北京脑中心高性能集群使用手册

2024 年 05 月 22 日

第一章 现有集群

北脑 1 号

平台海量数据处理与存储集群采用 GPU+CPU 异构系统，共 3368 处理器，拥有 70 个计算节点，18 台胖节点，7 台 GPU 节点，拥有近 400 万亿次每秒的计算能力， 4.2P 可用容量的高性能存储系统，实现高达 40GB/s 的实测聚合读写带宽，来满足脑科学领域的数据量巨大、高带宽、高 IOPS 等应用，为保证集群高效及稳定运行，计算中心还制定了资源 quota 限制策略，机房环境,硬件及软件故障实时报警.

节点类别	队列	节点数	处理器规格	单节点核心数	单节点内存数	qos	GPU数	算力
cpu普通节点	**q_cn_2**	**10**	**2* Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz**	**54核（申请一核，可用4.3G内存）**	**256G**	**high**	**/**	
cpu普通节点	q_cn	60	2*Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz	36 核 (申请一核，可用 4.9G 内存)	192G	high	/	0.4PF lops
四路胖节点	q_fat_2	4	4*Intel(R) Xeon(R) Gold 6328H CPU @ 2.80GHz	64 核 (申请一核，可用 46G 内存)	3072G	high	/	
四路胖节点	q_fat	2	4*Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz	72 核 (申请一核，可用 40G 内存)	3072G	high	/	
并入公共资源池课题组队列	q_fat_c	8	4*Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz	72 核(申请一核，可用 20G 内存)	1536G	high/high_c	/	
	q_fat_l	3	4*Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz	72 核(申请一核，可用 20G 内存)	1536G	high/high_l	/	
	q_fat_z	1	4*Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz	56 核(申请一核，可用 25600M 内存)	1536G	high/课题组 high_z	/	

	q_gpu_c	1	2* AMD EPYC 9754 128-Core Processor	256 核（申请一核，可用 5.7G 内存）	1536G	High	8*NVIDIA RTX 6000 Ada	
GPU胖节点	q_ai8	2	2*Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz	36 核 （申请一核，可用 27G 内存）	1024G	high	8*NVIDIA Tesla V100 32GB	
GPU节点	q_ai4	4	2*Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz	36 核 （申请一核，可用 14G 内存）	512G	high	4*NVIDIA Tesla V100 32GB	
存储可用容量合计：4.2P ， 聚合读写带宽：40GB/s								

第二章 集群登录

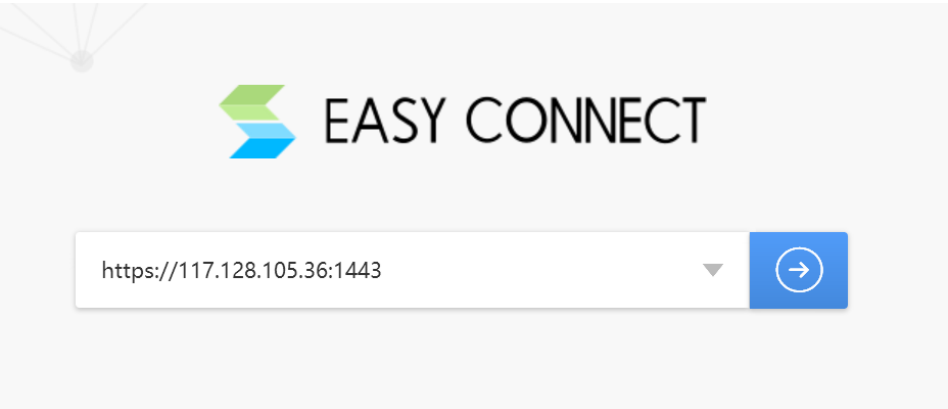
1.1 VPN 登录

1.1.1vpn 下载

打开 IE 浏览器或者 safari 浏览器访问 https://117.128.105.36:1443 或者 https://bbsi.cibr.ac.cn:1443 打开网页后会提示安全警告信息，点击“详细信息”然后再点击“转到此网页”，最后会显示 VPN 登陆窗口，输入 VPN 账号密码后点击登录。

第一次登录会下载安装 Easyconnect 客户端，后续登录用该客户端登录即可，不需再从网页登录。

打开 Easyconnect 客户端输入访问地址



登录集群的 vpn 账号，密码 然后输入手机接收到的的验证码

第一次是管理员设置好的手机号码，如果使用期间想更换手机号登录成功之后可自行修改

1.1.2 vpn 支持的版本

windows 操作系统	Linux 操作系统	Mac	移动终端
Windows XP SP3	Ubuntu 12.04.5（32、64 位）	MacOS 10.9-11.0	Android4.x-10.x
windows 7 专业版/旗舰版 32 位	Ubuntu 14.04.5（32、64 位）		iOS8.x-13.x
Windows 7 专业版/旗舰版 64 位	Ubuntu 16.04.6（32、64 位）		
Windows 8（Windows8.1）32 位和 64 位	Ubuntu 17.04（32、64 位）		
Windows 10 32 位和 64 位	Ubuntu 18.04.1（32、64 位）		
	中标麒麟 v6.0（32、64 位）		
	中标麒麟 v7.0（64 位）		

1.2 主机登录

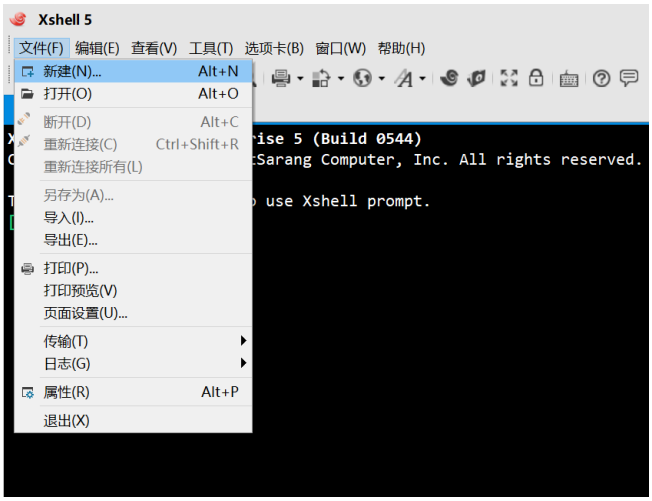
1.2.1 集群 IP 地址

集群 IP 地址为：10.12.100.88，用户通过该地址可以登录到集群的登录节点。登录节点主要用于文件上传下载、程序编写、软件安装以及作业提交等操作，**登录节点不能运行程序（需要在登录节点用 slurm 去调度）**，否则将会影响到其他用户的登录及操作。

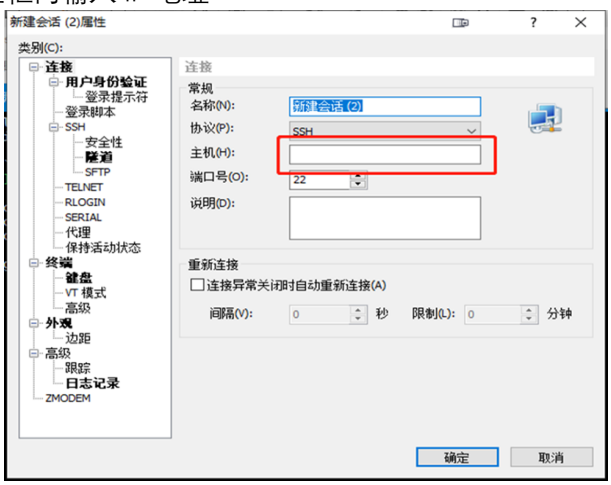
1.2.2 Windows 用户主机登录

Windows 用户可以用 MobaXterm，Xshell，SSH Secure Shell Client，PuTTY，SecureCRT 等 SSH 客户端软件登录集群。下面以 xshell 为例介绍如何登录。xshell 是付费商业软件，但有免费的教育家庭版可以下载。

1) 打开 xshell，点击“文件”中的“新建会话”

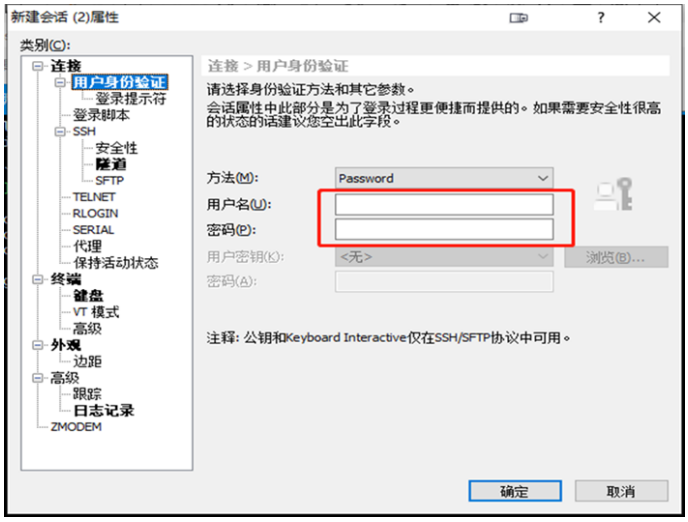


2) 编辑会话，在红框内输入 IP 地址

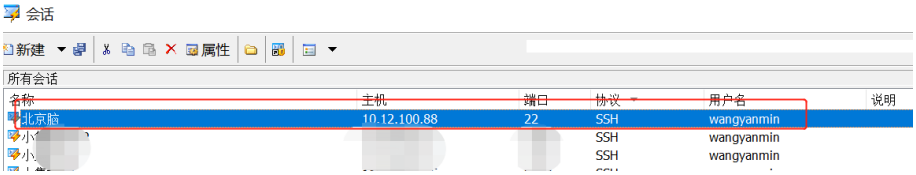


3) 输入集群主机账号和密码

点击“用户验证”，输入主机账号和密码，然后点击“确定”完成会话新建工作。



4) 选择刚创建的会话，然后点击“连接”登录集群



5) 首次登陆时会弹出窗口，询问是否保存密钥，选择“接受保持”后即可登录集群。

1.2.3 Linux、Mac 用户主机登录

直接在命令行终端直接在命令行终端中执行 ssh 命令进行登录:

```
$ ssh username@10.12.100.88
```

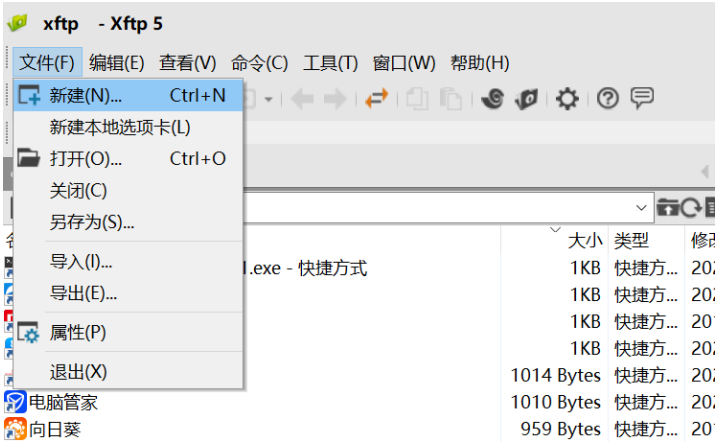
1.3 文件上传下载

1.3.1 Windows 用户文件上传下载

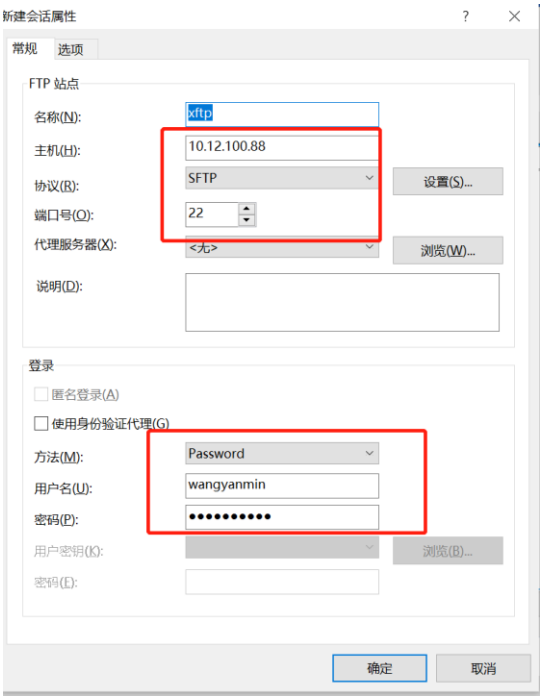
Windows 用户可以用 MobaXterm，Xftp, SSH Secure Shell Client，winscp 等软件实现文件的上传下载。下面以 Xftp 为例，介绍文件上传和下载使用方法。xshell 是商业付费商业软件，但有免费的教育家庭版可以下载。

1) 新建会话

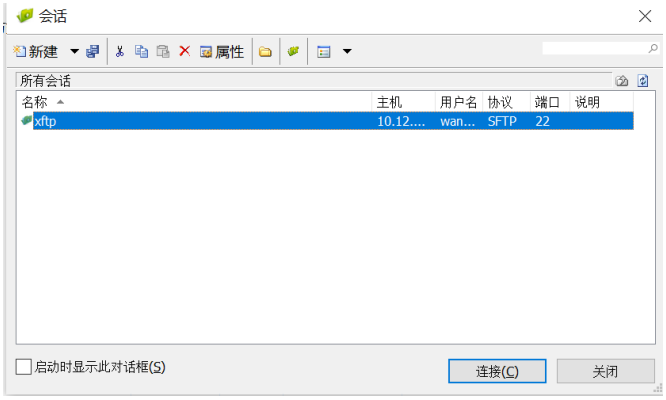
打开 xftp，点击“文件”中的“新建”。



2) 编辑回话，输入 IP、账号及密码

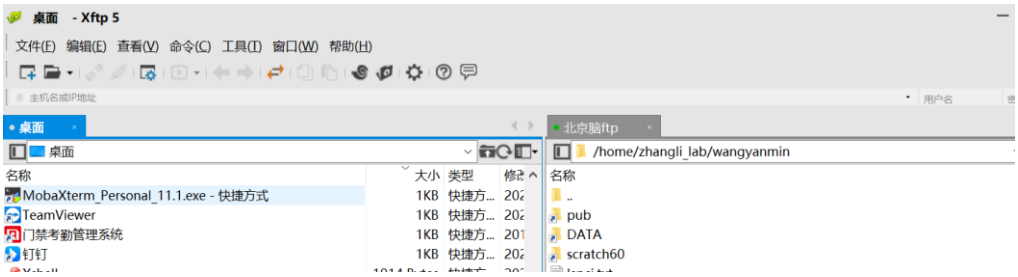


3) 选择创建完成的会话，然后点击“连接”登录集群



4) 文件上传下载

集群登录成功之后，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。



1.3.2 Linux、Mac 用户文件上传下载

Linux、Mac 用户可直接使用命令进行文件上传下载。文件都需上传到 DATA 目录下。

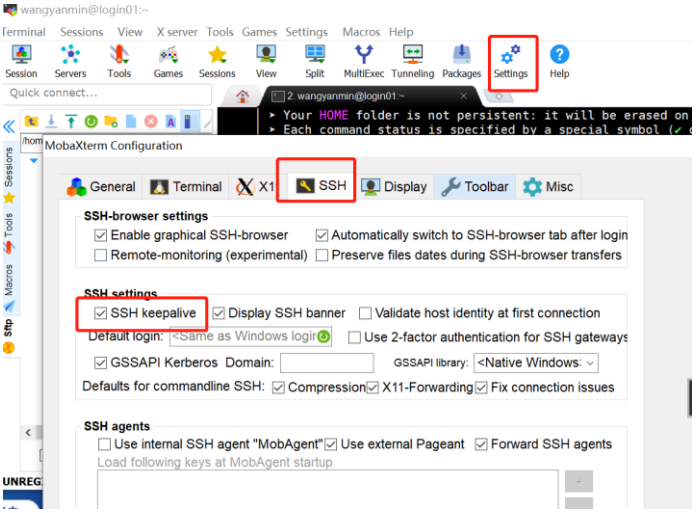
```
scp  filename  username@10.12.100.88:~/DATA
```

1.4 图形转发

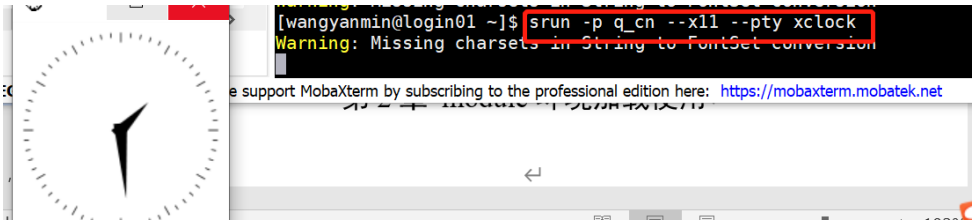
1.4.1 Window 用户图形转发

Windows 用户可以用 MobaXterm（推荐），Xshell+Xmanager(商业版)， putty+xming 等软件实现软件的图形转发。下面以 MobaXterm 为例，介绍图形转发的使用方法。

1） 设置终端保持在线



2）登录集群运行测试程序，跳出图形界面



1.4.2 Mac 用户图形转发

Mac 用户需要单独下载 xquartz　X11 终端程序

1） 修改配置文件

$ sudo vim /etc/ssh/sshd_config

#X11Forwarding no　把#去掉 no　改成 yes

2） 重启 sshd 服务

停止>$ sudo launchctl unload -w /System/Library/LaunchDaemons/ssh.plist

启动>$ sudo launchctl load -w /System/Library/LaunchDaemons/ssh.plist

查看是否启动>$ sudo launchctl list | grep ssh

3）登录终端

$ ssh -Y user@10.12.100.88

4）运行测试

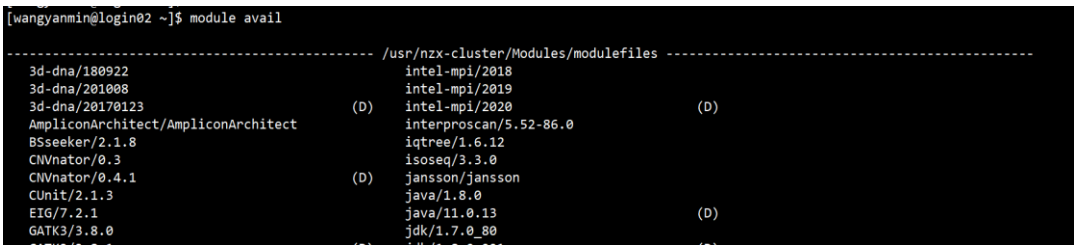
$ srun -p q_cn --x11 --pty xclock

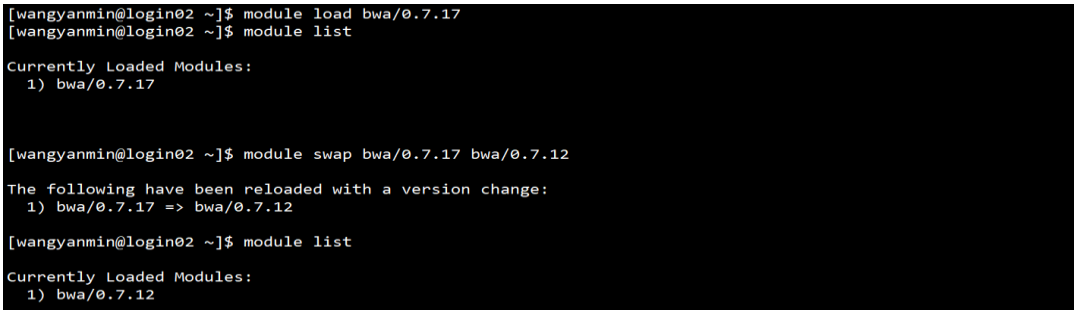
第三章 module 环境加载

集群已经安装了部分常用软件，这些软件通过 Module 来管理和使用

使用 module 命令可在同一软件的不同版本之间切换，也可以在同一功能的不同软件之间切换，以此来选择最合适的编程环境和运行环境。

module avail	查看所有 module 管理的软件
module load bwa/0.7.17	加载相应版本的软件
	如果写到~/.bashrc，重新登录的终端都会自动加载相应的软件
	如果写到 sbatch 的作业提交脚本中，只会在脚本中生效，脚本之外的 shell 环境无效
module list	显示目前已经加载的软件
module swap bwa/0.7.17 bwa/0.7.12	切换软件版本
module unload bwa/0.7.17	卸载相应版本的软件
module spider bwa	搜索模块的完整列表
module purge	将已经加载的软件全部清除





```
[wangyanmin@login02 ~]$ module spider bwa

----------------------------------------------------------------------------------------------------
  bwa:
----------------------------------------------------------------------------------------------------
    Description:
      Lmod: An Environment Module System

     Versions:
        bwa/0.7.12
        bwa/0.7.17
     Other possible modules matches:
```

第四章 作业提交

slurm 作业调度系统分为 srun 、sbatch 、salloc 3 种作业提交方式

作业提交方式	使用方式	优点	缺点	试用场景
srun 交互式提交	srun+资源申请+程序运行命令 srun -J test -p q_cn -c 1 python hello.py	快捷简单 程序输出直接打印到屏幕，便于观察程序的运行日志和错误信息	终端与集群断开连接，作业将会中断	前期作业调试
sbatch 批处理式提交	作业提交参数写在脚本 run.slurm，执行 sbatch run.slurm 提交作业	计算稳定，作业交由计算节点控制，与终端状态无关 批量式提交	需要写几行脚本，略繁琐	正式计算
salloc 分配式提交	salloc+资源申请 sallloc -J test -p q_cn -c 1	持续占用节点，不用重复排队(不退出会一直计费) 实时从屏幕看到程序的输出	终端与集群断开连接，作业将会中断	大量同规模的作业要提交但不想重复排队

4.1 单线程提交作业

srun 交互式提交命令

程序输出直接打印到屏幕，便于观察程序的运行日志和错误信息

先介绍一个简单例子:

我们在计算过程中运行 hostname 命令提交申请资源 1 个任务 1 个核，那么使用 srun 提交命令:

```
srun -J hostname -p q_cn -o job.%j.out -n 1 hostname
```

sbatch 批处理式提交 ，脚本名为 hostname.sh(日常推荐)

```
#!/bin/bash
#SBATCH -J hostname
#SBATCH -p q_cn
#SBATCH -o job.%j.out
#SBATCH -n 1

hostname
```

编辑完脚本下面就可以直接提交到计算节点上运行了

```
sbatch hostname.sh
```

salloc 分配式提交

```
salloc -p q_cn -n 1

srun -n 1 -o job.%j.out    hostname     #仍然需要 srun 去提交，不需要指定分区，不需要在排队
```

例子中涉及的参数:

```
-J hostname        #hostname 为提交作业的名称,自定义
-p q_cn            #作业提交的指定分区为 q_cn 队列;
-o job.%j.out      #脚本执行的输出将被保存在 job.%j.out 文件里，%j 表示作业号;
-n   1              #每个节点上运行一个任务（进程）
```

4.2 多线程提交（采用 OpenMP 编程的并行程序）

以下以 sbatch 提交方式为例

我们在计算过程中运行 multithread 命令,启动 1 个任务（进程）,36 核心，那么使用 sbatch 提交命令（**脚本名为 multithread.sh**）：

```
#!/bin/bash
#SBATCH -J multithread
#SBATCH -o job.%j.out
#SBATCH -p q_cn
#SBATCH -n 1
#SBATCH -c 36

module load anaconda3/4.8.2       #module 加载下需要的软件

./multithread
```

编辑完脚本下面就可以直接提交到计算节点上运行了

```
sbatch multithread.sh
```

例子中涉及的参数:

```
-J multithread   # multithread 为提交作业的名称,自定义
-p q_cn          #作业提交的指定分区为 q_cn 队列;
-o job.%j.out    #脚本执行的输出将被保存在 job.%j.out 文件里，%j 表示作业号;
-n   1           #每个节点上运行一个任务(进程)
-c   36          #每个进程使用 36 个核心
```

4.3 多进程提交（采用 MPI 编程的并行程序）

以下以 sbatch 提交方式为例

我们在计算过程中运行 multiprocess 命令,启动 100 个任务（进程），那么使用 sbatch 提交命令（**脚本名为 multiprocess.sh**）：

```
#!/bin/bash
#SBATCH -J multiprocess
#SBATCH -o job.%j.out
#SBATCH -p q_cn
#SBATCH -n 100

module load anaconda3/4.8.2       #module 加载下需要的软件

srun -n 100 ./multiprocess
```

编辑完脚本下面就可以直接提交到计算节点上运行了

```
sbatch multiprocess.sh
```

例子中涉及的参数:

```
-J multiprocess   # multithread 为提交作业的名称,自定义
-p q_cn           #作业提交的指定分区为 q_cn 队列;
-o job.%j.out     #脚本执行的输出将被保存在 job.%j.out 文件里，%j 表示作业号;
-n  1             #每个节点上运行一个任务(进程)
-c  36            #每个进程使用 36 个核心
```

4.4 多进程+多线程（采用 MPI+OpenMP 编程的并行程序）

以下以 sbatch 提交方式为例

我们在计算过程中运行 hybrid-pro-thr 命令,申请资源 2 个节点，每个节点运行一个进程，每个进程运行 36 核心，那么使用 sbatch 提交命令（**脚本名为 hybrid-pro-thr.sh**）：

```
#!/bin/bash
#SBATCH -J hybrid-pro-thr
#SBATCH -o job.%j.out
#SBATCH -p q_cn
#SBATCH -N 2
#SBATCH --ntasks-per-node=1
#SBATCH -c 36

module load anaconda3/4.8.2       #module 加载下需要的软件

srun -n 2 ./hybrid-pro-thr
```

编辑完脚本下面就可以直接提交到计算节点上运行了

```
sbatch hybrid-pro-thr.sh
```

例子中涉及的参数:

```
-J hybrid-pro-thr        # hybrid-pro-thr 为提交作业的名称,自定义
-p q_cn                  #作业提交的指定分区为 q_cn 队列;
-o job.%j.out            #脚本执行的输出将被保存在 job.%j.out 文件里，%j 表示作业号;
--ntasks-per-node=1      #每个节点上运行一个任务(进程)
-c  36                   #每个进程使用 36 个核心
-N  2                    #2 个节点
```

常见提交参数

```
--help                          # 显示帮助信息;
-D, --chdir=<directory>         # 指定工作目录;
--get-user-env                  # 获取当前的环境变量;
--gres=<list>                   # 使用 gpu 卡时需要需要的参数 如申请 1 块 gpu --
gres=gpu:1
-J, --job-name=<jobname> # 指定该作业的作业名;
--mail-type=<type>              # 指定状态发生时，发送邮件通知，有效种类为（NONE,
BEGIN, END, FAIL, REQUEUE, ALL）;
--mail-user=<user>              # 发送给指定邮箱;
-n, --ntasks=<number>           #默认情况下一个任务一个核心;
-c, --cpus-per-task=<ncpus> # 每个任务所需要的核心数，默认为 1;
--ntasks-per-node=<ntasks>      # 每个节点的任务数，--ntasks 参数的优先级高于该
参数，如果使用--ntasks 这个参数，那么将会变为每个节点最多运行的任务数;
-o, --output=<filename pattern>   # 输出文件，作业脚本中的输出将会输出到该文
件;
-p, --partition=<partition_names>   # 将作业提交到对应分区;
-t, --time=<time>                 # 允许作业运行的最大时间，目前集群默认时间为 2 天
-w, --nodelist=<node name list>   # 指定申请的节点;
-x, --exclude=<node name list>    # 排除指定的节点;
--mem-per-cpu=<size[units]>       #每个核心分配的内存大小   可以使用后缀[K|M|G|T]
来指定不同的单位
```

4.5 GPU 资源申请

- q_ai8、q_ai4、q_gpu_c 队列资源申请只需指定使用的 gpu 数量，无需指定 cpu 数量和内存大小。如需更多 cpu 或者内存只需申请多个 gpu 数量。
- q_ai8 队列:申请 1 个 gpu 同时会申请 4 个 core、108G 内存，以此类推...
- q_ai4 队列:申请 1 个 gpu 同时会申请 9 个 core、126G 内存，以此类推...
- q_gpu_c 队列：申请 1 个 gpu 同时申请 32 个 core, 183G 内存，以此类推...

例如：srun -p q_ai8 --gres=gpu:1

4.6 screen

如果用户使用 srun 交互模式可以使用 screen 进行后台运行，避免终端退出导致任务终止

screen -S 屏幕名称	Screen 创建
ctrl+a+d	从当前窗口切回主屏幕(不关闭屏幕)
ctrl+a+k	强行关闭当前的窗口
screen -ls	显示已创建的 screen 终端，获得作业名称

screen -r 屏幕名称	进入指定 screen

4.7 dSQ 批量提交

借助 Job Array dSQ 可以快速批量式提交**一组使用资源和执行任务非常相似，只是某些参数不相同的作业**。以下是 Job Array dSQ 使用说明:

编写计算任务列表文件

新建文件 joblist.txt，然后在文件中输入要计算的任务，每一行对应一个计算任务，如:

```
gatk GenomicsDBImport --genomicsdb-workspace-path ./AKCR1;
gatk GenomicsDBImport --genomicsdb-workspace-path ./AKCR2;
gatk GenomicsDBImport --genomicsdb-workspace-path ./AKCR3;
```

使用 dSQ 生成 Slurm 作业提交脚本

首先执行 module load dSQ 加载平台已安装的 dSQ 到当前终端窗口，然后执行如下命令生成 Slurm 作业提交脚

```
dsq --job-file joblist.txt -p q_cn -n 1 --mem-per-cpu 40g
```

joblist.txt 为上一步骤编写的任务列表文件；-p q_cn 表示作业提交到 q_cn 队列；-n 1 表示每一个计算任务使用的核心；--mem-per-cpu 40g 表示每一个计算任务使用 40g 内存

命令执行成功后，会在当前目录生成一个`dsq-joblist-yyyy-mm-dd.sh`文件，`yyyy-mm-dd`为创建日期。

```
dsq-joblist-2019-08-01.sh:
```

```
#!/bin/bash
#SBATCH --array 0-9999
#SBATCH --output dsq-joblist-%A_%4a-%N.out
#SBATCH --job-name dsq-joblist
#SBATCH -p q_cn -n 1 --mem-per-cpu 40g

# DO NOT EDIT LINE BELOW
/usr/nzx-cluster/apps/dSQ/dSQBatch.py /GPFS/zhangli/DATA/vcf.call.dsq/joblist.txt
/GPFS/zhangli/DATA/vcf.call.dsq
```

提交作业

执行如下命令提交作业

```
sbatch dsq-joblist-yyyy-mm-dd.sh
```

计算任务列表 `joblist.txt` 文件中有多少行（多少个计算任务），将会提交多少个作业。

作业管理

当有一个作业运行结束，当前目录下将会有一个 job_jobid_status.tsv 文件，该文件记录了每个作业的如下信息:

Job_ID: 作业号

Exit_Code: 程序退出码

Hostname: 占用节点名

Time_Started: 开始时间

Time_Ended: 结束时间

Time_Elapsed: 总耗时

Job: 运行命令

另外通过 slurm 的 squeue、scancel 命令可以查杀作业。

作业检查

运行如下命令：

```
dsqa jobsfile.txt job_2629186_status.tsv > failedjobs.txt 2> report.txt
```

使用 dSQ 前要先执行 module load dSQ，加载该软件到当前终端环境

会生成 failedjobs.txt 和 report.txt 文件，这两个文件中会记录运行成功和失败的作业个数，并且还会记录哪些作业运行失败。

4.8 本地/tmp 目录使用

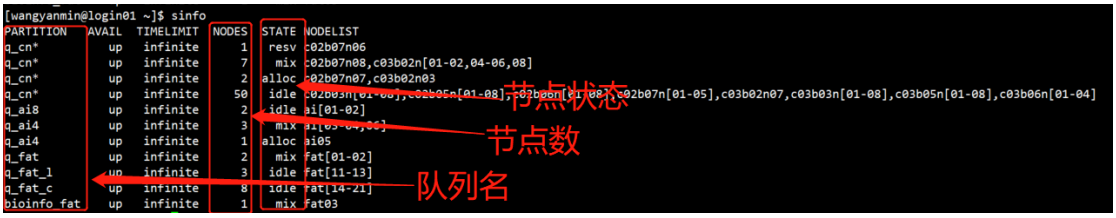
计算节点临时目录/tmp 下，磁盘总空间是 160G，如果产生的临时文件过大导致 tmp 目录磁盘空间爆满，就会影响程序的正常运行，为了不影响用户的工作进度和运行结果，需注意以下几点:

- 在运行之前可以评估下运行的程序大约可以产生多少临时文件，如果超出本地空间可以直接指定 tmp 输出路径到自己家目录下的 DATA 目录下如 1> mkdir $HOME/DATA/tmp 2> export TMPDIR=$HOME/DATA/tmp 添加到 .bashrc 或者.bash_profile 3>生效 source .bashrc 或者.bash_profile。
- 运行的时候可以多观察下程序输出的内容是否有报错信息。
- 可以 ssh 到申请的节点，看下/tmp 下的空间余量。
- 管理员发现/tmp 空间不足时也会通知相对应的用户，指定 tmp 输出路径，重新运行程序。

4.9 作业管理

sinfo

通过 sinfo 可查询各分区节点的空闲状态；显示集群的所有分区节点的空闲状态，`idel` 为空闲，`mix` 为节点部分核心可以使用，`alloc` 为已被占用;队列状态会不断调整，具体更新信息可关注计算中心网站：http://hpc.cibr.ac.cn



sinfo 常用参数

```
-a, --all                 # 显示所有分区 ((包括隐藏的和那些无法访问)
-d, --dead                #查看集群中没有响应的节点
-l, --long                 #长输出——显示更多信息
-n, --nodes=NODES         # 显示指定节点的信息，如果指定多个节点的话用逗号隔开
-o, --format=format        #按照指定格式输出
-p, --partition=PARTITION #显示指定分区的信息，如果指定多个分区的话用逗号隔开;
Help options:
--help                          # 显示 sinfo 命令的使用帮助信息;
```

job/squeue

查看提交作业的排队情况；

```
job         #查看自己提交的作业信息
squeue     #查看所有用户提交的作业信息
```

默认情况下 job 和 squeue 输出的内容如下，分别是作业号，分区，作业名，用户，作业状态，运行时间，节点数量，**申请的 cpu 数，申请的内存数**，运行节点

```
JOBID   PARTITION     NAME     USER ST       TIME  NODES   CPUS  MIN_M  NODELIST
```

默认情况下 squeue 输出的内容如下，分别是作业号，分区，作业名，用户，作业状态，运行时间，节点数量，运行节点

```
JOBID   PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

squeue 的常见参数

```
--help              # 显示 squeue 命令的使用帮助信息;
-A <account_list> # 显示指定账户下所有用户的作业，如果是多个账户的话用逗号隔开;
-i <seconds>      # 每隔相应的秒数，对输出的作业信息进行刷新
-j <job_id_list>   #显示指定作业号的作业信息，如果是多个作业号的话用逗号隔开;
-n <name_list>    #显示指定节点上的作业信息，如果指定多个节点的话用逗号隔开;
-t <state_list>    #显示指定状态的作业信息，如果指定多个状态的话用逗号隔开;
-u <user_list>     #显示指定用户的作业信息，如果是多个用户的话用逗号隔开;
-w <hostlist>     #显示指定节点上运行的作业，如果是多个节点的话用逗号隔开;
-l, --long        #输出长报告
```

通过 sacct 和 scontrol show job/node 显示作业/节点信息；

通过 sacct 查询已经结束作业的相关信息，如下所示：

```
sacct -j 899775
```

指定格式输出作业信息；

```
sacct --format=jobid,user,alloccpu,allocgres,state%15,exit -S 2022-08-01
```

备注：详细参数可通过 sacct -help 查看

通过 scontrol show job 查看正在运行作业的 jobid 资源：

```
[wangyanmin@login01 ~]$ scontrol show jobid=899774
JobId=899774 JobName=6-OHc4
   UserId=zhouxiangyu(5018) GroupId=zhangli_lab(5002) MCS_label=N/A
   Priority=1000 Nice=0 Account=zhangli_lab QOS=high
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=01:09:54 TimeLimit=7-00:00:00 TimeMin=N/A
   SubmitTime=2022-08-09T08:51:16 EligibleTime=2022-08-09T08:51:16
   AccrueTime=2022-08-09T08:51:16
   StartTime=2022-08-09T08:51:17 EndTime=2022-08-16T08:51:17 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2022-08-09T08:51:17
   Partition=q_cn AllocNode:Sid=login02:132385
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=c03b02n05
   BatchHost=c03b02n05
   NumNodes=1 NumCPUs=6 NumTasks=1 CPUs/Task=6 ReqB:S:C:T=0:0:*:*
   TRES=cpu=6,mem=120G,node=1,billing=6
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=6 MinMemoryCPU=20G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/GPFS/zhangli_lab_permanent/zhouxiangyu/projects/CIBR-SHIWEI/CIBR-ZH-C4-20220805/6-OHc4.slurm.sh
   WorkDir=/GPFS/zhangli_lab_permanent/zhouxiangyu/projects/CIBR-SHIWEI/CIBR-ZH-C4-20220805
   StdErr=/GPFS/zhangli_lab_permanent/zhouxiangyu/projects/CIBR-SHIWEI/CIBR-ZH-C4-20220805/6-OHc4.err
```

申请的计算资源

通过 scontrol show node 查看所占用节点的申请资源：

```
[wangyanmin@login01 ~]$ scontrol show node=c02b05n03
NodeName=c02b05n03 Arch=x86_64 CoresPerSocket=18
   CPUAlloc=35 CPUTot=36 CPULoad=2.27
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=(null)
   NodeAddr=c02b05n03 NodeHostName=c02b05n03 Version=18.08
   OS=Linux 3.10.0-957.el7.x86_64 #1 SMP Thu Nov 8 23:39:32 UTC 2018
   RealMemory=191891 AllocMem=171500 FreeMem=135845 Sockets=2 Boards=1
   State=MIXED ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=q_cn
   BootTime=2021-08-03T07:39:11 SlurmdStartTime=2022-06-02T09:56:09
   CfgTRES=cpu=36,mem=191891M,billing=36
   AllocTRES=cpu=35,mem=171500M
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

此节点最大资源

此节点所有用户申请的资源

scancel

取消队列中已提交的作业；

```
scancel   jobid
```

scancel 常见参数;

```
--help                  # 显示 scancel 命令的使用帮助信息;
-n <job_name>           # 取消指定作业名的作业;
-p <partition_name>     # 取消指定分区的作业;
-t <job_state_name>     # 取消指定作态的作业，"PENDING", "RUNNING" 或
"SUSPENDED";
-u <user_name>          # 取消指定用户下的作业;
```

第五章 存储使用查看

每位用户的存储目录分为四个部分

- **~/**
- 限额 20G
- 目录下的文件长期保存
- 建议仅用来保存用户的环境变量等设置信息。
- **~DATA2**
- 华为存储
- 使用建议：大量小文件 io 密集型计算建议使用 DATA2 存储，申请队列为 q_cn_2 和 q_fat_2
- **~/DATA**
- DDN 存储

- 目录下的文件长期保存
- 主要的计算程序和数据存储空间
- **~/scratch60**
- **DDN 存储**
- 额外的，临时的数据存储空间
- **~/pub**
- 公共数据库存放路径
- 用户只有读权限
- 请小心操作文件数据
- 误删除、误覆盖的文件无法恢复!

DDN 存储使用查看

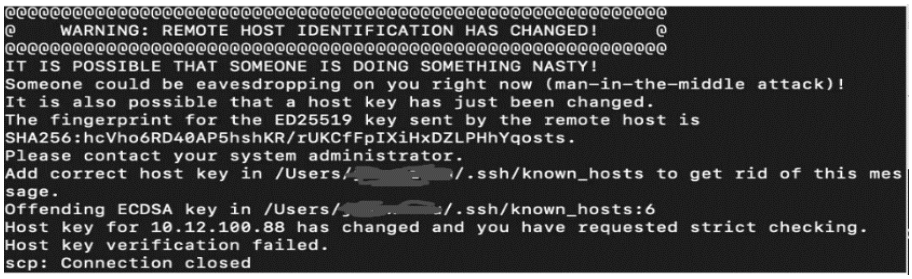
1>查看组使用情况 mmlsquota -g `groups`(为默认 DATA 2T+scratch60 10T)

2>查看 DATA 目录使用情况 mmlsquota -j `groups`_permanent gpfs

3>查看 scratch60 目录使用情况 mmlsquota -j `groups`_temp gpfs

第六章 常见问题

1. ssh 连接时报错



问题：ssh 无法登陆集群，报错 WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!

解答：1>登录节点的主机密钥变化导致与用户之前保存的登录节点密钥不同，SSH 客户端会阻止此登录，正确的做法是编辑 ~/.ssh/known_hosts 文件，删除登录节点对应的那一行并保存退出就可以了，然后重新连接，选择信任新的主机密钥。

2>这很可能是一种中间人攻击。因此 ssh 就会提示出上面的那个警告，告诉我们服务器的密钥与之前储存的不同,这种情况比较罕见

2. 无法在登录节点启动图形化界面

问题：运行图形化程序报错

解答： 集群是支持图形程序跳转

1>检查下自己使用的终端是否支持图形跳转，或者直接在登录节点测试运行 xclock 是否可以跳出钟表图标，如不能跳转大概率说明你的终端不支持图形跳转，或者没有安装插件

2>如果可以再进行提交集群测试 srun –p q_cn --x11 --pty xlcock 如果不行可以从以下 2 个方面去考虑问题

查看 home 空间是否已满,可以再 DATA 下做个软连接到 home 下，这样就不会占用 home 下空间

否则重新生成下本地秘钥（1>ssh-keygen 2> cat id_dsa.pub > authorized_keys 3>chmod 600 authorized_keys ）

3. Disk quota exceeded

问题： 本地无法写入，报错 Disk quota exceeded

解答：1>查看 home 下空间

cd $HOME && du -sh && du -sh .[!.]*

查看比较大的目录可以创建一个软连接，这样就不会占用 home 空间

mv $HOME/.local DATA/

ln -sf /GPFS/zhangli_lab_permanent/wangyanmin/.local /home/zhangli_lab/wangyanmin/

4. 任务自动停止

问题：提交的任务自动停止

解答： 1>程序报错导致/内存溢出

```
[wangyanmin@login01 ~]$ job
     JOBID PARTITION     NAME             USER ST       TIME  NODES   CPUS  MIN_M   NODELIST
   3412739      q_cn    sleep      wangyanmin  R       0:57      1      1  4900M  c02b07n05
[wangyanmin@login01 ~]$ squeue -l |grep wangyan
           3412739      q_cn    sleep wangyanm  RUNNING       0:58 2-00:00:00      1 c02b07n05
[wangyanmin@login01 ~]$
```

5. 请问可以给我 sudo(root) 权限吗?

问题：安装软件需要管理严权限

解答: root 或者 sudo 权限，任何情况下，都不会给用户。

如果你是想执行 sudo yum install 这样的操作，系统绝大部分依赖包都已经安装，

如果你是想执行 sudo apt install，计算中心集群使用的是 CentOS 而不是 Debian/Ubuntu，你要安装的包在 CentOS 里可能叫另外一个名字 yum。同样因为绝大部分依赖包都已经安装，你只需要跳过这一步即可。

如果你后续的安装使用步骤提示确实缺少依赖包，请把这个依赖包的名字告诉计算中心系统管理员，让管理员来安装。

用户需要 root(sudo) 权限的另外一大原因是在安装软件的时候，没有修改默认的安装路径(通常是 /opt， /usr/local 这样需要 root 权限的系统目录)。要解决这类问题，如果是从源代码编译软件，一般是在 configure 的时候使用 --prefix= 这个选项把安装目录指定到用户自己的目录下。如果是安装型的软件，在安装向导里修改默认的安装目录为用户自己的目录。

第七章 用户支持

1. 可以直接发送邮件：cdsc@cibr.ac.cn 或者 用户微信群
2. 计算与数据科学中心网址：http://cdsc.cibr.ac.cn

有问题或者需求一定要联系计算与数据科学中心