**高性能集群用户使用手册**

**--------成都中医药大学**

**2021年3月**

目录

[1 集群概述 3](#_Toc67307983)

[1.1 集群登录地址 3](#_Toc67307984)

[1.2 集群计算资源 3](#_Toc67307985)

[1.3 集群队列划分 3](#_Toc67307986)

[1.4 系统及软件 3](#_Toc67307987)

[2 集群作业管理系统 4](#_Toc67307988)

[2.1 应用软件环境管理 4](#_Toc67307989)

[2.2 PBS查看节点信息及状态 5](#_Toc67307990)

[2.3 PBS命令介绍 6](#_Toc67307991)

[2.4 PBS环境变量 8](#_Toc67307992)

[2.5 PBS脚本文件 9](#_Toc67307993)

[2.6 PBS环境下运行示例 10](#_Toc67307994)

[3 命令行作业PBS脚本 12](#_Toc67307995)

[3.1 Intel-mpi 12](#_Toc67307996)

[3.2 Ansys 13](#_Toc67307997)

[3.3 Fluent 13](#_Toc67307998)

[3.4 CFX 14](#_Toc67307999)

[3.5 Vasp 14](#_Toc67308000)

[3.6 Lammps 15](#_Toc67308001)

[3.7 OpenFOAM 15](#_Toc67308002)

[3.8 Gaussian-g09 16](#_Toc67308003)

[3.9 Gaussian-g16 16](#_Toc67308004)

[3.10 Anaconda 17](#_Toc67308005)

[4 Web浏览器端作业管理 17](#_Toc67308006)

[4.1 集群监控 17](#_Toc67308007)

[4.2 作业提交 18](#_Toc67308008)

[4.3 文件下载 19](#_Toc67308009)

[4.4 文件管理 20](#_Toc67308010)

# 集群概述

高性能集群(High performance cluster,HPC),它是利用一个集群中的多台机器共同完成同一件任务,使得完成任务的速度和可靠性都远远高于单机运行的效果。

## 集群登录地址

当前集群开放ssh登录及web端登录

* **ssh登录：**

IP1：10.200.143.253 端口：2223

* **Web控制台登录**

https://10.200.143.253:2225

## 集群计算资源

集群计算资源主要包括，cpu计算节点64台，胖计算节点2台，gpu计算节点3台，主要配置信息如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 节点类型 | 节点数量 | 节点名称 | 内存大小（单个节点） | CPU核心数（单个节点） |
| 1 | 计算节点 | 22 | cu01-cu22 | 192G | 48 |
| 2 | 胖节点 | 1 | fat01 | 768G | 80 |
| 4 | 其他信息 | cpu计算节点cpu型号：Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz  smp胖节点cpu型号：Intel(R) Xeon(R) Gold 6248 CPU @ 2.5GHz | | | |

## 集群队列划分

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **队列名称** | **包含资源** | **说明** |
| **1** | batch | cu01-cu22，fat01 | 所有计算节点 |

## 系统及软件

集群节点操作系统版本centos7.5

集群中全局共享存储/public和/home

其中/public目录提供管理员用户统一安装软件，普通用户无写入权限，只可进行读取。

其中/home目录为用户家目录，用户登录后即在该目录下

在集群中软件必须均按照到共享目录下，因此不可使用yum安装软件，普通用户安装软件源码编译安装到自己用户目录下即可

在系统预安装部分基础编译环境及应用软件如下表所示

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **软件名称** | **软件版本** | **安装目录** |
| **1** | anaconda | 3 | /public/software/anaconda3 |
| **2** | intel-mpi | 2018 | /public/software/intel2018 |
| **3** | intel-mpi | 2020 | /public/software/intel2020 |
| **4** | gcc | 9.2.0 | /public/software/gcc/gcc-9.2.0/ |

# 集群作业管理系统

集群作业调度管理系统，采用pbs进行资源调度及任务下发，基础应用软件环境使用module进行管理。

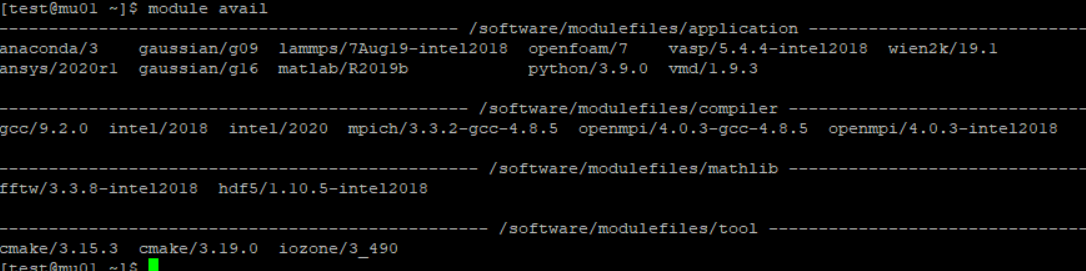
## 应用软件环境管理

通常情况下，我们在linux上安装软件后，需要使用export来添加PATH和LD\_LIBRARY\_PATH路径来使软件生效。

在高性能集群中，为了方便各类软件的环境变了管理，使用module来统一管理，module的使用非常的简单，如下面介绍。

* **查看所有可用软件**

module avail



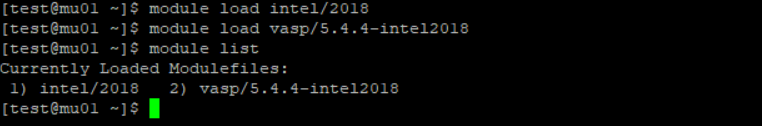
* **加载和卸载软件**

module load/unload

module purge 清除所有加载软件

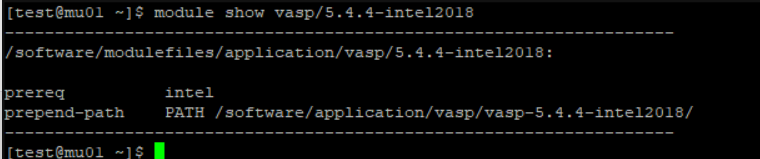
* **列出已经加载软件**

module list



* **查看环境配置，即modulefile文件**

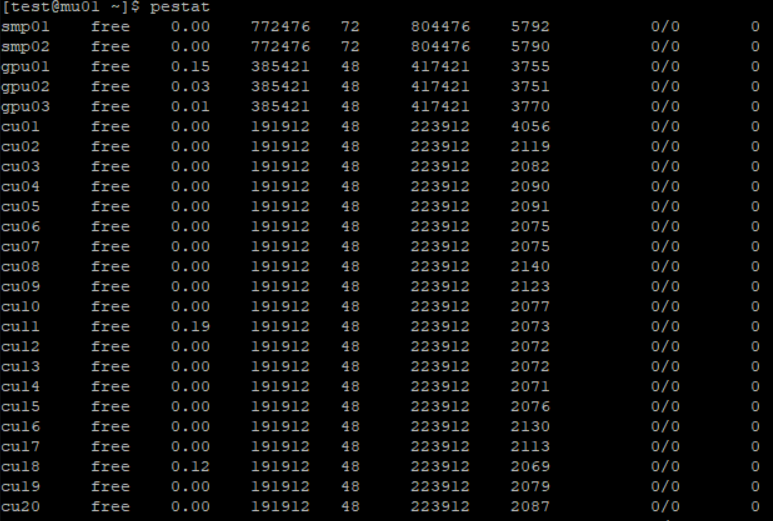
module show



以上module的相关操作用户可加入.bashrc文件中，实现默认配置，而无需每次登录都执行

## PBS查看节点信息及状态

**命令格式**：pestat或pbsnodes



## PBS命令介绍

PBS提供4条命令用于作业管理。

1. **qsub —— 用于提交作业脚本**

**命令格式：**

qsub [script]

1. **qstat —— 用于查询作业状态信息**

**命令格式：**

qstat [-f][-a][-i][-n][-s][-R][-Q][-q][-B][-u]

**参数说明：**

-f jobid 列出挃定作业的信息

-a 列出系统所有作业

-i 列出不在运行的作业

-n 列出分配给此作业的结点

-s 列出队列管理员不scheduler所提供的建议

-R 列出磁盘预留信息

-Q 操作符是destination id，挃明请求的是队列状态

-q 列出队列状态，并以alternative形式显示

-au userid 列出挃定用户的所有作业

-B 列出PBS Server信息

-r 列出所有正在运行的作业

-Qf queue 列出挃定队列的信息

-u 若操作符为作业号，则列出其状态。若操作符为destination id， 则列出运行在其上的属于user\_list中用户的作业状态。

**常用命令示例：**

1. **查看空闲节点信息**（提交作业前需要查看一下各队列节点空闲状况）

[test@mu01 ~]$ pbsnodes -l free

cu03 free

cu04 free

cu05 free

cu06 free

cu07 free

cu08 free

1. **查看指定作业信息**（提交作业后查看作业状态信息）

[test@mu01 ~]$ qstat -f 24951

Job Id: 24951.mu01

Job\_Name = myjob

Job\_Owner = test@mu01

resources\_used.cput = 00:08:42

resources\_used.energy\_used = 0

resources\_used.mem = 106772kb

resources\_used.vmem = 769292kb

resources\_used.walltime = 00:08:50

job\_state = R

queue = batch01

server = mu01

Checkpoint = u

ctime = Mon Sep 7 15:04:11 2020

……

……

1. **查看所有队列作业状态**

[test@mu01 ~]$ qstat -q

server: mu01

Queue Memory CPU Time Walltime Node Run Que Lm State

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

Batch01 -- -- -- 7 16 0 -- E R

Batch02 -- -- -- 12 20 0 -- E R

Batch03 -- -- -- -- 2 0 -- E R

Batch04 -- -- -- -- 0 0 -- E R

Batch05 -- -- -- 14 11 3 -- E R

Batch06 -- -- -- -- 0 0 -- E R

----- -----

49 3

1. **qdel —— 用于删除已提交的作业**

**命令格式：**

qdel [-W 间隔时间] 作业号 命令行参数：

qdel -p 强制清除某个作业号，一般不建议使用

例：# qdel -W 15 211 15秒后删除作业号为211的作业

1. **qhold & qrls —— 作业挂起 & 作业释放**

使用qhold命令可以挂起作业，使其不被调度执行；

使用qrls命令可以将挂起的作业释放，使其可以被调度执行；

**命令格式：**

qhold jobid1 jobid2 ...

qrls jobid1 jobid2 ...

其中 jobidX 代表需要操作的作业号，可以一次操作多个作业。

**注意事项**：

qhold只能将排队状态的作业挂起，如果是运行状态的作业是无法挂起的

## PBS环境变量

用户可以通过PBS环境变量获取到作业相关信息，从而更好的跟用户程序进行灵活的配置

|  |  |
| --- | --- |
| **变量名称** | **说明** |
| **登录shell后继承的变量** | 包括$HOME/$LANG/$LOGNAME/$PATH/$MAIL/$SHELL/$TZ |
| **$PBS\_O\_HOST** | qsub提交的节点名称 |
| **$PBS\_O\_QUEUE** | qsub提交的作业的最初队列名称 |
| **$PBS\_O\_WORKDIR** | qsub提交的作业的绝对路径 |
| **$PBS\_JOBID** | 作业被PBS系统所指定的作业号 |
| **$PBS\_NODENAME** | PBS系统指定的作业运行节点名称 |
| **$PBS\_JOBNAME** | 用户指定的作业名 |
| **$PBS\_QUEUE** | PBS在执行时的队列名称 |

## PBS脚本文件

1. **PBS作业脚本选项**

-a date\_time date\_time 格式为：[[[[CC]YY]MM]DD]hhmm[.SS] 表示经过 date\_time 时间后作业才可以运行。

-e path 将标准错误信息重定向到 path

-I 以交互方式运行

-j join 将标准输出信息不标准错误信息合并到一个文件 join 中

-k keep 定义在执行结点上保留标准输出和标准错误信息中的哪个文件。

keep 为o表示保留前者，e表示后者，oe或eo表示二者都保留，

n表示皆不保留。若忽略此选项，二者都不保留。

-l resource\_list 定义资源列表，几个常用的资源种类：

cput=N 请求N秒的CPU时间，也可以是hh:mm:ss的形式。

mem=N[K|M|G][B|W] 请求N {k|m|g}{bytes|words}大小的内存。

nodes=N:ppn=M 请求N个结点，每个结点M个处理器。

-m mail\_option mail\_option 为a：作业abort时给用户发信

为b：作业开始运行发信

为e：作业结束运行时发信

若无此选项，默认为a

-M user\_list 定义有关此作业的mail发给哪些用户

-N name 作业名，限15个字符，首字符为字母，无空格。

-o path 重定向标准输出到 path

-p priority 任务优先级，整数，[-1024，1023]，若无定义则为0

-q destination 指定队列名称

destination 有三种形式 queue；

@server;

queue@server

-r y|n 指明作业是否可运行，y为可运行，n为不可运行。

-S shell 指明执行运行脚本所用的shell，须包含全路径。

-u user\_list 定义作业将在运行结点上以哪个用户名来运行。

-v variable\_list 定义export到本作业的环境变量的扩展列表。

-V 表明qsub命令的所有环境变量都export到此作业。

-W additional\_attributes 作业的其它属性

1. **运行脚本同Linux下一般的运行脚本文件**

**[注]：**脚本文件中的mpirun命令行中的节点列表文件要用环境变量表示。

$PBS\_NODEFILE，这个环境变量表示由PBS自动分配给作业的节点列表；

节点数为命令行中挃定的进程数。

**命令格式：**

mpirun-np 进程数 -hostfile $PBS\_NODEFILE 可执行程序名

## PBS环境下运行示例

1. **脚本文件编辑示例**

**实例1：运行mpi程序**

命令行：#vi aaa.pbs

编辑的内容：

#PBS -N myjob

#PBS -o /home/test/my.out

#PBS -e /home/test/my.err

#PBS -l nodes=2:ppn=2

cd $PBS\_O\_WORKDIR（进入工作目录，即脚本所在的目录）

mpirun -np 4 -hostfile $PBS\_NODEFILE /home/test/helloworld

解释：

原来我们都是在终端输入mpirun这些命令执行程序的，现在只要把这些提交命令放在.pbs配置文件的最后，由PBS来调度执行（自动分配节点和其它资源）。

myjob 是为要运行的程序起的任务名，可以改成你自己想要的名字。

原先输出信息都是直接在屏幕上显示的，现在屏幕上的显示全部输出到文件中。

上例中输出文件 是/home/test/my.out文件，可以根据自己的需要修改（目录，文件名）。程序运行时遇到的一些错误会记录在.err文件中。这样的好处是，因为对每个任务都设定了不同的输出文件，所以看结果只要打开相应文件看就可以了，不需要开多个终端，而且 里面有任务的详细信息，比如实际分配的是哪些节点计算，运行时间等。

pbs -l nodes=2:ppn=2 规定使用的节点数，以及每个节点能跑多少核。

mpirun –np 4 –hostfile $PBS\_NODEFILE /home/test/helloworld

此例中-np后的4是并行数（2×2＝4个cpu），-hostfile $PBS\_NODEFILE不需要改变。/home/test/helloworld 是你编译好的可执行文件名，需修改。

对于每个你要运行的mpi程序都需要这样一个.pbs配置文件，也就是说原来的 操作是：mpirun……，现在改成2步走：

1）写个PBS配置文件（比如xxx.pbs）；

2）向PBS提交（qsub xxx.pbs）

**实例2：运行非mpi程序**

有些用户并不是mpi程序，同样也可以用PBS提交。

命令行：#vi job.pbs

编辑的内容：

#PBS -N myjob

#PBS -o /home/test/my.out

#PBS -e /home/test/my.err

#PBS –q 队列名称

#PBS -l nodes=1:ppn=2

#PBS -r y

cd $PBS\_O\_WORKDIR（原来直接在节点上运行时所在的目录）

sh helloword

解释：

把原来在终端直接输入的命令放到PBS配置文件中，因为只要一个节点，所以 nodes=1

至于用哪个节点系统自动分配，可以用qstat命令查询（比如qstat -n）。

1. **提交作业示例**

命令行：#qsub aaa.pbs

1. **作业状态查询示例**

qstat后加不同参数可以查看不同的信息，

查看作业的状态。

命令行：#qstat -a

解释：

Job id 211是给提交的任务分配的任务号，

S（状态：R代表运行，Q代表 排队，E代表正在退出，H代表挂起，C代表运行完毕）

命令行：#qstat -n 查看作业使用的节点

命令行：#qstat jobid1 jobid2 ... 查看指定作业号的作业（可一次查看多个作业）

命令行：#qstat -u user1 查看指定用户的作业

解释： 该方式输出和默认略有不同，但大同小异。

命令行：#qstat -f jobid 查看特定作业详细信息

解释： 该命令将会输出作业号为 jobid 的作业的详细信息。

# 命令行作业PBS脚本

## Intel-mpi

|  |
| --- |
| **intelmpi.pbs**  **注意事项，intel/2018与intel/2020运行参数不同**  **intel2018**：mpirun -rdma  **intel2020**：mpirun -genv I\_MPI\_FABRICS shm:ofi -genv FI\_PROVIDER mlx |
| #PBS -N job\_vasp  #PBS -l nodes=2:ppn=2  #PBS -l walltime=1200:00:00  #PBS -q batch  #PBS -V  #PBS -S /bin/bash  module load intel/2018  #module load intel/2020  EXEC=xxx  cd $PBS\_O\_WORKDIR  NP=`cat $PBS\_NODEFILE | wc -l`  NN=`cat $PBS\_NODEFILE | sort | uniq | tee /tmp/nodes.$$ | wc -l`  cat $PBS\_NODEFILE > /tmp/nodefile.$$  #intel/2018  mpirun -rdma -machinefile /tmp/nodefile.$$ -n $NP $EXEC > log.$$  #intel/2020  #mpirun -genv I\_MPI\_FABRICS shm:ofi -genv FI\_PROVIDER mlx -machinefile /tmp/nodefile.$$ -n $NP $EXEC > log.$$  rm -rf /tmp/nodefile.$$  rm -rf /tmp/nodes.$$ |

## Ansys

|  |
| --- |
| **ansys.pbs** |
| #!/bin/bash  #PBS -N job\_ansys201  #PBS -l nodes=2:ppn=2  #PBS -q part1  module load ansys/2020r1  input=thermal.txt  export MPI\_WORKDIR=$PBS\_O\_WORKDIR  export MPI\_REMSH=ssh  export MPI\_IC\_ORDER=IBV:TCP  export MPIRUN\_OPTIONS="-prot"  cd $PBS\_O\_WORKDIR  cat $PBS\_NODEFILE > ansys.host  sed -i s/^/ib/g ansys.host  machines=`uniq -c ansys.host | awk 'BEGIN {H=""}{if(H==""){ H=$2":"$1 } else {H=H":"$2":"$1 } } END {print H}'`  ansys201 -b -s -dis -machines $machines -i thermal.txt -o log.$$  rm -rf ansys.host |

## Fluent

|  |
| --- |
| **fluent.pbs** |
| #!/bin/sh -f  #PBS -N job\_fluent  #PBS -q part1  #PBS -l nodes=2:ppn=2  module load intel/2018  module load ansys/2020r1  input=eddy\_417k.jou  nprocs=`wc -l < $PBS\_NODEFILE`  cd $PBS\_O\_WORKDIR  export MPI\_REMSH=ssh  export MPIRUN\_OPTIONS="-prot"  export I\_MPI\_DEVICE=rdssm  cd $PBS\_O\_WORKDIR  date  fluent -pinfiniband -g 3ddp -mpi=intel -t $nprocs -cnf=$PBS\_NODEFILE -i $input -ssh > log.$$  date  rm -rf cleanup-fluent\* |

## CFX

|  |
| --- |
| **cfx.pbs** |
| #!/bin/sh -f  #PBS -N job\_cfx  #PBS -q part1  #PBS -l nodes=2:ppn=2  module load ansys/2020r1  module load intel/2018  input=740\_test.def  cd $PBS\_O\_WORKDIR  export MPIRUN\_OPTIONS=-prot  export CFX5RSH=ssh  export MPI\_IC\_ORDER=IBV:TCP  export I\_MPI\_DEVICE=rdssm  machine=`uniq -c $PBS\_NODEFILE | awk 'BEGIN {H=""}{if(H==""){ H=$2"\*"$1 } else { H=H","$2"\*"$1 } } END {print H}'`  echo "par-dist = $PAR\_MACH"  cfx5solve -def $input -double -start-method "Intel MPI Distributed Parallel" -par-dist $machine |tee log.$$ |

## Vasp

|  |
| --- |
| **vasp.pbs** |
| #PBS -N job\_vasp  #PBS -l nodes=2:ppn=2  #PBS -l walltime=1200:00:00  #PBS -q part1  #PBS -V  #PBS -S /bin/bash  module load intel/2018  module load vasp/5.4.4-intel2018  cd $PBS\_O\_WORKDIR  NP=`cat $PBS\_NODEFILE | wc -l`  NN=`cat $PBS\_NODEFILE | sort | uniq | tee /tmp/nodes.$$ | wc -l`  cat $PBS\_NODEFILE > /tmp/nodefile.$$  mpirun -rdma -machinefile /tmp/nodefile.$$ -n $NP vasp\_std > log.$$  rm -rf /tmp/nodefile.$$  rm -rf /tmp/nodes.$$ |

## Lammps

|  |
| --- |
| **lammps.pbs** |
| #PBS -N job\_lammps  #PBS -q batch  #PBS -l nodes=2:ppn=2  #PBS -S /bin/bash  #PBS -V  module load intel/2018  module load lammps/7Aug19-intel2018  input=PdP.in  cd $PBS\_O\_WORKDIR  NP=`cat $PBS\_NODEFILE | wc -l`  NN=`cat $PBS\_NODEFILE | sort | uniq | tee /tmp/nodes.$$ | wc -l`  cat $PBS\_NODEFILE > /tmp/nodefile.$$  mpirun -rdma -machinefile /tmp/nodefile.$$ -n $NP lmp\_intel\_cpu\_intelmpi < $input |tee log.$$  rm -rf /tmp/nodefile.$$  rm -rf /tmp/nodes.$$ |

## OpenFOAM

|  |
| --- |
| **openfoam.pbs** |
| #PBS -N job\_OpenFOAM  #PBS -l nodes=2:ppn=2  #PBS -l walltime=1200:00:00  #PBS -q batch  ###############set path and library  module load intel/2018  module load openfoam/7  source /software/application/OpenFOAM/OpenFOAM-7/etc/bashrc  #################work  cd $PBS\_O\_WORKDIR  NP=`cat $PBS\_NODEFILE | wc -l`  NN=`cat $PBS\_NODEFILE | sort | uniq | tee /tmp/nodes.$$ | wc -l`  cat $PBS\_NODEFILE > /tmp/nodefile.$$  ./makeMesh  decomposePar  mpirun -rdma -np $NP -machinefile /tmp/nodefile.$$ pimpleFoam -parallel >log.$$  rm -rf /tmp/nodefile.$$  rm -rf /tmp/nodes.$$ |

## Gaussian-g09

|  |
| --- |
| **g09.pbs** |
| #!/bin/sh  #PBS -N job\_g09  #PBS -l nodes=1:ppn=4  #PBS -l walltime=999:00:00  #PBS -q batch  #PBS -V  module load gaussian/g09  input=test0001.com  cd $PBS\_O\_WORKDIR  g09 <$input > $input.log |

## Gaussian-g16

|  |
| --- |
| **g16.pbs** |
| #PBS -N job\_gauss  #PBS -l nodes=2:ppn=2  #PBS -l walltime=12:00:00  #PBS -q batch  #PBS -V  #PBS -S /bin/bash  module load gaussian/g16  cd $PBS\_O\_WORKDIR  input=test0001.com  if [ ! -e $FILENAME ]  then  echo "$FILENAME does not exist,g16 can not run"  exit 1  fi  LINE=`cat $PBS\_NODEFILE | sort | uniq | tee nodelist.$$ | wc -l `  i=1  string="%lindaworker="  while [ $i -le $LINE ]  do  node=`sed -n "$i p" nodelist.$$`  if [ $i -eq $LINE ]; then  string=$string$node  else  string=$string$node","  fi  let i+=1  done  string2="%mem=$MEM"  string3="%nprocshared=$NprocShared"  rm -f nodelist.$$  g16 < $input> $input.log |

## Anaconda

|  |
| --- |
| **conda.pbs** |
| #PBS -N job\_torch  #PBS -l nodes=1:ppn=4  #PBS -l walltime=1200:00:00  #PBS -q batch  #PBS -V  #PBS -S /bin/bash  module load anaconda/3  source activate pytorch-1.4.0  cd $PBS\_O\_WORKDIR  python demo.py > demo.log |

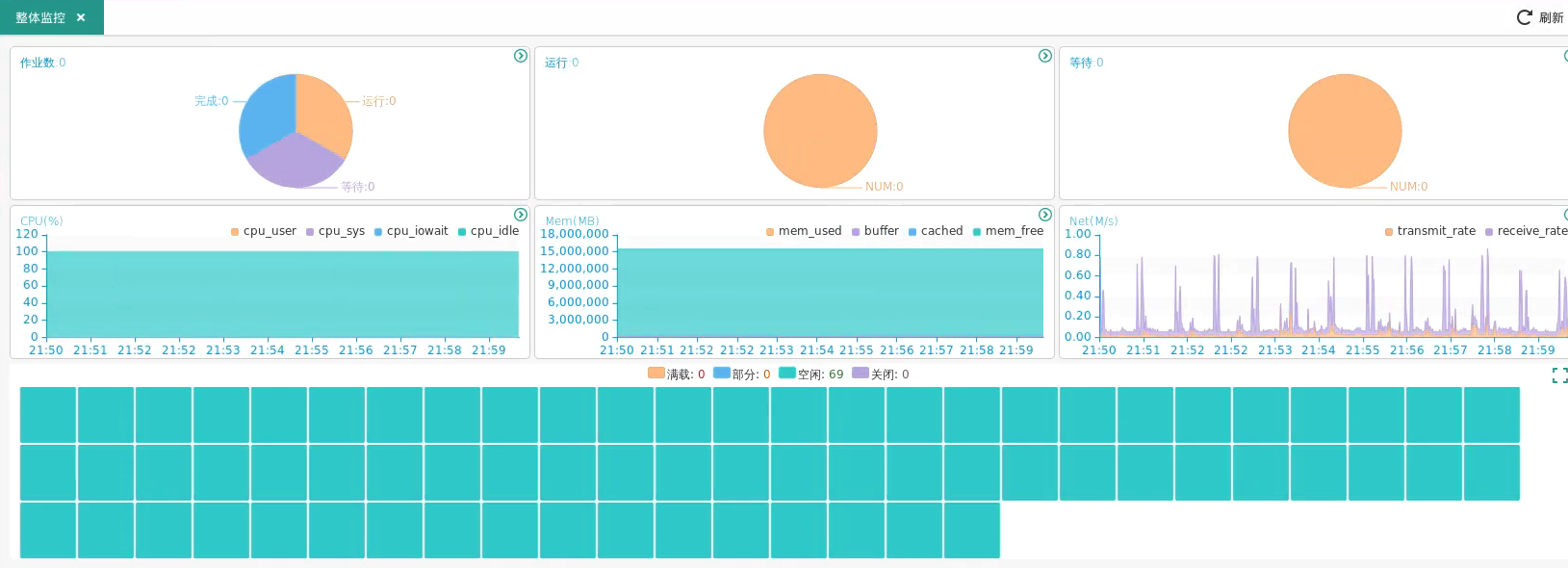
# Web浏览器端作业管理

登录地址：http://202.115.63.31:8080

如果遇到安全警告，选择高级-添加例外

## 集群监控

通过web端可查集群状态，资源使用情况，如下所示



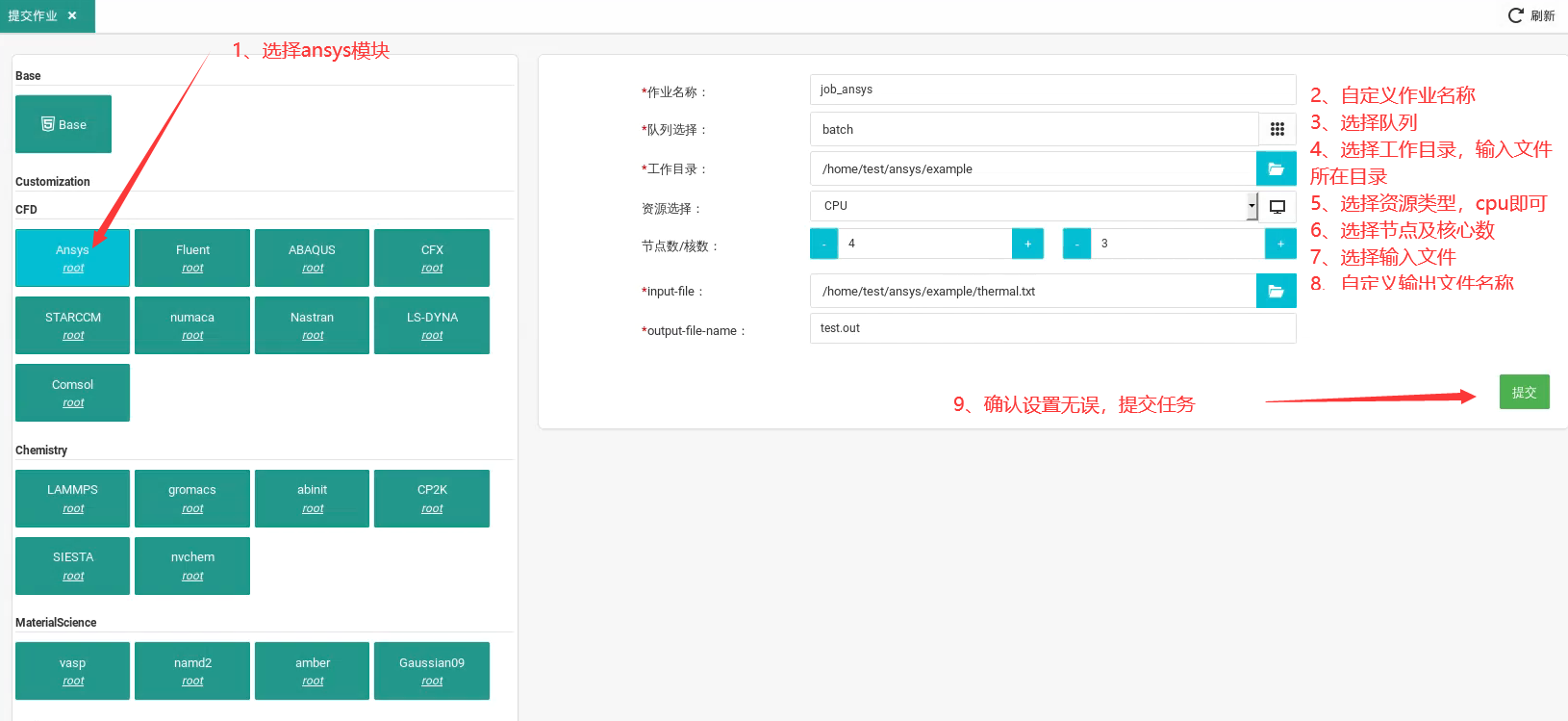


## 作业提交

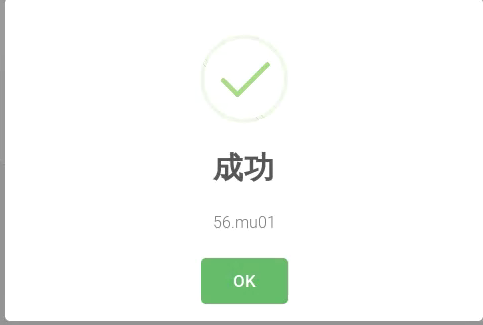
以下以ansys模块为例

选择作业管理->作业提交->选择要提交的作业模块（ansys）

左侧根据提示进行对应的项目的输入，确认无误后提交即可



提交成功有以下提示



通过作业查看，可以看到作业运行状态，如下R表示正常运行

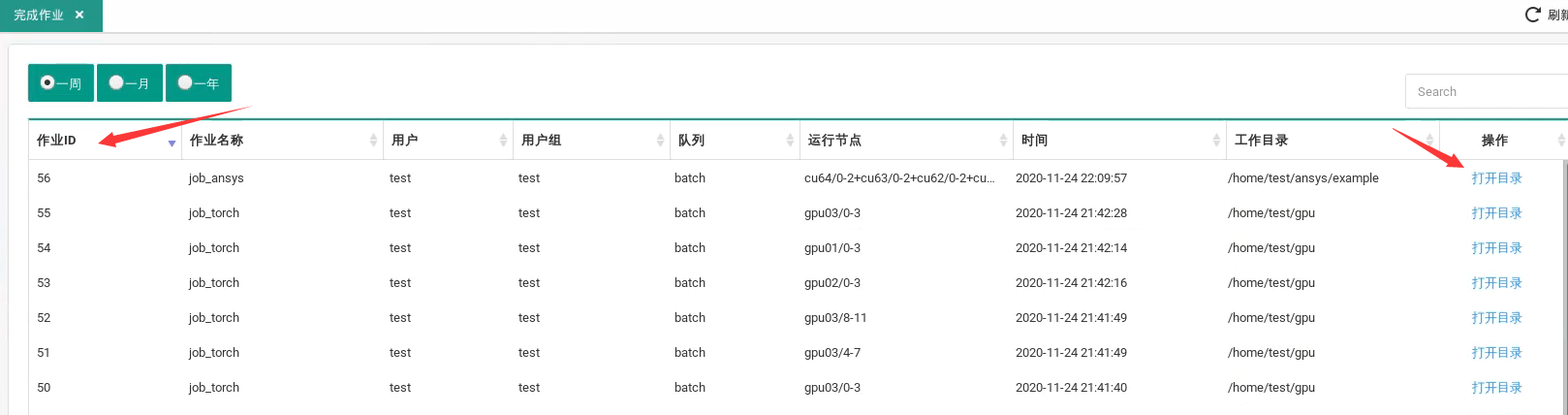


在作业查看左侧操作栏中，可查看作业信息、日志输出、资源利用监控

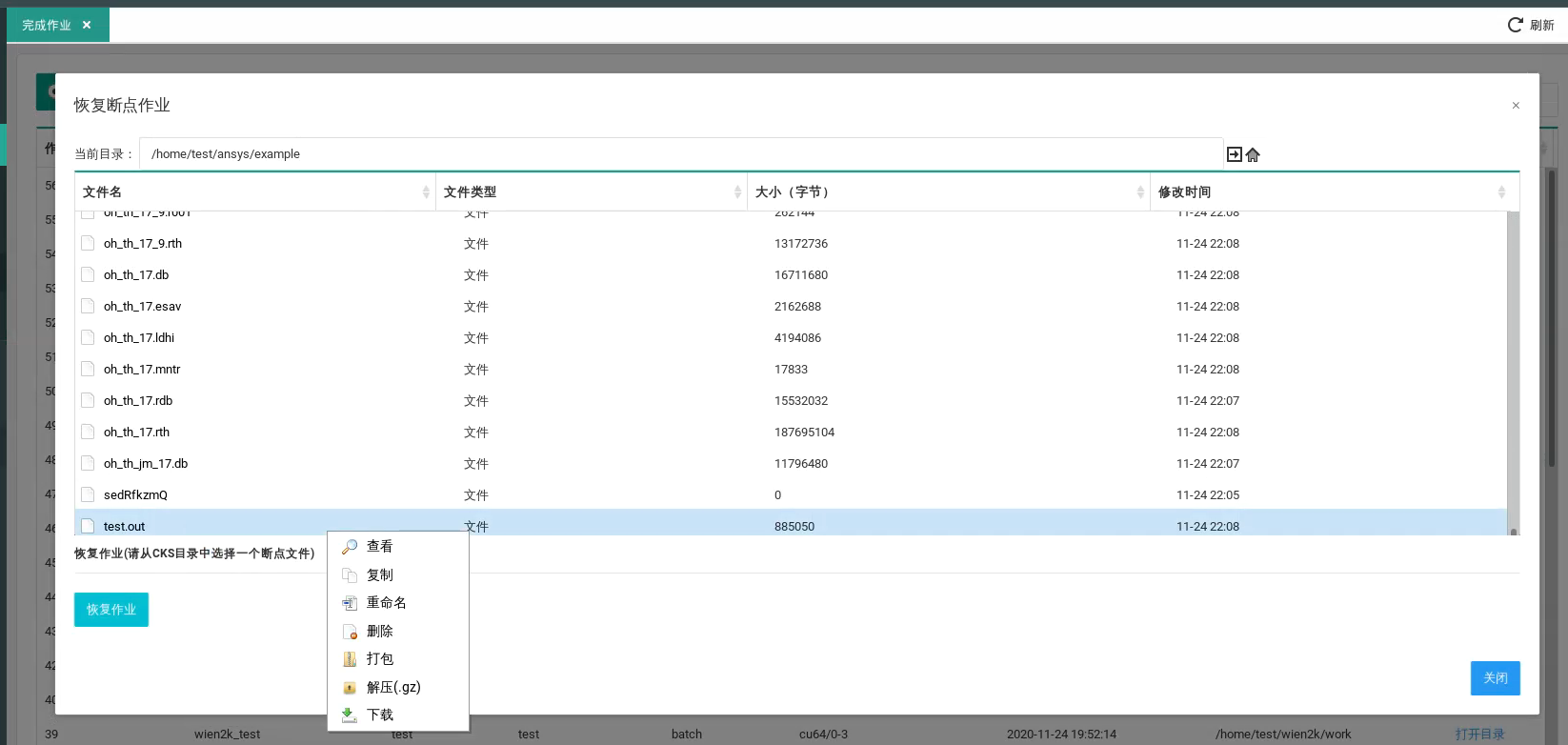


## 文件下载

作业完成后如果要下载输出文件，在完成作业栏中，可点击作业ID进行排序，点击打开目录，



选择要操作的文件，点击右键进行相应的操作



## 文件管理

在管理工具中选择文件管理，即可进入家目录，选择文件进行相应操作，或空白处点击右键进行文件上传等操作

