Beijing Brain Center High Performance Cluster User Manual

October 25, 2022

Chapter 1 Existing Clusters

The platform's massive data processing and storage cluster adopts GPU+CPU heterogeneous system, with a total of 2504 processors, 60 computing nodes, 2 fat nodes, and 6 GPU nodes. The total double-precision computing capacity of the cluster is about 0.4PFlops. P is a high-performance storage system with raw capacity, achieving an aggregated read and write bandwidth of up to 40GB/s, to meet applications in the field of brain science such as huge data volume, high bandwidth, and high IOPS. In order to ensure the efficient and stable operation of the cluster, the computing center also Formulated resource quota restriction strategy, real-time alarm of computer room environment, hardware and software faults

Node class	queue/partition name	number of nodes	Processor Specifications	Number of cores per node	memory per node	Number of GPUs	Cluster computing power
cpu common node	q_cn	60	2*Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz	36 cores (Apply for one core, about 4.9G memory is available)	192G	/	0.4 PFlops
Four-way fat node	q_fat	2	4*Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz	72 cores (Apply for one core, about 20G memory is available)	1536G	/	
	q_fat_c_	8	4*Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz	72 cores (Apply for one core, about 20G memory is available)	1536G	/	
	q_fat_l	3	4*Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz	72 cores (Apply for one core, about 20G memory is available)	1536G	/	
GPU fat node	q_ai8	2	2*Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz	40 cores (Apply for one core, about 24G memory is available)	1024G	8*NVIDIA Tesla V100 32GB	
GPU node	q_ai	2	2*Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz	36 cores (apply for one core, about 4.9G memory is available)	192G	4* NVIDIA Tesla V100 32GB	
	q_ai24	1	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	24cores (apply for one core, about 7G memory is available)			
	q_ai48	1	2*Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz	48 cores (Apply for one core, about 3.6G memory is available)			
Total available storage capacity: 3.1P, aggregate read and write bandwidth: 40GB/s							

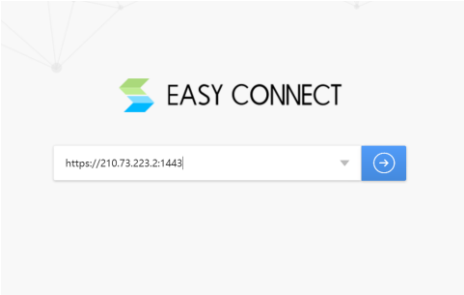
Chapter two Cluster login

1.1 VPN login

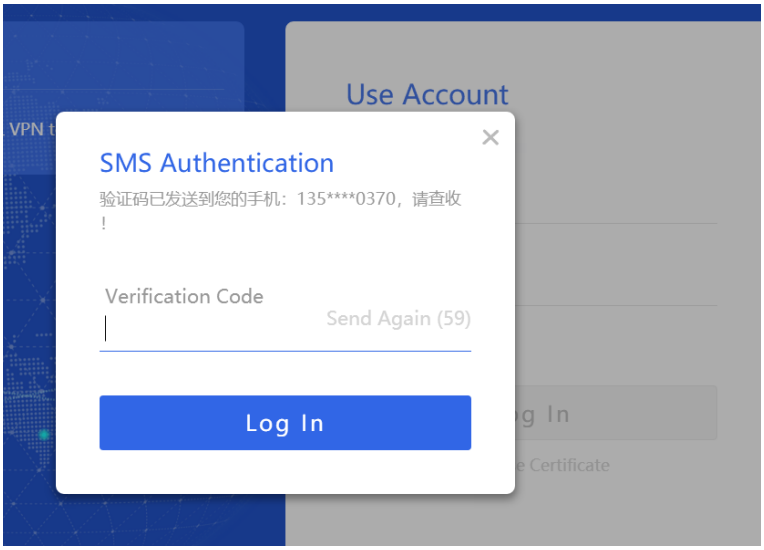
Open IE browser or safari browser and visit https://210.73.223.2:1443 or https://bbsi.cibr.ac.cn:1443 , a security warning message will be displayed after opening the webpage, click "Details" and then click "Go to this page", the VPN login window will be displayed at the end, enter the VPN account password and click login.

The Easyconnect client will be downloaded and installed for the first login , and the client can be used for subsequent logins without logging in from the web page.

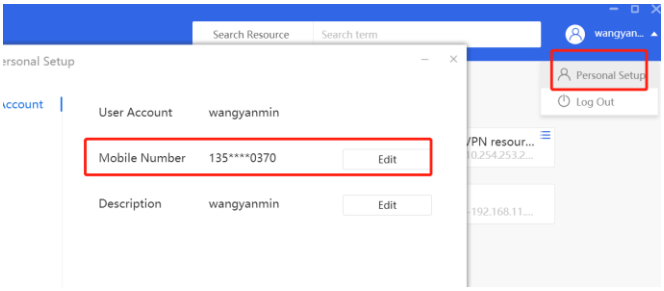
Open the Easyconnect client and enter the access address



vpn account of the cluster , password and then enter the verification code received by the mobile phone



The first time is the mobile phone number set by the administrator. If you want to change the mobile phone number during use, you can modify it yourself after logging in successfully.



1.2 Host login

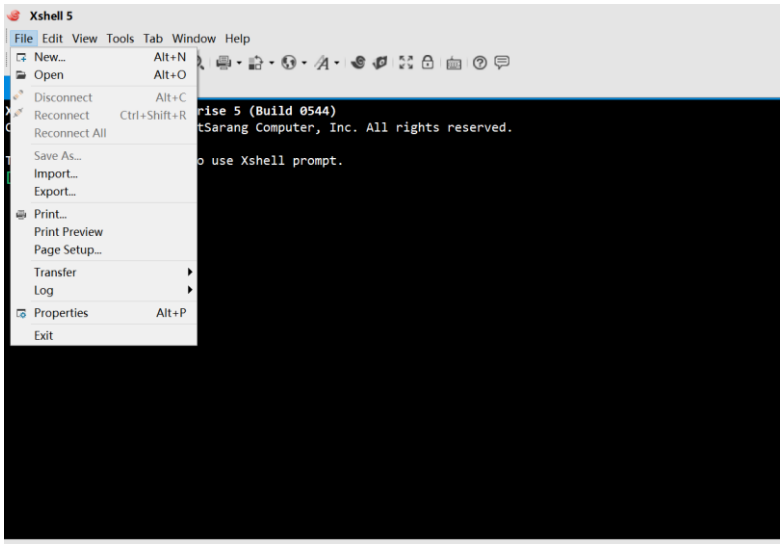
1.2.1 Cluster IP Address

cluster IP address is: 10.12.100.88 , through which users can log in to the login node of the cluster. The login node is mainly used for operations such as file uploading and downloading, program writing, software installation and job submission. The login node **cannot run the program (slurm needs to be used to schedule the login node)** , otherwise it will affect the login and operation of other users.

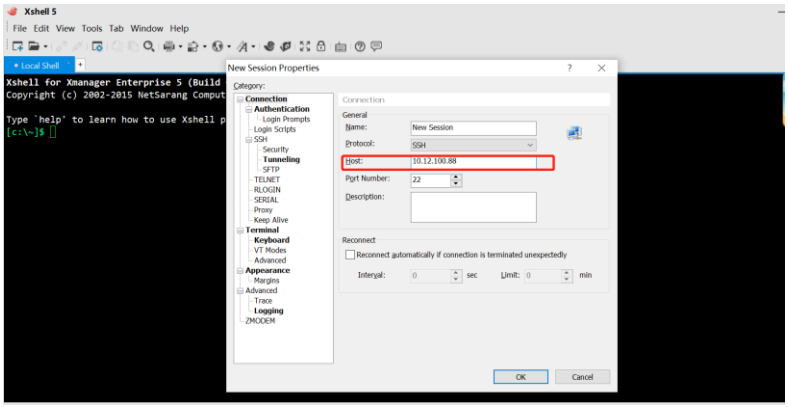
1.2.2 Windows user host login

Windows users can log in to the cluster with SSH client software such as MobaX term , Xshell , SSH Secure Shell Client, PuTTY, and SecureCRT . The following uses xs hell as an example to introduce how to log in. xshell is paid commercial software, bu t there is a free educational home version available for download.

1) Open x shell , click "New Session" in "File"

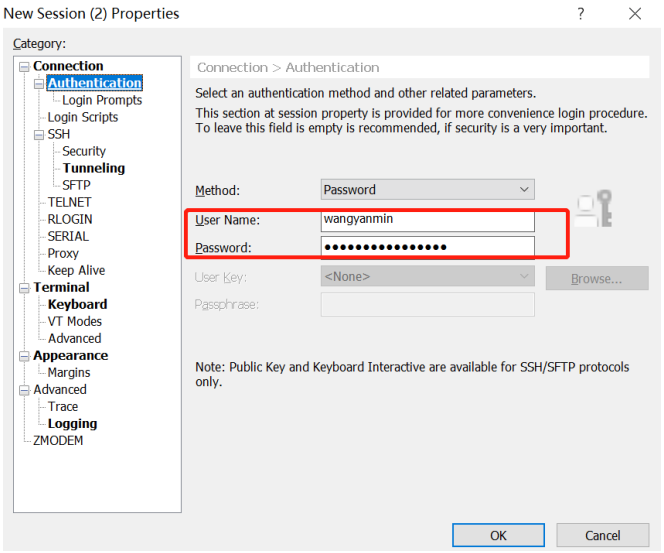


2) Edit the session, enter the IP address in the red box

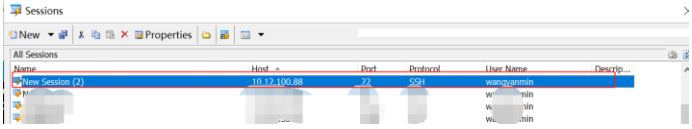


3) Enter the cluster host account and password

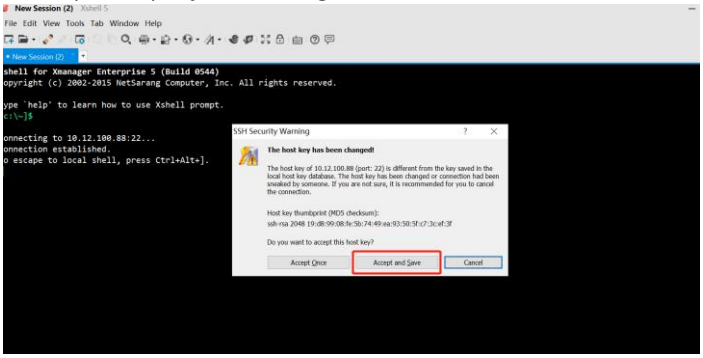
Click "User Authentication", enter the host account and password, and then click "OK" to complete the session creation.



4) Select the session you just created and click "Connect" to log in to the cluster



5) When you log in for the first time, a window will pop up asking whether to save the key. After selecting "Accept Keep", you can log in to the cluster.



1.2.3 Linux , Mac user host login

ssh command directly in the command line terminal to log in directly:

```
$ ssh username@10.12.100.88
```

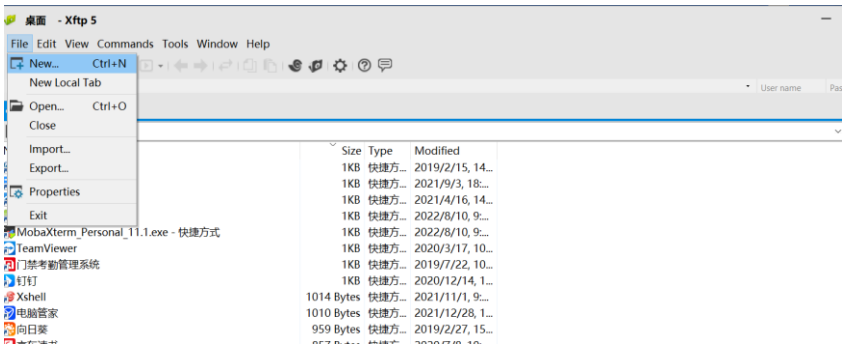
1.3 File upload and download

1.3.1 Windows user file upload and download

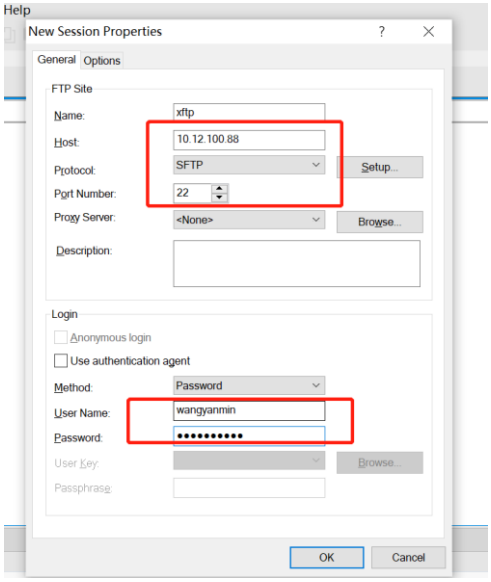
Windows users can use MobaXterm , Xftp , SSH Secure Shell Client, winscp and other software to upload and download files. The following uses Xftp as an example to introduce how to upload and download files. xshell is commercial paid commercial software, but there is a free educational home version available for download.

1) new session

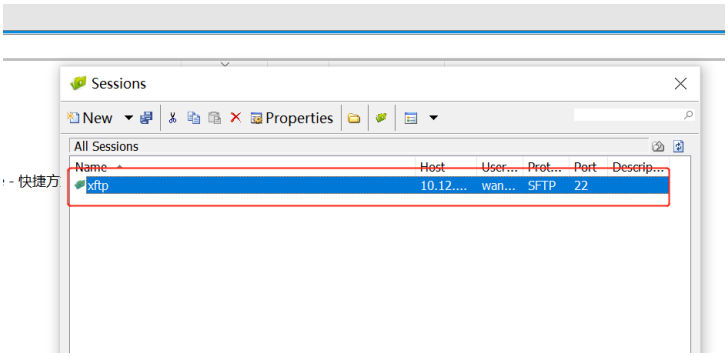
Open xftp and click "New" in "File".



2) Edit the call, enter the IP, account and password

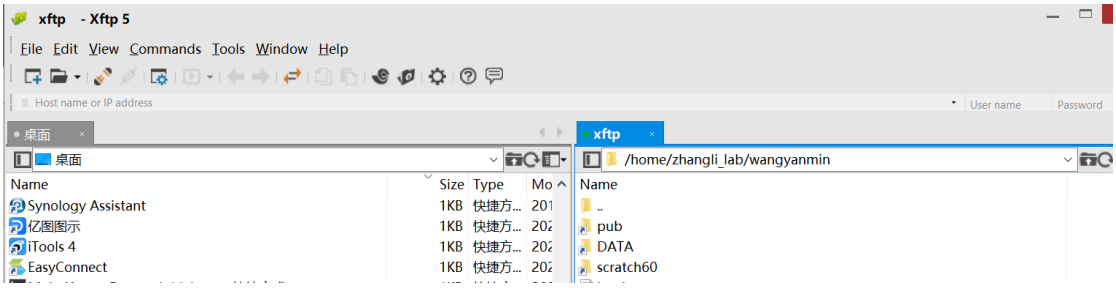


3) Select the created session and click "Connect" to log in to the cluster



4) File upload and download

After the cluster is successfully logged in , the left side is the machine, and the right side is the high-performance cluster, which can be directly dragged to upload and download files.



1.3.2 Linux , Mac user file upload and download

Linux and Mac users can directly use commands to upload and download files. All files need to be uploaded to the DATA directory .

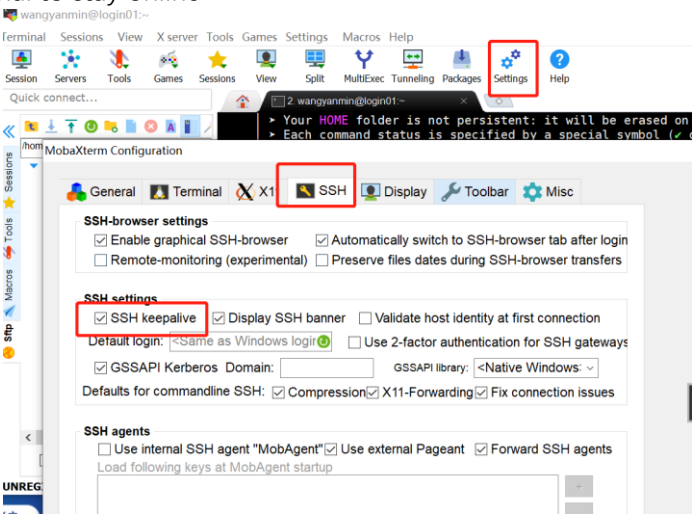
```
scp filename    username @10.12.100.88:~/DATA
```

1.4 Graphics Forwarding

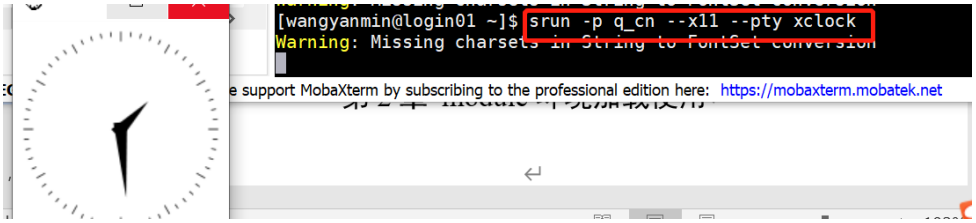
1.4.1 Window user graphics forwarding

Windows users can use MobaXterm , X shell+Xmanager (commercial version), putty+xming and other software to realize software graphics forwarding. The following takes Moba Xterm as an example to introduce how to use graphics forwarding.

1) Set the terminal to stay online



2) Log in to the cluster to run the test program and jump out of the graphical interface



1.4.2 Graphical forwarding for Mac users

Mac users need to download the xquartz X11 terminal program separately

1) Modify the configuration file

$ sudo vim / etc / ssh /sshd_config

#X11Forwarding noRemove # and change no to y es

2) restart sshd service

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

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

See if it starts > $ sudo launchctl list | grep ssh

3) Log in to the terminal

$ ssh -Y user@10.12.100.88

4) Run the test

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

Chapter 3 module environment loading

The cluster has installed some common software, which are managed and used by Module

Use the module command to switch between different versions of the same software, or switch between different software with the same function, so as to choose the most suitable programming environment and operating environment.

module avail	View all module managed software
Module load bwa /0.7.17	Load the appropriate version of the software
	If you write to ~/ .bashrc , the terminal that you log in will automatically load the corresponding software
	If it is written into the job submission script of sbatch , it will only take effect in the script, and the shell environment outside the script will be invalid.
module list	Show currently loaded software
module swap bwa/0.7.17 bwa/0.7.12	Switch software version
module unload bwa /0.7.17	Uninstall the corresponding version of the software
module spider bwa	Full list of search modules
module purge	Clear all loaded software

```
[wangyanmin@login02 ~]$ module avail

---------------------------------------------- /usr/nzx-cluster/Modules/modulefiles ----------------------------------------------
   3d-dna/180922                                  intel-mpi/2018
   3d-dna/201008                                  intel-mpi/2019
   3d-dna/20170123                      (D)       intel-mpi/2020                      (D)
   AmpliconArchitect/AmpliconArchitect            interproscan/5.52-86.0
   BSseeker/2.1.8                                 iqtree/1.6.12
   CNVnator/0.3                                   isoseq/3.3.0
   CNVnator/0.4.1                       (D)       jansson/jansson
   CUnit/2.1.3                                    java/1.8.0
   EIG/7.2.1                                      java/11.0.13                        (D)
   GATK3/3.8.0                                    jdk/1.7.0_80
```

```
[wangyanmin@login02 ~]$ module load bwa/0.7.17
[wangyanmin@login02 ~]$ module list

Currently Loaded Modules:
  1) bwa/0.7.17



[wangyanmin@login02 ~]$ module swap bwa/0.7.17 bwa/0.7.12

The following have been reloaded with a version change:
  1) bwa/0.7.17 => bwa/0.7.12

[wangyanmin@login02 ~]$ module list

Currently Loaded Modules:
  1) bwa/0.7.12
```

```
[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:
```

Chapter 4 Assignment Submission

s lurm job scheduling system is divided into s run , s batch , salloc 3 ways to submit assignments

How to submit assignments	How to use	advantage	shortcoming	Trial scene
srun interactive commit	srun + resource application + program running command srun -J test -p q_cn -c 1 python hello.py	Quick and easy Program output is printed directly to the screen, making it easy to observe program running logs and error messages	The terminal is disconnected from the cluster and the job will be interrupted	Pre-work debugging

sbatch batch submission	The job submission parameters are written in the script run.slurm , which executes sbatch run.slurm submit job	The calculation is stable, and the job is controlled by the computing node, regardless of the terminal state batch submission	Need to write a few lines of script, slightly cumbersome	formal calculation
salloc allocation commit	salloc + resource application sallloc -J test -p q_cn -c 1	Continue to occupy the node without repeated queuing (no exit will always be billed) See the output of the program from the screen in real time	The terminal is disconnected from the cluster and the job will be interrupted	A large number of jobs of the same size need to be submitted but do not want to be queued repeatedly
Note: The default duration of the task is 7 days. If you want to extend a longer time, you can contact the administrator to apply				

4.1 single-threaded job submission

srun interactively submits commands

Program output is printed directly to the screen, making it easy to observe program running logs and error messages

Let's start with a simple example:

During the calculation process, we run the hostname command to submit application resources for 1 task and 1 core, then use srun to submit the command:

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

sbatch batch submission , the script name is hostname.sh (daily recommended)

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

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

hostname
```

After editing the script, you can submit it directly to the computing node to run

```
sbatch hostname.sh
```

salloc allocation commit

```
salloc -p q_cn -n 1

srun -n 1 -o job.%j.out    hostname    #Still need s run to submit, no need to specify partition, no need to queue
```

Parameters involved in the example：

```
-J hostname #hostname is the name of the submitted job, custom
-p q_cn              #The specified partition for job submission is the q_cn queue;
-o job.%j.out        #The output of the script execution will be saved in the job.%j.out file, where %j represents the job number;
-n 1             #Run a task (process) on each node
```

4.2 Multithreaded submission (parallel program programmed with OpenMP)

The following takes the sbatch submission method as an example

multithread command during the calculation process , start 1 task (process), 36 cores, then use sbatch to submit the command (the **script name is 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 #m odule loads the required software

./multithread
```

After editing the script, you can submit it directly to the computing node to run

```
sbatch multithread.sh
```

Parameters involved in the example：

```
-J multithread    # multithread is the name of the submitted job, custom
-p q_cn           #The specified partition for job submission is the q_cn queue;
-o job.% j.out     #The output of the script execution will be saved in the job.% j.out file,
where %j represents the job number;
-n 1          #Run a task ( process ) on each node
-c 36 # use 36 cores per process
```

4.3 Multi-process submission (parallel programs programmed with MPI)

The following takes the sbatch submission method as an example

multiprocess command during the calculation , start 100 tasks (processes), then use sbatch to submit the command (the **script is named 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 #m odule load the required software

srun -n 100 ./multiprocess
```

After editing the script, you can submit it directly to the computing node to run

```
sbatch multiprocess.sh
```

Parameters involved in the example：

```
-J multiprocess    # multithread is the name of the submitted job, custom
-p q_cn            #The specified partition for job submission is the q_cn queue;
-o job.% j.out     #The output of the script execution will be saved in the job.% j.out file,
where %j represents the job number;
-n 1          #Run a task ( process ) on each node
-c 36 #Use 36 cores per process
```

4.4 Multi-process + multi-thread (parallel program programmed with MPI+OpenMP)

The following takes the sbatch submission method as an example

hybrid **-pro-thr** command during the calculation process , apply for 2 nodes for resources, each node runs a process, and each process runs 36 cores, then use sbatch to submit the command (the **script name is 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 #m odule loads the required software

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

After editing the script, you can submit it directly to the computing node to run

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

Parameters involved in the example：

```
-J hybrid-pro- thr          # hybrid-pro- thr is the name of the submitted job, custom
-p q_cn                     #The specified partition for job submission is the q_cn queue;
-o job.% j.out               #The output of the script execution will be saved in the job.%
j.out file, where %j represents the job number;
--n tasks - per-node = 1     #Run a task ( process ) on each node
-c 36 # use 36 cores per process
-N 2 # 2 nodes
```

Common submission parameters

```
--help # Display help information;
-D, --chdir =<directory> # Specify the working directory;
--get-user-env # Get the current environment variables;
--gres =<list> #Required parameters when using gpu card, such as applying for 1 gpu --gres = gpu:1
-J, --job-name=<jobname> # Specify the job name of the job;
--mail-type=<type> # When the specified state occurs, send email notification, the valid types are (NONE, BEGIN, END, FAIL, REQUEUE, ALL);
--mail-user=<user> #Send to the specified mailbox ;
-n, -- ntasks =<number> #By default, one task is one core ;
-c, --cpus -per-task=< ncpus > # The number of cores required by each task, the default is 1;
-- ntasks -per-node=< ntasks > # The number of tasks per node, the priority of the -- ntasks parameter is higher than this parameter, if the -- ntasks parameter is used , it will become the most running one per node number of tasks;
-o, --output=<filename pattern> # Output file to which the output from the job script will be output;
-p, --partition=< partition_names > # Submit the job to the corresponding partition;
-t, --time=<time> # The maximum time allowed for the job to run, the current cluster default time is 7 days
-w, -- nodelist =<node name list> # Specify the node to apply for;
-x, --exclude=<node name list> # Exclude the specified node;
- -mem-per- cpu =<size[units]>     #The memory size allocated by each core can use the suffix [K|M|G|T] to specify different units
```

4.5 screen

If the user uses s run Interactive mode can use screen to run in the background to avoid the termination of the task caused by the terminal exit

screen -S screen name	S creen
ctrl+ a + d	Switch back to the main screen from the current window (without closing the screen)
ctrl+a+k	force close the current window
screen - ls	Display the created screen terminal and get the job name
screen -r screen name	Enter the specified screen

4.6 dSQ batch submission

With the help of Job Array dSQ, you can quickly batch submit **a group of jobs that use resources and execute tasks that are very similar, but with different parameters** . The following are the instructions for using the Job Array dSQ :

Write a calculation task list file

Create a new file joblist.txt, and then enter the tasks to be calculated in the file, each line corresponds to a calculation task, such as:

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

Generate Slurm Job Submission Script Using dSQ

First execute module load dSQ to load the installed dSQ of the platform to the current terminal window, and then execute the following command to generate the Slurm job submission script

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

joblist.txt is the task list file written in the previous step; -p q_cn indicates that the job is submitted to the q_cn queue; -n 1 indicates the core used by each computing task; --mem-per- cpu 40g indicates that each computing task uses 40g Memory

After the command is executed successfully, a `dsq-joblist-yyyy-mm-dd.sh` file will be generated in the current directory, and ` yyyy -mm -dd` is the creation date.

```
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
```

submit homework

Execute the following command to submit the job

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

computing jobs) are in the joblist `joblist.txt` file and how many jobs will be submitted.

Job management

When a job ends, there will be a job_jobid_status.tsv file in the current directory, which records the following information about each job :

Job_ID : Job ID

Exit_Code : program exit code

Hostname: occupies the node name

Time_Started : start time

Time_Ended : end time

Time_Elapsed : total time elapsed

Job: run command

In addition, you can check and kill jobs through slurm 's squeue and scancel commands.

homework check

Run the following command:

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

using dSQ , execute module load dSQ to load the software into the current terminal environment

Failedjobs.txt and report.txt files will be generated, which will record the number of jobs that run successfully and fail, and which jobs fail to run.

4.7 local / tmp directory use

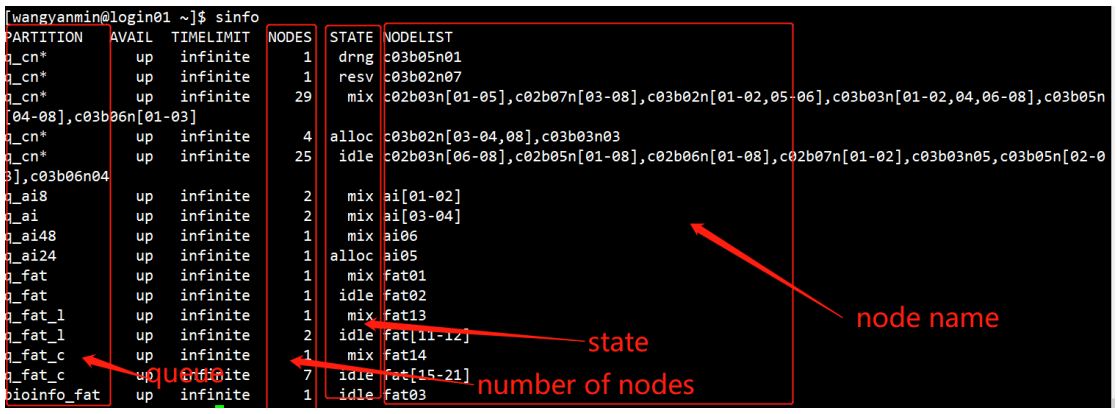
tmp of the computing node , the total disk space is 160G. If the temporary file generated is too large and the disk space of the tmp directory is full, it will affect the normal operation of the program. In order not to affect the user's work progress and operation results, the following should be noted A few points :

- Before running, you can evaluate how many temporary files the running program can generate. If it exceeds the local space, you can directly specify the tmp output path to the DATA directory under your home directory .
- When running, you can observe whether there is any error message in the output of the program .
- You can ssh to the requested node and see the space allowance under / tmp .
- administrator finds that the / tmp space is insufficient, the corresponding user will also be notified, specify the tmp output path, and re-run the program.

4.8 Job management

sinfo

The idle state of each partition node can be queried through sinfo ; the idle state of all partition nodes in the cluster is displayed , `idel` is idle, `mix` is part of the core of the node that can be used, and `alloc` is occupied; (available queues q _cn , q_ai , q_ai48, q_fat , among which q_cn _lyz, q_ai1024g_lyz, lab_fat_c, lab_fat_l , bioinfo_ai, bioinfo_fat are exclusive queues of other laboratories) The queue status will



be adjusted continuously, and the specific update information can be paid to the computing center website: h ttp :// h pc.cibr.ac.cn

Common parameters of sin fo

```
-a, --all # show all partitions ( (including hidden and those inaccessible)
-d, --dead #View unresponsive nodes in the cluster
-l, --long #long output -- show more information
```

```
-n, --nodes=NODES # Display information about the specified node, separated by commas if multiple nodes are specified
-o, --format=format #Output in the specified format
-p, --partition=PARTITION #Display the information of the specified partition, if multiple partitions are specified, separate them with commas;
Help options:
--help # Display the help information of the sinfo command;
```

job / squeue

View the queuing of submitted jobs;

```
job #View the job information submitted by yourself
squeue     #View job information submitted by all users
```

By default, the output contents of j ob and squeue are as follows: job number, partition, job name, user, job status, running time, number of nodes, number **of CPUs requested, number of memory requested** , and running nodes

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

By default, the output of squeue is as follows, namely job number, partition, job name, user, job status, running time, number of nodes, running node

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

Common parameters of squeue

```
--help # Display the help information of the squeue command;
-A < account_list > # Display the jobs of all users under the specified account, separated by commas if there are multiple accounts;
- i <seconds> # Refresh the output job information every corresponding number of seconds
-j < job_id_list > #Display the job information of the specified job number, if there are multiple job numbers, separate them with commas;
-n < name_list > #Display job information on the specified node, separated by commas if multiple nodes are specified;
-t < state_list > #Display the job information of the specified state, if multiple states are specified, separate them with commas;
-u < user_list > #Display the job information of the specified user, if there are multiple users, separate them with commas;
-w < hostlist > #Display jobs running on the specified node, separated by commas if there are multiple nodes;
-l, --long       # output long report
```

Display job /node information through sacct and scontrol show job / node ;

Use `sacct to` query information about jobs that have ended, as follows:

```
sacct -j 899775
```

Output job information in a specified format;

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

Note: Detailed parameters can be viewed through sacct –help
```

jobid resource of the running job through scontrol show job :

```
[wangyanmin@login01 ~]$ scontrol show jobid=2483843
JobId=2483843 JobName=sleep
   UserId=wangyanmin(5012) GroupId=zhangli_lab(5002) MCS_label=N/A
   Priority=666 Nice=0 Account=zhangli_lab QOS=high
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
   RunTime=00:00:23 TimeLimit=2-00:00:00 TimeMin=N/A
   SubmitTime=2022-11-10T14:22:56 EligibleTime=2022-11-10T14:22:56
   AccrueTime=Unknown
   StartTime=2022-11-10T14:22:56 EndTime=2022-11-12T14:22:56 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2022-11-10T14:22:56
   Partition=q_cn AllocNode:Sid=login01:10165
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=c03b03n04
   BatchHost=c03b03n04
   NumNodes=1 NumCPUs=20 NumTasks=1 CPUs/Task=20 ReqB:S:C:T=0:0:*:*
   TRES=cpu=20,mem=98000M,node=1,billing=20
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=20 MinMemoryCPU=4900M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=sleep
   WorkDir=/home/zhangli_lab/wangyanmin
   Power=
```

resources requested

show node via scontrol View the application resources of the occupied node :

```
[wangyanmin@login01 ~]$ scontrol show node c03b03n04
NodeName=c03b03n04 Arch=x86_64 CoresPerSocket=18
   CPUAlloc=36 CPUTot=36 CPULoad=0.01
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=(null)
   NodeAddr=c03b03n04 NodeHostName=c03b03n04 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=115184 FreeMem=178278 Sockets=2 Boards=1
   State=ALLOCATED ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=q_cn
   BootTime=2022-11-03T14:42:42 SlurmdStartTime=2022-11-03T14:58:15
   CfgTRES=cpu=36,mem=191891M,billing=36
   AllocTRES=cpu=36,mem=115184M
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

node maximum resources

total resources requested by nodes

scancel

Cancel submitted jobs in the queue;

```
scancel  jobid
```

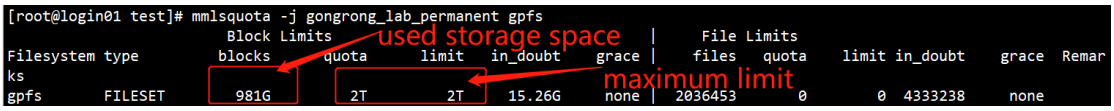
scancel common parameters;

```
--help # Display the help information of scancel command;
-n < job_name > # Cancel the job of the specified job name;
-p < partition_name > # Cancel the job of the specified partition;
-t < job_state_name > # Cancel the job of the specified state, "PENDING", "RUNNING" or "SUSPENDED";
-u < user_name > # Cancel the job under the specified user;
```

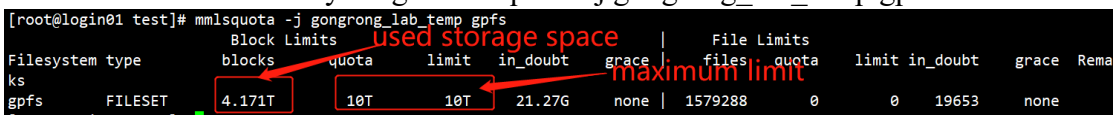
Chapter 5 View storage space

View group usage mmlsquota –g gongrong_lab (default DATA 2T+scratch60 10T)

View DATA directory usage mmlsquota –j gongrong_lab_permanent gpfs



View scratch60 directory usage mmlsquota –j gongrong_lab_temp gpfs



Note: If you need more storage space, you need to fill in the storage expansion application form

Chapter 6 User Support

1 . You can ask questions directly in the forum
2. You can also send emails directly to the specified email address, which can be synchronized to the forum. The email address is cibrhpc@mail.cibr.ac.cn
3. Forum website: http://bbs.cibr.ac.cn /
4. E- mail of Computing Center : hpc@cibr.ac.cn or wangyanmin@cibr.ac.cn
5. Wang Yanmin Tel: 13505420370 (Wechat synchronization)

Remarks: The user manual will be updated from time to time, the latest version can be downloaded from the website of the Computing Center h ttp://hpc.cibr.ac.cn

If you have any questions or needs, you must contact the computing center