMP并行作业提交

使用**openmp**关键字

**例1：**提交作业16核openmp，并保证作业独占该计算节点。

**bsub -a openmp -n 16 -R "span[hosts=1]" myOpenMPJob**

#### MPI并行作业提交

用**mpirun.lsf**关键字提交作业，并使用**-a**选项指定所选用的mpi。**不同mpi要使用不同的关键字。**

**例1：**提交intelmpi并行作业

bsub -a **intelmpi** –o output.%J –e error.%J -n 16 **mpirun.lsf** /cpi

**例2：**提交mvapich并行作业

bsub -a **mvapich** –o output.%J –e error.%J -n 16 **mpirun.lsf** /examples/cpi

**例3：**提交openmpi并行作业

bsub –a **openmpi** –o output.%J –e error.%J -n 16 **mpirun.lsf** /examples/cpi

#### 大内存并行作业提交

系统计算节点内存有两种配置，48G和32G。需要大内存的用户在提交作业时必须使用-R选项把作业提交到大内存节点(内存48G)上，使用方法如下：

**例1**：将作业提交到内存剩余总量超过42G的计算计算节点上

bsub -a intelmpi –R "select [mem>42000]" -n 16 mpirun.lsf /examples/large\_mem

其中，单位为MB

用户可以根据自身需求设定剩余总量的限制。

此外，为防止内存不足造成的计算缓慢或系统死机等问题。系统设置计算节点剩余内存不足500M时，节点上相关作业将会被挂起，作业状态为SSUSP。用户如果发现作业为SSUSP，请及时和管理员联系，确认挂起原因。

#### 使用脚本提交作业

为使用方便，用户可以自行撰写脚本提交作业，每次直接运行脚本即可。

撰写脚本有两种方式：

**方法1：**建立包含bsub的脚本

创建文件（如job），在job中写入bsub提交命令，如：

bsub -a **intelmpi** –o output.%J –e error.%J -n 12 **mpirun.lsf** /examples/cpi

然后chmod +x job

直接运行./job，就可以提交作业。

**方法2：**使用bsub 脚本多次提交具有相同参数的作业，其格式如下：

#!/bin/sh

#BSUB -q QUEUENAME

#BSUB –a MPITYPE

#BSUB -n Z

#BSUB -o OUTPUTFILE

#BSUB -e ERRFILE

mpirun.lsf program

用户根据实际需求可以添加其他选项。

提交脚本,运行命令bsub <脚本名，即可提交作业。

该脚本等同于命令：

**bsub -q QUEUENAME –a MPITYPE -n Z -o OUTPUTFILE -e ERRFILE mpirun.lsf program**

**推荐用户使用方法2“bsub脚本模式”提交作业。**

**提交作业如果需要其他选项，如-J、-R、-M、-W、等请按照以上格式自己添加。**

例如：提交intelmpi作业

1. 创建文件job,内容如下：

#BSUB -q TINY

#BSUB -a intelmpi

#BSUB –n 24

#BSUB –o output.%J

#BSUB –e error.%J

mpirun.lsf ./mpi\_openmp\_hello

1. 用bsub提交作业：

bsub < job

### 状态查看

#### 查看作业状态(bjobs)

作业提交后，用户使用bjob命令查看作业ID和状态

**$ bjob**s

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

55167 gyfeng RUN HUGE login01 node27 sleep 60 Nov 18 15:45

一个作业提交后，将可能为以下几种状态之一：

|  |  |
| --- | --- |
| STAT | 状态 |
| PEND | 任务在队列中排队等待 |
| RUN | 任务正在执行 |
| PSUSP | 任务在队列中排队等待时被用户挂起 |
| SSUSP | 任务被系统挂起 |
| DONE | 作业正常结束，exit代码为0 |
| EXIT | 作业退出，exit代码不为0 |

**常用选项：**

-a: 除了可以查看已提交及尚未结束的作业，还可以看到刚结束不久的作业信息

-u：查看系统其它用户作业情况，如：

查看user1的作业：bjobs –u users1

查看所有人的作业：bjobs –u all

-l :查看某个作业详细信息

查看作业JOBID 详细信息：bjobs –l JOBID

#### 查看运行作业的标准（屏幕）输出（bpeek）

**$ bpeek 55167**

#### 查看作业历史运行情况(bhist)

**$ bhist –l 55167**

#### 查看用户状态（busers）

**$ busers**

USER/GROUP JL/P MAX NJOBS PEND RUN SSUSP USUSP RSV

gyfeng - - 0 0 0 0 0 0

Max：用户可用核数上限

NJOBS:已提交作业所需要的全部核数

PEND:在队列中等待执行的所有作业的核数

RUN: 正在运行作业的核数

SSUSP: 系统挂起用户作业核数

USUSP：用户自行挂起作业的核数

RSV：系统预约保留的核数

#### 查看队列状态（bqueues）

**$ bqueues**

QUEUE\_NAME PRIO STATUS MAX JL/U JL/P JL/H NJOBS PEND RUN SUSP

SUILAB 60 Open:Active 160 - - - 160 0 160 0

APPION2.2 60 Open:Active 160 - - - 0 0 0 0

APPION2.3 60 Open:Active 160 - - - 0 0 0 0

NICHE 30 Open:Active 480 - - - 0 0 0 0

normal 30 Open:Active 0 - - - 0 0 0 0

GPU 30 Open:Active 10 - - - 0 0 0 0

BNODE 30 Open:Active 32 - - - 0 0 0 0

GIANT 30 Open:Active 480 - - - 0 0 0 0

HUGE 30 Open:Active 480 - - - 208 0 208 0

MID 30 Open:Active 480 - - - 160 0 160 0

SMALL 30 Open:Active 480 - - - 352 160 192 0

TINY 30 Open:Active 80 - - - 0 0 0 0

TEST 30 Open:Active 480 - - - 0 0 0 0

TOTAL 30 Open:Active 480 - - - 0 0 0 0

bqueues –l: 查询某个队列的详细信息

**$ bqueues -l TINY**

#### 查询系统各主机状态（bhosts）

**$ bhosts**

HOST\_NAME STATUS JL/U MAX NJOBS RUN SSUSP USUSP RSV

bnode01 ok - 32 0 0 0 0 0

gpunode01 ok - 16 0 0 0 0 0

gpunode02 ok - 16 0 0 0 0 0

login01 closed - 0 0 0 0 0 0

mgt ok - 8 0 0 0 0 0

node01 ok - 16 0 0 0 0 0

node02 ok - 16 0 0 0 0 0

node03 ok - 16 0 0 0 0 0

node04 ok - 16 0 0 0 0 0

node05 ok - 16 0 0 0 0 0

OK：该节点可以接收用户作业

Closed：已经有作业运行或负载过高。

#### 查询各主机系统状态（lsload）

**$ lsload**

HOST\_NAME status r15s r1m r15m ut pg ls it tmp swp mem

node09 ok 0.0 0.0 0.0 0% 0.0 0 38784 424G 1G 61G

node27 ok 0.0 0.1 1.0 0% 0.2 0 10264 424G 450M 61G

node04 ok 0.0 0.1 0.0 0% 0.0 0 8552 422G 1G 61G

node06 ok 0.0 0.1 0.0 0% 0.0 0 8552 424G 1G 61G

node02 ok 0.0 0.0 0.0 0% 0.0 0 5484 422G 1G 61G

node10 ok 0.0 0.1 0.0 0% 0.0 0 38784 424G 1G 61G

node03 ok 0.0 0.1 0.1 0% 0.0 0 8552 422G 1G 61G

node08 ok 0.0 0.0 0.0 0% 0.0 0 8544 424G 1G 61G

login01 ok 0.0 0.1 0.1 1% 0.0 4 1 250G 1G 28G

node58 ok 0.0 0.1 1.4 0% 0.4 0 38784 424G 524M 61G

bnode01 ok 0.0 0.1 0.1 0% 0.0 0 1392 252G 1024M 502G

### 控制作业执行

#### 删除作业（bkill）

用bkill停止作业运行。

**$ bkill 55167**

Job <55167> is being terminated

使用bkill删除并行作业时，lsf需要收集信息、发送信号等处理，用户执行bkill命令后，作业可能没有立即删除，使用bjobs命令还可以看到作业。**请用户耐心等待（大约1分钟），lsf将完整作业删除工作。**

#### 作业挂起（bstop）

用bstop挂起正在运行的作业，需要指明作业ID：

**$ bstop 55167**

Job <55167> is being stopped

**$ bjobs**

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

55167 gyfeng USUSP TOTAL login01 node58 sleep 60 Nov 18 15:51

挂起之后，STAT为USUSP。

#### 作业恢复（bresume）

用bresume恢复作业运行

**$ bresume 1266**

Job <55167> is being resumed

**$ bjobs**

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

55167 gyfeng RUN TOTAL login01 node58 sleep 60 Nov 18 15:55

#### 调整队列（bwitch）

用bswitch将正在运行的作业调度到其他队列中

**$bswitch MID 55179**

Job <55179> is switched to queue <MID>

#### 改变作业排队次序（btop/bbot）

 用户可以使用btop/bbot改变本用户提交且处于“PEND”状态的作业调度次序。

 btop: 指定队列中，所有同优先级作业最先获得调度。

 bbot: 指定队列中，所有同优先级作业最后获得调度。

**bjobs**

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

55179 gyfeng RUN MID login01 node58 /s500 Nov 18 15:59

55180 gyfeng PEND MID login01 /s200 Nov 18 16:15

55181 gyfeng PEND MID login01 /s700 Nov 18 16:23

**btop 55181**

Job <55181> has been moved to position 1 from top.

**bjobs**

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

55179 gyfeng RUN MID login01 node58 /s500 Nov 18 15:59

55180 gyfeng PEND MID login01 /s200 Nov 18 16:15

55181 gyfeng PEND MID login01 /s700 Nov 18 16:23

### 作业提交脚本

**典型脚本**

#!/bin/sh

#BSUB -q MID

#BSUB -a openmpi

#BSUB -o output.%J

#BSUB -e error.%J

#BSUB -n XXXcoresXXX

#BSUB -R "span[ptile=16]"

# Environment

source ~/.bashrc

rm -rf ./hosts

touch ./hosts

#construct the hosts file for the job

j=''

k=0

for i in `echo $LSB\_HOSTS`

do

 ((k = k + 1))

 if [ $((k % 16)) -eq 1 ]

 then

 echo $k

 echo $i slots=16>>./hosts

 fi

done

#job submission

mpirun -machinefile ./hosts --bynode -n XXXmpinodesXXX XXXcommandXXX

## SGE作业管理系统使用说明

C集群使用SGE作业管理系统进行作业的管理与分配。用户只需用sge提交命令（qsub）将作业提交到集群，系统就会按照管理员制定的作业分配策略自动进行调度，决定何时以及在哪些计算结点运行程序。作业管理系统不仅方便用户使用，更提高了整个系统使用效率。

### 提交作业(qsub)

#### 简单作业

脚本示例：test.sh

#!/bin/sh

sleep 20

echo 'hello'

date

键入提交命令：qsub < test.sh

查看状态：qstat

收到的状态报告应能提供 Grid Engine 系统当前可识别到的所有作业的信息。对于每
项作业，状态报告列出以下各项：
■ 作业 ID， 包含在提交信息中的唯一编号
■ 作业脚本的名称
■ 作业的拥有者
■ 状态指示符，例如， r 表示正在运行
■ 提交或启动时间
■ 运行作业的队列名称
如果 qstat 没有产生输出，则表示系统没有识别到作业。例如，作业已完成。
通过检查 stdout 和 stderr 重定向文件可以控制已完成作业的输出。默认情况
下，这些文件在运行作业的主机上作业拥有者的根目录下生成。文件名称由作业脚本
文件名称、扩展名 .o （对于 stdout 文件）或扩展名 .e （对于 stderr 文件）和
唯一的作业 ID 组成。

#### 并行作业

脚本示例：在/Share/ptest中的PI.qsub

#!/bin/sh

#$ -N PI-test

#$ -S /bin/bash

#$ -q relion.q

##$ -e test.err

##$ -o test.log

#$ -cwd

#$ -pe make 16

mpirun -v PI

键入提交命令：qsub < PI.qsub

查看状态：qstat

-N : 作业名称

-q: 指定队列

-e: 指定报错文件名（）

-o： 指定输出文件名

-cwd: 在当前路径执行

-pe: 指定并行环境并指定申请核数

### 状态查看

#### 查看作业状态(qstat)

作业提交后，用户使用qstat命令查看作业ID和状态

$qstat

job-ID prior name user state submit/start at queue slots ja-task-ID

-----------------------------------------------------------------------------------------------------------------

447851 0.60500 PI-test gyfeng qw 01/16/2015 10:48:51 16

一个作业提交后，将可能为以下几种状态之一：

|  |  |
| --- | --- |
| STAT | 状态 |
| qw | 作业排队等待 |
| r | 作业正在执行 |
| s | 作业被挂起 |
| R | 作业重启 |
| d | 作业被删除 |
| E | 作业发生错误 |

**常用选项：**

-f: 查看所有队列作业

-u：查看系统其它用户作业情况，如：

查看user1的作业：qstat –u users1

查看所有人的作业：qstat –u ‘\*’

-j: 查看某个作业详细信息

查看作业JOBID 详细信息：qstat –j JOBID

#### 查询系统各主机状态（qhost）

**$qhost**

HOSTNAME ARCH NCPU LOAD MEMTOT MEMUSE SWAPTO SWAPUS

------------------------------------------------------------------------------- -

cluster linux-x64 8 0.33 15.6G 6.4G 31.2G 16.2G

node-0-1 linux-x64 16 16.15 63.0G 31.1G 1000.0M 94.1M

node-0-10 linux-x64 16 16.21 63.0G 31.9G 1000.0M 189.8M

node-0-11 linux-x64 16 16.12 63.0G 31.4G 1000.0M 35.8M

node-0-12 linux-x64 16 15.99 63.0G 27.7G 1000.0M 29.3M

node-0-13 linux-x64 16 14.99 63.0G 25.5G 1000.0M 26.7M

node-0-14 linux-x64 16 16.05 63.0G 29.9G 1000.0M 41.4M

node-0-15 linux-x64 16 15.98 63.0G 29.5G 1000.0M 43.0M

node-0-16 linux-x64 16 16.13 63.0G 30.9G 1000.0M 136.2M

node-0-17 linux-x64 16 16.06 63.0G 23.5G 1000.0M 14.3M

node-0-18 linux-x64 16 16.08 63.0G 25.0G 1000.0M 14.6M

node-0-19 linux-x64 16 16.02 63.0G 24.6G 1000.0M 16.0M

node-0-2 linux-x64 16 16.07 63.0G 29.1G 1000.0M 15.3M

node-0-20 linux-x64 16 14.81 63.0G 27.2G 1000.0M 34.6M

node-0-3 linux-x64 16 16.01 63.0G 29.2G 1000.0M 19.8M

node-0-4 linux-x64 16 16.03 63.0G 32.1G 1000.0M 38.5M

node-0-5 linux-x64 16 15.25 63.0G 31.8G 1000.0M 119.5M

node-0-6 linux-x64 16 16.05 63.0G 34.2G 1000.0M 28.8M

node-0-7 linux-x64 16 16.01 63.0G 33.9G 1000.0M 24.4M

node-0-8 linux-x64 16 14.86 63.0G 27.0G 1000.0M 67.5M

node-0-9 linux-x64 16 16.06 63.0G 31.4G 1000.0M 71.7M

### 控制作业执行

#### 删除作业（qdel）

用qdel停止作业运行。

$qdel 447851

gyfeng has deleted job 447851

#### 作业挂起及恢复（qmod）

 **挂起：**

用qmod挂起正在运行的作业，需要指明作业ID：

**$qmod -s 447976**

gyfeng - suspended job 447976

**$qstat**

job-ID prior name user state submit/start at queue slots ja-task-ID

-----------------------------------------------------------------------------------------------------------------

447976 0.60500 relion\_tes gyfeng  **s** 01/16/2015 11:35:38 relion.q@node-0-6.local 4

作业的状态变成s。被挂起

 **恢复：**

**$qmod -us 447976**

gyfeng - unsuspended job 447976

**$qstat**

job-ID prior name user state submit/start at queue slots ja-task-ID

-----------------------------------------------------------------------------------------------------------------

447976 0.60500 relion\_tes gyfeng r 01/16/2015 11:35:38 relion.q@node-0-6.local 4

作业状态重新变为r

附录1：Linux基本命令

1. **目录操作**

**名称：cd**

**语法：**cd [directory]

**说明：**把当前工作目录转到” directory”指定的目录。

**实例：** 进入目录 /usr/bin/：

cd /usr/bin

**名称 : ls**

**语法：**ls [options] [pathname-list]

**说明：**显示目录内的文件名和“pathname-list”中指定的文件名

**实例：**列出目前工作目录下所有名称是 s 开头的文件：

ls s\*

**名称：pwd**

**语法：**pwd

**说明：**显示当前目录的绝对路径。

**名称： mkdir**

**语法：**mkdir [options] dirName

**说明：**创建名称为 dirName 的子目录。

**实例：**在工作目录下，建立一个名为 AA的子目录：

mkdir AA

**名称：rmdir**

**语法：** rmdir [-p] dirName

**说明：** 删除空的目录。

**实例：**将工作目录下，名为 AA的子目录删除 :

 rmdir AA

1. **文件操作**

**名称：cp**

**语法：**cp [options] file1 file2

**说明：**复制文件file1到file2。

**常用选项：**-r 整个目录复制

**实例：**将文件 aaa 复制(已存在)，并命名为 bbb :

cp aaa bbb

**名称：mv**

**语法：**mv [options] source... directory

**说明：**重新命名文件，或将数个文件移至另一目录。

**范例：**将文件 aaa 更名为 bbb :

mv aaa bbb

**名称：rm**

**语法：**rm [options] name...

**说明：**删除文件及目录。

**常用选项：**-f 强制删除文件

**实例：**删除除后缀名为.c的文件

 rm \*.c

**名称：cat**

**语法：**cat[options] [file-list]

**说明：**在标准输出上连接、显示文件列表file-list里的文件

**实例1：**显示file1和file2的内容

 cat file1 file2

**实例2：**将file1和file2合并成file3

 cat file1 file2 > file3

**名称：more**

**语法：**more[options] [file-list]

**说明：**在标准输出上连接、分页显示文件列表file-list里的文件

**实例：**分页显示文件AAA

 more AAA

**名称：head**

**语法：**head[options] [file-list]

**说明：**显示文件列表file-list中的文件的起始部分，默认显示10行；

**实例：**显示文件AAA起始部分

 head AAA

**名称：tail**

**语法：**tail[options] [file-list]

**说明：**显示文件列表file-list中的文件的尾部；默认显示10行；

**实例：**显示文件AAA尾部

 tail AAA

**名称：ln**

**语法：**ln[options] existing-file new-file

ln[options] existing-file-list directory

**说明：**为“existing-file”创建链接，命名为new-file

在directory目录，为existing-file-list”中包含的每个文件创建同名链接

**常用选项：**-f 不管new-file是否存在，都创建链接

 **-s** 创建软链接

**实例1：**建立软连接temp.soft,指向Chapter3

 ln –s Chapter3 temp.soft

**实例2**：为examples目录下的所有文件和子目录建立软连接

 ln –s ~/linuxbook/examples/\* /home/faculty/linuxbook/examples

**名称：chmod**

**语法：**chmod [option] mode file-list

**说明：**改变或设置参数file-list中的读、写或执行权限

**实例：**添加文件job的可执行权限

 chmod +x job

 **名称： tar**

**语法：**chmod [option] [files]

**说明：**备份文件。可用来建立备份文件，或还原备份文件。

**实例1：**备份test目录下的文件，并命名为test.tar.gz，可执行命令：

tar –zcvf test.tar.gz test

**实例2**：解压缩相关的test.tar.gz文件，可执行命令：

tar –zxvf test.tar.gz

1. **其他**

**名称：echo**

**语法：**echo $variable

**说明：**显示变量variable的值。

**实例1：**显示当前用户路径PATH的值

echo $PATH

**名称：ps**

**语法：**$ps [options]

**说明：**用于查看当前系统中的活跃进程

**实例1：**显示当前所有进程

ps –aux

**名称：kill**

**语法：**$kill [-signal] pid

**说明：**终止指定进程

**实例1：**终止1511号进程

kill 1511

附录2：Vi使用

1. **简要使用流程**
2. 使用 "vi [选项] [文件 ..]" 命令打开要编辑的文件
3. 使用 "方向键" 浏览文件
4. 按下 "i" 进入编辑模式
5. 编辑
6. 按 "Esc" 键退出编辑模式
7. 输入 ":w" 回车保存，再输入":q" 回车退出。或者直接输入 ":wq" 回车，代表保存并退出
8. **两种操作模式**
* 编辑模式：对文本进行编辑处理
* 命令模式: 接收按键指令执行操作，如复制、粘贴、搜索、替换、保存、另存为等
1. **编辑模式**

i: 进入编辑模式

a: 进入编辑模式，将光标向后移动一位

o: 进入编辑模式，在光标处插入一个空行

r: 按下 r 键，再按任意字符键，将光标所在处的字符替换成新输入的字符

Esc: 退出编辑模式

1. **命令模式**
* **移动光标**

↑或k：把当前光标向上移动一行，保持光标的列位置。

↓或j：把当前光标向下移动一行，保持光标的列位置。

→或l：把当前光标向右移动一个字符。

←或h：把当前光标向左移动一个字符。

$：把当前光标移动到该行行末。

^：把当前光标移动到该行行首。

w：把当前光标移动到该行的下一个字的首字符上。

b：把当前光标移动到该行的上一个字的首字符上。

e：把当前光标移动到该行的该字的末尾字符上。

^F：向前滚动一整屏正文。

^D：向下滚动半个屏正文。

^B：向后滚动一整屏正文。

^U：向上滚动半个屏正文。

* **搜索与替换**

/word: 从光标处开始，向后搜索文本中出现word的字符串

?word: 从光标处开始，向前搜索文本中出现word的字符串

:1,$s/word1/word2/g: 在第1行与最后一行之间搜索word1，并将其替换为word2

:n1,n2s/word1/word2/g: 在第 n1行与第n2行之间搜索word1，并将其替换为word2

* **删除 (剪切)、复制与粘贴**

x,X: x 为向后删除一个字符，X 为向前删除一个字符

dd: 删除光标所在行

yy: 复制光标所在行的内容

nyy: 复制光标到第 1 行的所有内容

y1G: 复制光标到第 1 行的所有内容

yG: 复制光标到最后一行的所有内容

p,P: p 为将复制或剪切的内容粘贴在光标下一行，P 为粘贴在上一行

u: 撤消上一操作

* **管理命令**

:w:保存

:w!: 强制保存

:q:退出 vi 编辑器

:q!:强制退出

:w [文件名]:另存为..

:r[文件名]:读取另一个文档的内容，内容追加到光标所在行之后

:set nu:显示正文的行号。

:set nonu:取消行号。

:![命令]：暂时离开 vi 编辑器，并在 shell 中执行命令