上海理工大學

UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

上 海 理 工 大 学 高 性 能 计 算 中 心

用 户 手 册

校内使用请勿公开传播

（使用指南初稿 v1.1）

2019 年 05 月

目 录

1 前言

本用户手册主要将对在上海理工大学超级计算中心曙光 TC4600E 百亿次超级计算系统上使用介绍，其它未详尽描述部分请参阅相关专项文献。

由于受水平和时间所限，错误和不妥之处在所难免，欢迎指出错误和改进意见，本人将尽力完善更新。

2 中心简介

上海理工大学以工学为主，工学、理学、经济学、管理学、文学、法学、艺术学等多学科协调发展，是一所上海市属重点建设的应用研究型大学。在国家建设“一流大学和一流学科”、上海市建设地方高水平大学的重要战略机遇期，上海理工大学正以未来光学、智能制造、医疗器械与康复工程 3 大国际实验室和系统管理 1 个特色平台为载体，建设光学工程、系统科学、动力工程及工程热物理、机械工程、生物医学工程 5 大一流学科。学校将抢抓机遇，改革创新，加快高水平大学建设，促进内涵发展，力争把学校建设成为特色显著的一流理工科大学。

契合各学科蓬勃发展，本中心拟建成一套高性能计算集群为全校师生提供计算平台，保障科学研究顺利开展。

系统采用 100Gbps 高速互联网络，共计 2 个管理节点、2 个用户登录节点、4 个存储节点、110 个 CPU 计算节点及 2 个 GPU 计算节点构成。CPU 节点整体理论峰值性能为 260 万亿次，GPU 双精度理论计算能力为 56TFLOPS。整套集群采用曙光 Gridview HPC 版集群软件统一进行调度与管理。

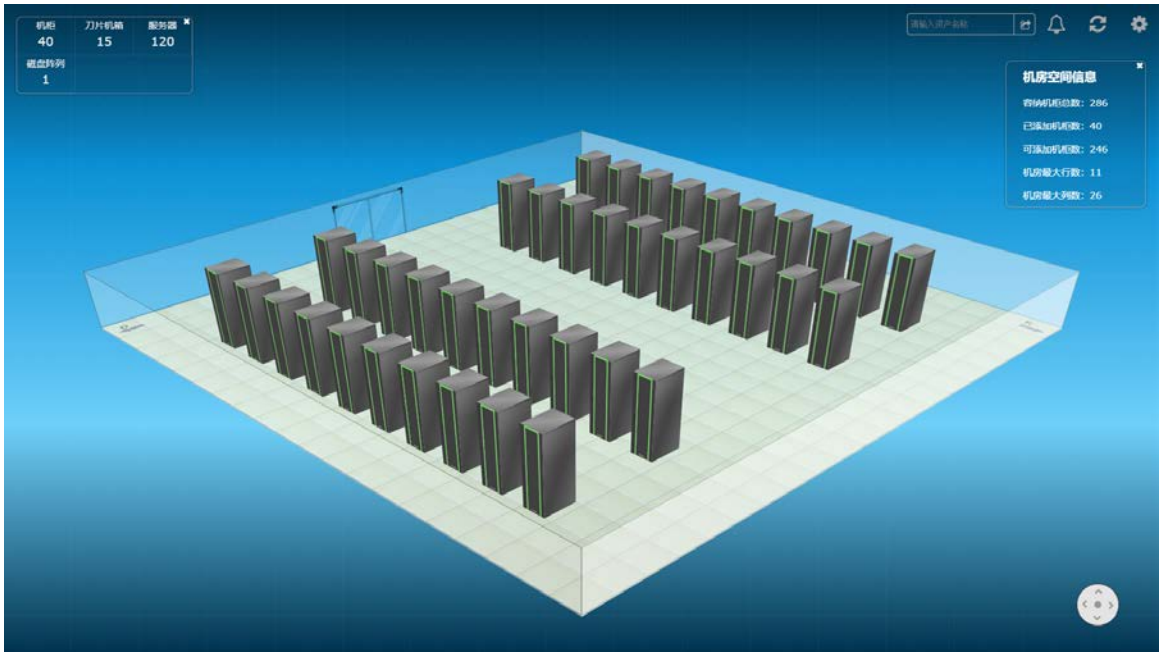
- 节点列表

功能	节点名	型号	CPU	内存	硬盘	GPU 型号
登陆节点	login 1-2	I620 -G30	2*Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	192GB DDR4 2666 MHz	480 GSSD	/
计算节点	comput 1-110	CX50 -G30	2*Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz	192GB DDR4 2666 MHz	240 GSSD	/

计算节点	gpu 1-2	X795-G30	2*Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz	192GB DDR4 2666 MHz	240 GSSD	4*Nvidia V100
存储节点	io 01-04	CX50-G35	2*Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	64GB DDR4 2666 MHz	924 T	/

- 实物展示

- 效果展示



3 使用集群进行计算业务

3.1 系统登陆

3.1.1 登陆 IP 地址

用户使用高性能计算集群，需要登陆集群的登陆节点 login1/login2，可以通过多种方式登录。login1 的 ip 地址是 http://10.10.114.203；login2 的 ip 地址是 http://10.10.114.204 ；两台登陆节点具有分流与冗余功能。

3.1.2 命令行终端登录

Windows 用户可以用 SSH Secure Shell Client，PuTTY，SecureCRT，Xshell 等 SSH 客户端软件登录。推荐使用 MobaXterm,它为开源的支持图形转发功能。要求任何软件均从其官方网站下载，避免潜在不安全因素。

Linux 用户可以直接在命令行终端中执行 ssh 命令进行登录：

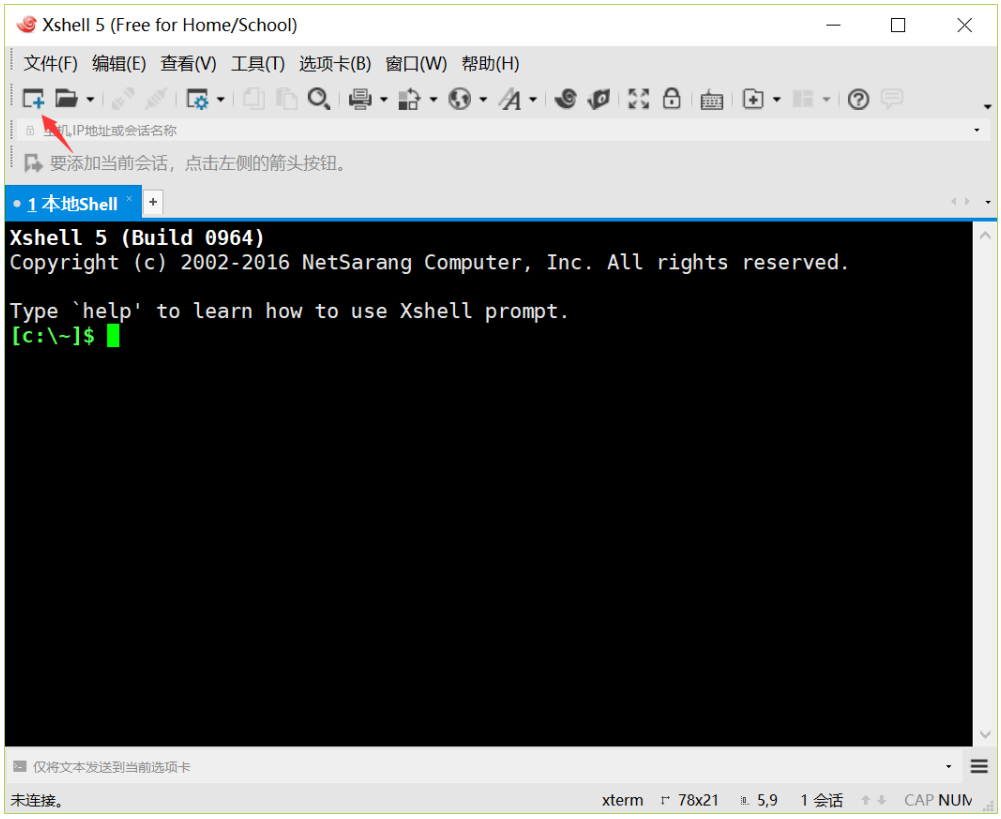
```
$ ssh username@10.10.114.203
```

下面以 xshell 为例介绍登陆方法

1：首先选择新建一个会话

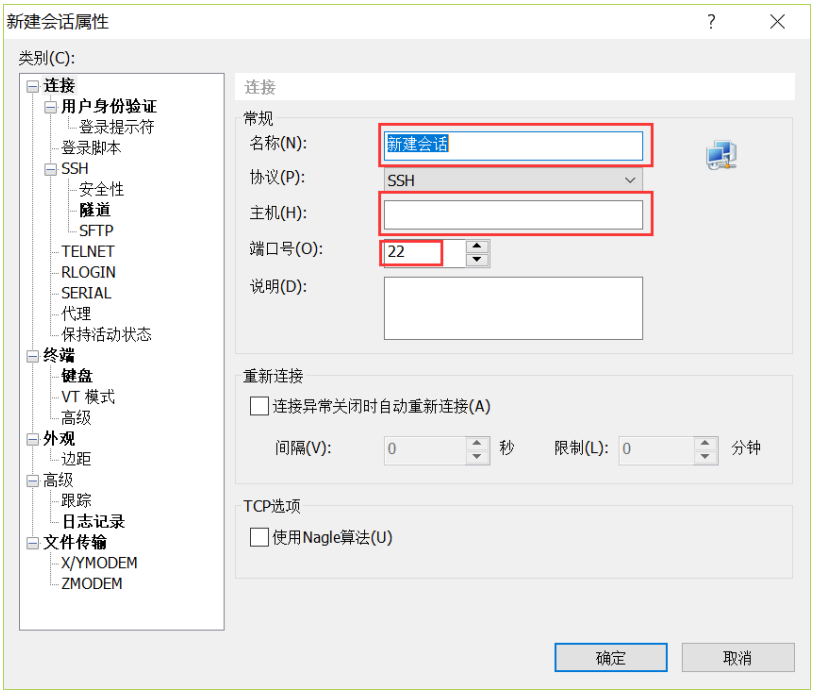
会话用来保存要登陆的 HPC 集群 IP 地址（或域名），SSH 端口号，用户名和密码（或密钥）信息，保存为 Profile 后，后续可直接点击保存好的 Profile

登陆，不需要每次都输入用户名和密码。



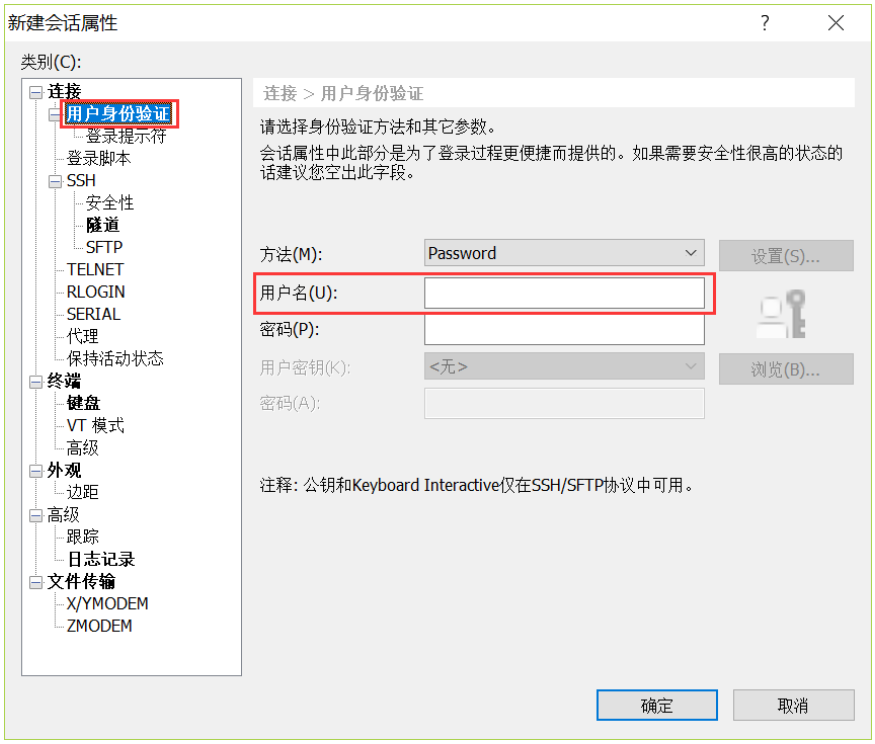
2．编辑会话，填写登陆节点 IP 地址

在下图中填写会话名称（可任意填写），主机部分添加高性能集群登陆节点的 IP 地址或域名，ssh 登陆端口一般选择默认端口 22.

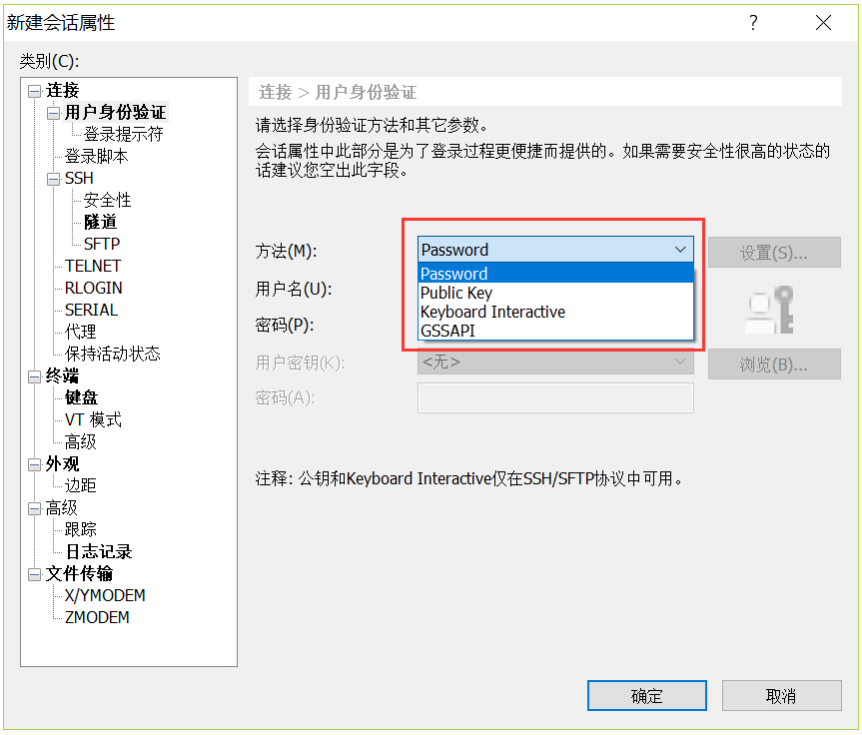


3．填写用户名和密码（或密钥）

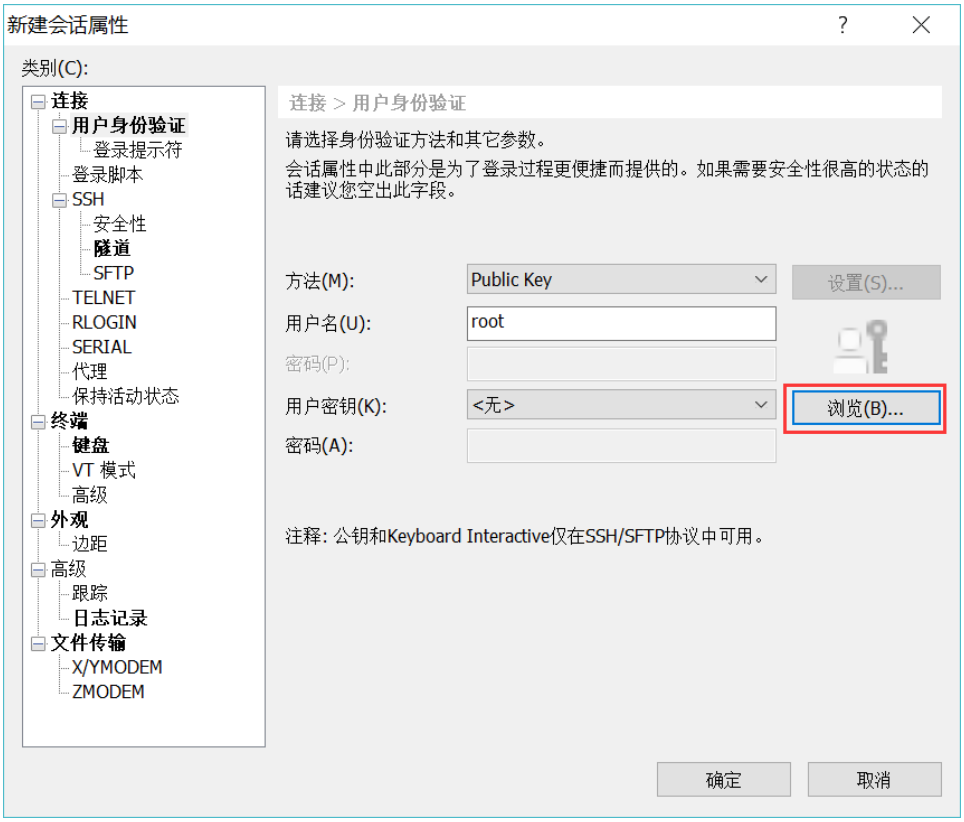
在该部分填写要登陆的用户密码或密钥。如使用密码登陆，可直接在下图中填写用户名及密码。

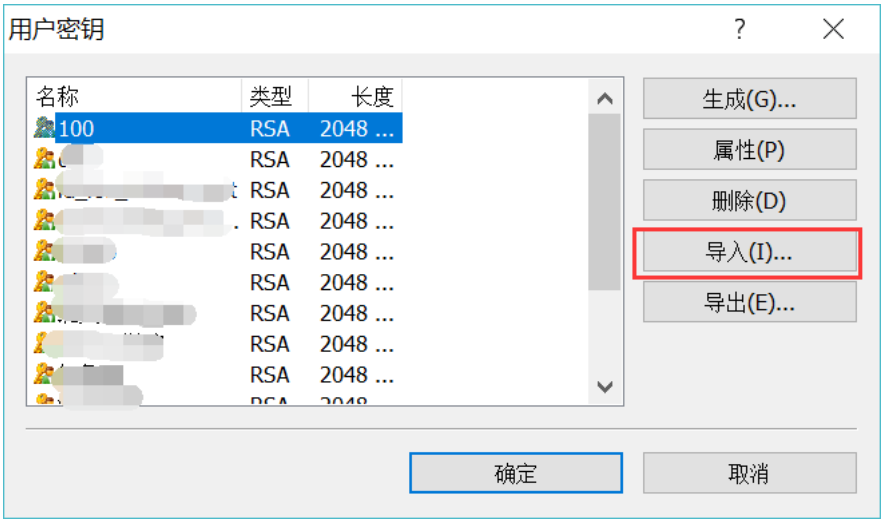


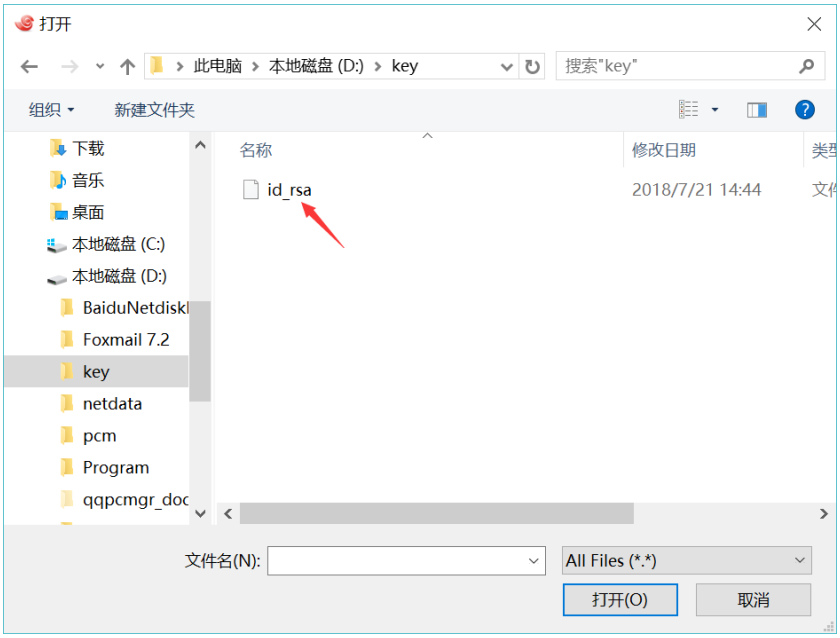
如果使用密钥登陆，可在方法中切换为 Public key 方式

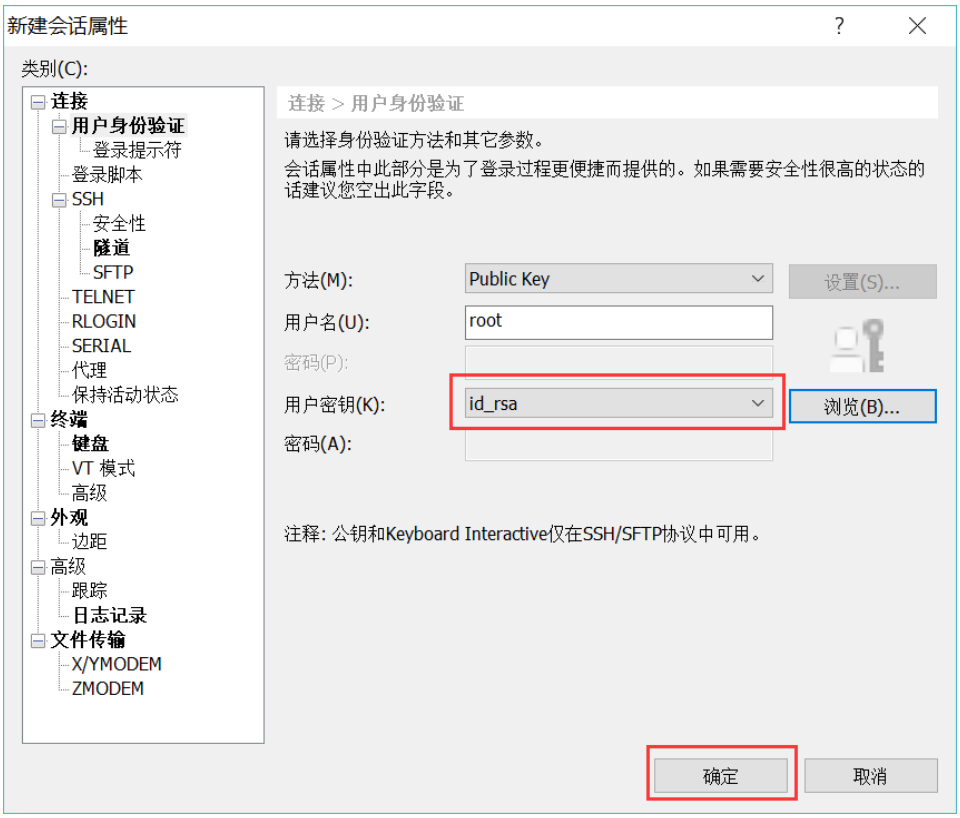


点击用户密钥旁的浏览，导入密钥

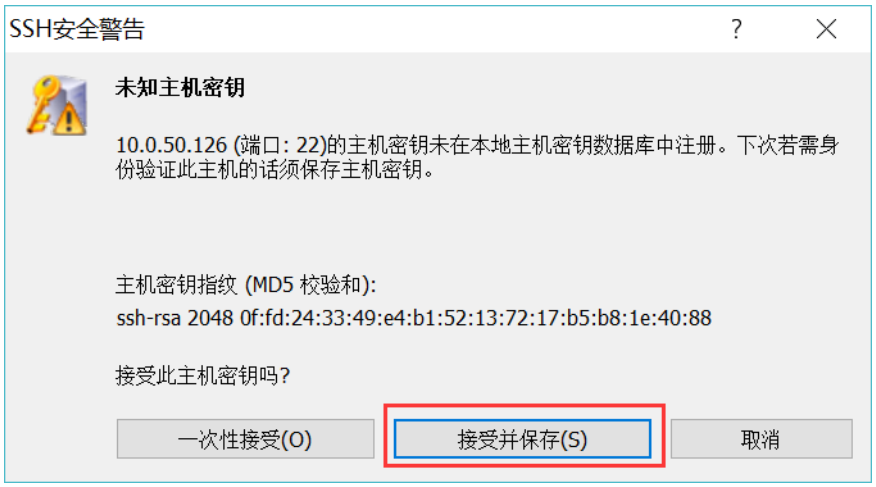




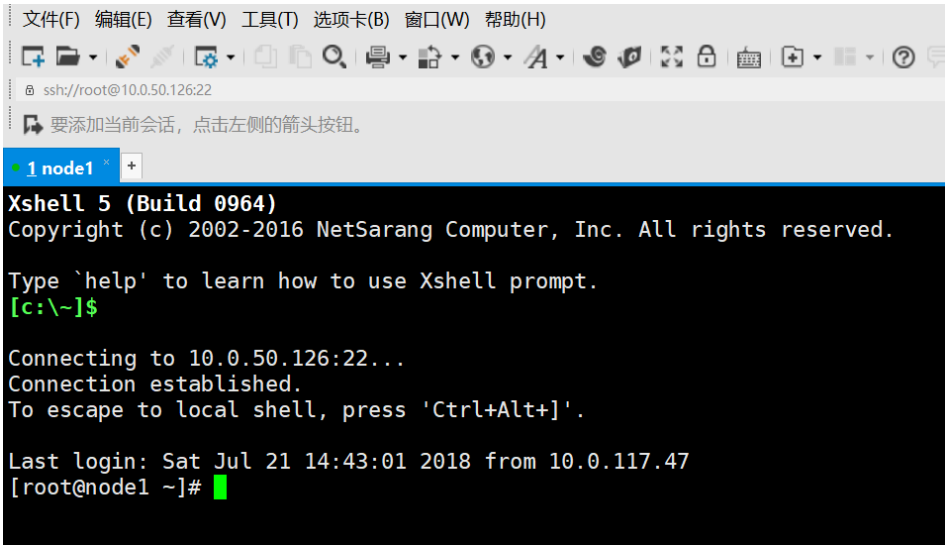




4.选择创建完成的会话，完成登陆。首次登陆时会弹出窗口，询问是否保存主机密钥，选择保存



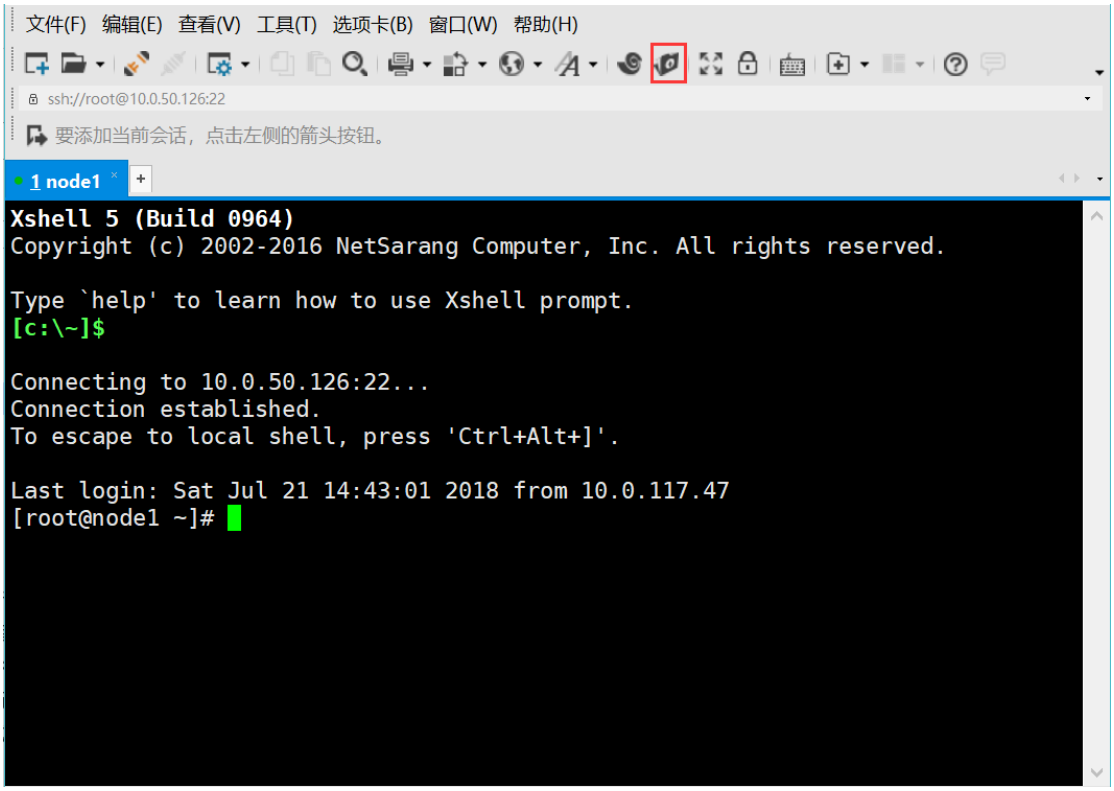
登陆完成，可以使用命令行界面。



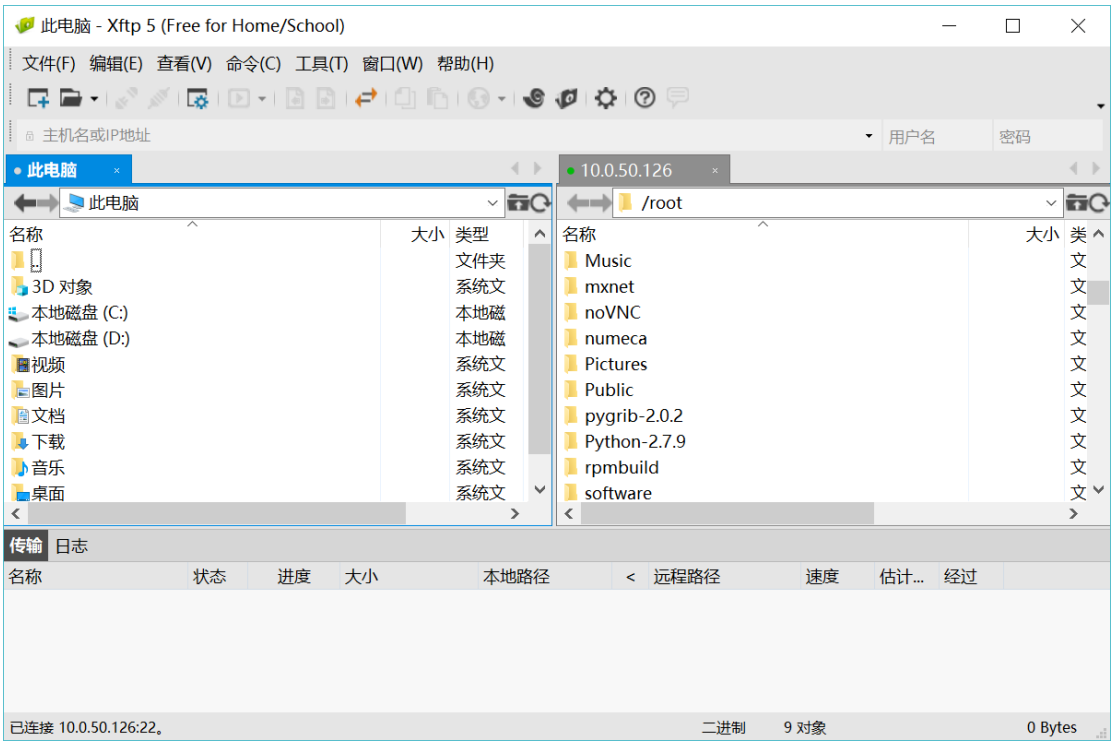
3.1.3 文件上传下载

Windows 用户可以用 SSH Secure Shell Client，winscp，Xftp 等软件实现文件的上传下载。

下面以 Xftp 为例，介绍文件上传和下载使用方法。Xftp 与 Xshell 进行了集成，使用 xshell 登陆到集群后，点击工具栏上文件夹的图标，可以打开 Xftp 文件传输的窗口。

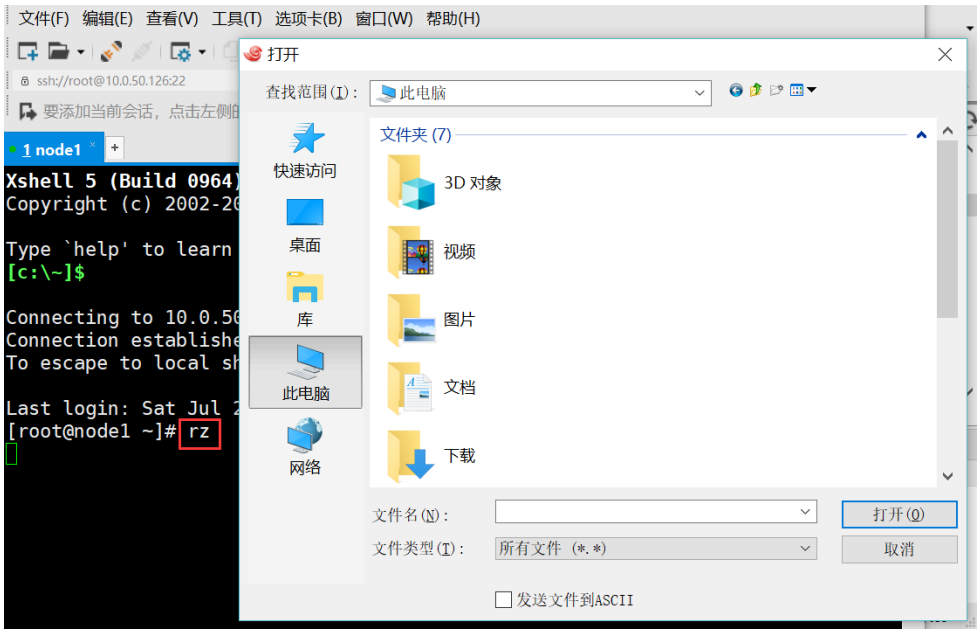


左侧为本机，右侧为高性能集群，可直接拖动进行文件上传和下载。

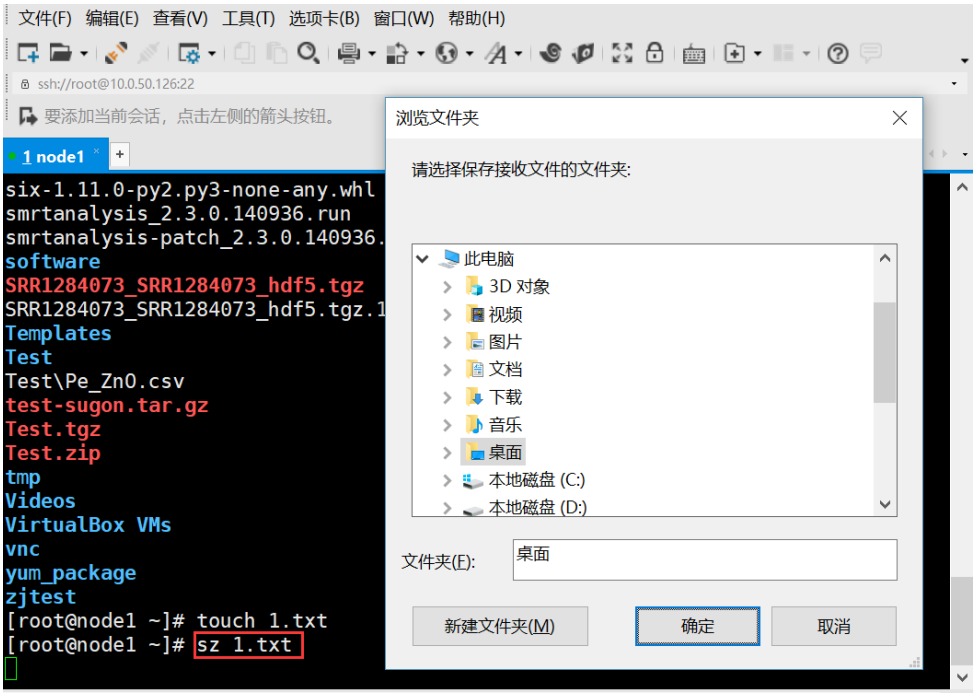


除上述工具外，在命令行中使用 `rz` 和 `sz` 命令也可以方便的进行小文件的上传和下载。

$ rz 上传文件



$ sz 下载文件



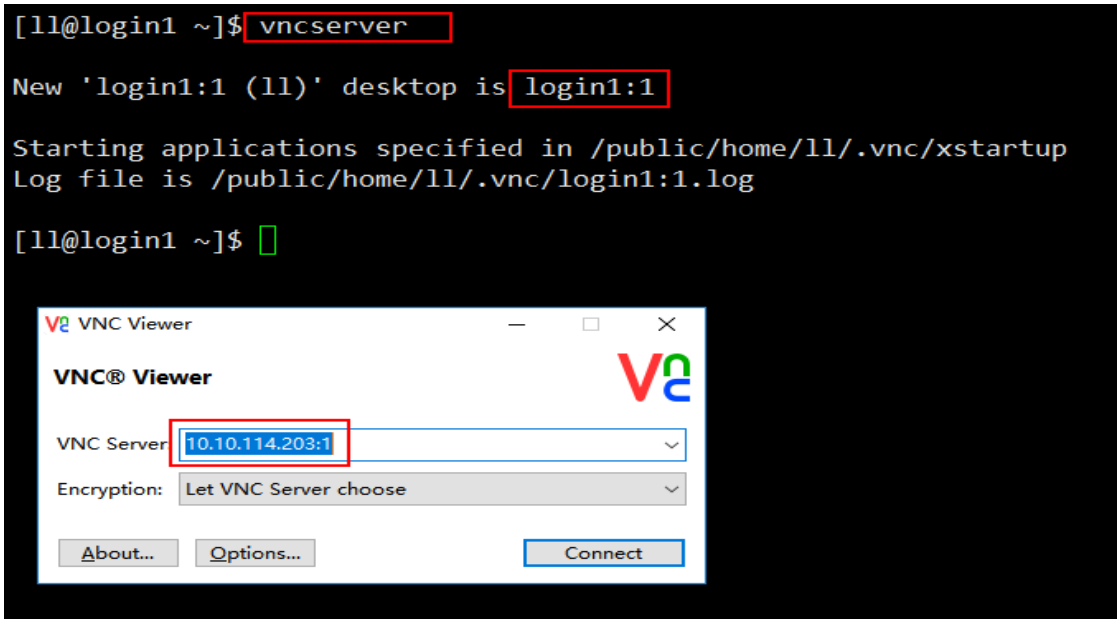
Linux 操作系统直接使用命令

```
scp  filename  test@ip:/home/test
```

3.1.4 图形界面登录

远程图形界面登录推荐采用 VNC 方式。第一次使用 VNC 登录前，需要参照

3.1.2 节以命令行终端方式登录到集群登录节点，执行 vncserver 命令，会提示用户输入 VNC 登录密码，输入后会得到一个 VNC 会话，一般是“主机名:VNC 会话号”格式，如“login1:1”。

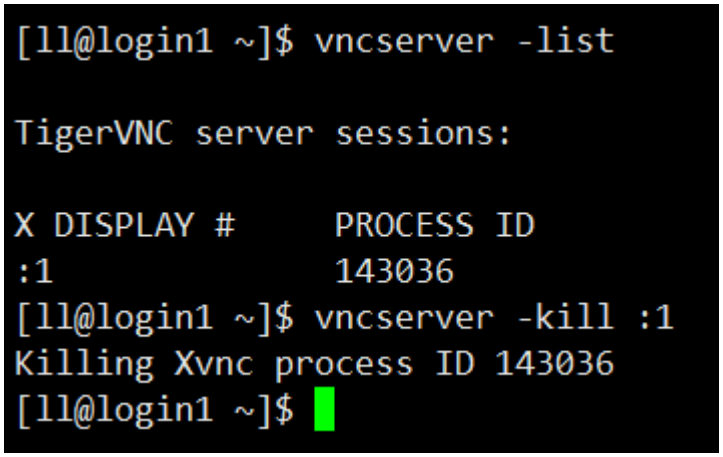


查看已经启动的 vnc 服务端，使用

```
vnserver -list
```

关闭 vnc 服务端使用

```
vncserver -kill :会话号 如 vncserver -kill :1
```



3.1.5 Gridview Web 登录

普通用户也可以通过 WEB 方式登入曙光 GridView 集群管理系统，运行配置和使用作业相关运算提交等操作。在 WEB 浏览器（推荐使用 FireFox 或 Chrome）

的地址栏中输入以下 URL 即可出现登录界面：http://10.10.114.203:6080

输入普通用户“用户名”、“密码”，即可以进入 gridview 作业管理 WEB 界面.

Gridview 详细使用方法请参考<<Gridview 用户手册>>。

3.2 编译、安装 OpenMPI 示例

以 OpenMPI 2.1.2 为例：

```
$ tar zxvf openmpi-2.1.2.tar.gz
$ cd openmpi-2.0.1
$ ./configure
--prefix=/public/software/mpi/openmpi/intel/2.1.2
--enable-mpirun-prefix-by-default -without-ucx CC=icc
CXX=icpc FC=ifort F77=ifort
$ make -j 8 && make install
```

设置环境变量脚本：

```
vim /public/software/profile.d/mpi_openmpi-intel-2.1.2.sh
```

```
#!/bin/bash
export MPI_HOME=/public/software/mpi/openmpi/intel/2.1.2
export PATH=${MPI_HOME}/bin:${PATH}
export LD_LIBRARY_PATH=${MPI_HOME}/lib:${LD_LIBRARY_PATH}
export MANPATH=${MPI_HOME}/share/man:${MANPATH}
```

☑ **Tips：**

1. OpenMPI 安装会自动检测编译节点本地可用的通信网络设备（如 Infiniband、Omni-Path 等），如需支持 InfiniBand 或 Omni-Path 等高速网络，请确保编译 MPI 前该节点已安装 Infiniband 或 OPA 驱动。

2. 执行 OpenMPI 安装目录$MPI_HOME/bin 下的 ompi_info 命令，可查询当前 OpenMPI 配置信息。

4 环境变量

集群约定用户默认家目录为：/public/home/“用户名”；

共享软件目录为：/public/software/“软件名”；

环境变量默认地址为：/public/software/profile.d/“配置文件”；

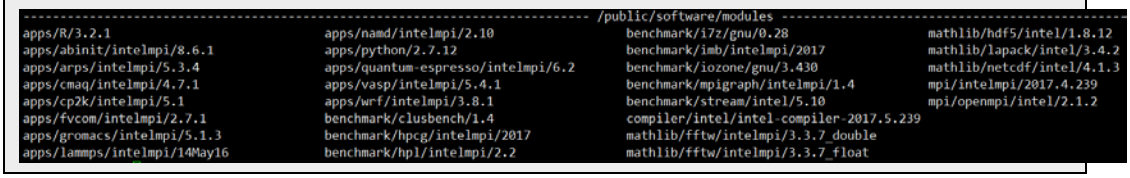
4.1 环境变量简介

“Environment module”(环境模块)是一组环境变量设置的集合。module可以被加载(load)、卸载(unload)、切换(switch)，这些操作会改变相应的环境变量设置，从而让用户方便地在不同环境间切换。 相比与将环境变量设置写入/etc/profile 或者~/.bashrc，Environment module 操作只影响当前用户的当前登录环境，不会因错误配置造成全局持续的破坏。 普通用户也可以自己编写 module，具有很好的定制性。

查看可用环境变量命令 module avail

```
module avail
```

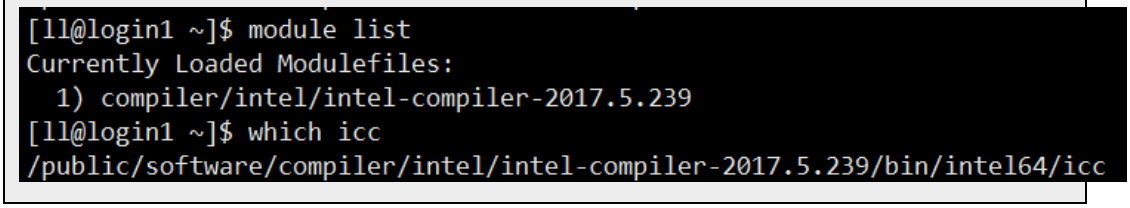
输出如下



示例加载 intel2017 编译器

```
module load
/public/software/modules/compiler/intel/intel-compiler-2017.5.239
```

查看加载 module list



其它用法

```
module -help
```

```
  Usage: module [ switches ] [ subcommand ] [subcommand-args ]

Switches:
        -H|--help               this usage info
        -V|--version            modules version & configuration options
        -f|--force              force active dependency resolution
        -t|--terse              terse    format avail and list format
        -l|--long               long     format avail and list format
        -h|--human              readable format avail and list format
        -v|--verbose            enable  verbose messages
        -s|--silent             disable verbose messages
        -c|--create             create caches for avail and apropos
        -i|--icase              case insensitive
        -u|--userlvl <lvl>      set user level to (nov[ice],exp[ert],adv[anced])
  Available SubCommands and Args:
        + add|load              modulefile [modulefile ...]
        + rm|unload             modulefile [modulefile ...]
        + switch|swap           [modulefile1] modulefile2
        + display|show          modulefile [modulefile ...]
        + avail                 [modulefile [modulefile ...]]
        + use [-a|--append]     dir [dir ...]
        + unuse                 dir [dir ...]
        + update
        + refresh
        + purge
        + list
        + clear
        + help                  [modulefile [modulefile ...]]
        + whatis                [modulefile [modulefile ...]]
        + apropos|keyword       string
        + initadd               modulefile [modulefile ...]
        + initprepend           modulefile [modulefile ...]
        + initrm                modulefile [modulefile ...]
        + initswitch            modulefile1 modulefile2
        + initlist
        + initclear
```

4.2 SOURCE 加载法

没有条件使用 module 模块时可使用 source 加载方法。

示例加载 cuda10.0

```
source /public/software/profile.d/cuda-10.0.sh
```

确认 Cuda 加载

```

```

```
[ll@login1 ~]$ source /public/software/profile.d/cuda-10.0.sh
[ll@login1 ~]$
[ll@login1 ~]$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Sat_Aug_25_21:08:01_CDT_2018
Cuda compilation tools, release 10.0, V10.0.130
```

4.3 默认变量加载

用户自定义环境变量可通过编写“~/.bashrc”文件定义，每次登陆自动加载。

示例格式如下

```
vim ~/.bashrc
```

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions


#绝对路径定义程序可执行程序位置，可多个写在一起中间以“：”分开，也可单独写。
#LINUX默认读取是从前往后读取的,推荐"$PATH"写在最后，而且不能不写否则系统下默认命令无法使用。
export PATH=/public/software/anaconda3/bin:/public/software/apps/R/3.2.1/bin:$PATH
export LD_LIBRARY_PATH=/public/software/anaconda3/lib:$LD_LIBRARY_PATH


#直接source方式也可生效
source /public/software/profile.d/cuda-10.0.sh
~
```

4.4 用户密码修改

系统默认使用 NIS 服务管理用户信息，修改用户密码执行“yppasswd”，输入旧密码后输入新密码并确认。密码设置尽量复杂，包含大小写字母、数字、特殊字符、密码长度大于 8 位。

yppasswd

```
[ll@login1 ~]$ yppasswd
Changing NIS account information for ll on iadmin1.
Please enter old password:
Changing NIS password for ll on iadmin1.
Please enter new password:
Please retype new password:
```

5 命令行运行程序

5.1 运行串行程序

方法一

```
cd /home/your_account/your_workdir
./your_code
```

方法二

```
cd $HOME
vi .bashrc
```

添加

```
export PATH=/home/your_account/your_workdir:$PATH
```

执行命令

```
your_code
```

5.2 使用 openmpi 运行并行程序

5.2.1 编译 MPI 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/public/software//mpi/openmpi/intel/2.1.2/include
-pthread -Wl,-rpath
-Wl,/public/software//mpi/openmpi/intel/2.1.2/lib
-Wl,--enable-new-dtags
-L/public/software//mpi/openmpi/intel/2.1.2/lib -lmpi
```

编译程序示例：

```
$ source
/public/software/profile.d/mpi_openmpi-intel-2.1.2.sh
$ mpicc -o hello hello.c
$ mpif90 -o hello hello.f90
```

5.2.2 运行 MPI 程序

OpenMPI 使用自带的 OpenRTE 进程管理器，启动命令为 mpirun，基本格式如下：

```
$ mpirun -np N -hostfile <filename> <program>
```

- -np N：运行 N 个进程
- -hostfile：指定计算节点，文件格式如下：

```
node1 slots=8
node2 slots=8
```

slots=8 代表可在该节点上执行 8 个进程，也可将 node1 和 node2 分别写 8 行。hostfile 中的主机名可使用以太网别名（nodeXXX）或 ib 网别名（inodeXXX）或者 IP 地址，但不要将以太网别名和 ib 网别名混用。

Openmpi 运行时会自动检测集群的网络配置，并自动选择 Infiniband 或 OPA 等高速网络进行通信，并不会依赖于 hostfile 中的主机名来选择不同网络。另外，mpirun 后支持多种参数来选择支持的网络，

```
mpirun -mca btl self,sm,openib      //用于 Inifiniband
mpirun -mca mtl psm2                //用于 Omni-Path（OPA）
```

5.3 使用 mpich 运行并行程序

5.3.1 编译 MPI 程序

MPICH2 编译 C、C++、Fortran77 和 Fortran90 的编译器分别为 mpicc,mpicxx,mpif70 和 mpif90。

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

编译程序示例：

```
source /public/software/profile.d/mpi_mpich-intel-3.2.sh
mpicc -o hello hello.c
mpif90 -o hello hello.f90
```

5.3.2 运行 MPI 程序

MPICH2 默认使用 hydra 进程管理器，使用 mpirun 启动 MPI 进程，命令格式如下：

```
$ mpirun -f hostfile -n 6 ./program
```

hostfile 文件格式如下：

```
node1:4      #<node name>:<number of cores>
node2:4
```

5.4 使用 mvapich2 运行并行程序

5.4.1 编译 MPI 程序

```
$ source
/public/software/profile.d/mpi_mvapich2-intel-2.3b.sh
$ mpicc -o hello hello.c
$ mpif90 -o hello hello.f90
```

MVAPICH2 编译 C、C++、Fortran77 和 Fortran90 的编译器分别为 mpicc,mpicxx,mpif70 和 mpif90。

5.4.2 运行 MPI 程序

MVAPICH2 默认使用 Hydra 进程管理器，启动进程直接使用 mpirun 命令

```
mpirun -f hosts -n 4 [-env ENV value]./program
```

- -f hosts：格式同 MPICH2 相同，<node name>:<proc num>
- -n 4：指定进程数
- -env ENV=value：设置运行环境变量

5.5 使用 intelmpi 运行并行程序

5.5.1 编译 MPI 程序

IntelMPI 提供了非常完整的 MPI 编译器，既支持 GNU 编译器，也支持 Intel 编译器，具体如下表所示：

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 -ldl
-I/public/software/intel/impi/4.1.0.024/intel64/include
-L/public/software/intel/impi/4.1.0.024/intel64/lib -Xlinker
--enable-new-dtags -Xlinker -rpath -Xlinker
/public/software/intel/impi/4.1.0.024/intel64/lib -Xlinker
-rpath -Xlinker /opt/intel/mpi-rt/4.1 -lmpi -lmpigf -lmpigi
-lrt -lpthread
```

编译示例：

```
$ mpiicc -o hello hello.c
```

5.5.2 运行 MPI 程序

IntelMPI 也使用 Hydra 进程管理器，进程启动使用 mpirun 命令

```
$ mpirun -np 16 -machinefile hosts -env I_MPI_DEVICE
rdma ./program
```

- -machinefile hosts，指定节点列表
- -n 4：指定进程数

- -env ENV=value：设置运行环境变量

另外，IntelMPI 也支持参数来选择高速网络，主要参数如下

```
// Infiniband 高速网络下推荐使用
-DAPL
-genv I_MPI_FABRICS=shm:dapl
// 使用 PSM2 底层通信协议，OPA 网络下推荐使用
-PSM2
-genv I_MPI_FABRICS=shm:tmi
// 使用 libfabric 提供的底层通信协议，OPA 网络下推荐使用（建议较新的驱动版本下使用）
-OFI
-genv I_MPI_FABRICS=shm:ofi
```

6 使用作业调度（SLURM 命令行模式）运行程序

6.1 SLURM 的基本命令

在超算系统中，用户使用 sbatch 命令提交用户程序。

```
sbatch xxx.job
```

用户运行程序的命令及 slurm 环境变量设置组成作业脚本，提交格式如下（注释,以“#”开头，sbatch 指令以“#SBATCH”开头并非注释）：

示例脚本“sleepAll.job”

```
#!/bin/bash
#SBATCH --job-name=SleepAll
#SBATCH --nodes=110
#SBATCH --ntasks=110
#SBATCH --partition=batch
##SBATCH --workdir=`$home`
#SBATCH --output=%j.out
#SBATCH --error=%j.err
#
###module load /etc/modulefiles/mpi/openmpi-x86_64
###module load 和 source 都是加载环境变量的方式
###source /public/software/profile.d/python3.sh
####source /public/software/profile.d/cuda-10.0.sh
#
################################################################
echo begin time is `date` | tee -a $SLURM_JOBID.out
sleep 600
echo end time is `date` | tee -a $SLURM_JOBID.out %j.out
```

算例规模的大小合理估算所需的 deadline 和 mem，把其写进作业脚本里，这样有助于更快、更有效地分配资源。更详细参数解释如下：

```
#!/bin/bash
```

```
###SBATCH 参数以#开头，非注释！
#SBATCH --job-name=Myjob
###作业名称

#SBATCH --nodes=1
###使用节点数量

#SBATCH --ntasks=8
###总的进程数(CPU 核数)

#SBATCH --ntasks-per-node=8
###每个节点的进程数，1 个节点此项无需指定

#SBATCH --gres=gpu:4
###每个节点使用的 GPU 数量，CPU 作业此项此项无需指定

##SBATCH --mem=10G
###申请预留内存大小，可选项

#SBATCH --partition=gpu
###使用的分区，目前有 3 个分区

##SBATCH --workdir=`$home`
###作业的工作目录，输出 log 在此路径,要求使用自己的家目录

#SBATCH --output=%j.out
###作业错误输出文件,%j 代表作业 ID

#SBATCH --error=%j.err
```

```
###作业正确输出文件

##SBATCH --begin=12:00
###作业开始执行时间，默认立即执行，可选项

##SBATCH --deadline=23:00
###作业强制终止时间，可选项

###module load
#/public/software/modules/compiler/intel/intel-compiler-201
7.5.239
#source source /public/software/profile.d/cuda-10.0.sh
###加载环境变量，根据实际情况可能需要加载多个如 intel 编译器，
intelmpi

echo -e "JOB NAME:$SLURM_JOB_NAME,Job
ID:$SLURM_JOBID,Allocate Nodes:$SLURM_JOB_NODELIST"
###显示作业名称，作业 ID，使用节点

#mpirun  caffe train -solver ./solver.prototxt -gpu all
-weights ./model1.bin,./model2.bin
###训练

#mpirun  caffe train -solver ./solver.prototxt -gpu all
-snapshot ./_iter_3000.solverstate
```

6.2 查询队列信息

```
sinfo
```

```
[ll@login1 test]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
low*         up   infinite    110   idle node[1-110]
batch        up   infinite    110   idle node[1-110]
gpu          up   infinite      2   idle gpu[1-2]
```

6.3 查询节点详细信息

scontrol show nodes

```
scontrol show nodes|more
```

如下输出

```
NodeName=gpu1 Arch=x86_64 CoresPerSocket=14
   CPUAlloc=0 CPUTot=56 CPULoad=2.27
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=gpu:4
   NodeAddr=gpu1 NodeHostName=gpu1 Version=18.08
   OS=Linux 3.10.0-693.el7.x86_64 #1 SMP Tue Aug 22 21:09:27 UTC 2017
   RealMemory=191908 AllocMem=0 FreeMem=187019 Sockets=2 Boards=1
   MemSpecLimit=10240
   State=IDLE ThreadsPerCore=2 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=gpu
   BootTime=2019-05-13T03:21:43 SlurmdStartTime=2019-05-18T11:07:57
   CfgTRES=cpu=56,mem=191908M,billing=56,gres/gpu=4
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s


NodeName=gpu2 Arch=x86_64 CoresPerSocket=14
   CPUAlloc=0 CPUTot=56 CPULoad=2.13
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=gpu:4
   NodeAddr=gpu2 NodeHostName=gpu2 Version=18.08
   OS=Linux 3.10.0-693.el7.x86_64 #1 SMP Tue Aug 22 21:09:27 UTC 2017
   RealMemory=191908 AllocMem=0 FreeMem=187452 Sockets=2 Boards=1
   MemSpecLimit=10240
   State=IDLE ThreadsPerCore=2 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=gpu
   BootTime=2019-05-13T03:21:44 SlurmdStartTime=2019-05-18T11:07:57
   CfgTRES=cpu=56,mem=191908M,billing=56,gres/gpu=4
   AllocTRES=
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s


NodeName=node1 Arch=x86_64 CoresPerSocket=14
   CPUAlloc=0 CPUTot=28 CPULoad=0.05
   AvailableFeatures=(null)
--More--
```

```
.....
```

节点状态

idle ： 资源空闲；

mix ： 有部分作业运行；

down ： 离线或 slurmd 服务关闭；

6.4 查询作业运行状态

查询当前作业命令

```
squeue
```

如下输出：

```
[ll@login1 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               147     batch SleepAll       ll  R       1:45    110 node[1-110]
[ll@login1 ~]$
```

当用户认为任务异常时，可使用这些 工具来追踪任务的信息。对于正在运行或排队的任务，可以使用

```
scontrol show job JOBID
```

如下输出：

```
[ll@login1 ~]$ scontrol show job 147
JobId=147 JobName=SleepAll
   UserId=ll(1006) GroupId=ll(1006) MCS_label=N/A
   Priority=2000 Nice=0 Account=ll QOS=ll_qos
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:07:57 TimeLimit=1-00:00:00 TimeMin=N/A
   SubmitTime=2019-05-20T13:09:46 EligibleTime=2019-05-20T13:09:46
   AccrueTime=2019-05-20T13:09:46
   StartTime=2019-05-20T13:09:46 EndTime=2019-05-21T13:09:46 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2019-05-20T13:09:46
   Partition=batch AllocNode:Sid=login1:151032
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=node[1-110]
   BatchHost=node1
   NumNodes=110 NumCPUs=110 NumTasks=110 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=110,mem=832590M,node=110,billing=110
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=7569M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/public/home/ll/test.job
   WorkDir=/public/home/ll
   StdErr=/public/home/ll/147.err
   StdIn=/dev/null
   StdOut=/public/home/ll/147.out
   Power=
```

作业完成后，需要用 `sacct` 命令来查看历史作业。默认情况下，用户仅能查看属于自己的历史作业。直接使用 `sacct` 命令会输出从当天 `00:00:00` 起到现在的全部作业

```
sacct
```

如下输出：

```

```

```
[ll@login1 ~]$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
136            SleepAll      batch         ll       2640 CANCELLED+      0:0
138          STDIN_052+        low         ll        999     FAILED      1:0
139          STDIN_052+        low         ll        110  COMPLETED      0:0
139.batch         batch                    ll          1  COMPLETED      0:0
140          STDIN_052+        low         ll          1  COMPLETED      0:0
140.batch         batch                    ll          1  COMPLETED      0:0
141            SleepAll      batch         ll        110  COMPLETED      0:0
141.batch         batch                    ll          1  COMPLETED      0:0
142            SleepAll      batch         ll        110  COMPLETED      0:0
142.batch         batch                    ll          1  COMPLETED      0:0
143            SleepAll      batch         ll        110  COMPLETED      0:0
143.batch         batch                    ll          1  COMPLETED      0:0
144            SleepAll      batch         ll        110  COMPLETED      0:0
144.batch         batch                    ll          1  COMPLETED      0:0
145            SleepAll      batch         ll        110  COMPLETED      0:0
145.batch         batch                    ll          1  COMPLETED      0:0
146            SleepAll      batch         ll        110  COMPLETED      0:0
146.batch         batch                    ll          1  COMPLETED      0:0
147            SleepAll      batch         ll        110  COMPLETED      0:0
147.batch         batch                    ll          1  COMPLETED      0:0
```

如果使用

```
sacct -S MMDD
```

则会输出从 MM 月 DD 日起的所有历史作业。

默认情况会输出作业 ID，作业名，分区，账户，分配的 CPU，任务结束状态，返回码。当然我们还可以使用 --format 参数来指定到底要输出那些指标。

```
```

```
[ll@login1 ~]$ sacct --format=jobid,user,alloccpu,allocgres,state%15,exitcode
       JobID      User  AllocCPUS    AllocGRES           State ExitCode
------------ --------- ---------- ------------ --------------- --------
136                  ll       2640              CANCELLED by 0      0:0
138                  ll        999                      FAILED      1:0
139                  ll        110                   COMPLETED      0:0
139.batch                        1                   COMPLETED      0:0
140                  ll          1                   COMPLETED      0:0
140.batch                        1                   COMPLETED      0:0
141                  ll        110                   COMPLETED      0:0
141.batch                        1                   COMPLETED      0:0
142                  ll        110                   COMPLETED      0:0
142.batch                        1                   COMPLETED      0:0
143                  ll        110                   COMPLETED      0:0
143.batch                        1                   COMPLETED      0:0
144                  ll        110                   COMPLETED      0:0
144.batch                        1                   COMPLETED      0:0
145                  ll        110                   COMPLETED      0:0
145.batch                        1                   COMPLETED      0:0
146                  ll        110                   COMPLETED      0:0
146.batch                        1                   COMPLETED      0:0
147                  ll        110                   COMPLETED      0:0
147.batch                        1                   COMPLETED      0:0
```

6.5 更新任务

有时我们很早就提交了任务，但是在任务开始前却发现作业的属性写错了（例如提交错 了分区，忘记申请 GPU 个数），取消了重新排队似乎很不划算。如果作业恰好没在运行 我们是可以通过 scontrol 命令来修改作业的属性。

使用以下命令可以修改 JOBID 任务的部分属性

```
scontrol update jobid=JOBID ...
```

例如我要更改当前的分区到 gpu，并且申请 1 块卡，可以输入

```
scontrol update jobid=138 partition=gpu gres=gpu:1
```

变更的时候仍然不能超过系统规定的上限。变更成功后，作业的优先级可能需要重新来计算。

6.6 删除作业

作业删除命令：scancel JOBID

```
scncel 148
```

注意事项

1、非 root 用户只能查看、删除自己提交的作业；

2、强制删除作业，当某些作业由于节点死机无法删除时，可由 root 用户来强制删除作业。

6.7 作业未运行原因查看

用户提交作业后，是否运行取决于用户申请的资源情况和当前系统的情况。建议使用 squeue 命令来查看所有已经提交和正在运行的作业。其中 NODELIST(REASON)一栏包含非常有用的信息，在作业未运行时，它会显示未运行的原因；当作业在运行时，它会显示作业是在哪个节点运行的。

下面的表格整理了常见作业未运行的原因，用户可根据此来调整自己的脚本。其中的加粗部分表示异常原因，用户需要修改 SLURM 脚本或联系管理员。

原因代码	详细说明
BeginTime	未到用户所指定的任务开始时间
Dependency	该作业所依赖的作业尚未完成
InvalidAccount	**用户的 SLURM 账号无效**
InvalidQOS	**用户指定的 QoS 无效**
ParitionTimeLimit	**用户申请的时间超过该分区时间上限**
QOSMaxCpuPerUserLimit	超过当前 QoS 用户最大 CPU 限制
QOSMaxGRESPerUser	超过当前 QoS 用户最大 GRES(GPU) 限制
Priority	存在一个或多个更高优先级的任务，该任务需要等待
ReqNodeNotAvail	所申请的部分节点不可用
Resources	暂无闲置资源，该任务需等待其他任务完成

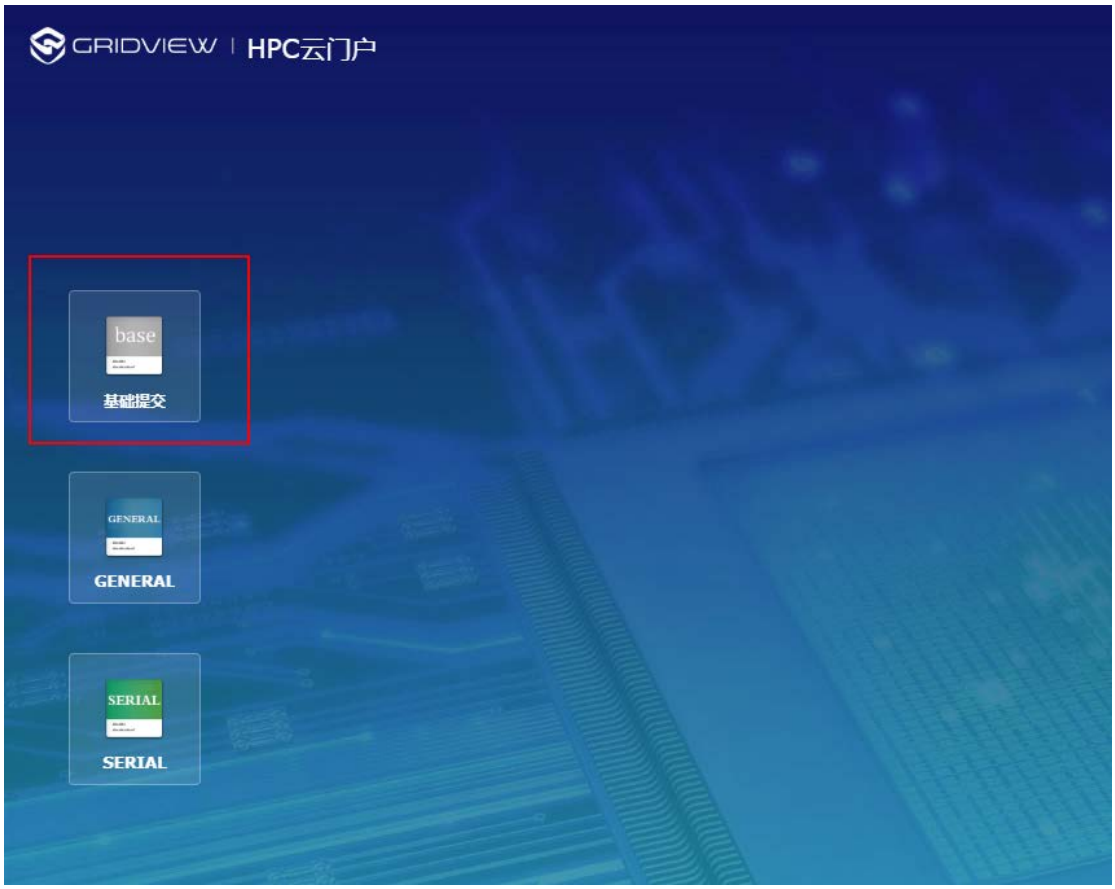
7 使用 Gridview 作业调度

使用 Linux 的普通用户登陆 Gridview 云门户，如使用用户 ll 登陆，登陆后可以看到如下界面，可以完成作业提交、作业查看、资源申请、WebShell 等操作。更多详细的使用方法请参考 gridview 使用手册。

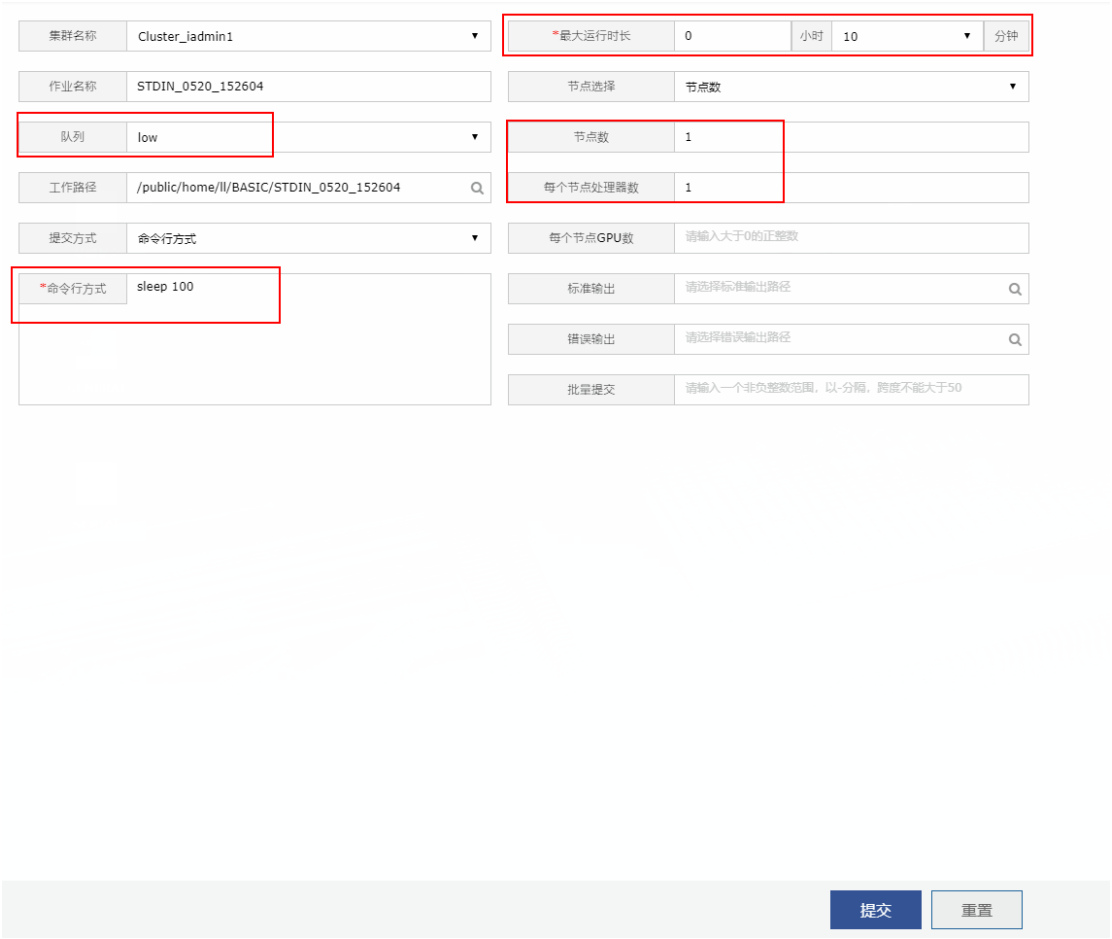
Gridview 中可集成应用的 web 作业提交 Portal，可以通过 Portal 更加便捷的使用高性能集群。

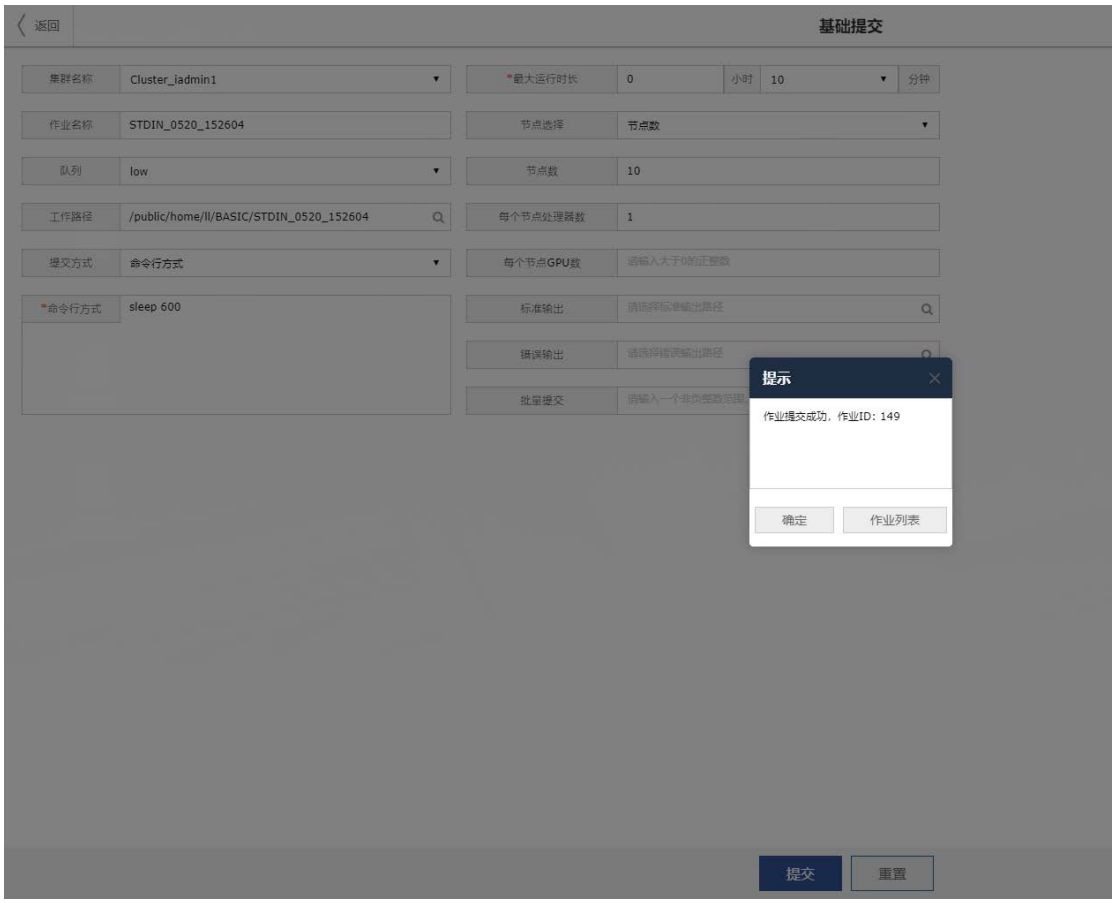
7.1 作业提交

点击基础提交，可以实现作业提交。



在作业提交界面中可以设置作业名称、提交到队列、核心数、墙钟时间、运行脚本等信息。

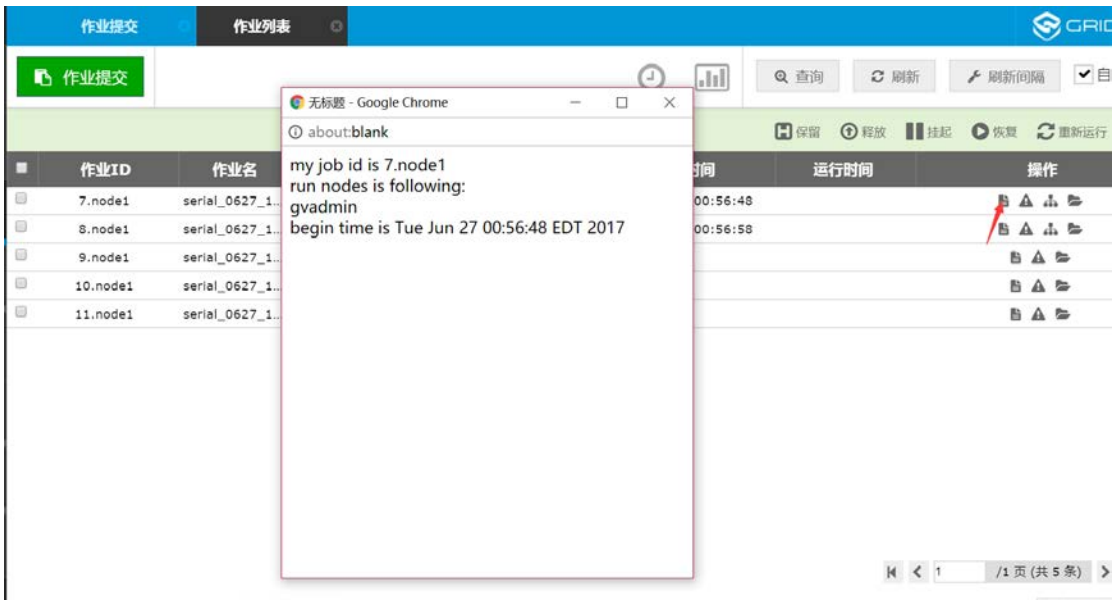

集群名称
Cluster_iadmin1
*最大运行时长
0
小时
10
分钟
作业名称
STDIN_0520_152604
节点选择
节点数
队列
low
节点数
1
工作路径
/public/home/ll/BASIC/STDIN_0520_152604
每个节点处理器数
1
提交方式
命令行方式
每个节点GPU数
请输入大于0的正整数
*命令行方式
sleep 100
标准输出
请选择标准输出路径
错误输出
请选择错误输出路径
批量提交
请输入一个非负整数范围，以-分隔，跨度不能大于50
提交
重置



7.2 查看作业状态

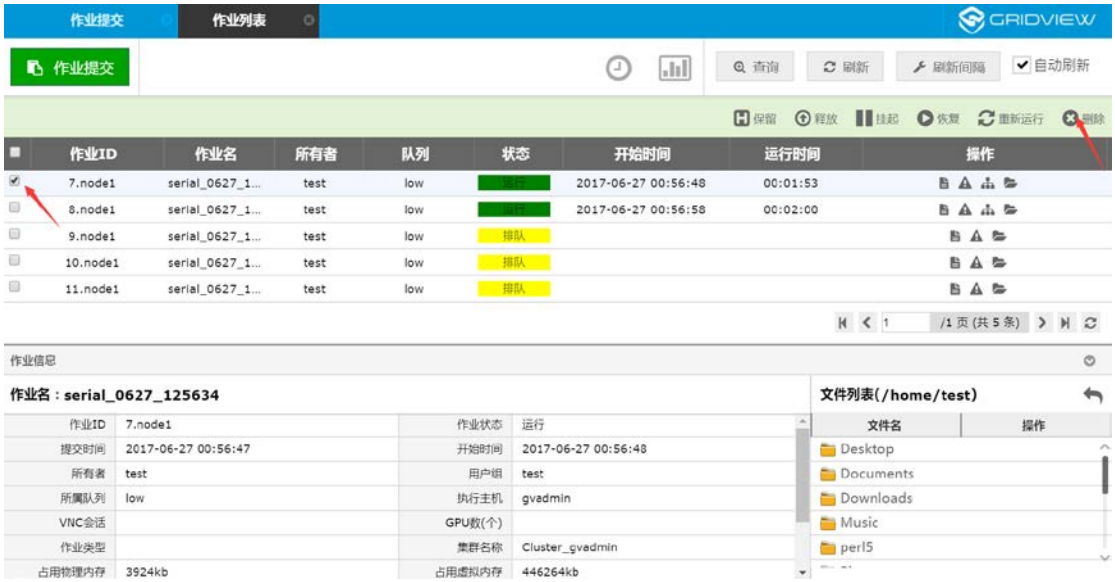
点击作业列表，可以看到集群内正在运行的作业。

查看作业的标准输出和标准错误输出



7.3 删除作业

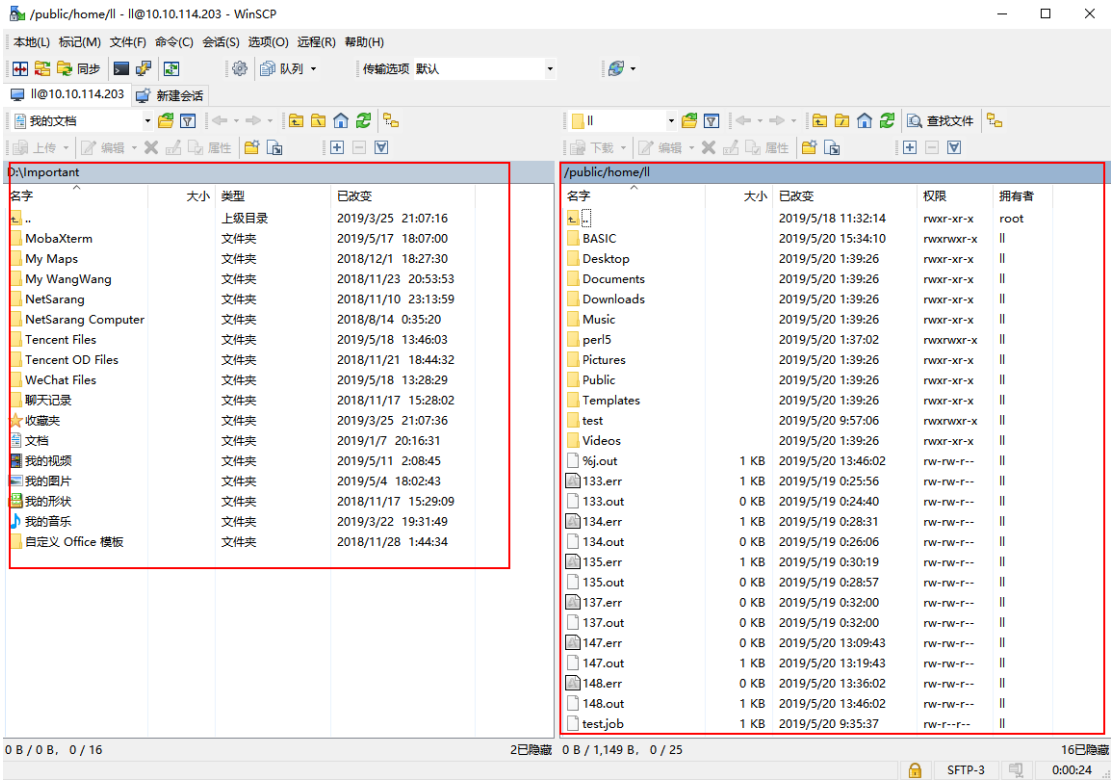
在作业列表中还可以删除作业



7.4 Winscp 文件传输

Web 界面作业提交也可与 winscp 文件传输结合使用，编辑好作业脚本上传到集群上进行调用。

以下左面为本地目录，右侧为服务器目录



8 SLURM 与 PBS 关系参考

考虑到已用户已熟悉 PBS 作业调度系统，这里整理了 PBS 和 SLURM 使用的对应关系。

功能	PBS	SLURM
任务名称	#PBS -N name	#SBATCH -J name
指定队列	#PBS -q cpu	#SBATCH -p cpu
指定 QoS	#PBS --qos=debug，需调度器支持	#SBATCH --qos=debug
最长运行时间	#PBS -l walltime=5:00	#SBATCH -t 5:00

指定节点数量	#PBS -l nodes=1	#SBATCH -N 1
指定 CPU 核心	#PBS -l ppn=4	#SBATCH --cpus-per-task=4
指定 GPU 卡	不支持	#SBATCH --gres=gpu:1
作业数组	#PBS -t 0-2	#SBATCH -a 0-2
输出文件	#PBS -o test.out	#SBATCH -o test.out
提交任务脚本	qsub run.pbs	sbatch run.slurm
查看任务状态	qstat	squeue
取消任务	qdel 1234	scancel 1234
交互式任务	qsub -I，自动切换	salloc，手动切换
指定特定节点	qsub -l nodes=comput1	#SBATCH --nodelist=comput1

9 联系方式

日常业务联系方式：

- 办公电话：021-55272153
- 地　　址：超算中心运维
- 公共邮箱：hpc@usst.edu.cn
- 中心主页：http://hpc.usst.edu.cn