高性能计算集群用户培训手册v1.3

目录

1 集群概况、集群登录及文件传输

2 slurm调度系统简介

3 slurm常用命令操作介绍

4 slurm脚本作业

5 交互式作业

6 常见问题

集群概况

浪潮高性能集群目前有1个管理登录节点，28个计算节点。

28个计算节分别配置了15个瘦计算节点,9个GPU节点和4个胖节点，分成三个分区，分别是cpu、gpu、fat分区。

登录节点是用户使用集群的唯一一个入口，用户的所有操作（代码及数据上传，作业脚本编写）均在此节点完成。

cpu分区配置了15个瘦计算节点，适用于纯CPU计算类型的应用程序。

gpu分区配置了9个GPU计算节点，适用于CPU+GPU异构计算，深度学习模块的训练。

集群概况

fat分区配置了4个胖计算节点，适用于消耗内存较大的应用程序。

集群存储采用宏杉存储，挂载到所有节点的home目录，home目录为共享目录。

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/home/software目录下，用户可直接调用/home/software目录下的所有软件，/home/software目录下的软件环境变量添加方法参考/home/sourcecode/env_file文件调用即可。

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该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。

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，上传文件，比如userA用户将文件上传到自己的家目录/home/userA/下。

3、查看软件安装路径（如/home/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，若没有，可以按需将软件安装到自己的家目录下。

4、在/home/userA/目录下编写一个创建一个程序执行目录，比如创建slurm_job，将程序执行需要用到的源数据，代码等文件放置slurm_job目录下，并且slurm_job目录下创建slurm作业脚本（该脚本就是一个普通文本文件），比如自定义命名为vasp.slurm 。脚本具体编写方法参考本手册的第4章节《slurm脚本作业》。slurm作业脚本模板可参考/home/sourcecode/slurm_sample/目录下文件。

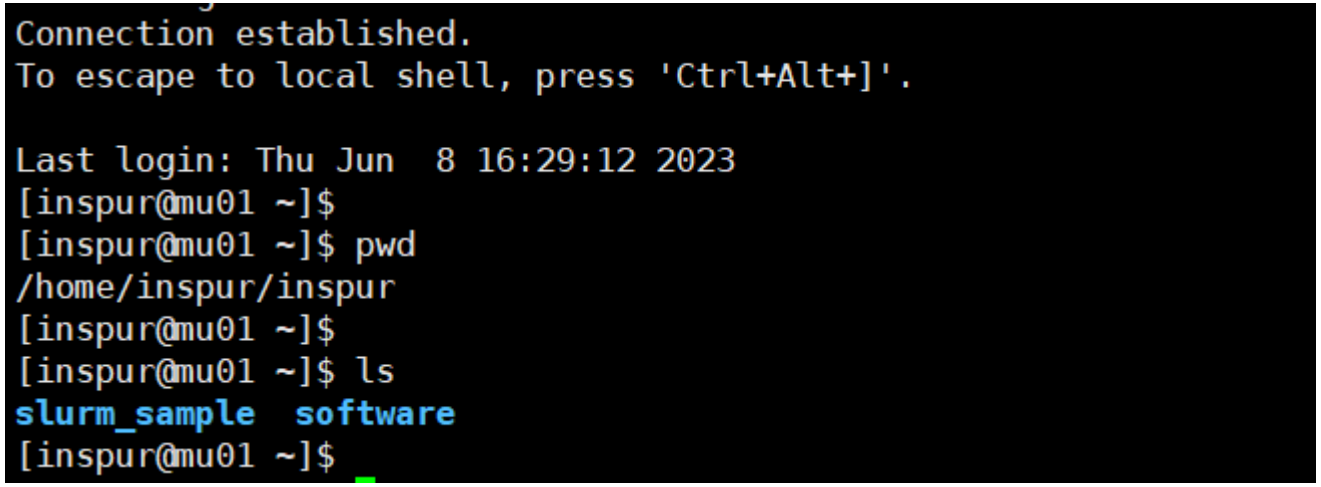
5、在slurm作业脚本vasp.slurm的路径上执行sbatch vasp.slurm命令提交作业。

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。

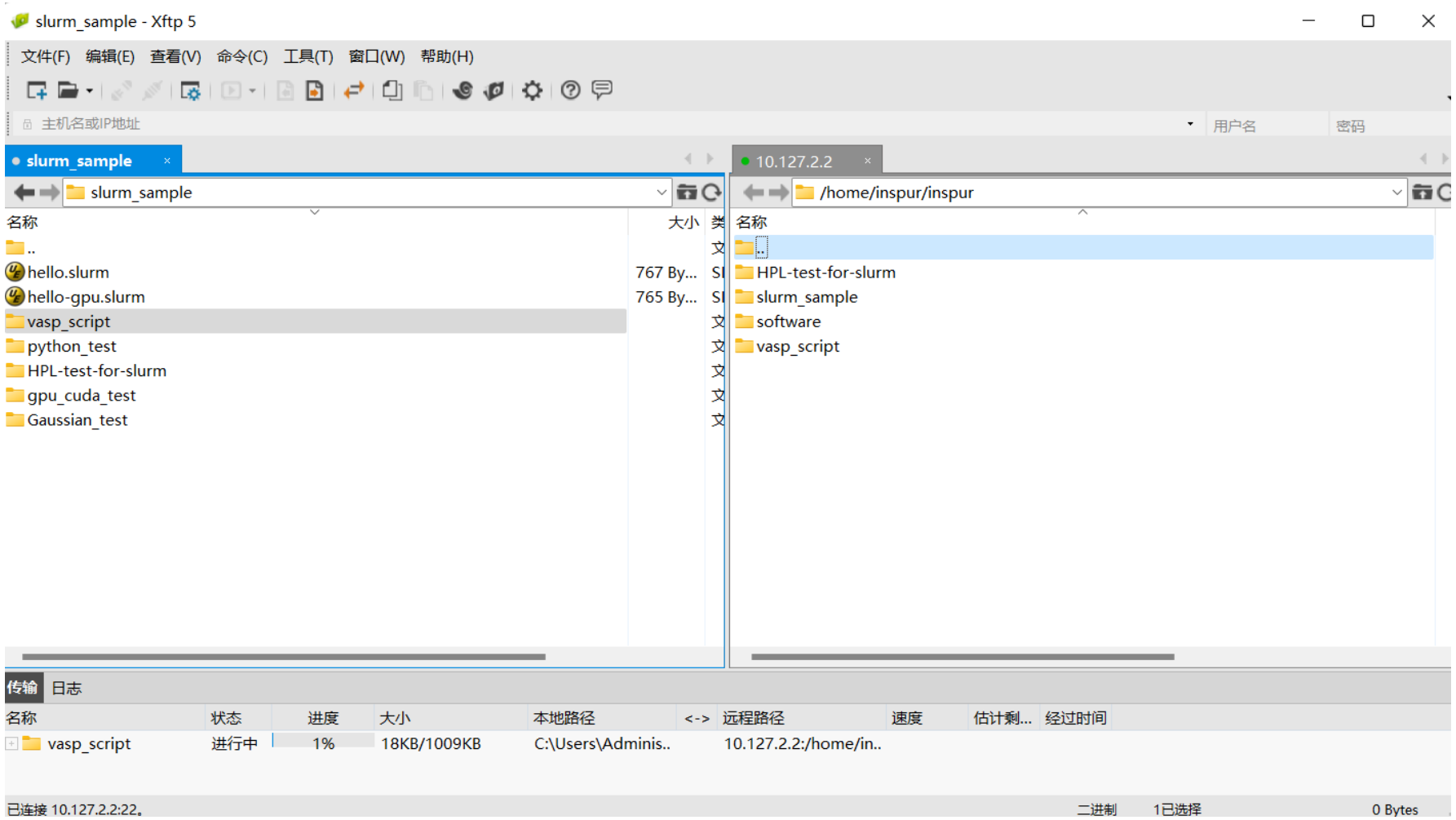
集群登录

集群基于CentOS系统搭建，用户可通过xshell，putty等常用终端工具登录集群的shell界面。shell登录界面如下



文件传输

文件传输如下，以xftp例



目录

1 集群概况、集群登录及文件传输

2 slurm调度系统简介

3 slurm常用命令操作介绍

4 slurm脚本作业

5 交互式作业

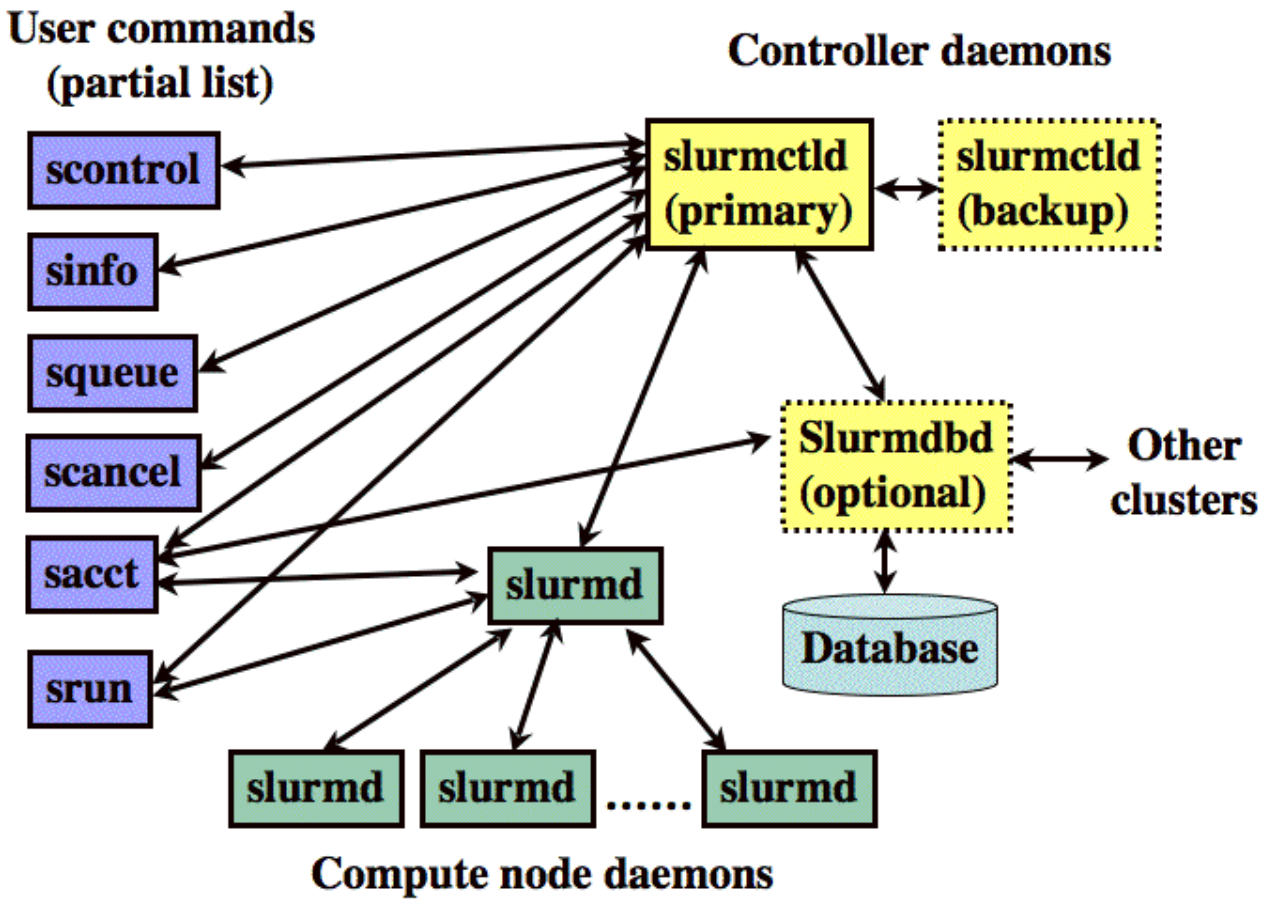
6 常见问题

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。

目前国防科大，中南大学，天河二号等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
- 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

1 集群概况、集群登录及文件传输

2 slurm调度系统简介

3 slurm常用命令操作介绍

4 slurm脚本作业

5 交互式作业

6 常见问题

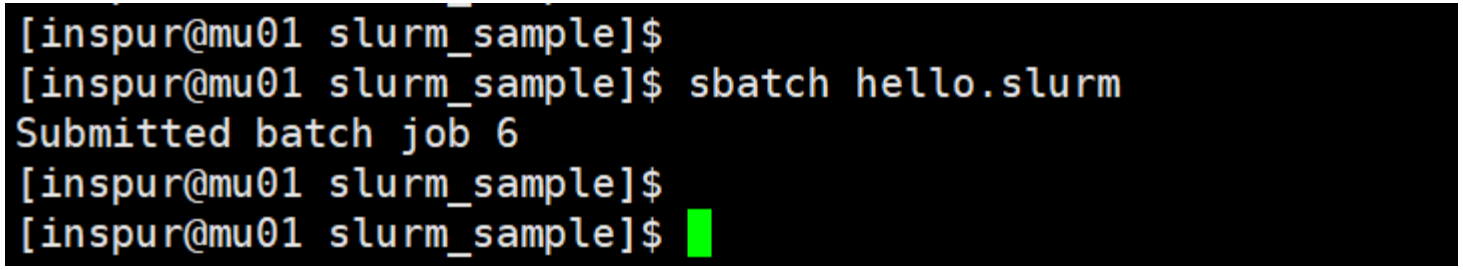
slurm常用命令操作介绍

命令	作用
sinfo	查看有关slurm节点和分区的信息
scontrol show partition	查看详细分区信息
scontrol show node	查看详细节点信息
scontrol show job	查看作业详细信息
sbatch	批处理方式提交作业
srun	交互式提交并行作业
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

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。



salloc

salloc将获取作业的分配后执行命令，当命令结束后释放分配的资源（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.执行用户指定的命令；4.命令执行结束，释放资源

```
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ salloc -N 1 --ntasks-per-node=52  -p cpu
salloc: Granted job allocation 7
salloc: Waiting for resource configuration
salloc: Nodes cu01 are ready for job
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ ssh cu01
inspur@cu01's password:
Last login: Tue Jun  6 21:02:29 2023 from mu01
[inspur@cu01 ~]$
[inspur@cu01 ~]$ hostname
cu01
[inspur@cu01 ~]$ exit
logout
Connection to cu01 closed.
[inspur@mu01 slurm_sample]$ exit
exit
salloc: Relinquishing job allocation 7
[inspur@mu01 slurm_sample]$
```

资源分配请求成功后会进入 bash shell 终端。在终端输入 exit 命令或 Ctrl+D 退出分配模式

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。

```
[inspur@mu01 slurm_sample]$ srun -N 2 -p cpu --ntasks-per-node=4 hostname
cu01
cu01
cu01
cu01
cu02
cu02
cu02
cu02
```

srun、sbatch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
- --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>
- -o，--output=file, 标准输出到指导的file文件中
- -e，--error=file，错误流输出到指定的文件中
- -p, --partition=<partition_names>：将任务提交到指导的partition_names分区中
- --exclusive：独占模式
- --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。
- -q, --qos=<qos>：指定作业使用的qos

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

```
Submitted batch job 19
[inspur@mu01 slurm_sample]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                10       cpu     test   inspur  R       0:02      2 cu[01-02]
                11       cpu     test   inspur  R       0:02      2 cu[01-02]
                12       cpu     test   inspur  R       0:02      2 cu[01-02]
                13       cpu     test   inspur  R       0:02      2 cu[01-02]
                14       cpu     test   inspur  R       0:02      2 cu[01-02]
                15       cpu     test   inspur  R       0:02      2 cu[01-02]
                16       cpu     test   inspur  R       0:02      2 cu[03-04]
                17       cpu     test   inspur  R       0:02      2 cu[03-04]
                18       cpu     test   inspur  R       0:02      2 cu[03-04]
                19       cpu     test   inspur  R       0:02      2 cu[03-04]
                 5       gpu     bash   inspur  R      13:13      1 gpu08
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$
```

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35的作业进行删除操作

```
[inspur@mu01 slurm_sample]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                10       cpu     test   inspur  R       0:24      2 cu[01-02]
                11       cpu     test   inspur  R       0:24      2 cu[01-02]
                12       cpu     test   inspur  R       0:24      2 cu[01-02]
                13       cpu     test   inspur  R       0:24      2 cu[01-02]
                14       cpu     test   inspur  R       0:24      2 cu[01-02]
                15       cpu     test   inspur  R       0:24      2 cu[01-02]
                16       cpu     test   inspur  R       0:24      2 cu[03-04]
                17       cpu     test   inspur  R       0:24      2 cu[03-04]
                18       cpu     test   inspur  R       0:24      2 cu[03-04]
                19       cpu     test   inspur  R       0:24      2 cu[03-04]
                 5       gpu     bash   inspur  R      13:35      1 gpu08
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ scancel 10
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                11       cpu     test   inspur  R       0:29      2 cu[01-02]
                12       cpu     test   inspur  R       0:29      2 cu[01-02]
                13       cpu     test   inspur  R       0:29      2 cu[01-02]
                14       cpu     test   inspur  R       0:29      2 cu[01-02]
                15       cpu     test   inspur  R       0:29      2 cu[01-02]
                16       cpu     test   inspur  R       0:29      2 cu[03-04]
                17       cpu     test   inspur  R       0:29      2 cu[03-04]
                18       cpu     test   inspur  R       0:29      2 cu[03-04]
                19       cpu     test   inspur  R       0:29      2 cu[03-04]
                 5       gpu     bash   inspur  R      13:40      1 gpu08
```

目录

1 集群概况、集群登录及文件传输

2 slurm调度系统简介

3 slurm常用命令操作介绍

4 slurm脚本作业

5 交互式作业

6 常见问题

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。脚本作业使用的具体流程在前面也介绍过，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=cpu-test        ##作业名称
#SBATCH --partition=cpu            ##作业申请的分区名称
#SBATCH --nodes=2                  ##作业申请的节点数
#SBATCH --ntasks-per-node=4        ##作业申请的每个节点使用的核心数
#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          ##CPU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=gpu            ##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 --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
```

通用脚本模板内容说明

CPU与GPU的脚本模板差异在于GPU脚本多了#SBATCH --gpus=1和 echo $SLURM_GPUS 两行参数，因此该部分内容以GPU脚本模板为例子做详细说明

####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=cpu　##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

通用脚本模板内容说明

####指定使用的节点数以及每个节点使用的CPU核心数

#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，则作业默认只使用一个来运行。

#SBATCH --ntasks-per-node=8 ##通过--ntasks-per-node参数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。

注意： #SBATCH --nodes=1 和#SBATCH --ntasks-per-node=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，ntasks-per-node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。

通用脚本模板内容说明

####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作业改参数必须不能写。

####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。

通用脚本模板内容说明

####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

通用脚本模板内容说明

```
####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 /home/software/inteloneapi/2022.2/setvars.sh
```

通用脚本模板内容说明

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。

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

```
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ sbatch hello.slurm
Submitted batch job 6
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$
```

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

```
[inspur@mu01 slurm_sample]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                10       cpu     test   inspur  R       0:24      2 cu[01-02]
                11       cpu     test   inspur  R       0:24      2 cu[01-02]
                12       cpu     test   inspur  R       0:24      2 cu[01-02]
                13       cpu     test   inspur  R       0:24      2 cu[01-02]
                14       cpu     test   inspur  R       0:24      2 cu[01-02]
                15       cpu     test   inspur  R       0:24      2 cu[01-02]
                16       cpu     test   inspur  R       0:24      2 cu[03-04]
                17       cpu     test   inspur  R       0:24      2 cu[03-04]
                18       cpu     test   inspur  R       0:24      2 cu[03-04]
                19       cpu     test   inspur  R       0:24      2 cu[03-04]
                 5       gpu     bash   inspur  R      13:35      1 gpu08
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ scancel 10
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                11       cpu     test   inspur  R       0:29      2 cu[01-02]
                12       cpu     test   inspur  R       0:29      2 cu[01-02]
                13       cpu     test   inspur  R       0:29      2 cu[01-02]
                14       cpu     test   inspur  R       0:29      2 cu[01-02]
                15       cpu     test   inspur  R       0:29      2 cu[01-02]
                16       cpu     test   inspur  R       0:29      2 cu[03-04]
                17       cpu     test   inspur  R       0:29      2 cu[03-04]
                18       cpu     test   inspur  R       0:29      2 cu[03-04]
                19       cpu     test   inspur  R       0:29      2 cu[03-04]
                 5       gpu     bash   inspur  R      13:40      1 gpu08
[inspur@mu01 slurm_sample]$
```

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=cpu               ##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 --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=/home/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=cpu         ##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 --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
/home/software/intel_parallel_studio_xe/2020u1/compilers_and_libraries_2020.1.217/linux/bin/compilervars.sh intel64
#source
/home/software/intel_parallel_studio_xe/2020u1/compilers_and_libraries_2020.1.217/linux/mkl/bin/mklvars.sh intel64
#source
/home/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=/home/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=cpu                 ##作业申请的分区名称
#SBATCH --nodes=1                       ##作业申请的节点数
#SBATCH --ntasks-per-node=32            ##作业申请的OpenMP所使用的线程数
#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=/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16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 --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 /home/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

1 集群概况、集群登录及文件传输

2 slurm调度系统简介

3 slurm常用命令操作介绍

4 slurm脚本作业

5 交互式作业

6 常见问题

salloc交互式作业提交

salloc提交交互式作业

作业salloc提交交互式作业必须要独占节点，否则会遇到无法ssh登录到分配的节点的情况出现，提交命令如下 salloc -N 2 -p

```
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ salloc -N 1 --ntasks-per-node=52  -p cpu
salloc: Granted job allocation 7
salloc: Waiting for resource configuration
salloc: Nodes cu01 are ready for job
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ ssh cu01
inspur@cu01's password:
Last login: Tue Jun  6 21:02:29 2023 from mu01
[inspur@cu01 ~]$
[inspur@cu01 ~]$ hostname
cu01
[inspur@cu01 ~]$ exit
logout
Connection to cu01 closed.
[inspur@mu01 slurm_sample]$ exit
exit
salloc: Relinquishing job allocation 7
[inspur@mu01 slurm_sample]$
[inspur@mu01 slurm_sample]$ squeo
```

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

```
[inspur@mu01 ~]$
[inspur@mu01 ~]$ srun -N 2 --ntasks-per-node=4 -p cpu hostname
cu02
cu02
cu02
cu02
cu01
cu01
cu01
cu01
[inspur@mu01 ~]$
```

srun运行多openmp或者多线程任务

```
[inspur@mu01 ~]$ srun -N 1 --cpus-per-task=4 -p fat hostname
fat01
[inspur@mu01 ~]$
```

-N, --nodes=<nodenum>：申请执行作业节点数，

-n , --ntasks 作业总的进程数

--ntasks-per-node=1，一个节点使用几个进程

--cpu-per-task=NCPUs：一个进程使用几个CPU核

目录

1 集群概况、集群登录及文件传输

2 slurm调度系统简介

3 slurm常用命令操作介绍

4 slurm脚本作业

5 交互式作业

6 常见问题

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

- **Batch job submission failed: Job violates accounting/QOS policy (job submit limit, user's size and/or time limits)**

答：通常是因为用户余额不足，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。

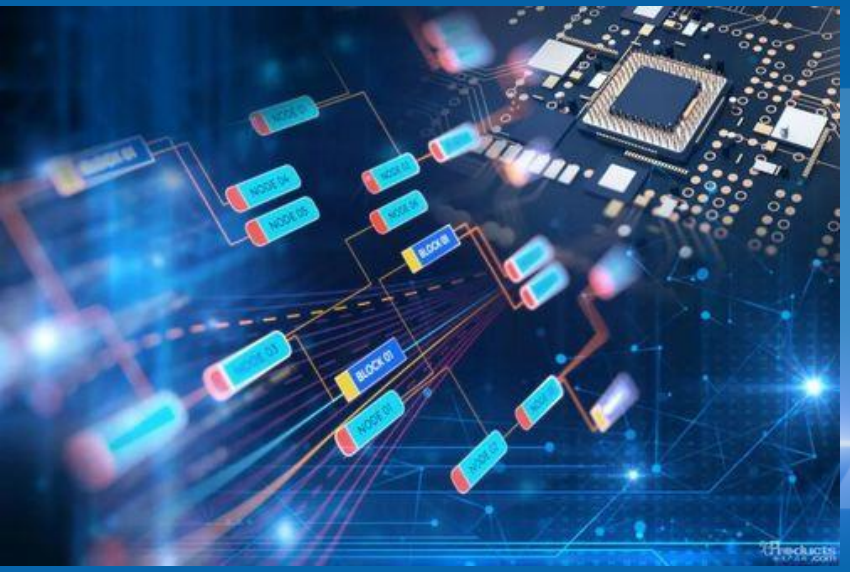
- **sbatch job.slurm 提交任务后无任何输出:**

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。

账号申请流程

- **有意向使用超算集群的老师或者同学，可通过发送邮件到以下邮箱进行账号开通申请**

hpcc@ncu.edu.cn



谢谢！

COMPUTING INSPIRES FUTURE