国家超级计算广州中心/广州超级计算中心

启明系统简明使用手册

V 1.3

HPC 技术支持服务团队

2023 年 3 月 31 日

使 用 须 知

1. **启明系统的结点使用方式为独占式，即计算结点分配给用户以后，不能为其他用户所用；机时的计算也将以结点为基本单元。**比如，在启明系统 work 分区中使用 2 个结点共 12 核（每结点使用 6 核）运行 1 小时的程序，机时为 2*36*1*权重（核*小时），而非 2*6*1*权重（核*小时）。因此用户需要合理利用自己的资源。

2. **用户所分配机时仅为机时使用。**如委托研发、修改程序代码、编写复杂脚本等额外服务，会产生一定的服务费用。如有需求，请发邮件到 techsupport@nscc-gz.cn 洽商。

3. **有少量机时但不能满足作业运行完成时，按照中心目前规定：账号有剩余机时且还有少量短时间作业需要运行，作业可正常提交；若剩余机时已不能满足少量短时间作业运行完成，对于已经提交的作业仍会保障运行完成。**对于不希望继续运行需结束作业，用户需自行停止需要结束的作业。对于已无机时的账号，使用 yhi 和 yhq 无法查看分区和作业状态，此时若还需查看作业状态信息，请使用“yhq -a”命令查看。

4. **超算中心不提供商业软件的安装适配、售后及应用支持服务(中心已购版权的商业软件除外)。**您可以自行安装和使用商业软件，或与该软件的售后服务团队联系寻求协助。用户自行安装软件带来的版权问题请自负责任。

5. 超算中心根据实际情况对基于“启明系统”部署的开源软件的安装进行一定程度上的协助，但软件的算法合理性、精度、并行效率、使用方法等软件自身问题需要自行解决。常用的开源软件的使用方法可以查阅超算中心官

网上的相关说明，如：http://www.nscc-gz.cn/newsdetail.html?7311（Quantum ESPRESSO）。

6.如您的任何行为对超算中心的财产和声誉等方面造成了任何损失,超算中心将依法追究相关责任。以上条例解释权归超算中心所有。

目　录

1 系统架构 6
1.1 计算结点 6
1.1.1 CPU 结点队列简介 6
1.1.2 GPU 结点队列简介 6
1.2 文件系统 7
1.3 软件环境 7
1.3.1 操作系统 7
1.3.2 作业调度系统 7
1.3.3 编译环境 7
2 登录 9
2.1 VPN 验证 9
2.2 终端登录 9
2.2.1 MobaXterm 登录 9
2.2.2 PUTTY 登录 10
2.2.3 Xshell（Xmanager-XShell）登录 12
2.2.4 Linux 或苹果系统登录 13
2.3 账户管理 14
2.3.1 账户秘钥重置 14
2.3.2 账户信息查询 14
3 数据传输 17
3.1 外部数据传输 17
3.1.1 网络传输 17
3.1.2 硬盘挂载拷贝 24
3.2 “启明系统”内部数据传输 24
3.2.1 同集群账户间的数据传输 24
3.2.2 跨集群或文件系统账户间的数据传输 24
4 环境变量管理 26
4.1 module 环境变量管理工具的使用 26
4.1.1 module 工具的基本命令 26
4.2 export / source 命令加载环境变量 27
4.3 .bashrc 的配置方法 28
5 程序编译及软件安装 29
5.1 编译器 29
5.1.1 GCC 编译器 29
5.1.2 Intel 编译器 29
5.1.3 CUDA 编译器 30
5.1.4 PGI 编译器 30
5.1.5 MPI 编译环境 30
5.2 程序编译示例 31

5.2.1 单个源文件示例 31
5.2.2 多个源文件示例 31
5.2.3 openmp 程序编译 32
5.2.4 MPI 程序编译 33
5.3 常规软件安装方法 34
6 作业提交与管理 36
6.1 作业管理 36
6.1.1 结点状态查看 yhinfo 或 yhi 36
6.1.2 作业状态信息查看 yhqueue 或 yhq 36
6.1.3 任务取消 yhcancel 37
6.2 作业提交 37
6.2.1 交互式作业提交 yhrun 38
6.2.2 批处理作业 yhbatch 40
6.2.3 结点资源抢占命令 yhalloc 42
6.2.4 部分典型脚本示例 44
6.3 备注 45
7 帮助与支持 47
7.1 名词解析 47
7.2 常见问题 48
7.2.1 登录问题 48
7.2.2 系统资源问题 48
7.2.3 作业提交及运行问题 49
附录 A：常用 unix 命令 51
A.1 浏览目录和文件命令 51
A.2 目录和文件操作命令 51
A.3 文本编辑工具 vi 52
A.3.1 移动光标类命令 52
A.3.2 输入控制命令 52

1 系统架构

启明系统主要包含 CPU 结点队列以及 GPU 结点队列。使用 CPU 结点队列的用户默认开通 work 分区；使用 GPU 结点队列的用户默认开通 gpu_v100 分区。

1.1 计算结点

1.1.1 CPU 结点队列简介

CPU 结点队列主要包含常规计算节点分区（work 与 work_8358P 分区），以及大内存节点分区（bigmem 分区）。

其中 work 分区的每个计算结点包含 2 个 Intel(R) Xeon(R) Gold 6150 18 核心的多核中央处理器（CPU），每个计算结点拥有 192G 内存（2 个 CPU 共用）。而 work_8358P 分区的每个计算结点包含 2 个 Intel(R) Xeon(R) Platinum 8358P 32 核心的多核中央处理器（CPU），每个计算结点拥有 512G 内存（2 个 CPU 共用）。

大内存结点（bigmem 分区）的每个计算结点包含 2 个 Intel(R) Xeon(R) Gold 6150 18 核心的多核中央处理器（CPU），每个 bigmem 分区计算结点拥有 768G 内存（2 个 CPU 共用）。

1.1.2 GPU 结点队列简介

GPU 结点队列主要包含 GPU_V100 分区，以及 GPU_A800 分区。

GPU_V100 结点队列（gpu_v100 分区）中每个计算结点包含 2 个 Intel(R) Xeon(R) Gold 6132 14 核心的多核中央处理器（CPU）和 4 个 GPU 卡。每个计算结点拥有 256GB 内存（2 个 CPU 共用）。所配 GPU 卡型号为 NVIDIA Tesla V100 SXM2，配备 16GB HBM2 显存。目前配置的 GPU 驱动为 460.73.01 版本。

GPU_A800 结点队列（GPU_A800 分区）中每个计算结点包含 2 个 Intel(R) Xeon(R) Gold 6348 28 核心的多核中央处理器（CPU）和 8 个 GPU 卡。每个计算结点拥有 1024GB 内存（2 个 CPU 共用）。所配 GPU 卡型号为 NVIDIA A800，配备 80GB HBM2e 显存。目前配置的 GPU 驱动为 525.85.12 版本。

1.2 文件系统

启明系统 GPUFS 文件系统信息如下表所示：

名称	集群	类型	存储默认限额	数据备份政策
GPUFS	tianhe2k	lustre	1TB	无，数据文件一旦删除后将无法恢复

“/GPUFS/系统账号名”为用户默认家目录，用于代码存储、程序编译、数据存放和作业运行，用户的所有文件将存放在各自的家目录下。所有登录服务结点和计算结点都可以访问用户家目录下的文件。对于试用用户，其存储限额每用户不超过 1TB；对于签约用户，其存储限额将根据用户的具体需求进行配置。如果您的数据量超出该限额，可发邮件到客户服务邮箱<service.nscc@nscc-gz.cn>商谈磁盘限额购买事宜。

1.3 软件环境

1.3.1 操作系统

启明系统（tianhe2k）操作系统为 CentOS Linux release 7.6.1810。

1.3.2 作业调度系统

启明系统采用 SLURM 作业调度系统，SLURM 系统介绍可见 SLURM 官网：https://slurm.schedmd.com/

1.3.3 编译环境

目前，启明系统已配置 GNU、Intel 以及 CUDA 和 PGIcompiler/17.1 编译环境，支持 C，C++，Fortran77 和 Fortran90 语言程序的开发；能够为运用大规模并行英伟达 GPU 的应用程序加速。同时，启明系统支持 OpenMP 和 MPI 两种并行编程模式。其中 OpenMP 为共享内存方式，仅能在一个计算结点内并行，最大线程数不能超过结点处理器核心数（常规 work 计算结点核心数为 36）；MPI 是分布式内存并行，计算作业直接跨结点运行，可以在一个或者若干个结点上进行，最大进程数仅受用户帐号所能调用的计算结点总数限制。

共享内存的 OpenMP 并行方式通常由编译器来支持，目前 GNU 和 Intel 的编译器均已实现了对该标准的支持。

目前启明系统上已部署的 GNU、Intel 以及 CUDA 编译器常用版本如下表所示：

编译器类型	版本号	模块名
Intel	15.0.1	intelcompiler/15.0.1
	18.0.0	intelcompiler/18.0.0
GNU	4.8.5	默认加载
	4.9.4	gcc/4.9.4
	5.5.0	gcc/5.5.0
	6.5.0	gcc/6.5.0
	9.2.0	gcc/9.2.0
CUDA	9.2.88	CUDA/9.2.88
	10.0	CUDA/10.0
	11.0	CUDA/11.0
	11.2	CUDA/11.2
PGI	17.1	PGIcompiler/17.1

目前启明系统上已部署的 MPI 编译器常用版本如下表所示，经测试，在启明系统上 IMPI 的并行效率普遍比其他 MPI 高，建议用户使用 IMPI 进行 MPI 程序的编译与运行。

MPI 类型	版本号
IMPI	15.0.1
	2018
OpenMPI	3.1.4
mvapich2	2.3.2

2 登录

2.1 VPN 验证

VPN 登录操作步骤请见《VPN 客户端使用手册》。

VPN 显示已连接后，可使用“ping 天河系统 IP”命令验证 VPN 是否连接成功，以 Windows 系统为例，可在系统始菜单-> 在 “ 搜索程序和文件 ” 框里输入 “cmd” 调出命令提示符 -> 输入“ ping 172.16.20.20 ”。如 ping 结果显示“连接超时”则说明 VPN 未成功连接。 Linux 或 MAC 系统可直接打开 terminal 窗口输入“ ping 172.16.20.20 ”命令进行验证。

2.2 终端登录

在成功与启明系统建立了 VPN 安全链接后，用户需要通过 ssh 登录方式来使用启明系统计算资源（为了进一步保证用户的数据安全，暂不允许通过 telnet 等方式登录服务器）。用户可以使用 ssh 客户端软件（如 MobaXterm、Putty、Xmanager、SecureCRT）来登录系统。登录步骤如下：

Step 1：从账户经办人处获取账户 Private Key 文件（后缀为.id），通常 Private Key 文件会随账号通知邮件附件给出，为保障账户安全，切忌传播。

Step 2：使用 ssh 客户端工具进行连接，通过使用 Private Key 文件进行认证登录，具体操作可参考下例。

注意：

（1）下文提到的**系统 IP** 为 **172.16.20.20** 。

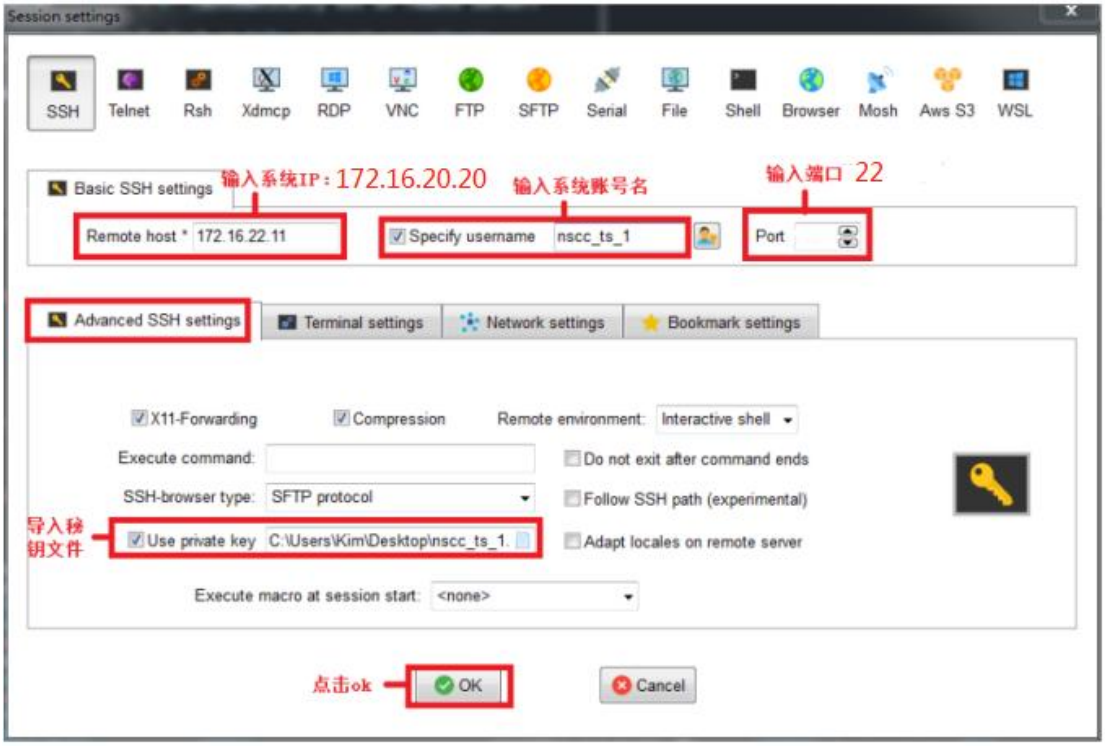
（2）登录系统账户后，请不要修改家目录、密钥所在文件夹（~/.ssh 目录）和密钥文件等的属主、属组及读写权限，否则可能会出现因密钥不符合安全规范而不能登陆的问题。

2.2.1 MobaXterm 登录

打开 MobaXterm，点击工具栏右上角的“Session”：

在“Session settings”弹框中点击“SSH”然后进行账户的登录设置，“Basic SSH settings”中在“Remote host”处填写系统 IP，“Specify username”中填写“用户账户名”，“Port”处填写“22”。

在“Advanced SSH settings”中勾选“Use private key”，载入从账户经办人处获取的 Private Key 文件（后缀为.id），并点击“OK”登录到启明系统。

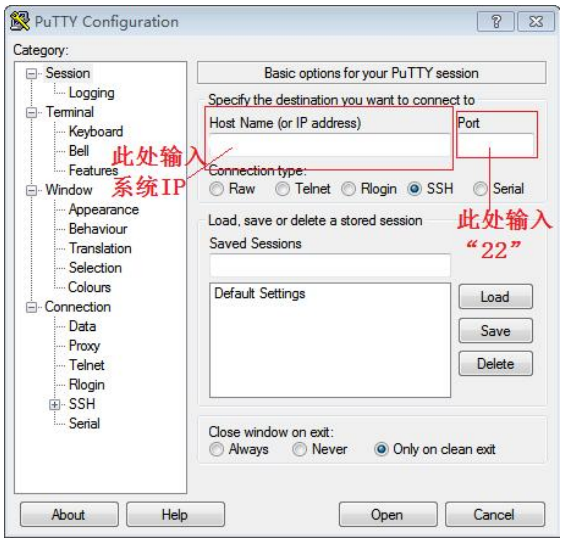


2.2.2 PUTTY 登录

用 PUTTY 登录首先需要转换 Private Key 文件。打开 PUTTY 的安装路径，运行 PUTTYGEN.EXE 程序进行 Private Key 文件转换。

点击“Load”载入 Private Key 文件，点击“Save private key”保存转换后的 Key 文件。

运行 PUTTY.EXE，选择“Session”。在“Host Name”处填写系统 IP：172.16.20.20，“Port”处填写“22”。

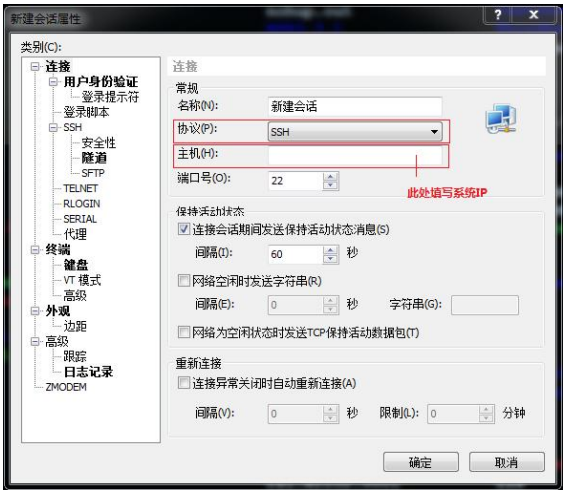


选择“Connection”->“SSH”->“Auth”，然后点击“Private key file for authentication”处的“Browse” 选择转换后的 Private Key 文件，然后点击“Open”打开登录界面。

在登录界面里输入系统账号，回车即可登录。

2.2.3 Xshell（Xmanager-XShell）登录

打开 XShell，点击工具栏的“新建”选项。在“连接”的“常规”里，“协议”选择 SSH，然后在“主机”处填写系统 IP：172.16.20.20。



在“用户身份验证”这里，“方法”选择 Public Key，点击“浏览”导入从账户经办人处获取的 Private Key 文件（后缀为.id），导入后，点击选择账户对应的秘钥，然后点击“确定”即可登录。

2.2.4 Linux 或苹果系统登录

如果是 Linux 或者苹果系统，首先需要给 Private Key 文件（从账户经办人处获取的密钥文件，后缀为.id）设置权限，命令如下：

chmod 400 PrivateKey

然后对系统文件做如下修改：

1）用 root 权限修改 ssh_config 文件（linux 系统路径为/etc/ssh/ssh_config，苹果系统则需根据本机系统配置而定，一般 ssh_config 文件所在路径为 /etc/ssh/ssh_config 或 /etc/ssh_config）：

Linux 系统：sudo vi /etc/ssh/ssh_config

苹果系统：sudo vi /etc/ssh/ssh_config 或 sudo vi /etc/ssh_config

2）在 ssh_config 文件中增加如下两行并保存：

StrictHostKeyChecking no

UserKnownHostsFile /dev/null

重新打开一个终端界面进行登录，使用“ssh 命令的 -i 选项”来指定 Private Key 文件，命令如下所示：

ssh -i PrivateKey 系统账号@系统 IP

2.3 账户管理

2.3.1 账户秘钥重置

为避免 Private Key 泄露导致数据泄露，建议用户在第一次登录时重新生成 Private Key。重置 Private Key 的步骤可参考如下：

第一步：

$ cd ~/.ssh

$ tar cvf bak.tar *

第二步：

$ ssh-keygen -t rsa（一直输入回车；若出现“Overwrite (y/n)?”，请输入“y”。）

$ cd ~/.ssh

$ cp id_rsa.pub authorized_keys

$ chmod 400 authorized_keys

$ cd ..

$ chmod 700 -R .ssh

第三步：

将 id_rsa（Private Key linux 系统版）的内容复制到本地的文本文件中，新建一个终端用此新文件用作 Private Key 文件登录。若登录成功则新的 Private Key 生成成功。若不成功，重复第二、三步操作。若多次操作不成功，请解压第一步中生成的压缩包（$ tar －xvf bak.tar），并使用旧 Private Key 登录。

2.3.2 账户信息查询

系统账户信息会集成在天河星光平台中。用户可通过天河星光平台进行账户信息的查询，包括机时用量、存储容量信息、历史作业等。

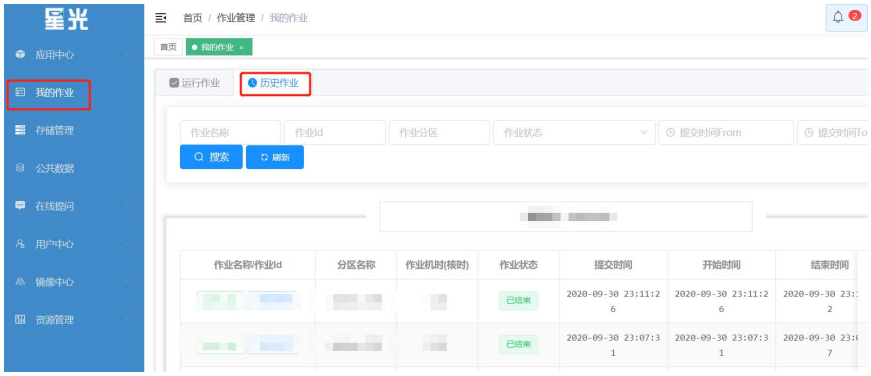
2.3.2.1 机时及存储容量信息查询

在浏览器（建议使用 Google Chrome 浏览器）中输入天河星光网址（https://starlight.nscc-gz.cn/），然后登录天河星光账号（首次登录需注册并按提示绑定 HPC 账号），机时及磁盘存储使用情况会在天河星光账户首页中显示。

2.3.2.2 历史作业查询

用户的历史作业可通过天河星光平台，作业列表中的“历史作业”进行查询。在登录天河星光平台后，可通过点击“我的作业”--“历史作业”中进行查看。

作业显示的信息包括作业名称、作业 id，作业运行使用的分区名称、作业消耗的机时信息、作业状态，以及作业提交\开始\结束时间。



3 数据传输

“启明系统”上的数据传输通常可分为外部数据传输和内部数据传输。其中，外部数据传输是指将数据从外部机器（如本地计算机/服务器）上传到“启明系统”上，或者从“启明系统”上将数据下载到外部机器；内部数据传输是指在“天河二号”或“启明系统”上进行的账号内或者账号之间的数据传输。下文将分别介绍这两种传输的操作方法。

3.1 外部数据传输

外部数据传输方法可分为网络传输和硬盘挂载拷贝。

网络传输：VPN 连接成功后，直接通过互联网向“启明系统”上传/下载数据。

硬盘挂载拷贝：先联系中心账户经办人（一般为发账号通知的人）申请硬盘挂载，将硬盘通过邮寄或者其他方式送到负责人处，然后工作人员将硬盘挂载到数据传输服务器上，用户登录据传输服务器上后通过使用 cp 等拷贝命令传输数据。

用户可根据个人网络传输状态选择传输方式。**若是传输过程花费时间超过 2 天，推荐使用硬盘挂载拷贝方式；2 天以内则推荐网络传输方式。**

3.1.1 网络传输

以下将针对用户本地机器操作系统分别说明外部数据网络传输的操作过程。在进行数据传输前，请留意如下注意事项：

1、外部数据传输服务需提前使用 VPN 验证登录。

2、启明系统中的存储空间只作为数据的临时存储，鉴于存储空间容量限制和数据安全的考虑，请用户及时把重要数据或敏感数据保存到自己的计算机中，并及时清理自己的存储空间。

3、在有大量文件需要下载时，建议使用 tar 命令进行打包，以加快下载速度，减少出错几率。该命令为：tar -cvf　file.tar file，其中 file 为需要打包压缩的文件或目录，file.tar 为打包后的文件。

3.1.1.1 Windows 操作系统

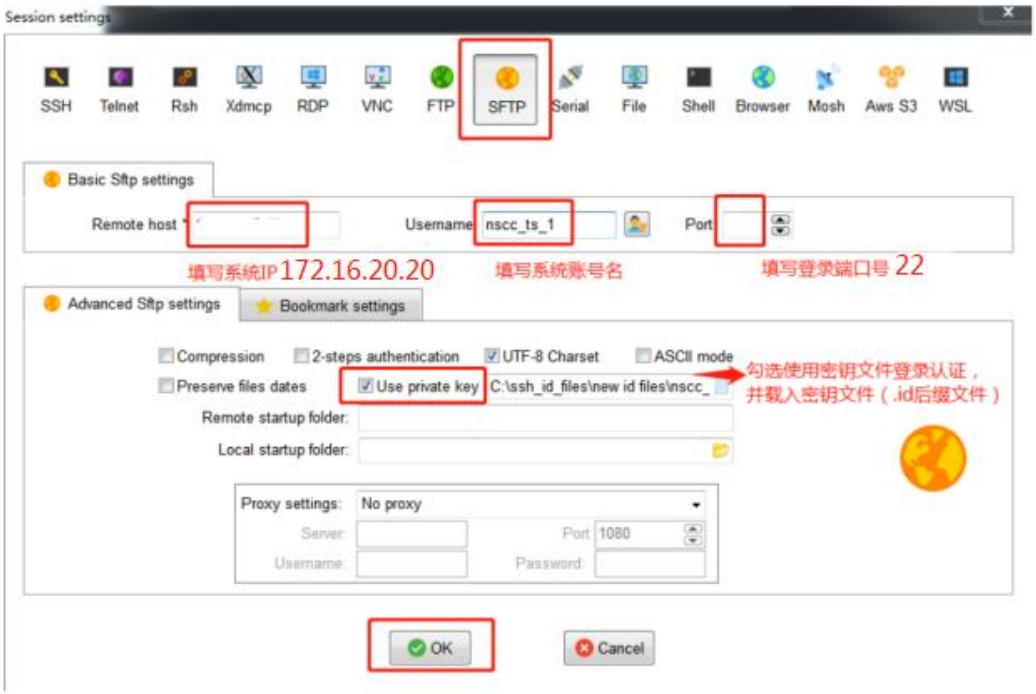
从外部机器向启明系统上传或下载文件，可以使用 sftp 客户端实现，例如 MobaXterm、Xmanager 等本身自带的文件传输功能，或者使用 FileZilla、WinScp 等的 sftp 数据传输软件。

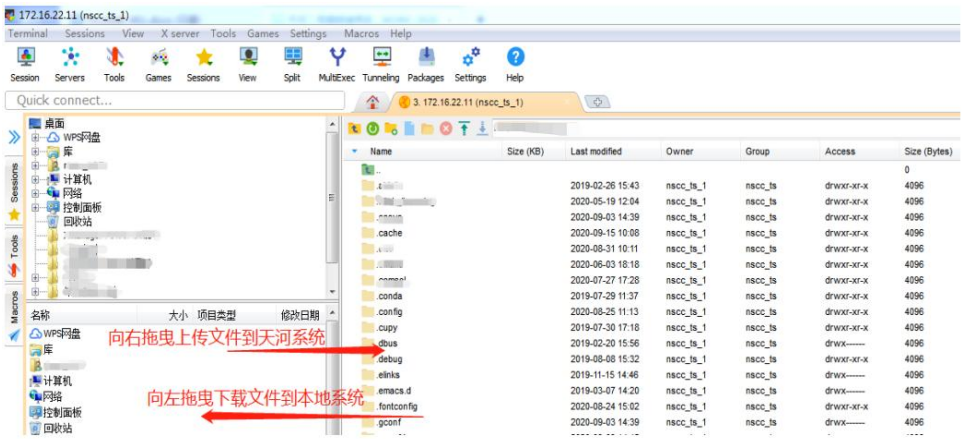
（1）MobaXterm 传输

打开 MobaXterm，点击工具栏右上角的“Session”：

在“Session settings”弹框中点击“SFTP”然后进行账户的登录设置，“Basic Sftp settings”中在“Remote host”处填写系统 IP:172.16.20.20，“Username”中填写“用户账户名”，“Port”处填写“22”。

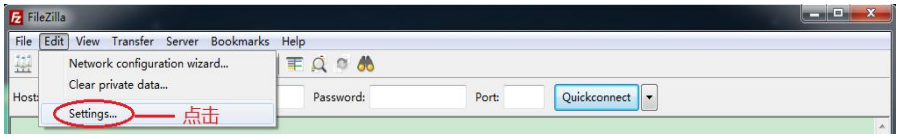
在“Advanced Sftp settings”中勾选“Use private key”，载入从账户经办人处获取的 Private Key 文件（后缀为.id），并点击“OK”进行登录。

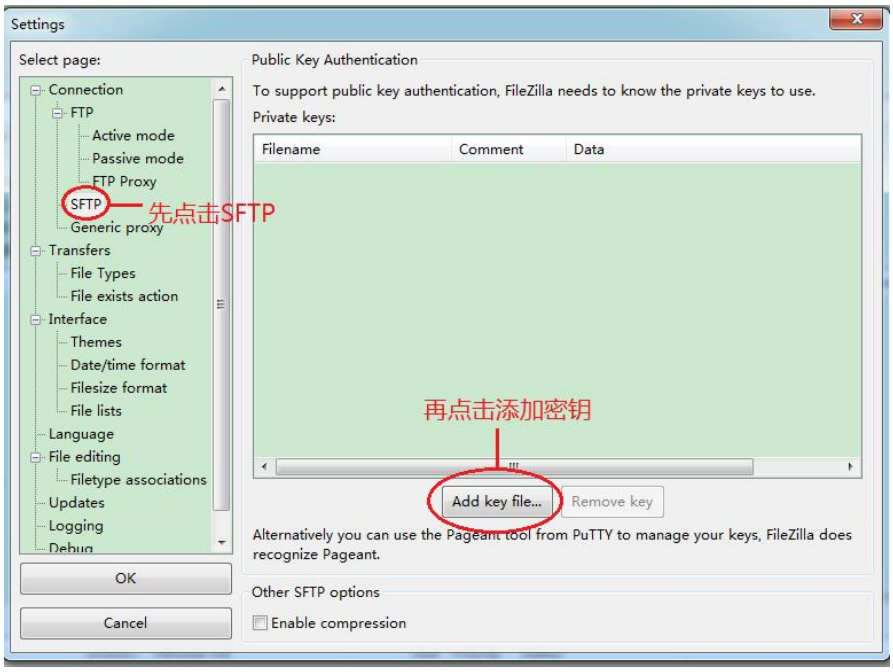


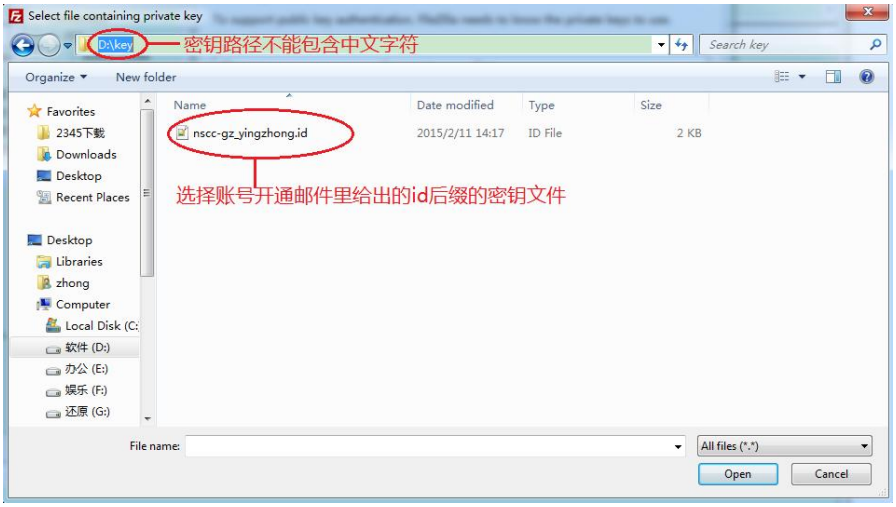


（2）FileZilla 传输

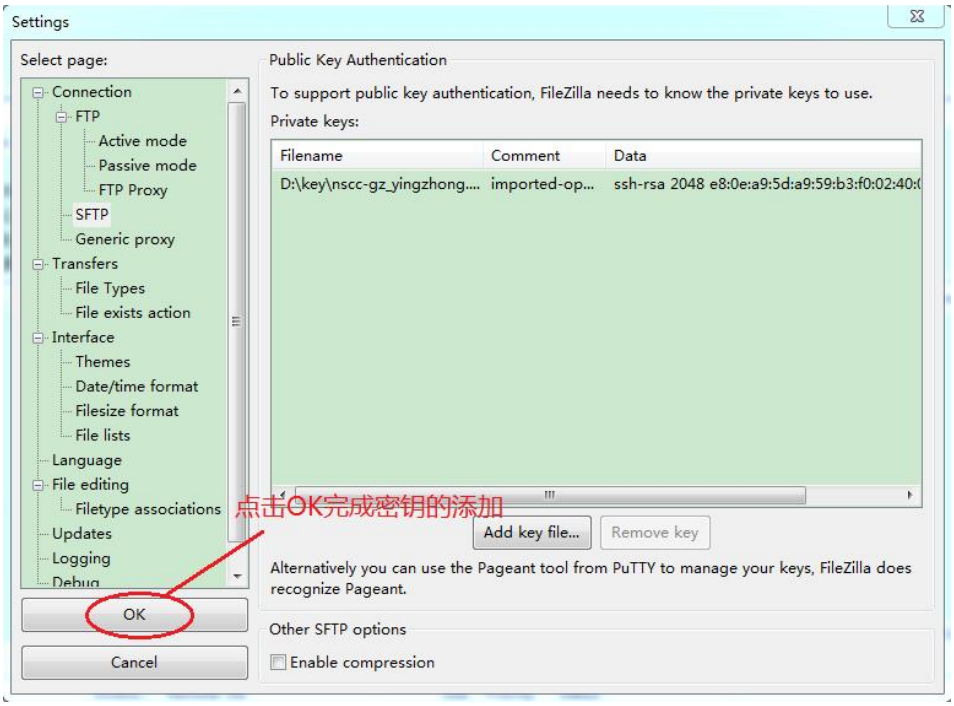
FileZilla 的登录步骤见下图：

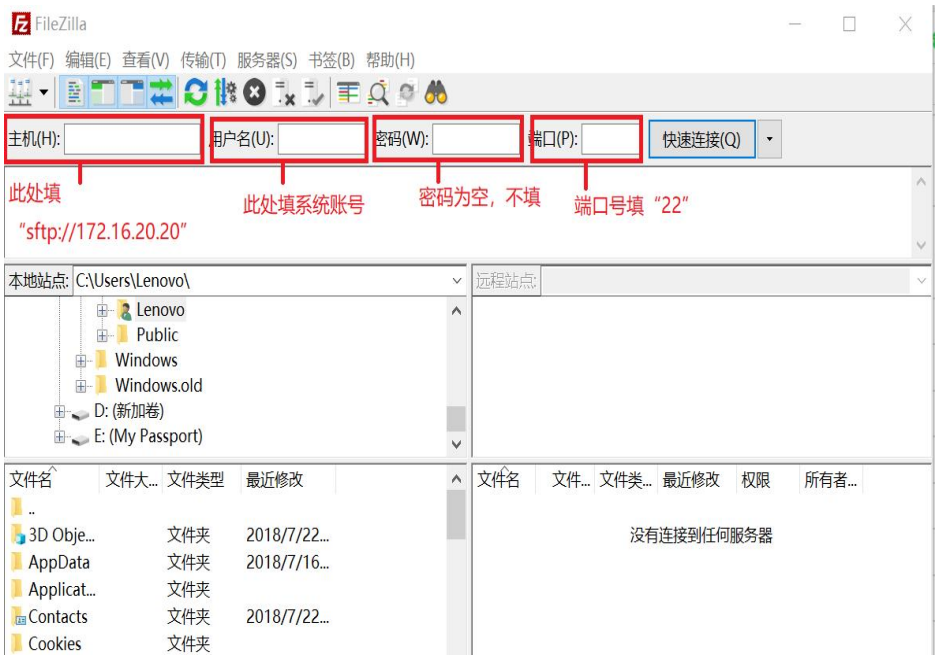




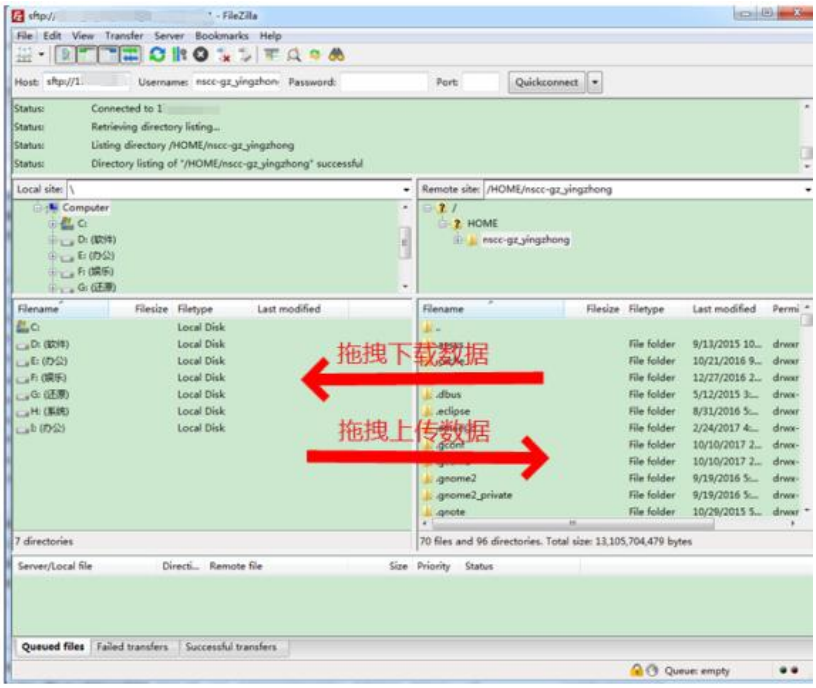


注意：密钥所在路径不能含有中文。



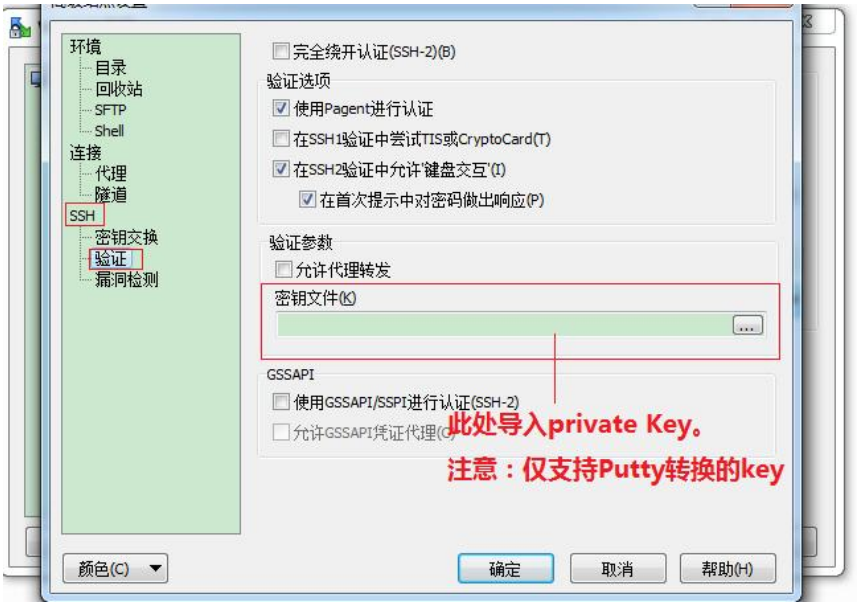

FileZilla
文件(F) 编辑(E) 查看(V) 传输(T) 服务器(S) 书签(B) 帮助(H)
主机(H):
用户名(U):
密码(W):
端口(P):
快速连接(Q)
此处填
"sftp://172.16.20.20"
此处填系统账号
密码为空，不填
端口号填"22"
本地站点: C:\Users\Lenovo\
Lenovo
Public
Windows
Windows.old
D: (新加卷)
E: (My Passport)
远程站点:
文件名 文件大... 文件类型 最近修改
3D Obje... 文件夹 2018/7/22...
AppData 文件夹 2018/7/16...
Applicat... 文件夹
Contacts 文件夹 2018/7/22...
Cookies 文件夹
文件名 文件... 文件类... 最近修改 权限 所有者...
没有连接到任何服务器


Host key mismatch
Warning: Potential security breach!
The server's host key does not match the key that has been cached.
This means that either the administrator has changed the host key,
or you are actually trying to connect to another computer
pretending to be the server.
If the host key change was not expected, please contact the server administrator.
Details for new key:
Host:
Hostkey algorithm: ssh-rsa 2048
Fingerprints: SHA256: c6EWqQ0khYU1XanNpVUh2fmBnI/Vwnf3gnqXwA8s1g0=
MD5: 42:bb:b9:70:88:11:99:19:84:f7:0c:36:bf:18:c9:8e
Trust the new key and carry on connecting?
Update cached key for this host
OK
Cancel
遇到此提示，点击OK


拖拽下载数据
拖拽上传数据

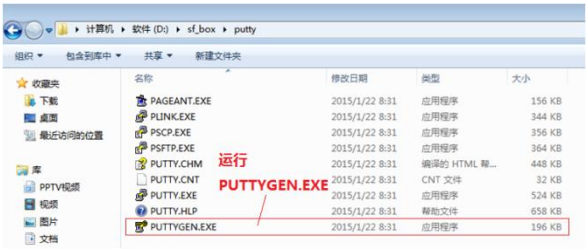
（3）WinSCP 传输

WinSCP 的登录步骤见下图：



Putty 转换可以过程如下：

用 Putty 登录首先需要转换 Private Key 文件。打开 Putty 的安装路径，运行 PUTTYGEN.EXE 程序进行 Private Key 文件转换。



点击“Load”载入 Private Key 文件，点击“Save private key”保存转换后的 Key 文件。

3.1.1.2 Linux 或 Mac 操作系统

VPN 登录成功后，打开一个终端界面进行文件传输。

注意：“系统账号.id”是账号通知邮件中的密钥文件

a）向启明系统上传数据：

scp -i 系统账号.id -r /原文件所在路径 系统账号@172.16.20.20:/账户上存储路径

b）从启明系统下载数据：

scp -i 系统账号.id -r 系统账号@172.16.20.20:/账户上文件所在路径 /本机存储路径

3.1.2 硬盘挂载拷贝

若需要挂载硬盘进行数据拷贝，请联系中心应用推广部（发账号通知的人）或发邮件到客户服务邮箱"客户服务"<service.nscc@nscc-gz.net>申请挂盘服务。

注意：挂盘系统目前支持的接口类型有（SATA，USB，千兆以太网口），可以正确处理的磁盘分区格式有（xfs，ntfs，ext*，nfs），请邮寄符合以上格式要求的磁盘。如果是新硬盘的话，请先进行格式化。

3.2 “启明系统”内部数据传输

内部数据传输是指在“启明系统”上进行的账号内或者账号之间的数据传输。账号内的数据传输是指在同一个账号内的数据转移。通常操作步骤为：登录系统后，使用命令 rsync、cp、mv 等命令进行数据传输，建议使用绝对路径。

账号之间的数据传输是指在两个不同账号之间进行的数据传输。其传输方法，以下将分集群进行叙述。

3.2.1 同集群账户间的数据传输

假设需要在 tianhe2k 集群 accountA 和 accountB 两个账号之间进行数据传输：

首先上传 accountB 账号的密钥（accountB.id）到 accountA 账号中，并在 accountA 账号中修改该密钥权限为 400（chmod 400 accountB.id）；

A）从 accountA 中传输数据到 accountB 中

scp -i accountB.id -r /accountA 中文件所在路径 accountB@localhost:/accountB 中存储路径

B）将 accountB 中传输数据到 accountA 中

scp -i accountB.id -r accountB@localhost:/accountB 中文件所在路径 /accountA 中存储路径

3.2.2 跨集群或文件系统账户间的数据传输

跨集群账户间的数据传输，这里指的是 tianhe2k 集群账户与天河二号的其他集群或文件系统账户（比如 tianhe2c 集群，tianhe2h 集群）进行的数据传输。如用户只使用启明系统 tianhe2k 集群的账户，如下部分内容可忽略。

假设需要在 accountA（tianhe2k）和 accountB（其他集群）两个账号之间进行数据传输：

首先上传 accountB 账号的密钥（accountB.id）到 accountA 账号中，并在 accountA 账号中修改该密钥权限为 400（chmod 400 accountB.id）；

A) 从 accountA 中传输数据到 accountB 中

scp -i accountB.id -r /accountA 中文件所在路径 accountB@tn4:/accountB 中存储路径

B）将 accountB 中传输数据到 accountA 中

scp -i accountB.id -r accountB@tn4:/accountB 中文件所在路径 /accountA 中存储路径

4 环境变量管理

由于不同用户在启明系统上可能需要使用不同的软件环境，配置不同的环境变量，默认配置无法满足所有用户的需要，因而在启明系统上，用户可通过系统上安装的 module 软件来进行对环境变量的管理，也可通过 Linux 命令 export 或 source 的方法设置环境变量。同时，用户也可对家目录下的.bashrc 文件进行配置，以便于账户登录时自动加载常用的计算环境。

4.1 module 环境变量管理工具的使用

module 通过配置 modulefile 支持环境变量的动态修改，能够控制软件不同版本对环境变量的依赖关系。用户通过简单的命令即可获得适于自己环境变量设置。

4.1.1 module 工具的基本命令

启明系统上已配置好 module 工具，主要用法如下：

module avail：查看可用的模块的列表。

```
[nscc_ts_4@ln102 ~]$ module avail
---------------------------------- /app/modulefiles/compiler ----------------------------------
gcc/4.9.4  gcc/5.5.0  gcc/6.5.0  gcc/9.2.0  intel/15.0.1  intel/18.0.1

------------------------------------ /app/modulefiles/MPI -------------------------------------
IMPI/5.0.2.044-icc-15.0.1   mvapich2/2.3.2-gcc-5.5.0   mvapich2/2.3.2-icc-18.0.1
IMPI/2018.1.163-icc-18.0.1  mvapich2/2.3.2-gcc-6.5.0   openmpi/3.1.4-icc-15.0.1
mvapich2/2.3.2-gcc-4.8.5    mvapich2/2.3.2-gcc-9.2.0   openmpi/3.1.4-icc-18.0.1
mvapich2/2.3.2-gcc-4.9.4    mvapich2/2.3.2-icc-15.0.1

---------------------------------- /app/modulefiles/application ----------------------------------
grads/2.2.0      lammps/31May19-icc-18.0.1  screen/4.8.0-gcc-4.8.5
HiQ/0.0.1-py3.7  NCL/6.6.2                  WRF/4.2-icc-15

---------------------------------- /app/modulefiles/library ----------------------------------
curl/7.62.0-icc-15  mpc/0.8.1             netcdf/4.5.0-icc-15     readline/8.0-gcc-4.8.5
gmp/4.2.4           mpfr/2.4.2            proxy/1.0               zlib/1.2.8-icc-15
hdf5/1.8.21-icc-15  ncurses/6.1-gcc-4.8.5  readline/6.3-gcc-4.8.5

---------------------------------- /app/modulefiles_GPU/compiler ----------------------------------
CUDA/9.2.88  CUDA/10.0  CUDA/10.1.2  CUDA/11.0  CUDA/11.2  PGIcompiler/17.1
```

module load [modulesfile]：能够加载需要使用的 modulefiles。

```
[nscc_ts_4@ln102 ~]$ icc
bash: icc: command not found
[nscc_ts_4@ln102 ~]$ module load  intel/18.0.1
[nscc_ts_4@ln102 ~]$ icc -v
icc version 18.0.1 (gcc version 4.8.5 compatibility)
[nscc_ts_4@ln102 ~]$
```

module 其它用法，可在 help 中查询。

```
[nscc_ts_4@ln102 ~]$ module --help
Modules Release 4.3.0 (2019-07-26)
Usage: module [options] [command] [args ...]

Loading / Unloading commands:
  add | load      modulefile [...]  Load modulefile(s)
  rm | unload     modulefile [...]  Remove modulefile(s)
  purge                             Unload all loaded modulefiles
  reload | refresh                  Unload then load all loaded modulefiles
  switch | swap   [mod1] mod2       Unload mod1 and load mod2

Listing / Searching commands:
  list            [-t|-l]           List loaded modules
  avail   [-d|-L] [-t|-l] [-S|-C] [--indepth|--no-indepth] [mod ...]
                                    List all or matching available modules
  aliases                           List all module aliases
  whatis          [modulefile ...]  Print whatis information of modulefile(s)
  apropos | keyword | search  str   Search all name and whatis containing str
  is-loaded       [modulefile ...]  Test if any of the modulefile(s) are loaded
  is-avail        modulefile [...]  Is any of the modulefile(s) available
  info-loaded     modulefile        Get full name of matching loaded module(s)

Collection of modules handling commands:
  save            [collection|file] Save current module list to collection
  restore         [collection|file] Restore module list from collection or file
  saverm          [collection]      Remove saved collection
```

4.2 export / source 命令加载环境变量

Linux 系统的 export 命令可用于设置或添加当前的环境变量，以供后续执行的程序使用。一般可通过配置 PATH 变量设置可执行程序的运行路径；可通过配置 LD_LIBRARY_PATH 及 LIBRARY_PATH 变量设置库环境；可通过配置 INCLUDE 及 CPATH 变量设置头文件环境。如下以添加 gromacs 软件运行环境为例：

假设 gromacs 软件安装的路径为：/GPUFS/app/gromacs/2020.3-icc-18，该路径下包含 bin 可执行程序目录，lib64 库目录，以及 include 头文件目录。

加载可执行命令的环境可执行：export PATH=$PATH:/GPUFS/app/gromacs/2020.3-icc-18/bin

加载运行库的环境可执行：export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/GPUFS/app/gromacs/2020.3-icc-18/lib64 及 export LIBRARY_PATH=$LIBRARY_PATH:/GPUFS/app/gromacs/2020.3-icc-18/lib64

加载头文件的环境可执行：export CPATH=$CPATH:/GPUFS/app/gromacs/2020.3-icc-18/include 及 export INCLUDE=$INCLUDE:/GPUFS/app/gromacs/2020.3-icc-18/include

另外，也可将以上的 export 命令写到一个 bash 脚本中，然后执行“source bash 脚本”来加载以上的变量环境。

4.3 .bashrc 的配置方法

.bashrc 文件是一个可以配置用户环境变量的系统文件,这是一个隐藏文件,位于账户家目录（/GPUFS/系统账号名）下。当每次执行登录操作时，系统都会首先去读取宿主目录下的.bashrc 文件。因此，如用户需要配置登录时默认加载的环境变量，可将该环境变量设置写入.bashrc 文件中。如下为配置.bashrc 的步骤：

Step1：.bashrc 位于账户家目录下，执行 ls -A 可看到该文件，编辑.bashrc 文件可执行 vi ~/.bashrc

Step2：保存.bashrc 文件中添加的设置后，可通过重新登录终端，或者使用命令 source ~/.bashrc 使之生效。

```
[nscc_ts_2@ln42%tianhe2-H ~]$ module list
No Modulefiles Currently Loaded.
[nscc_ts_2@ln42%tianhe2-H ~]$ source ~/.bashrc
[nscc_ts_2@ln42%tianhe2-H ~]$ module list
Currently Loaded Modulefiles:
 1) intelcompiler/18.0.0
[nscc_ts_2@ln42%tianhe2-H ~]$
```

注意：.bashrc 文件是系统配置文件，其中的环境变量设置要谨慎添加。某些环境变量的设置可能会影响其他软件应用的正常使用，比如添加 anaconda 的环境会影响可视化作业的启动。**如您的系统账户同时拥有多个使用者，建议将各自的计算环境写在不同的 env.sh 脚本中，登录天河系统时分别执行 source env.sh 进行计算环境的加载。**

5 程序编译及软件安装

目前，集群上已配置的 GNU 和 Intel 编译器，支持 C，C++，Fortran77 和 Fortran90 语言程序的开发。支持 OpenMP 和 MPI 两种并行编程模式。其中 OpenMP 为共享内存方式，仅能在一个计算结点内并行，最大线程数不能超过结点处理器核心数；MPI 是分布式内存并行，计算作业可以在一个或者若干个结点上进行，最大进程数仅受用户帐号所能调用的计算结点总数限制。共享内存的 OpenMP 并行方式通常由编译器来支持，目前 GNU 和 Intel 的编译器均已实现了对该标准的支持。同时系统上也配置了相应的 CUDA 编译环境和 PGIcompiler/17.1。集群目前没有设置 GPU 默认编译器版本，编译前需使用 module 命令加载相关环境。

注意：用户在进行程序编译时请不要使用超过 8 核、或同时编译 20 多个程序等操作，这类操作会耗费大量系统资源，影响用户的体验。若经发现，可能会取消用户相关进程；严重者将被暂时禁止登录。

5.1 编译器

5.1.1 GCC 编译器

系统默认安装的 GNU 编译器版本是 4.8.5，相关的编译命令安装在/usr/bin 目录中。

5.1.2 Intel 编译器

集群上已配置 15.0.1 以及 18.0.1 版本的 Intel 编译器。没有默认加载，用户若想使用 Intel 编译器可使用 module 进行环境加载。具体命令如下：

使用 Intel 18 编译器：module load intel/18.0.1

注意：查找编译命令所在的路径可以使用 which 命令，例如“which icc”将返回当前使用的 icc 命令所在的具体路径。确认编译器的版本请在编译命令后使用 -v 或者-V 参数，例如“icc -v”、“ifort -V”，Intel 编译器的详细命令行调用则可以用“icc --help”获得。

5.1.3 CUDA 编译器

CUDA 编译器可支持 C、C++ 以及 Fortran 语言，能够为运用大规模并行英伟达 GPU 的应用程序加速。启明系统上已安装了 CUDA/11.2 版本编译器，没有默认加载，需使用 module 进行环境加载。具体命令如下：

使用 CUDA/11.2 编译器：module load CUDA/11.2

5.1.4 PGI 编译器

系统配置了 PGI/17.1 版本编译器。没有默认加载，用户若需加载 PGI 编译器可使用 module 进行环境加载。具体命令如下：

使用 PGI 17.1 编译器：module load PGIcompiler/17.1

5.1.5 MPI 编译环境

目前系统上已安装了 IMPI，mvapich 和 openmpi 三种 MPI 编译环境，没有默认加载，需使用 module 进行环境加载。具体命令如下：

使用 IMPI/2018.1.163-icc-18.0.1 编译器：module load IMPI/2018.1.163-icc-18.0.1

使用 mvapich2/2.3.2-icc-18.0.1 编译器：module load mvapich2/2.3.2-icc-18.0.1

使用 openmpi/3.1.4-icc-18.0.1 编译器：module load openmpi/3.1.4-icc-18.0.1

请注意：使用 openmpi 编译的程序，提交作业时需要使用 mpirun 且指定网卡来提交，提交脚本内容形式如下(其中“程序名”请根据实际计算情况作修改)：

```
#!/bin/bash
yhrun -N $SLURM_NNODES -n $SLURM_NTASKS hostname > hostlist
mpirun --mca btl openib,self,vader --mca btl_openib_if_include mlx5_2 --machinefile hostlist -np $SLURM_NTASKS 程序名
```

作业提交命令为“**yhbatch -N 结点数 -n 核数 -p 分区名 脚本名**”，如“yhbatch -N 1 -n 36 -p work run.sh”。其中结点数 N 和核数 n 对应脚本中的 $SLURM_NNODES 和 $SLURM_NTASKS ，用户需根据实际计算规模在提交命令中进行 N，n 参数的设置；“分区名”可通过 yhi 命令查看（yhi 命令的使用方法可参考本手册 5.1 小节）。关于 openmpi 程序的作业提交方式，如有疑问，欢迎咨询 techsupport@nscc-gz.cn 。

非 openmpi 程序的作业提交方法可参考本手册第 6 章节内容。

5.2 程序编译示例

编译器是通过命令行执行，最基本的用法（以 icc 为例）： icc [options] [filenames]，实际编译过程中，比较常用的是分成编译和链接两步。如下可见是几个编译示例。

注意：如下示例中运行编译后程序均使用的是 yhrun 的交互式方式，在实际作业运行过程中，建议用户使用批处理作业 yhbatch 的作业提交方式，具体使用方法见本手册 6.2.2 章节。

5.2.1 单个源文件示例

hello.c 源码：

```
#include<stdio.h>
int main(int argc, char* argv[]){
printf("Hello world");
return 0;
}
```

编译：icc -c hello.c -o hello.o

链接：icc hello.o -o hello

运行编译出的程序：yhrun -n 1 ./hello

5.2.2 多个源文件示例

hello_main.c 源码：

```
#include<stdio.h>
int main(int argc,char* argv[])
{
   sayHello();
   return 0;
}
```

hello_sub.c 源码：

```
#include<stdio.h>
```

```
int sayHello()
{
 printf("hello");
 return 0;
}
```

编译：

```
icc -c hello_main.c -o hello_main.o
icc -c hello_sub.c -o hello_sub.o
```

链接：

```
icc hello_main.o hello_sub.o -o hello
```

运行编译出的程序：

```
yhrun -n 1 ./hello
```

5.2.3 openmp 程序编译

openmp 程序是共享内存式多线程并行,是通过编译器支持，icc 的编译参数是为-openmp，gcc 的编译参数为-fopenmp

omp.c 源码如下：

```
#include <stdio.h>
#include <omp.h>
int main (int argc, char *argv[])
{
int tid=-1;
#pragma omp parallel
{
tid = omp_get_thread_num();
printf("%d : Hello World!\n", tid);
}
 return 0;
}
```

编译命令：

icc -openmp omp.c -o omp

设置环境变量：

export OMP_NUM_THREADS=4

运行编译出的程序：

yhrun -N 1 -n 1 -c 4 ./omp

5.2.4 MPI 程序编译

mpi_hello.c 源码如下：

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
    int MyID, NumProcess, NameLen;

    // MPI_MAX_PROCESSOR_NAME Maximun computer name
    char Processor_Name[MPI_MAX_PROCESSOR_NAME];
    // MPI program starts
     MPI_Init(&argc, &argv);
     MPI_Comm_rank(MPI_COMM_WORLD, &MyID); // Tag of current process
     MPI_Comm_size(MPI_COMM_WORLD, &NumProcess); // Total number of
processes
     MPI_Get_processor_name(Processor_Name, &NameLen); // Computer name of current
process

     printf("Process No.%d of %d on %s \n\n", MyID, NumProcess, Processor_Name);
    // MPI program ends
      MPI_Finalize();
    return 0;
}
```

编译过程：

mpicc -c mpi_hello.c -o mpi_hello.o

mpicc mpi_hello.o -o mpi_hello

运行编译出的程序：

yhrun -n 4 ./mpi_hello

5.3 常规软件安装方法

软件通常是以源代码或者预编译包的形式提供，需要将其编译为二进制的机器代码才能进行使用。如果软件提供预编译包，可以根据平台操作系统下载相应的预编译包直接进行使用；如果软件提供的是源码，需要进行编译，不同服务器操作系统、硬件等环境是有差异的，建议不要在服务器之间拷贝可执行文件，而是重新编译。

一般软件的编译包含下面几个步骤：

1）下载源码文件，并上传至系统账号下

2）解压安装包

（1）. tar 或 tar.gz 或 tar.bz2 格式的解压命令：tar -xf xxx.tar 或 tar -xf xxx.tar.gz 或 tar -xf xxx.tar.gz.bz2

（2）. zip 格式的解压命令：unzip xxx.zip

3) 进入解压后的目录，配置、编译和安装软件

（1）查看是否有 configure 或 Makefile 文件，有 configure 文件的话执行命令“./configure --prefix=/GPUFS/系统账号/安装路径”来配置软件，其中--prefix 参数用于指定安装路径，需指定为自己账号下目录。如果直接有 Makefile，可跳过 configure 过程。

（2）使用“make 命令”编译软件

（3）编译完成后执行 make install 命令安装软件。

如用户需要安装 python 库，可使用 pip / pip3 方法将 python 库安装在账户目录下，pip / pip3 支持离线安装 wheel 和 tar.gz 格式的包，如：

pip install --user /path/to/ase-3.14.1.tar.gz

或

pip3 install --user /path/to/ase-3.14.1-py2-none-any.whl

注意：

1、可能会有额外的步骤需要设置，请详细查看官网上给出的安装步骤。

2、用户是没有管理员权限的，无法使用 sudo 命令。

3、安装的时候注意将软件安装路径执行在自己的系统账号下。

6 作业提交与管理

启明系统使用 slurm 进行资源管理与调度，用户提交作业后，作业进入调度状态，资源管理系统根据启明系统的实时资源使用情况来决定该作业的状态，系统有资源时，作业即可进入运行状态。目前启明系统的结点使用方式是独占式，各个作业之间使用的结点是独立的，不存在相互干扰的情况。

为了高效利用系统资源，目前对用户账号的资源使用情况进行了一定的限制。若需要查看或调整资源限制，可以登录天河星光（https://starlight.nscc-gz.cn）的资源管理页面；或者联系应用推广部账号负责人。

6.1 作业管理

用户登录后，建议看一下系统的情况，例如查看一下分区结点的空闲情况以及账号下作业运行情况。以便进行下一步的操作。

6.1.1 结点状态查看 yhinfo 或 yhi

yhi 为 yhinfo 命令的简写，用于查看结点状态。

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhi
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
bigmem       up   infinite      5   idle fn[04-08]
dcv-lava     up   infinite      3   idle lava[01-03]
gpu_v100     up   infinite      1   drng gpu54
gpu_v100     up   infinite     29  alloc gpu[1-6,10,12,14-16,20-23,25-32,49-51,55-56,58]
gpu_v100     up   infinite      6   idle gpu[9,11,13,17-18,53]
Delta        up   infinite     18  alloc cpn[1-3,5-6,8-9,12,14,17,19-21,30,33-34,37,41]
Delta        up   infinite     24   idle cpn[4,7,10-11,13,15-16,18,22-29,31-32,35-36,38-40,42]
Gold         up   infinite     31  alloc cpn[1-3,5-6,8-9,12,14,17,19-21,30,33-34,37,41,103-105,1
08-110,112,115-118,120,122]
```

其中 PARTITION 表示分区，NODES 表示结点数，NODELIST 为结点列表，STATE 表示目前结点所处的状态。在 STATE 中，idle 表示结点处于空闲状态，该状态的结点可以被分配作业；alloc 表示结点已经被占用，有作业在运行。当 idle 状态的结点较少时提交作业，作业可能会处于 PD（排队）状态。

6.1.2 作业状态信息查看 yhqueue 或 yhq

yhq 为 yhqueue 命令的简写，用于查看作业运行情况。

```
[nscc_ts_4@ln102 test]$ yhq
             JOBID PARTITION     NAME         USER ST       TIME  NODES NODELIST(REASON)
              8171  gpu_v100     bash    nscc_ts_4  R       0:43      1 gpu6
```

其中 JOBID 表示任务 ID，NAME 表示任务名称，USER 为用户，TIME 为已运行时间，NODES 表示占用结点数，NODELIST 为任务运行的结点列表。

作业状态通常有 R、PD、S、CG、JobHeldAdmin 或者 AssociationResourceLimit 几种。R 表示作业正在执行；PD 表示目前没有资源，作业处于排队状态，等待资源释放即可；S 表示作业被挂起，系统正在维护中，维护完毕会恢复作业运行状态；**CG 表示作业已结束，但是没有正常退出，该状态下作业不耗机时，管理员会定期处理 CG 状态的作业，耐心等待该作业释放即可**；JobHeldAdmin 表示作业因为资源调度由管理员暂时锁定，待资源满足时会被释放并进入作业调度；AssociationResourceLimit 表示用户总的结点数已经达到资源限制。

注意：推荐使用“yhq -a”查看作业状态信息。对于已无机时的账号，使用 yhi 和 yhq 无法查看分区和作业状态，此时若还需查看作业状态信息，使用“yhq -a”命令可查看。

6.1.3 任务取消 yhcancel

用户使用 yhcancel 命令取消自己的作业。命令格式如下：yhcancel jobid

jobid 可通过 yhq 获得。对于排队作业，取消作业将简单地把作业标记为 CANCELLED 状态而结束作业。对于运行中的作业，取消作业将终止作业的所有作业步，包括批处理作业脚本，将作业标记为 CANCELLED 状态，并回收分配给作业的结点。

```
[nscc_ts_2@lon26%tianhe2-H ~]$ yhq -a
             JOBID PARTITION     NAME         USER ST       TIME  NODES NODELIST(REASON)
           3632961  bigdata     bash     nscc_ts_2  R       0:45      1 cn7524
[nscc_ts_2@lon26%tianhe2-H ~]$ yhcancel 3632961
[nscc_ts_2@lon26%tianhe2-H ~]$ yhq -a
             JOBID PARTITION     NAME         USER ST       TIME  NODES NODELIST(REASON)
[nscc_ts_2@lon26%tianhe2-H ~]$ yhacct -j 3632961
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
3632961            bash    bigdata    nscc_ts         24 CANCELLED+      0:0
[nscc_ts_2@lon26%tianhe2-H ~]$
```

6.2 作业提交

启明系统的提交方式主要包括批处理作业提交方式 yhbatch 和交互式作业提交方式 yhrun，也可以使用 yhalloc 抢占结点进行提交。下述对几种提交方式进行简要的介绍。

6.2.1 交互式作业提交 yhrun

在 shell 窗口中执行 yhrun 命令，主要命令格式如下： yhrun [options] program，该提交方式资源分配与任务加载均通过 yhrun 命令进行。yhrun 是交互式的作业提交方式，终端中断，提交的作业也会中断，如果用户程序需要交互，请选择该方式，如果不需要，请使用批处理作业提交方式 yhbatch。

6.2.1.1 yhrun 常用选项

yhrun 包括多个选项，其中最常用的选项主要有以下几个：

- **-n, --ntasks=number**

指定运行的进程数，缺省的情况下，默认值为 1

- **-N, --nodes=minnodes[-maxnodes]**

指定作业运行时至少分配多少个结点。可以通过 maxnodes 限制最多分配的结点数目（例如“-N 2-4”或“--nodes=2-4”）。如果作业使用结点数超出账号下的总结点数限制，作业将被拒绝。如果没有指定-N，是按照-n 和-c 参数的值来进行分配结点满足需求。

- **-p, --partition=partition name**

在指定分区中分配资源。如未指定，则在系统默认分区中分配资源。

- **-w, --nodelist=node name list**

指定结点列表。作业分配的资源中将至少包含指定的这些结点。列表可以用逗号分隔的结点名或结点范围（如 cn[1-5,7,...]）指定，或者用文件名指定。如果参数中包含“/”字符，则会被当作文件名。如果指定了最大结点数如-N 1-2，但是文件中有多余 2 个结点，则请求列表中只使用前 2 个结点。

- **-x, --exclude=node name list**

不要将指定的结点分配给作业。如果包含“/”字符，参数将被当作文件名。

- **-c, --cpus-per-task=ncpus**

指定线程数，缺省的情况下，默认值为 1

- **-h, --help**

若需使用 yhrun 更多选项，可通过“yhrun –h”或“yhrun --help”查看。

6.2.1.2 使用示例

1） 在 work 分区上的使用示例：

A: 在分区 work 上指定作业数运行 hostname：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -n 4 -p work hostname
cpn225
cpn225
cpn225
cpn225
[nscc_ts_4@ln102%tianhe2-K ~]$
```

B：在分区 work，结点 cpn[212,232]上运行 hostname：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -n 4 -p work -w cpn[212,232] hostname
cpn212
cpn232
cpn212
cpn232
[nscc_ts_4@ln102%tianhe2-K ~]$
```

C: 在 work 分区，运行 4 作业的 hostname，每个结点一个作业，分配的结点中至少包含结点 cpn[212,232]：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -N 4 -n 4 -p work -w cpn[212,232] hostname
cpn212
cpn232
cpn248
cpn244
[nscc_ts_4@ln102%tianhe2-K ~]$
```

D: 在 work 分区，运行 4 作业的 hostname，每个结点一个作业，分配的结点中不包含结点 cpn[212,232]：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -N 4 -n 4 -p work -x cpn[212,232] hostname
yhrun: job 86307 queued and waiting for resources
yhrun: job 86307 has been allocated resources
cpn248
cpn203
cpn202
cpn244
```

备注：以上例子中是将作业提交至 work 分区下运行。用户作业也可以根据实际计算需要将作业提交到其他计算分区，比如将-p 参数改为“-p bigmem”，即可将作业提交至 bigmem 分区下运行。另外，例子 B、C 中使用-w 参数指定的结点需在-p 参数指定的结点分区下，否则作业提交会出错，如结点 cpn[212,232]属于 work 分区下的结点（yhi 命令可查看此项信息）。

2） 在 gpu_v100 分区上的使用示例：

A: 在分区 gpu_v100 上指定作业数运行 hostname：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -n 4 -p gpu_v100 hostname
gpu6
gpu6
gpu6
gpu6
[nscc_ts_4@ln102%tianhe2-K ~]$
```

B: 在分区 gpu_v100，结点 gpu[53-54]上运行 hostname：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -n 4 -w gpu[53,54] -p gpu_v100 hostname
gpu54
gpu53
gpu53
gpu54
[nscc_ts_4@ln102%tianhe2-K ~]$
```

C: 在 gpu_v100 分区，运行 4 作业的 hostname，每个结点一个作业，分配的结点中至少包含结点 gpu[53-54]：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -N 4 -n 4 -w gpu[53,54] -p gpu_v100 hostname
gpu53
gpu54
gpu61
gpu62
[nscc_ts_4@ln102%tianhe2-K ~]$
```

D: 在 gpu_v100 分区，运行 4 作业的 hostname，每个结点一个作业，分配的结点中不包含结点 gpu[53-54]：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhrun -N 4 -n 4 -x gpu[53,54] -p gpu_v100 hostname
gpu57
gpu55
gpu56
gpu58
[nscc_ts_4@ln102%tianhe2-K ~]$
```

备注：以上例子中是将作业提交至 gpu_v100 分区下运行。用户作业也可以根据实际计算需要将作业提交到其他计算分区，比如将-p 参数改为“-p GPU_A800”，即可将作业提交至 GPU_A800 分区下运行。另外，例子 B、C 中使用-w 参数指定的结点需在-p 参数指定的结点分区下，否则作业提交会出错，如结点 gpu[53-54] 属于 gpu_v100 分区下的结点（yhi 命令可查看此项信息）。

6.2.2 批处理作业 yhbatch

批处理作业是指用户编写作业脚本，yhbatch 提交脚本到后台执行作业。用户在登录结点使用 yhbatch 提交作业后，在命令行显示作业号，作业进入调度状态，在资源满足要求时，所提交的作业开始运行。在该过程中，yhbatch 负责资

源分配，在资源满足要求分配完计算结点之后，系统将在所分配的第一个计算结点（而不是登录结点）上加载执行用户的作业脚本。使用 yhbatch 提交作业终端关闭不受影响，登录结点维护也不受影响，如果没有交互式的需求，请使用该方式进行作业提交。

批处理作业的脚本为一个文本文件，脚本第一行以“#!”字符开头，并制定脚本文件的解释程序，如 sh，bash。由于计算结点为精简环境，只提供 sh 和 bash 的默认支持。

yhbatch 提交时的各项参数，可以使用 yhbatch --help 进行查看，常用的参数有：-N 指定结点数，-n 指定核数，-p 指定分区名，-w 将指定的结点列表分配给作业，**-x** 不要将指定的结点分配给作业。

6.2.2.1 使用示例

1）在 work 分区上的使用示例：

例如用户的脚本名为 myjob.sh，内容如下：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ vi myjob.sh

#!/bin/bash
yhrun -n 4 -N 4 -p work hostname
~
```

根据该脚本用户提交批处理作业，需要明确申请的资源为 work 分区的 4 个结点。

```
[nscc_ts_4@ln102%tianhe2-K ~]$ ll myjob.sh
-rw-r--r-- 1 nscc_ts_4 nscc_ts 45 Nov  8 11:35 myjob.sh
[nscc_ts_4@ln102%tianhe2-K ~]$
```

注意：需给该文本文件设置 myjob.sh 可执行权限，利用命令：chmod +x myjob.sh

```
[nscc_ts_4@ln102%tianhe2-K ~]$ chmod +x myjob.sh
[nscc_ts_4@ln102%tianhe2-K ~]$ ll myjob.sh
-rwxr-xr-x 1 nscc_ts_4 nscc_ts 45 Nov  8 11:35 myjob.sh
[nscc_ts_4@ln102%tianhe2-K ~]$
```

用户 yhbatch 批处理命令为： yhbatch -N 4 -p work ./myjob.sh

计算开始后，工作目录中会生成以 slurm 开头的.out 文件为输出文件。

更多选项，用户可以通过 yhbatch --help 命令查看。

备注：以上例子中是将作业提交至 work 分区下运行。用户作业也可以根据实际计算需要将作业提交到其他计算分区，比如将-p 参数改为“-p bigmem”，即可将作业提交至 bigmem 分区下运行。

2）在 gpu_v100 分区上的使用示例：

例如用户的脚本名为 mybash.sh，内容如下：

```
[nscc_ts_4@ln102%tianhe2-K test]$ vi mybash.sh

#!/bin/bash

yhrun -n 4 -N 4 -p gpu_v100 hostname
```

根据该脚本用户提交批处理作业，需要明确申请的资源为 gpu_v100 分区的 4 个结点。

```
[nscc_ts_1@ln102%tianhe2-K test]$ ll
total 4
-rw-r--r-- 1 nscc_ts_1 nscc_ts 44 Jul 17 11:17 mybash.sh
[nscc_ts_1@ln102%tianhe2-K test]$
```

注意：需给该文本文件设置 mybash.sh 可执行权限，利用命令：chmod +x mybash.sh

```
[nscc_ts_1@ln102%tianhe2-K test]$ chmod +x mybash.sh
[nscc_ts_1@ln102%tianhe2-K test]$ ll
total 4
-rwxr-xr-x 1 nscc_ts_1 nscc_ts 44 Jul 17 11:17 mybash.sh
[nscc_ts_1@ln102%tianhe2-K test]$
```

用户 yhbatch 批处理命令为： yhbatch -N 4 -p gpu_v100 ./mybash.sh

计算开始后，工作目录中会生成以 slurm 开头的.out 文件为输出文件。

更多选项，用户可以通过 yhbatch --help 命令查看。

备注：以上例子中是将作业提交至 gpu_v100 分区下运行。用户作业也可以根据实际计算需要将作业提交到其他计算分区，比如将-p 参数改为“-p GPU_A800”，即可将作业提交至 GPU_A800 分区下运行。

6.2.3 结点资源抢占命令 yhalloc

该命令支持用户在提交作业前，抢占所需计算资源（此时开始计算所用机时）。使用该命令后，如果终端中断，作业也会中断。

6.2.3.1 使用示例

1）yhalloc 提交方式在 work 分区上的使用示例：

首先申请资源，执行如下命令：

```
[nscc_ts_4@ln102%tianhe2-K ~]$ yhalloc -N 1 -p work
yhalloc: Granted job allocation 87544
[nscc_ts_4@ln102 ~]$
```

通过 yhq 查看相应的 jobID 为 87544，结点为 cpn209。

```
[nscc_ts_4@ln102 ~]$ yhq
             JOBID PARTITION     NAME         USER ST       TIME  NODES NODELIST(REASON)
             87544      work     bash    nscc_ts_4  R       0:37      1 cpn209
[nscc_ts_4@ln102 ~]$
```

用户可以选择用 ssh 命令登陆到计算节点：

```
[nscc_ts_4@ln102 ~]$ ssh cpn209
Last login: Fri Nov  8 09:14:22 2019 from 89.72.31.3
[nscc_ts_4@cpn209%tianhe2-K ~]$
```

切换到 cpn209 结点，之后执行程序。

备注：以上例子是申请 work 分区下的结点资源。用户作业也可以根据实际计算需要申请其他计算分区的资源，比如将-p 参数改为“-p bigmem”，即可申请 bigmem 分区的结点资源。

2）yhalloc 提交方式在 gpu_v100 分区上的使用示例：

首先申请资源，执行如下命令：

```
[nscc_ts_4@ln102%tianhe2-K test]$ yhalloc -N 1 -p gpu_v100
yhalloc: Granted job allocation 8171
[nscc ts 4@ln102 test]$
```

通过 yhq 查看相应的 jobID 为 8171，结点为 gpu6。

```
[nscc_ts_4@ln102 test]$ yhq
             JOBID PARTITION     NAME         USER ST       TIME  NODES NODELIST(REASON)
              8171  gpu_v100     bash    nscc_ts_4  R       0:43      1 gpu6
[nscc_ts_4@ln102 test]$
```

用户可以选择用 ssh 命令登陆到计算节点：

```
[nscc_ts_4@ln102 test]$ ssh gpu6
Last login: Wed Oct 31 09:33:39 2018 from ln102
[nscc_ts_4@gpu6%tianhe2-K ~]$
```

切换到 gpu6 结点，之后执行程序。

备注：以上例子中是将作业提交至 gpu_v100 分区下运行。用户作业也可以根据实际计算需要将作业提交到其他计算分区，比如将-p 参数改为“-p GPU_A800”，即可将作业提交至 GPU_A800 分区下运行。

6.2.4 部分典型脚本示例

6.2.4.1 单结点多任务提交

如果有需求在单个结点内提交多个任务，可参照如下脚本 run.sh：

```
#!/bin/bash
yhrun -N 1 -n 1  程序名 1 &
yhrun -N 1 -n 1  程序名 2 &
yhrun -N 1 -n 1  程序名 3 &
……
yhrun -N 1 -n 1  程序名 num &
wait
```

$yhbatch -N 1 -p 分区名 run.sh（脚本名）#作业提交命令

注意：脚本中的任务数请不要超过 10 个，如果单个结点内提交过多的任务，会导致系统调度卡顿，影响系统的健康运行，若影响较大，会被管理员杀掉作业，并暂时禁止提交作业。

6.2.4.2 前一个作业完成，下一个作业自动提交

示例如下：
假设有 a.sh 和 b.sh 两个脚本作业需要运行，其中 a.sh 运行完成后，b.sh 才可运行。
a.sh

```
#!/bin/bash
yhrun -n 1 program1
```

b.sh

```
#!/bin/bash
yhrun -N 2 -n 48 program2
```

run.sh

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

```
LAST=$(yhbatch -N 1 -p 分区名 a.sh | awk '{print $4}')
yhbatch -d afterok:$LAST  -N 2 -p 分区名 b.sh
$chmod +x run.sh   #脚本加可执行权限
$./run.sh   #直接./运行 run.sh 脚本
```

说明：
“--dependency=afterok:${SLURM_JOB_ID} ”参数实现等待前一个作业完成了，下一个作业在自动提交。

6.2.4.3 csh 脚本提交

脚本第一行可写为：
#!/usr/bin/csh

6.3 备注

由于手册篇幅限制，如您有其他需求也可以联系超算中心技术人员。

1） 合同、资源申请、产品咨询相关问题联系方式：

邮箱： service.nscc@nscc-gz.cn

电话: （020）37106026

2） 系统使用、作业运行相关问题联系方式：

邮箱： techsupport@nscc-gz.cn

电话：（020）37106031

重要提示：

1） 请不要在登录结点直接运行可执行程序（极大的影响其他用户的登录和使用效率）。

2） 如无特殊需求，请使用批处理方式（yhbatch）提交任务，如有问题请联系超算中心技术人员。

3） 请保存好运行程序的 log 文件，从而方便超算中心技术人员在作业出问题后，协助解决问题。

4） 提交时需在提交命令中加入参数选项“-p 分区名”，即提交命令应为“yhrun -p 分区名 ...”或者“yhbatch -p 分区名”。同时，推荐用户使用 yhbatch 方式提交作业。分区名请通过 yhi 命令查看获得，其中 PARTITION 一栏对应的就是分区，如下图所示。

```
[nscc_ts_4@ln102 test]$ yhi
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
gpu_v100     up   infinite     18  alloc gpu[1-8,10-11,15-17,19,32,50,60,63]
gpu_v100     up   infinite     26   idle gpu[9,12-14,18,20-30,51,53-59,61-62]
work         up   infinite     59   idle cpn[1-2,4-60]
test         up   infinite      0    n/a
[nscc_ts_4@ln102 test]$
```

分区名

7 帮助与支持

7.1 名词解析

编译：将程序的源代码转化为计算机可以识别运行的 2 进制文件（0 和 1），是软件安装（Install）的前置步骤。如果习惯了 Windows 下的软件安装，你大可以将编译理解为在 Linux 安装软件的一个必须步骤就可以了。

提交作业：即一般通俗理解的提交研究“任务”。为了更准确的理解本文档，我们这里先定义好“作业”，“任务”，“进程”和“线程”的含义以及他们之间的关系。“作业”的结果是我们想要的结果，也是最终结果，而为了完成“作业”必须选择怎样将“任务”们分派出去。而一个“任务”对应一个“进程”，一个“进程”又能对应多个“线程”。“线程”之于“进程”就如“任务”之于“作业”。

举个简单的例子，你的“作业”是用一堆书填满一个空的图书馆，这时你可以设定为了完成这个“作业”而分配 3 个“任务”，任务 1 填满图书馆第一层，任务 2 填满图书馆第二层，任务 3 填满图书馆第三层，每个任务找 2 个人来完成。最后结果就是一个“作业”开 3 个“任务”，每个“任务”等于一个“进程”，每个“进程”开 2 个“线程”。只有当所有“任务”都正常完成后，你的“作业”才能完成。

加速比：按照上面的例子，似乎一个作业只要不停提高进程和线程的数量就能无限线性提高作业完成速度，但在实际中并没那么简单。设定一个作业开一个进程一个线程时的完成时间为 1，那么同样的作业，开 4 个进程，每个进程开 4 个线程，是不是完成时间只需要原来的 1/16？答案是否定的，实际是不可能达到，甚至有可能出现比原来花了更多时间的情况。单进程完成的时间除以多进程完成的时间就是所谓的**加速比**。加速比与程序的算法，并行度设计，作业的规模相关。如何选择合理的任务数和线程数以最少的价钱在可以接受的时间内完成作业，要根据经验来决定。

进程并行和线程并行：参考图书馆的例子，这个作业同时打开了进程并行和线程并行。但在实际上，作业能够采取什么并行是有限制的，这个和程序设计有关，请在选择前先查看程序说明书。如果程序没有说明，这里提供一个简单粗略的判定方法：如果程序在编译时调用了-openmp，就表示这个程序可以开线程并行，如果没有就表示一个进程只能开一个线程；如果程序的并行执行命令调用是mpirun，就表示这是一个进程并行的程序。这些概念在后面设置提交作业参数时需要用到。

交互式作业提交：交互式作业提交就像两个人面对面对话，当一个人离开的时候，对话就无法继续了。天河 2 号设定为，一个交互式作业在客户断开与天河 2 号连接时，视为客户主动取消。所以长时间的作业请务必使用 yhbatch 提交。

7.2 常见问题

7.2.1 登录问题

（1） VPN 账号连接成功，但是终端工具连接不了启明系统

若出现该现象，首先请查看您的电脑是否安装 360 卫士、安全卫士等软件，若安装了请先将软件关闭，再重新连接 VPN；若上一步完成后仍无法连接，请 ping 系统 IP，查看丢包率，若丢包率很高则是您的网速导致，若丢包率低，则请联系中心相关人员排查。

（2） 系统登录时报“用户密钥未在远程主机上注册”

首先检查账号与秘钥是否匹配，检查账号前后是否空格等特殊字符；检查端口是否填写正确；检查秘钥所在路径是否有中文字符，如果有的话，请转移到没有中文字符的位置。

7.2.2 系统资源问题

（1） 使用“yhi”命令查看没有分区显示

出现该问题可能是账号没有机时，建议按照上面的操作方式查看机时，看一下机时是否超额，若已超额，请联系应用推广部账号负责人购买机时。

（2） 账号下无法写入，出现 Disk quota exceeded

这是由于账号下数据量已经超过存储限额，建议清理一下账号下数据。

（3） “ls”等访问文件夹操作很慢

出现“ls”等访问文件夹操作慢的原因主要有 3 个：一是网络慢，网络时延大；二是有大量的 IO 操作正在进行，造成 IO 阻塞；三是该文件夹下的文件过多（有成千上万个文件）。若是原因一和二，通常等一段时间后即可恢复正常；若是原因三，则需用户将自己文件夹下的文件分开存放。

7.2.3 作业提交及运行问题

（1） 提交作业报“Invalid partition name specified”。

报该错时，建议用户先用“yhi”查看是否可以看见自己所在的分区。若无法看见分区，则是您的机时已到限制。

（2） 提交作业报“Failed to allocate resources: User's group not permitted to use this partition”。

用户提交作业时通常需要加“-p 分区名”这一参数，同时该参数应写在程序名前。分区可用“yhi”来查看所在分区。

（3） 使用“yhi”查看有空闲结点，但是提交作业后在排队

这是因为在您前面有其他用户的作业在排队，该作业提交时间比您提交的早，所以优先满足该作业的需求，即使该作业需求的结点数较多，空闲结点无法满足该作业。

（4） 采用 yhrun 提交作业，关闭界面后，再次登录时发现作业被 killed。

yhrun 是交互式提交作业模式，一旦作业提交的界面关闭作业就会被 killed。若需要较长时间运行的作业，建议用户采用 yhbatch 批处理提交方式。yhbatch 负责资源分配，yhbatch 获取资源后会在获取资源的第一个结点运行提交的脚本，当前登录 shell 断开后，加载作业仍可正常运行。

（5） yhalloc 分配资源，退出 yhalloc 后发现作业断掉。

yhalloc 与 yhbatch 最主要的区别是，yhalloc 命令资源请求被满足时，直接在提交作业的结点执行相应任务，适合需要指定运行结点和其他资源限制，并有特定命令的作业。当当前登录 shell 断开后，申请获得的资源以及加载作业任务会退出。

（6） 计算结点无法登录。

目前我们对计算结点做了限制，除非用户分配了计算结点，否则无法登录。用户若想登录计算结点再算题，首先需要用 yhalloc 分配结点，方可登录结点算题。

（7） 作业退出后仍显示 CG 状态，是否影响作业退出？

CG 状态是作业退出时，部分结点上的进程没有完全停止导致，并不影响作业的正常退出。

（8） 作业完成退出时显示部分进程被 killed，然后退出。

这种情况下，用户首先应检查所需的输出是否已正常输出完成。导致这种情况出现的原因是有部分进程先完成了计算而提前结束，而当一个作业的部分进程结束，系统默认为作业已完成，在一定时间内其他进程若不结束，则会被强制结束。

（9） 作业运行时报库无法找到，如何处理？

可以使用“module ava”查看是否已经安装这个库文件，安装了的话，可以直接“module load 相应版本的库文件”来加载环境再提交。没有安装，可以下载库的源码，在自己的系统账号下进行安装，安装完后设置环境变量 ：“export LD_LIBRARY_PATH=库安装路径:$LD_LIBRARY_PATH”。

（10） 作业报错 forrtl: severe (41): insufficient virtual memory

若是有临时文件输出，请将文件输出位置指定到自己账号下。若您的程序是 MPI 的可以同增加结点数来扩大内存试试。

（11） 作业运行中报错，提示 node faild

结点运算过程中发生了死机，请跟我们联系处理。

（12） 作业运行提示“forrtl: Input/output error”

可能是存储某一时刻压力较大，造成 IO 错误，请重新提交作业，并跟我们进行反馈。

（13） 作业运行一段时间后无输出

出现这个现象的原因很多，需要具体问题具体分析，其中一种可能原因是用户磁盘配额已满，无法写入数据导致的，您可以登录天河星光查看存储，如果确实已超限额，建议清理数据后重新提交作业；如果并非这种情况请您邮件或者电话与我们联系，我们将协助您进行查看。

附录 A：常用 unix 命令

A.1 浏览目录和文件命令

ls ：用来显示目录的内容。配合参数的使用，能以不同的方式显示目录内容。命令格式：ls [参数] [目录名或文件名]

pwd:可以用来显示当前所在目录的绝对路径。命令格式：pwd

cd ：可以让用户切换当前工作的目录。 命令格式：cd [目录名]

cat：用于显示文件的内容,也可以将多个文件合并成一个文件。命令格式：cat [参数] <文件名>

less：用来浏览超过一页的文件。命令格式：less [参数] <文件名>

head：用于显示文件前几行的内容。命令格式：head [参数] <文件名>

tail：用于显示文件最后几行的内容。命令格式：tail [参数] <文件名>

A.2 目录和文件操作命令

chmod：用于修改文件或目录的权限。命令格式：chmod <参数> <文件或目录名>

mkdir(make directory)：可用来创建新目录。命令格式：mkdir [参数] <目录名>

rm(remove directory)：用来删除目录或文件。命令格式：rm [参数] <目录名或文件名>。慎用该命令，文件删除后是无法找回的。

mv（move）：可以将文件或目录移到另一个目录下，或更改文件及目录的名称。命令格式：mv <源文件或目录> <目标文件或目录>

cp(copy)：可以用来复制文件或目录，使用时需要指定源文件路径与目标文件路径或源目录路径与目标目录路径。 命令格式：cp [参数] <源文件路径> <目标文件路径> 或 cp [参数] -r <源目录路径> <目标目录路径>

注：复制目录时必须使用-r 参数。

find：用来查找文件或目录的位置。命令格式：find [<路径>] [匹配条件]

tar：可以将用户所指定的文件或目录打包成一个文件，提高文件传输的效率。也可以对打好包的文件将里面的文件会目录解出来。命令格式：tar <参数> <打包成的文件名> <要打包的文件或目录> 或 tar <参数> <打包成的文件名>

A.3 文本编辑工具 vi

vi：是 Linux 下面的命令行文本编辑工具，具有强大的功能。进入 vi 的命令 vi filename :打开或新建文件，并将光标置于第一行首。

A.3.1 移动光标类命令

上:k nk:向上移动 n 行

下:j nj:向下移动 n 行

左:h nh:向左移动 n 列

右:l nl:向右移动 n 列

gg 可以移到第一行

G 移到最后一行

A.3.2 输入控制命令

：w ：保存当前文件

：q：退出 vi

：q!：不保存文件并退出 vi

：wq：保存当前文件并退出

：x：保存当前文件并退出

dd：删除当前行

yy：复制当前行

p：粘贴复制或删除的内容

/pattern：从光标开始处向文件尾查找 pattern

?pattern：从光标开始处向文件首查找 pattern

n：在同一方向重复上一次搜索命令

N：在反方向上重复上一次搜索命令

：s/p1/p2/g：将当前行中所有 p1 均用 p2 替代

：n1,n2s/p1/p2/g：将第 n1 至 n2 行中所有 p1 均用 p2 替代

：g/p1/s//p2/g：将文件中所有 p1 均用 p2 替换