华南农业大学 South China Agricultural University

华南农业大学
基因组学与生物信息学创新研究中心
高性能计算集群用户培训手册v1.4

发布时间：2024年6月17日

目录

集群概况

基础概念

高性能计算集群：高性能计算集群（High Perfermance Computing），即超级计算机。它是将众多服务器，网络，存储，以及并行消息传递库，作业调度器等软硬件融合在一起的计算机设备。其广泛应用于生命科学、计算化学、高能物理、气象预报、航空航天、能源勘探、工业制造、人工智能等领域。

节点：一个包含有限CPU、内存等计算资源的独立物理服务器

分区：多个节点（通常是相同配置的节点）组成的计算资源池

集群概况

华南农业大学农学院高性能集群目前有1个管理节点，2个登录节点，66个计算节点。

管理节点是整个集群的控制核心，不对用户开放。登录节点是用户登录集群的唯一一个入口。登录节点命名为ln01和ln02，配置32颗intel 4314 2.4GHz CPU，128G内存，用户的所有操作（代码及数据上传，作业脚本编辑及提交）均在此两个节点完成。

66个计算节点一共有6种配置，共划分为6个计算分区。分别是56c503g、 52c191g、28c128g、192c11t、144c4t、56c1t8n这6个分区。

56c503g分区配置了30个计算节点（主机名为cu01至cu30），每个计算节点配置了56个Intel 6348 2.6GHz CPU核心，503G内存，适用于内存需求不高，计算要求高的纯CPU计算应用程序。

52c191g分区配置了13个计算节点（主机名为cu31至cu43），每个计算节点配置了52个Intel 6230R 2.1GHz CPU核心，191G内存，适用于内存需求不高，计算要求高的纯CPU计算应用程序。

28c128g分区配置了20个计算节点（主机名为cu51至cu70），每个计算节点配置了28个Intel E5-2680 2.4GHz CPU核心，128G内存，适用于内存需求不高，计算要求高的纯CPU计算应用程序。

集群概况

192c11t分区配置了1个计算节点（主机名为fat01），该计算节点配置了192个Intel 8260L 2.4Hz CPU核心，12093G内存，适用于内存需求非常高的纯CPU计算应用程序。

144c4t分区配置了1个计算节点（主机名为fat02），该计算节点配置了144个Intel E7-8860 2.2Hz CPU核心，4031G内存，适用于内存需求较高的纯CPU计算应用程序。

56c1t8n分区配置了1个计算节点（主机名为gpu01），该计算节点配置了56个Intel 6348 2.6GHz CPU核心，1007G内存，8张NVIDIA A800 PCIE（单张显卡显存80G）显卡，适用于CPU+GPU异构计算的应用程序。

集群存储采用分布式共享存储系统，挂载到所有节点的/public目录， /public目录为共享目录，用于存放用户家目录（ /public/home），共享软件（ /public/software）等，其空间约为1.4P。

用户的应用软件可安装在用户自己的家目录下面。常用的并行库及编译器及部分通用软件一般是集群管理员安装在/public/software目录下，用户可直接调用/public/software目录下的所有软件。

集群使用前提

HPC集群使用需要具备以下三个知识点，若用户当前不具备，则需要用户自行学习掌握相应的知识点后再使用集群。

1、集群使用者需要具备linux操作系统的基本知识概念，如目录、文件、权限等。熟悉linux操作系统的常用命令操作，如vi、cd、cat、ls、mkdir、rm、touch、mv、cp、pwd、chmod、tar、find、grep等常用命令。

2、集群使用者需要熟悉你所使用到的科研软件（如python，vasp，ansys，gaussian，bwa软件多线程，并行计算的用法）。

3、集群使用者需要对HPC集群的调度系统有一定的了解，如slurm、pbs、lsf、sge等。

集群使用步骤

集群使用步骤如下

1、请求管理员开通一个专属于你的个人账号。

2、使用个人账号登录集群，上传数据或程序代码，比如userA用户将其数据集代码上传到自己的家目录/public/home/groupname/userA/下。

3、查看软件安装路径（如/public/software）目录内容，确认你所需要用到的软件是否在集群内都已经安装，或者咨询管理员老师集群集群已安装的软件及使用方式。也可以按需将软件安装到自己的家目录下。

4、在/public/home/groupname/userA用户家目录下编写作业脚本或者程序执行脚本。slurm作业脚本模板可参考第4章或者集群的/public/sourcecode/slurm_samples/目录下作业脚本模板文件。

5、在slurm作业脚本路径上执行sbatch命令提交作业，或者使用srun，salloc提交交互式作业。

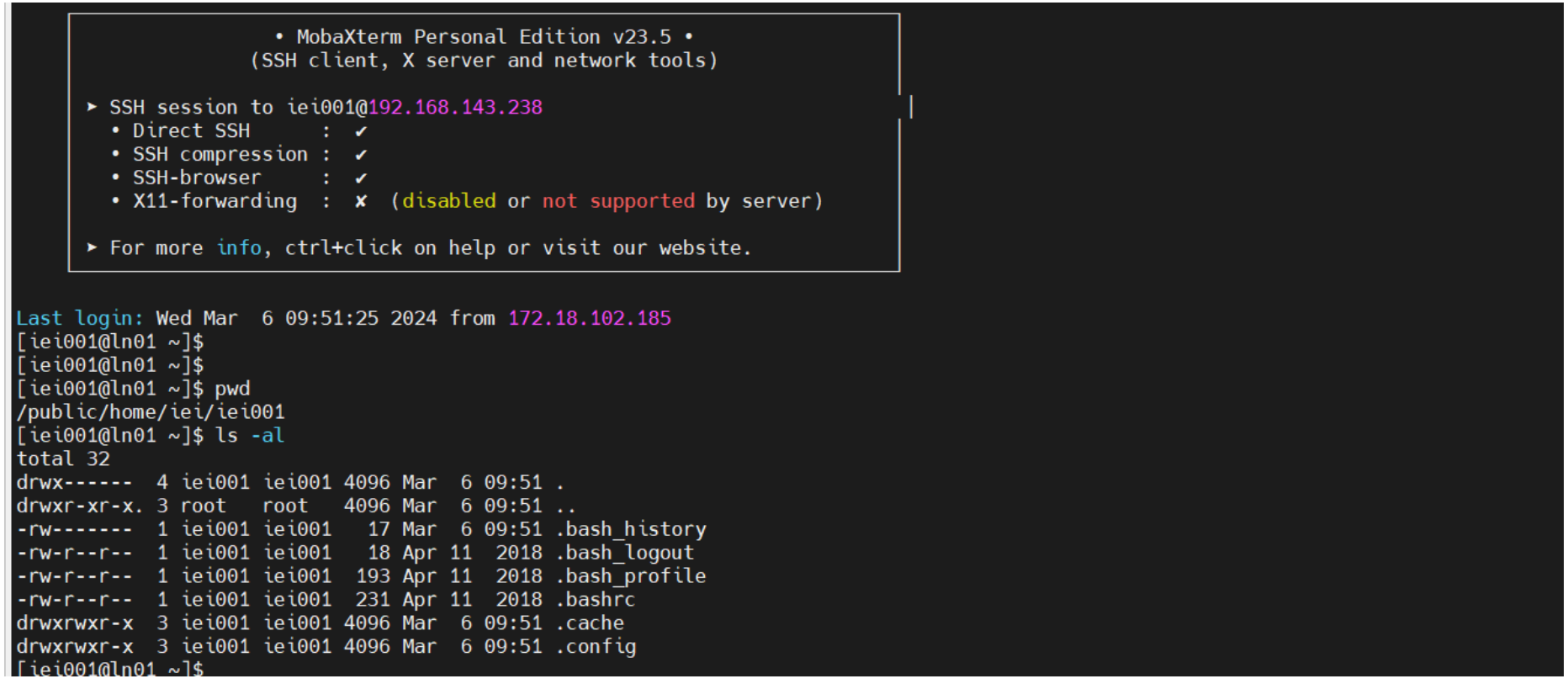
6、使用squeue命令查看作业运行状态，或者使用scancel删除有问题的作业 。

7、作业运行完毕，使用cat、less等文件查看命令查看计算结果。

集群登录

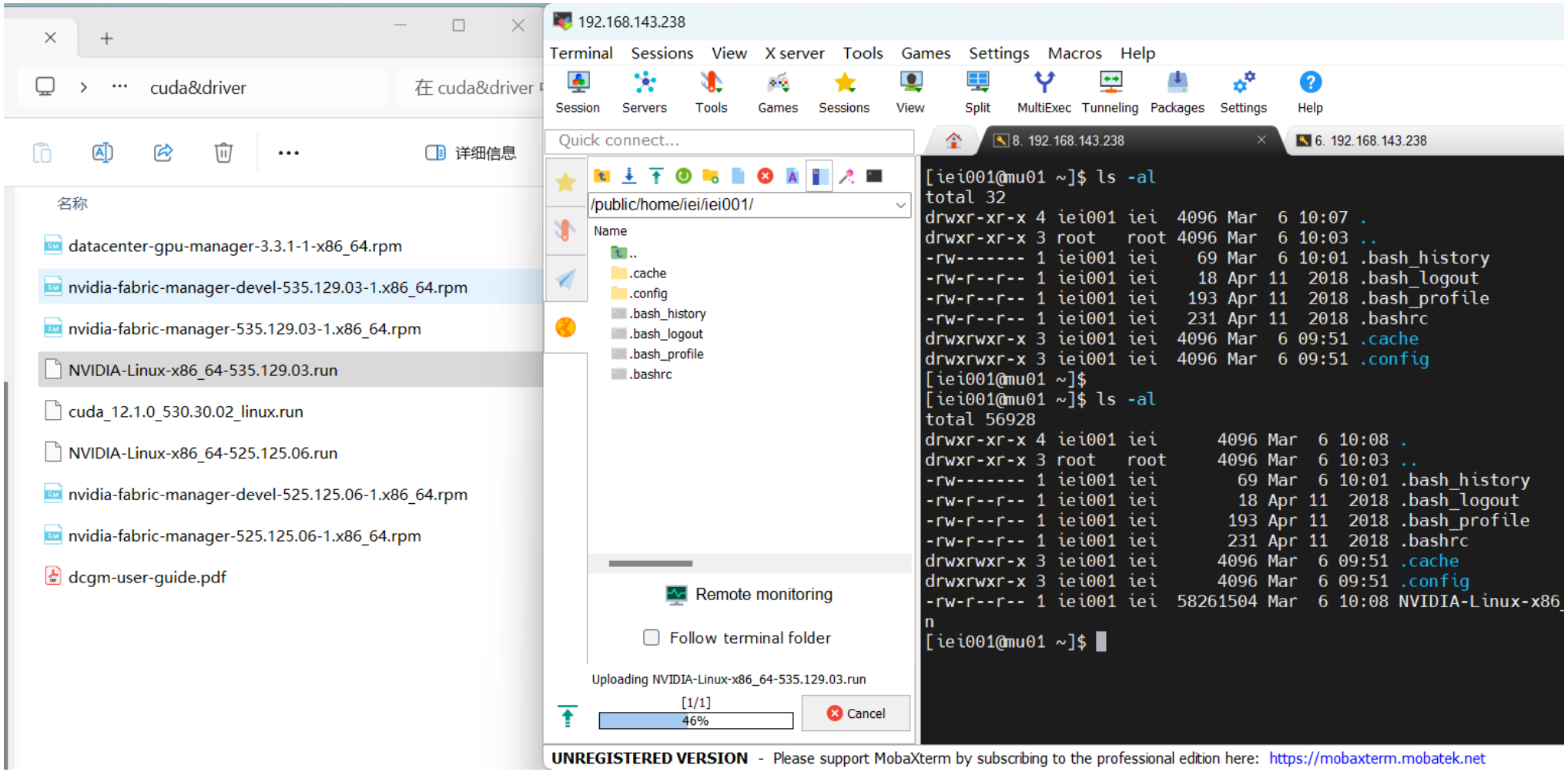
集群基于CentOS系统搭建，windows用户可通过Mobaxterm，xshell，putty等常用终端工具登录集群的shell界面，Linux或者macOS用户可直接通过terminal工具直接使用ssh username@192.168.87.200登录集群，登录地址是192.168.87.200端口号22。

shell登录界面如下



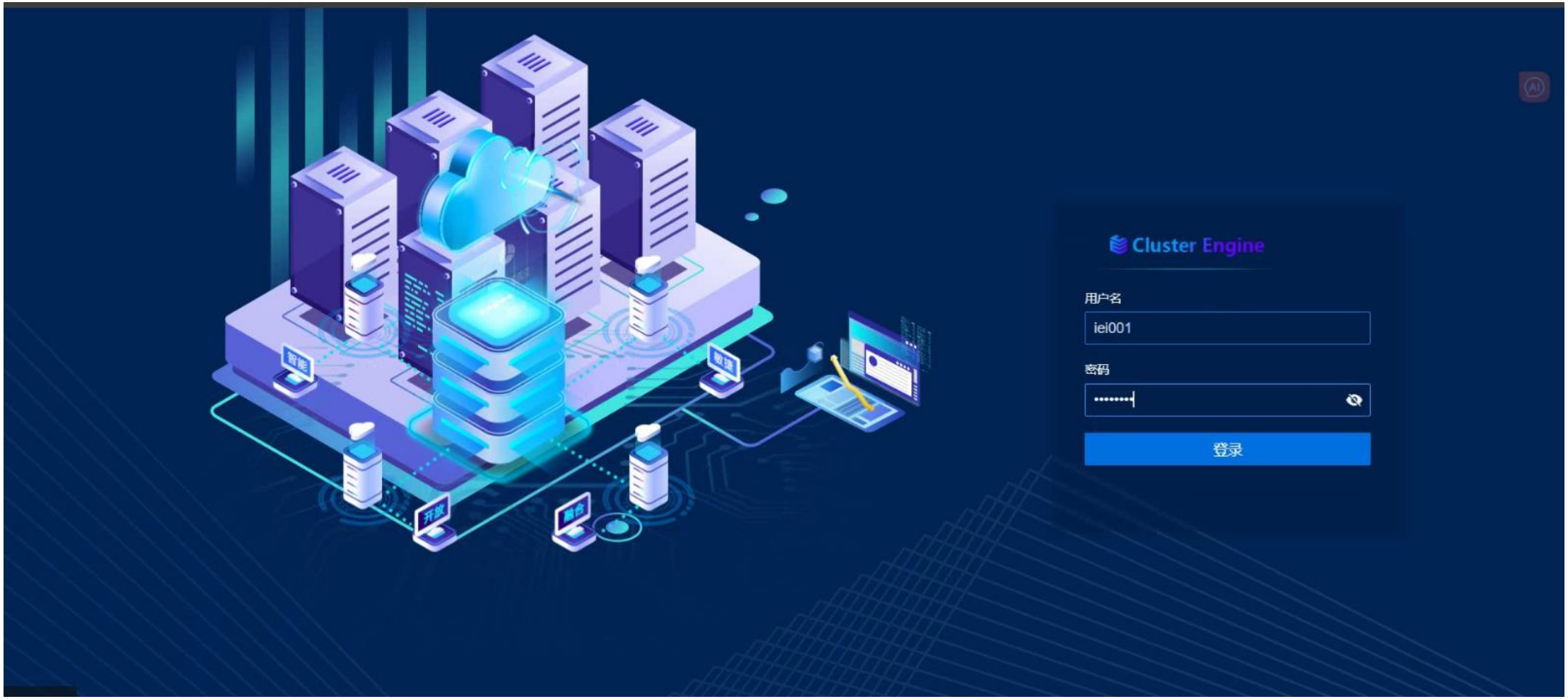
文件传输

文件传输如下，以Mobaxterm为例，将本地电脑文件往集群上传或者下载



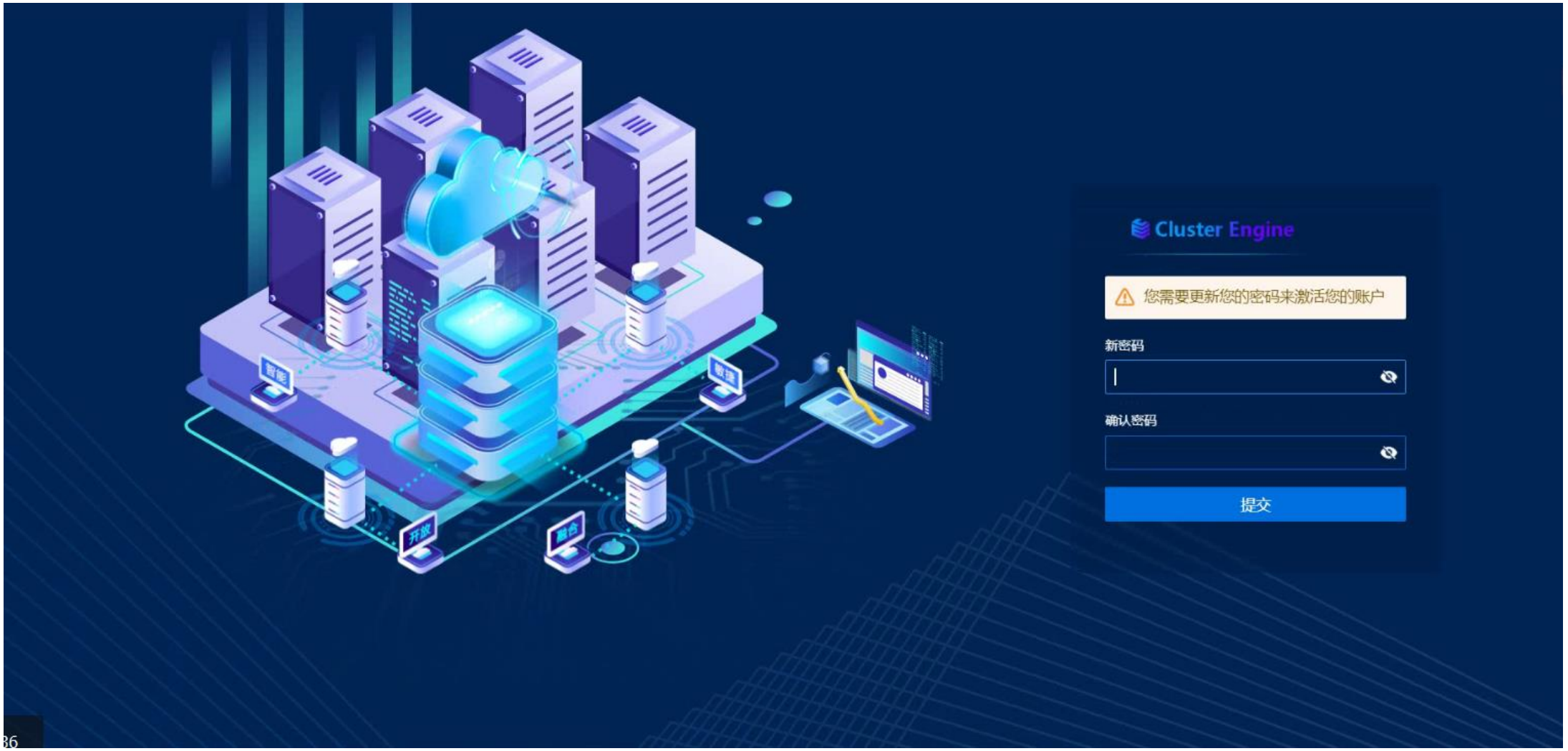
密码修改

首次登录集群，需要登录web页面修改密码，web登录地址为https://192.168.87.200



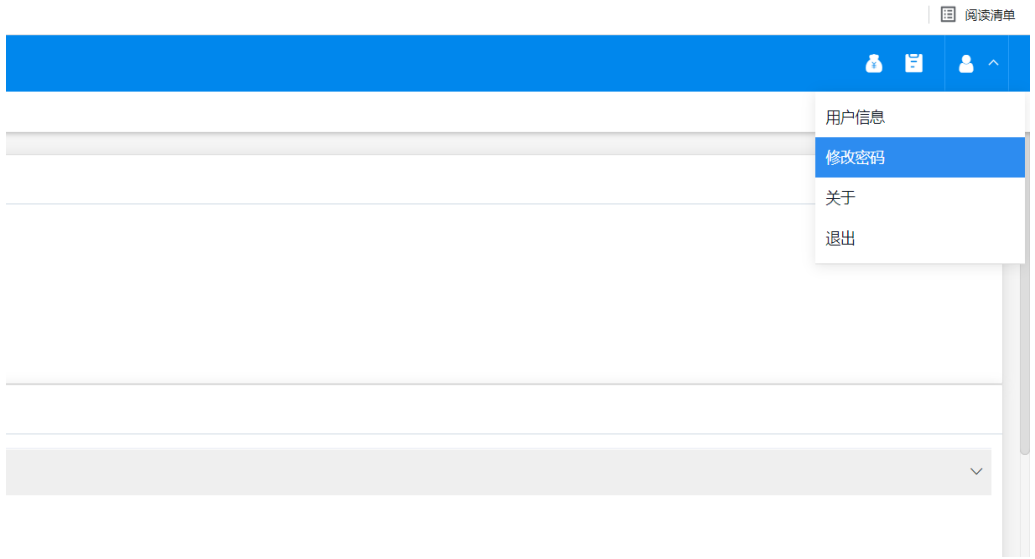
密码修改

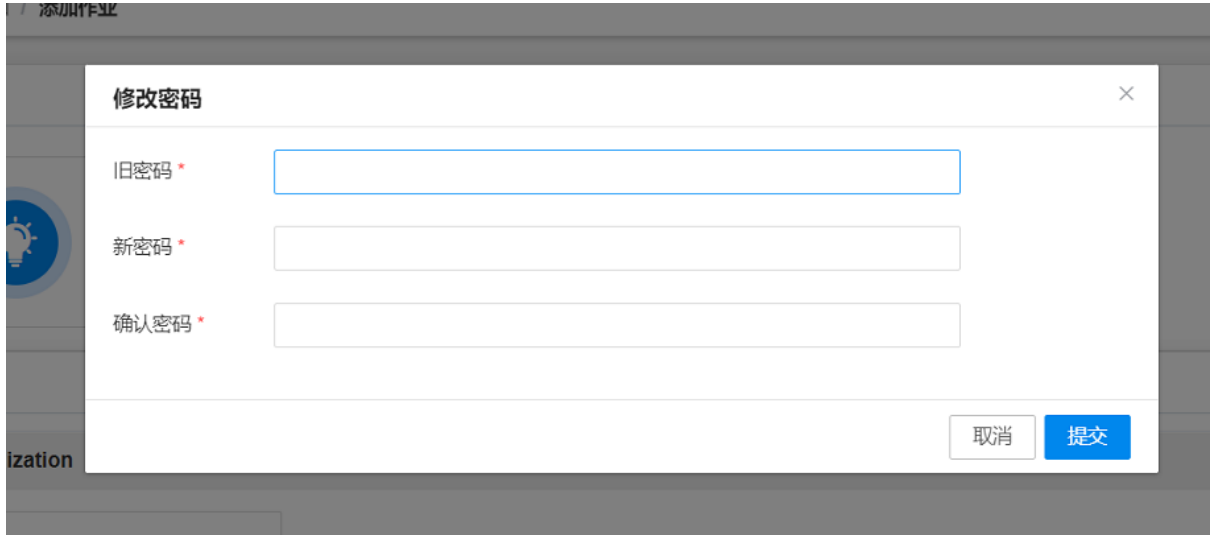
首次登录会强制用户修改密码，重置后可用新密码登录web以及shell（注意：在web上修改账号密码后会同步到shell终端）



密码修改

若非首次修改密码，则需要普通用户登录进去web后在右上方，点击修改密码，根据提示输入原有密码以及修改新的密码，点击提交（注意密码复杂度，防止被盗用）。修改完密码后可以用新的密码通过shell工具重新登录集群





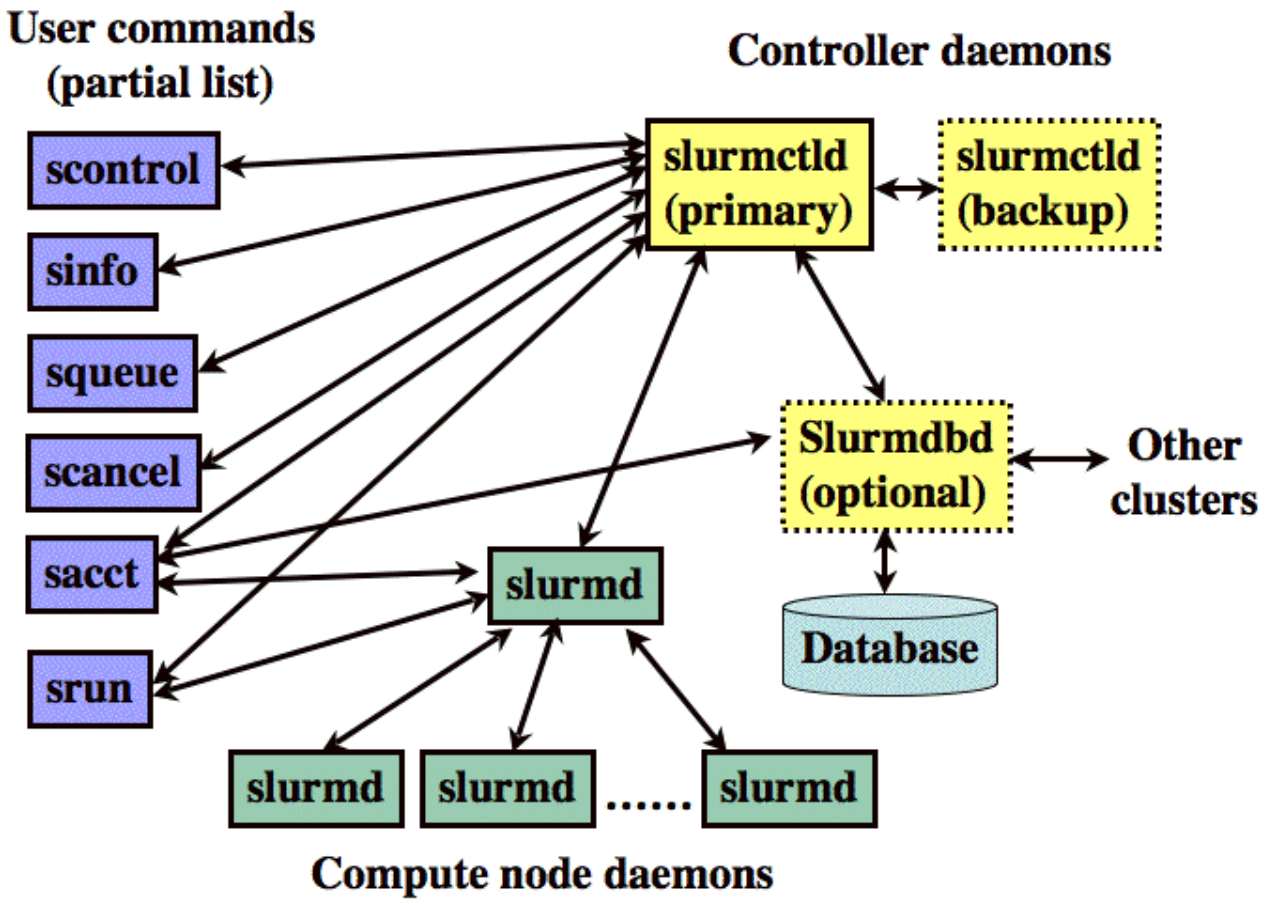
目录

slurm简介

SLURM （Simple Linux Utility for Resource Management）是一种可用于大型计算节点集群的高度可伸缩和容错的集群管理器和作业调度系统，被世界范围内的超级计算机和计算集群广泛采用。它提供了三个关键功能。第一，为用户分配一定时间的专享或非专享的资源(计算机节点)，以供用户执行工作。第二，它提供了一个框架，用于启动、执行、监测在节点上运行着的任务(通常是并行的任务，例如 MPI)。第三，为任务队列合理地分配资源。

目前国防科大，北京大学，上海交大，天河二号等众多高校及超算中心均采用slurm作为其作业调度系统。

slurm架构



- slurmctld：slurm调度系统服务端服务
- slurmd： slurm调度系统客户端服务
- slurmdbd：记录account信息
- sinfo：查看集群分区和节点的状态
- sbatch：运行slurm脚本式作业
- srun：交互式运行作业
- salloc：交互式申请资源
- squeue：报告作业或作业步骤的状态
- scancel：取消挂起或等待或者运行的作业
- scontrol：查看集群配置和状态信息，查看和修改作业（已提交且未运行）参数。
- sacct：查看已完成的作业

目录

slurm常用命令操作介绍

命令	作用
sinfo	查看有关slurm节点和分区的信息
scontrol show partition/node/job	查看详细分区/节点/作业信息
sbatch	批处理方式提交作业
srun	交互式提交并行作业
salloc	交互式申请资源
squeue	报告作业或作业步骤的状态
scancel	取消挂起或等待或者运行的作业
sacct	查看已完成的作业

sinfo节点状态

- STATE：节点状态，可能的状态包括以下几种：
  - idle：节点完全空闲，可以接收新作业
  - alloc：节点CPU核心已全部被分配
  - mix：混合，节点在运行作业，但有些空闲CPU核，可接受新作业。
  - down：服务异常或者节点OS已宕机。
  - drain：已失去活力。

```
[iei001@ln01 ~]$ sinfo -Nla
Sun May 19 18:06:10 2024
NODELIST   NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
cu01           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu02           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu03           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu04           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu05           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu06           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu07           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu08           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu09           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu10           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu11           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu12           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu13           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu14           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu15           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu16           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu17           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu18           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu19           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu20           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
cu21           1  56c503g*        idle 56     2:28:1 515400        0      1   (null) none
```

sbatch

sbatch命令采用批处理方式运行作业，sbatch提交完脚本后，立即退出，同时获取到一个作业号,作业等所需资源满足后开始运行（详细操作，请参考：man sbatch） 。具体过程如下：

1. 用户编写slurm作业脚本（该脚本为linux普通shell脚本文件）；
2. 使用*sbatch 脚本文件名称*命令提交作业；
3. 作业排队等待资源分配；
4. 分配资源后执行作业；
5. 脚本执行结束，释放资源；
6. 运行结果定向到指定的文件中记录（JobID.out 和JobID.err两个文件），其中JobID.out 文件为标准输出， JobID.err 文件为错误输出。

```
[iei001@ln01 General_script]$ pwd
/public/home/iei/iei001/slurm_samples/General_script
[iei001@ln01 General_script]$ ls
hello-gpu.slurm  hello.slurm
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$ sbatch hello.slurm
Submitted batch job 1095
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$
```

salloc

salloc在获取分配的资源后ssh跳转到节点内执行程序命令，当命令结束后需手动退出释放分配的资源（详细操作，请参考：man salloc）。

1.提交资源分配请求； 2.作业排队等待资源分配；3.查看分配资源详细情况 4、执行用户指定的命令

```
[iei001@ln01 ~]$ salloc -N 1 --ntasks-per-node=56 -p 56c503g --mem-per-cpu=8G --account=iei
salloc: Granted job allocation 48
salloc: Waiting for resource configuration
salloc: Nodes cu14 are ready for job
[iei001@ln01 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                48   56c503g interact   iei001  R       3:23      1 cu01
[iei001@ln01 ~]$ ssh cu14
iei001@cu14's password:
Last login: Sun May 19 18:12:28 2024 from 12.12.12.202
[iei001@cu14 ~]$ cd slurm_samples/General_script/
[iei001@cu14 General_script]$ pwd
/public/home/iei/iei001/slurm_samples/General_script
[iei001@cu14 General_script]$
```

salloc

若程序执行完毕，需手动执行两次exit退出程序执行节点并释放作业所申请的资源

#注意：salloc交互式作业，如果关闭的当前shell窗口，或者电脑异常重启，运行的作业将会被自动中止，（nohup & 参数在后台执行的程序也无效）

srun

srun可以交互式提交运行并行作业，提交后，作业等待运行，等运行完毕后，才返回终端（详细操作，请参考：man srun） 。流程如下：

1. 在终端提交资源分配请求，指定资源数量与限制；
2. 等待资源分配；
3. 获得资源后，加载计算任务；
4. 运行中，任务 I/O 传递到终端；
5. 可与任务进行交互，包括 I/O，信号等；
6. 任务执行结束后，资源被释放。

```
[iei001@ln01 ~]$
[iei001@ln01 ~]$ srun -N 2 --ntasks-per-node=2 -p 56c503g --mem-per-cpu=8G --account=iei hostname
cu01
cu01
cu02
cu02
[iei001@ln01 ~]$
```

sbatch、salloc、srun主要参数

- -N, --nodes=<nodenum>：申请执行作业节点数，
- -n , --ntasks 作业总的进程数 (不建议使用)
- --ntasks-per-node：每个节点使用的进程数，
- -c, --cpus-per-task：每个进程使用的CPU核心数。
- -J, --job-name=<jobname>：设定作业名<jobname>
- -p, --partition=<partition_names> ：将任务提交到指导的partition_names分区中
- --gres=gpu:1 ：每个节点的GPU数量
- -w, --nodelist=<node name list>：指定作业运行在某些节点上。
- --exclusive：独占模式(不建议使用，若使用会产生更多费用)
- --array ： 数组作业
- --dependency：依赖作业

squeue

squeue：显示分区中的作业信息(详细操作，请使用man squeue)。如squeue显示

```
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1160 cpu-low-m hello-te   iei001 PD       0:00      2 (None)
              1159 cpu-low-m hello-te   iei001 PD       0:00      2 (None)
              1145 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1144 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1143 cpu-low-m hello-te   iei001 PD       0:00      2 (Resources)
              1131 cpu-low-m hello-te   iei001  R       0:09      2 cu[01-02]
              1132 cpu-low-m hello-te   iei001  R       0:09      2 cu[01-02]
              1133 cpu-low-m hello-te   iei001  R       0:09      2 cu[03-04]
              1134 cpu-low-m hello-te   iei001  R       0:09      2 cu[03-04]
[iei001@ln01 General_script]$
```

scancel

scancel：删除分区中的作业(详细操作，请使用man scancel)。如scancel显示，对正在运行的作业号为1133的作业进行删除操作

```
[iei001@ln01 General_script]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1160 cpu-low-m hello-te   iei001 PD       0:00      2 (None)
              1159 cpu-low-m hello-te   iei001 PD       0:00      2 (None)
              1145 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1144 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1143 cpu-low-m hello-te   iei001 PD       0:00      2 (Resources)
              1131 cpu-low-m hello-te   iei001  R       0:09      2 cu[01-02]
              1132 cpu-low-m hello-te   iei001  R       0:09      2 cu[01-02]
              1133 cpu-low-m hello-te   iei001  R       0:09      2 cu[03-04]
              1134 cpu-low-m hello-te   iei001  R       0:09      2 cu[03-04]
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$ scancel 1133
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1160 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1159 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1145 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1144 cpu-low-m hello-te   iei001 PD       0:00      2 (Resources)
              1143 cpu-low-m hello-te   iei001  R       0:00      2 cu[03-04]
              1131 cpu-low-m hello-te   iei001  R       0:17      2 cu[01-02]
              1132 cpu-low-m hello-te   iei001  R       0:17      2 cu[01-02]
              1134 cpu-low-m hello-te   iei001  R       0:17      2 cu[03-04]
[iei001@ln01 General_script]$
```

目录

slurm脚本作业

slurm调度系统常用的方法是通过脚本方式提交作业。脚本作业使用方式具体流程如下：

1. 用户编写slurm作业脚本（该脚本为linux普通shell脚本文件）；
2. 使用*sbatch 脚本文件名称*命令提交作业；
3. 作业排队等待资源分配；
4. 分配资源后执行作业；
5. 脚本执行结束，释放资源；
6. 运行结果定向到指定的文件中记录（JobID.out 和JobID.err两个文件），其中JobID.out 文件为标准输出， JobID.err 文件为错误输出。若用户程序执行命令部分将结果重定向到指定文件，则JobID.out 文件不存储程序计算结果的输出。

通用CPU计算任务作业脚本模板

CPU计算任务通用脚本内容如下，所有CPU计算任务均可在该模板的基础上进行相应的修改

```
#!/bin/bash
#SBATCH --job-name=python-test       ##作业名称
#SBATCH --partition=56c503g        ##作业申请的分区名称
#SBATCH --nodes=2                    ##作业申请的节点数
#SBATCH --ntasks-per-node=56         ##作业申请的每个节点使用的核心数
#SBATCH --mem-per-cpu=8500MB     ##作业单个CPU核心使用的内存
#SBATCH --account=iei                ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
```

通用CPU计算任务作业脚本模板

```
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++++++"

##以下几行为加载软件环境变量（注意：你在该任务里面需要用到的所有软件均需要添加到这个位置，务必根据实际情况按需添加或者删除）

#setting environment for your software          ##设置你在本作业需要用到的软件环境变量
Program excute Command                          ##GPU程序执行命令语句，每个软件的执行命令都是互不相同的，请自行查询你所需要使用到的软件程序执行的命令格式，务必根据实际情况修改。软件跑串行、单节点多和并行、多节点多核并行、多线程请参考你所用到的软件的使用说明。

echo "++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

通用GPU计算任务作业脚本模板

GPU计算任务通用脚本内容如下，所有GPU计算任务均可在该模板的基础上进行相应的修改

```
#!/bin/bash
#SBATCH --job-name=gpu-test          ##作业名称
#SBATCH --partition=56c1t8n          ##作业申请的分区名称
#SBATCH --nodes=1                    ##作业申请的节点数
#SBATCH --ntasks-per-node=8          ##作业申请的每个节点使用的核心数
#SBATCH --gres=gpu:1                 ##作业申请的每个节点GPU卡数量
#SBATCH --mem-per-cpu=17500MB        ##作业单个CPU核心使用的内存
#SBATCH --account=iei                ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

通用GPU计算任务作业脚本模板

```
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++++++"

##以下几行为加载软件环境变量（注意：你在该任务里面需要用到的所有软件均需要添加到这个位置，务必根据实际情况按需添加或者删除）

#setting environment for your software          ##设置你在本作业需要用到的软件环境变量
Program excute Command                          ##GPU程序执行命令语句，每个软件的执行命令都是互不相同的，请自行查询你所需要使用到的软件程序执行的命令格式，务必根据实际情况修改。软件跑串行、单节点多和并行、多节点多核并行、多线程请参考你所用到的软件的使用说明。

echo "++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

指定节点作业脚本模板

指定节点作业脚本内容如下，可在该模板的基础上进行相应的修改

```
#!/bin/bash
#SBATCH --job-name=nodelist-test        ##作业名称
#SBATCH --partition=56c503g          ##作业申请的分区名称
#SBATCH --ntasks-per-node=8             ##作业申请的每个节点使用的核心数
#SBATCH --nodelist=cu01                    ##作业指定节点
#SBATCH --mem-per-cpu=8500MB        ##作业单个CPU核心使用的内存
#SBATCH --account=iei                      ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

指定节点作业脚本模板

```
echo "process will start at : "
date
echo "+++++++++++++++++++++++++++++++++++++++++++++"

##以下几行为加载软件环境变量（注意：你在该任务里面需要用到的所有软件均需要添加到这个位置，务必根据实际情况按需添加或者删除）

#setting environment for your software          ##设置你在本作业需要用到的软件环境变量
Program excute Command                          ##GPU程序执行命令语句，每个软件的执行命令都是互不相同的，请自行查询你所需要使用到的软件程序执行的命令格式，务必根据实际情况修改。软件跑串行、单节点多和并行、多节点多核并行、多线程请参考你所用到的软件的使用说明。

echo "+++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

数组作业脚本模板

数组作业脚本内容如下，可在该模板的基础上进行相应的修改

```
#!/bin/bash
#SBATCH --job-name=array-test          ##作业名称
#SBATCH --partition=56c503g       ##作业申请的分区名称
#SBATCH --nodes=2                      ##作业申请的节点数
#SBATCH --ntasks-per-node=8            ##作业申请的每个节点使用的核心数
#SBATCH --mem-per-cpu=8500MB        ##作业单个CPU核心使用的内存
#SBATCH --array   1-10                 ##数组范围
#SBATCH --account=iei                  ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

数组作业脚本模板

```
echo "process will start at : "
date
echo "+++++++++++++++++++++++++++++++++++++++++++++"

##以下几行为加载软件环境变量（注意：你在该任务里面需要用到的所有软件均需要添加到这个位置，务必根据实际情况按需添加或者删除）

#setting environment for your software          ##设置你在本作业需要用到的软件环境变量
Program excute Command                          ##GPU程序执行命令语句，每个软件的执行命令都是互不相同的，请自行查询你所需要使用到的软件程序执行的命令格式，务必根据实际情况修改。软件跑串行、单节点多和并行、多节点多核并行、多线程请参考你所用到的软件的使用说明。

echo "+++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

依赖作业脚本模板

依赖作业脚本内容如下，可在该模板的基础上进行相应的修改

```
#!/bin/bash
#SBATCH --job-name=dep-test                ##作业名称
#SBATCH --partition=56c503g          ##作业申请的分区名称
#SBATCH --nodes=2                          ##作业申请的节点数
#SBATCH --ntasks-per-node=8                ##作业申请的每个节点使用的核心数
#SBATCH --mem-per-cpu=8500MB               ##作业单个CPU核心使用的内存
#SBATCH --dependency=afterok:job_id   ##当指定ID的作业正常结束（退出码为0）时开始运行。
#SBATCH --account=iei                       ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

依赖作业脚本模板

```
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++++++"

##以下几行为加载软件环境变量（注意：你在该任务里面需要用到的所有软件均需要添加到这个位置，务必根据实际情况按需添加或者删除）

#setting environment for your software          ##设置你在本作业需要用到的软件环境变量
Program excute Command                          ##GPU程序执行命令语句，每个软件的执行命令都是互不相同的，请自行查询你所需要使用到的软件程序执行的命令格式，务必根据实际情况修改。软件跑串行、单节点多和并行、多节点多核并行、多线程请参考你所用到的软件的使用说明。

echo "++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

作业脚本模板参数内容说明

作业脚本模板的参数说明如下

####指定脚本使用/bin/sh来解释执行

#!/bin/sh

####指定作业名

#SBATCH --job-name=gpu-test　　##通过--job-name 参数指定作业名为gpu-test，若不写该参数，则作业名默认跟slurm作业脚本的名称一致

####指定所要执行的分区名称

#SBATCH --partition=gpu　　##通过--partition参数指定作业所选择的分区，该参数必须要写，需要咨询管理员确认你可使用的分区名称

作业脚本模板参数内容说明

####指定使用的节点数

#SBATCH --nodes=1　　##通过--nodes参数指定作业要申请的节点数，--nodes=1表示随机启动1个节点每个节点2核心。若不写该参数，则作业默认只申请一个节点的资源来运行作业。

####指定每个节点启动的进程数

#SBATCH --ntasks-per-node=8　　##指定作业要申请的每个节点所使用的CPU核心数， --ntasks-per-node=8表示随机在每个节点内使用8个CPU核心资源。若不写该参数，则作业在每个节点内默认只使用一个CPU核心资源。

####指定单个节点每个进程使用的CPU数（线程数），一般是单节点多线程任务或者openmp任务

#SBATCH --cpus-per-task=8　##指定单节点多线程的程序使用的CPU数（线程数）

注意： #SBATCH --nodes=1 和（#SBATCH --ntasks-per-node=8 或#SBATCH --cpus-per-task=8）两个参数需要配合使用。如果是单核串行作业，则nodes参数和ntasks-per-node参数均写1。如果是单节点多线程作业，则nodes参数写1， cpus-per-task参数写线程数。如果是多节点多核并行作业，则nodes参数和ntasks-per-node根据实际情况修改，两个参数的值应都大于等于2。

作业脚本模板参数内容说明

####指定作业在使用的GPU卡总数

#SBATCH --gres=gpu:1 ##作业在每个节点申请的GPU卡数，GPU作业该参数必须要写，CPU作业该参数务必去掉。

####指定作业单个CPU核心使用的内存

#SBATCH --mem-per-cpu=7500MB

注意：单核CPU核心使用的内存需要根据不同的分区进行调整，参考下面表格里面的值

分区名称	单个CPU核心使用的参考内存容量
56c503g	8500MB
52c191g	3500MB
28c128g	4500MB
192c11t	62500MB
144c4t	27500MB
56c1t8n	17500MB

作业脚本模板参数内容说明

####指定用户所在的用户组名称（可以使用id username查看自己所在的用户组）
#SBATCH --account=iei

####指定作业的标准输出文件

#SBATCH --error=%j.err

#SBATCH --output=%j.out

作业运行后会生成两个标准文件，一个是JobID.err文件，一个是JobID.out文件。其中JobID.err为错误输出文件，通常情况下，若作业执行失败，报错信息会存储在JobID.err文件内；该文件常用于判断作业失败的原因。其中JobID.out文件为标准输出文件，若脚本内程序执行命令语句没有将结果重定向到指定文件内，则程序执行的结果的标准输出会存储在该文件内。

作业脚本模板参数内容说明

####计算slurm作业所需要用到的CPU总核心数以及GPU总卡数

```
CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

##这部分内容无需修改，是slurm自动根据前面两页的资源申请数自动计算出来的CPU总核心数以及GPU总卡数

作业脚本模板参数内容说明

```
####记录作业开始运行的时间
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++++++"

####设置程序执行所需要用到的软件环境变量

#setting environment for your software

##这部分根据实际情况修改，需要添加在作业脚本里面程序执行所需要用到的所有软件环境变量


 如下面使用intel oneapi软件，则按如下添加软件环境变量。
#setting environment for inteloneapi2022.2
source  /public/software/intel/oneapi/2022.2/setvars.sh
```

作业脚本模板参数内容说明

Program excute Command　##程序执行命令语句，每个软件的执行命令都是互不相同的，请自行查询你所需要使用到的软件程序执行的命令格式，务必根据实际情况修改。软件跑串行、单节点多和并行、多节点多核并行、多线程请参考你所用到的软件的使用说明。

如下为用intel oneapi执行一个并行输出主机名的例子。

```
mpirun -np $SLURM_NPROCS hostname

####记录作业运行结束时间，该部分内容无需修改
echo "++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date

####删除slurm软件自动生成的临时文件，该部分内容无需修改
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

作业脚本提交

写完slurm作业脚本后，如果脚本内部已经设定好相关slurm参数，可以在作业脚本路径下直接使用sbatch 作业脚本名称格式提交作业，如用户的作业脚本名称为hello.slurm，则使用sbatch hello.slurm命令提交作业

```
[iei001@ln01 General_script]$ pwd
/public/home/iei/iei001/slurm_samples/General_script
[iei001@ln01 General_script]$ ls
hello-gpu.slurm  hello.slurm
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$ sbatch hello.slurm
Submitted batch job 1095
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$
```

作业查看和取消

提交作业后可使用squeue命令查询作业当前的状态，以及使用scancel命令删除作业

```
[iei001@ln01 General_script]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1160 cpu-low-m hello-te   iei001 PD       0:00      2 (None)
              1159 cpu-low-m hello-te   iei001 PD       0:00      2 (None)
              1145 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1144 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1143 cpu-low-m hello-te   iei001 PD       0:00      2 (Resources)
              1131 cpu-low-m hello-te   iei001  R       0:09      2 cu[01-02]
              1132 cpu-low-m hello-te   iei001  R       0:09      2 cu[01-02]
              1133 cpu-low-m hello-te   iei001  R       0:09      2 cu[03-04]
              1134 cpu-low-m hello-te   iei001  R       0:09      2 cu[03-04]
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$ scancel 1133
[iei001@ln01 General_script]$
[iei001@ln01 General_script]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1160 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1159 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1145 cpu-low-m hello-te   iei001 PD       0:00      2 (Priority)
              1144 cpu-low-m hello-te   iei001 PD       0:00      2 (Resources)
              1143 cpu-low-m hello-te   iei001  R       0:00      2 cu[03-04]
              1131 cpu-low-m hello-te   iei001  R       0:17      2 cu[01-02]
              1132 cpu-low-m hello-te   iei001  R       0:17      2 cu[01-02]
              1134 cpu-low-m hello-te   iei001  R       0:17      2 cu[03-04]
[iei001@ln01 General_script]$
```

基于CPU计算的python单核串行作业脚本模板

```
#!/bin/bash
#SBATCH --job-name=python-cpu-test     ##作业名称
#SBATCH --partition=56c503g          ##作业申请的分区名称
#SBATCH --nodes=1                          ##作业申请的节点数
#SBATCH --ntasks-per-node=1              ##作业申请的每个节点使用的核心数
#SBATCH --mem-per-cpu=7500MB         ##作业单个CPU核心使用的内存
#SBATCH --account=iei                      ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

基于CPU计算的python单核串行作业脚本模板

```
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++++++"

##以下按照python在CPU单核串行计算的需求，添加所需要用到的python软件环境变量

#setting environment for python3.9.15
export PATH=/public/software/python/python-3.9.15/bin:$PATH

python3 test.py  ##该命令为python软件基于CPU跑单核串行模式的命令格式。

echo "++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

基于CPU计算的vasp多节点多核并行任务作业脚本

```
#!/bin/bash
#SBATCH --job-name=vasp-cpu-test    ##定义作业名称为vasp-cpu-test
#SBATCH --partition=56c503g       ##定义作业运行的分区为cpu
#SBATCH --nodes=2                       ##定义作业使用2个节点进行多节点并行计算
#SBATCH --ntasks-per-node=24          ##定义作业在每个节点使用24个CPU核心进行多核并行计算
#SBATCH --mem-per-cpu=7500MB       ##作业单个CPU核心使用的内存
#SBATCH --account=iei                    ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

基于CPU计算的vasp多节点多核并行任务作业脚本

```
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++"

##以下按照vasp在CPU多节点多核并行计算的需求，添加所需要用到的intel以及vasp两个软件环境变量

#setting environment for intel2020u1
#source
/public/software/intel_parallel_studio_xe/2020u1/compilers_and_libraries_2020.1.217/linux/bin/compilervars.sh intel64
#source
/public/software/intel_parallel_studio_xe/2020u1/compilers_and_libraries_2020.1.217/linux/mkl/bin/mklvars.sh intel64
#source
/public/software/intel_parallel_studio_xe/2020u1/compilers_and_libraries_2020.1.217/linux/mpi/intel64/bin/mpivars.sh

#setting environment for vasp-6.2.1-intel2020
#export PATH=/public/software/vasp/vasp-6.2.1-intel2020/bin:$PATH
```

基于CPU计算的vasp多节点多核并行任务作业脚本

mpirun -machinefile $CURDIR/nodelist.$SLURM_JOB_ID -np $SLURM_NPROCS vasp_std > ./log ##
该命令为vasp软件基于CPU跑多节点多核并行模式的命令格式，其中将程序计算结果重定向到log文件内。

```
echo "++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

基于CPU计算的gaussian单节点OpenMP作业脚本模板

```
#!/bin/bash
#SBATCH --job-name=gaussian-test       ##作业名称
#SBATCH --partition=56c503g        ##作业申请的分区名称
#SBATCH --nodes=1                          ##作业申请的节点数
#SBATCH --cpus-per-task=32                 ##作业申请的OpenMP所使用的线程数
#SBATCH --mem-per-cpu=7500MB       ##作业单个CPU核心使用的内存
#SBATCH --account=iei                      ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

基于CPU计算的gaussian单节点OpenMP作业脚本模板

```
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++++++"

##以下为gaussian软件使用方法，添加gaussian软件的临时目录

#create a scratch directory for Gaussian
GAUSS_SCRDIR=$CURDIR/tmp
mkdir -p $GAUSS_SCRDIR
export GAUSS_SCRDIR

##以下按照gaussian基于CPU单节点OpenMP计算的需求，添加所需要用到的gaussian软件环境变量

# setup environment for Gaussian
export g09root=/public/home/inspur/inspur/software/gaussian       #定义高斯09软件home目录
source $g09root/g09/bsd/g09.profile                               #定义高斯09软件的环境变量
```

基于CPU计算的gaussian单节点OpenMP作业脚本模板

##该部分命令为gaussian跑单节点OpenMP模式的命令格式。

```
#Run a Gaussian command file
echo "Starting Gaussian Program at"`date`
for i in `ls *.gjf`
do
   g09 $i
   name=`basename $i| awk -F "." '{print $1}'`
done
echo "Gaussian Program Finished at"`date`
echo "Removing scratch files ...."

echo "++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

基于GPU计算的cuda单节点单卡计算任务作业脚本

```
#!/bin/bash
#SBATCH --job-name= test-gpu-cuda    ##定义作业名称为vasp-gpu-test
#SBATCH --partition=gpu              ##定义作业运行的分区为gpu
#SBATCH --nodes=1                    ##定义作业使用1个节点进行并行计算
#SBATCH --ntasks-per-node=2          ##定义作业在每个节点使用2个CPU核心用于程序跑GPU所需要
的用到的CPU开销，程序员自行定义跑该GPU程序所需要用到的CPU资源，按需修改改参数即可。
#SBATCH --gres=gpu:1                 ##定义作业在每个节点所需要用到的GPU卡数
#SBATCH --mem-per-cpu=15000MB        ##作业单个CPU核心使用的内存
#SBATCH --account=iei                ##用户所在用户组名称
#SBATCH --error=%j.err
#SBATCH --output=%j.out

CURDIR=`pwd`
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
NODES=`scontrol show hostnames $SLURM_JOB_NODELIST`
for i in $NODES
do
echo "$i:$SLURM_NTASKS_PER_NODE" >> $CURDIR/nodelist.$SLURM_JOB_ID
done
```

基于GPU计算的cuda单节点单卡计算任务作业脚本

```
echo "process will start at : "
date
echo "++++++++++++++++++++++++++++++++++++++++++++"

##以下按照cuda单机单卡计算的需求，直接进入到batchCUBLAS可执行程序所在的路径下执行batchCUBLAS程序

cd /public/software/cuda_samples/NVIDIA_CUDA-11.4_Samples/7_CUDALibraries/batchCUBLAS
./batchCUBLAS -m8192 -n8192 -k8192
./batchCUBLAS -m8192 -n8192 -k8192
./batchCUBLAS -m8192 -n8192 -k8192
./batchCUBLAS -m8192 -n8192 -k8192

echo "++++++++++++++++++++++++++++++++++++++++++++"
echo "process will sleep 30s"
sleep 30
echo "process end at : "
date
rm -rf $CURDIR/nodelist.$SLURM_JOB_ID
```

目录

salloc交互式作业提交

salloc提交交互式作业

通过salloc命令申请CPU资源，ssh到对应的计算节点，使用申请的资源运行程序，（切记，salloc运行程序不能关闭当前窗口）

```
[iei001@ln01 ~]$ salloc -N 1 --ntasks-per-node=64 -p cpu-high-mem --mem-per-cpu=15000MB --account=iei
salloc: Granted job allocation 1096
[iei001@ln01 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1096 cpu-high- interact   iei001  R       0:02      1 cu14
[iei001@ln01 ~]$ ssh cu14
iei001@cu14's password:
[iei001@cu14 ~]$ cd slurm_samples/job-dir/
[iei001@cu14 job-dir]$ hostname
cu14
[iei001@cu14 job-dir]$exit
logou
Connection to cu14 closed.
[iei001@ln01 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1096 cpu-high- interact   iei001  R       1:40      1 cu14
[iei001@ln01 ~]$ exit
exit
salloc: Relinquishing job allocation 1096
[iei001@ln01 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
[iei001@ln01 ~]$
```

srun交互式作业提交

srun提交交互式作业命令如下：

srun运行多进程并行任务

```
[iei001@ln01 ~]$ srun -N 1 --ntasks-per-node=8 -p cpu-low-mem --mem-per-cpu=7500MB --account=iei hostname
cu01
cu01
cu01
cu01
cu01
cu01
cu01
cu01
[iei001@ln01 ~]$
```

srun运行多openmp或者多线程任务

```
[iei001@ln01 ~]$ srun -N 1 --cpus-per-task=8 -p cpu-low-mem --mem-per-cpu=7500MB --account=iei hostname
cu01
[iei001@ln01 ~]$
```

-N, --nodes=<nodenum>：申请执行作业节点数，

--ntasks-per-node=1，一个节点使用几个进程

--cpus-per-task=1：一个进程使用几个CPU核

目录

常见问题解答

- **error: Job submit/allocate failed: Invalid partition name specified**

答：作业脚本未指定正确的分区名称

- **batch job submission failed: Requested node configuration is not available**

答:申请资源的节点配置不匹配，通常是作业申请的单个主机的CPU核心数超过该主机的总核心数

- **error: Unable to allocate resources: Unspecified error**

答：通常是因为未指定account或者是用户余额不足，请先充值。

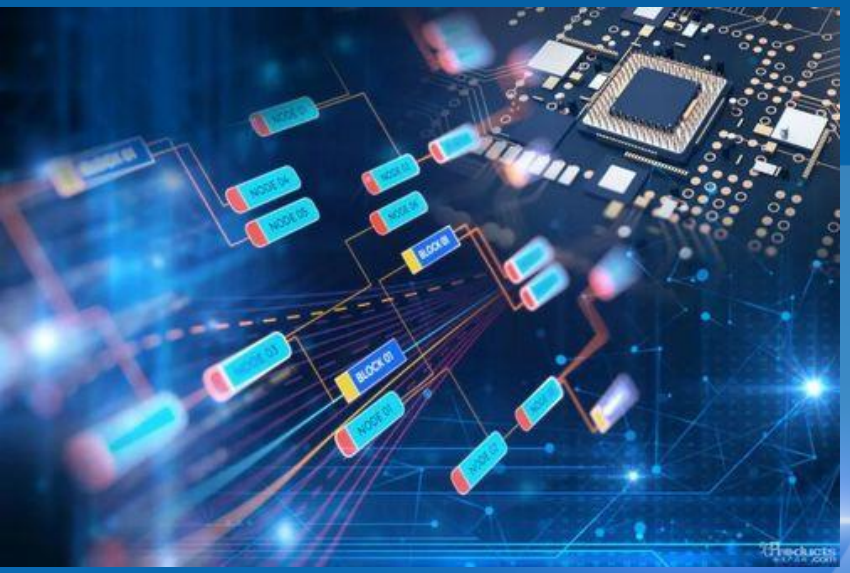
常见问题解答

- **作业pending错误QOSMaxCpuPerUserLimit：**

答：作业申请的CPU资源超过qos允许用户使用最大资源数量。

- **sbatch job.slurm 提交任务后无任何输出:**

答：通过sbatch 提交任务后，没有生成 JobID.err及JobID.out文件，一般为节点共享存储异常导致，请联系管理员。



谢谢！