1 基础配置建议

- 集群网站：`pbdcc.gzu.edu.cn`
- 集群访问IP：`210.40.56.85`
- 集群访问端口：`21563`

ssh密钥登录

每次登录都需要输入密码，麻烦且耗时！

解决办法：设置密钥登录。

step 1 生成密钥对

在本地操作，二选其一即可：

```
ssh-keygen -t ed25519 -C "your_email@example.com"

ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

执行完之后用户家目录的`.ssh`目录中(形如`C:\Users\XXX/.ssh/`)会产生公钥密钥对(如：公钥`id_ed25519.pub`密钥`id_ed25519`).

step 2 上传公钥

将刚刚生成的公钥内容复制粘贴到超算的指定文件。

```
# 登录超算
ssh [username]@210.40.56.85 -p 21563
# 在超算上创建 .ssh 目录（如果没有）
mkdir -p ~/.ssh
chmod 700 ~/.ssh

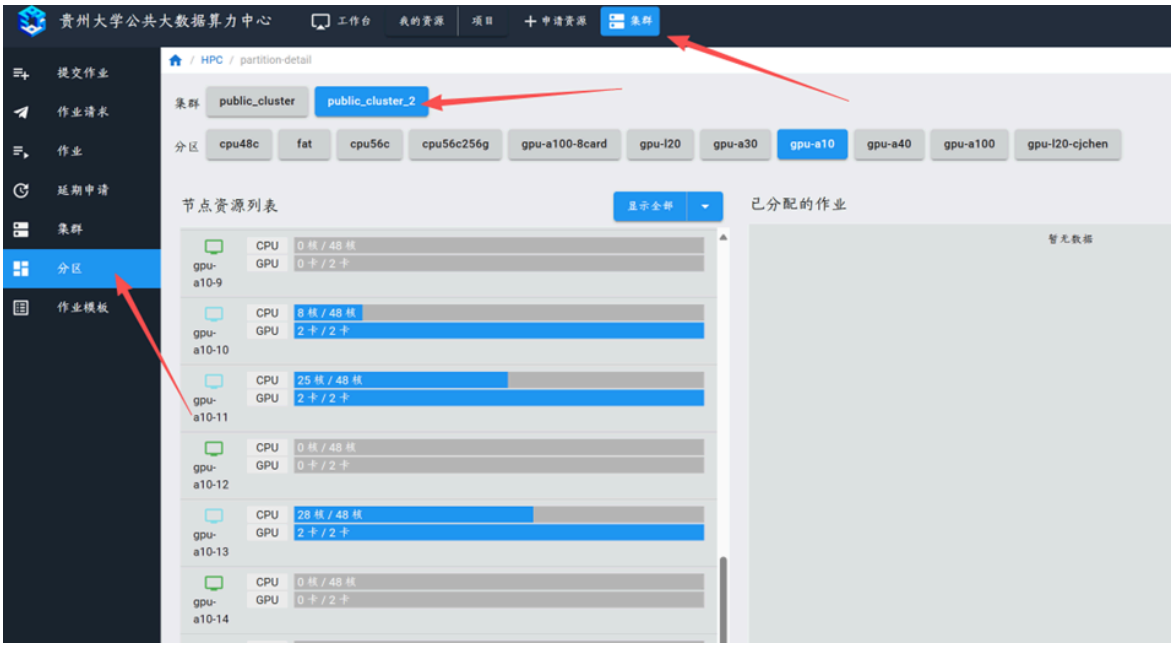
# 粘贴内容：
# 1打开/新建文件
touch ~/.ssh/authorized_keys
# 2.写入内容，注意替换内容！！！
echo "公钥" >> ~/.ssh/authorized_keys
# 文件赋权
chmod 600 ~/.ssh/authorized_keys
```

经上述两步，执行`ssh [username]@210.40.56.85 -p 21563`可无密码登录。

查看资源空闲情况

两种方式：

i. 进入集群官网，依次点击：集群->分区->public_cluster_2



ii. 命令行输入`sinfo`，idle为空闲节点，mix为部分已分配节点，alloc为完全已分配节点

```
[demo_@workstation ~]$ sinfo
PARTITION      AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu48c*           up 7-00:00:00      2    mix cpu48c-[1,5]
cpu48c*           up 7-00:00:00      4  alloc cpu48c-[2-4,6]
cpu48c*           up 7-00:00:00     10   idle cpu48c-[7-16]
fat               up 7-00:00:00      1    mix fat-1
cpu56c            up 7-00:00:00      1    mix cpu56c-1
cpu56c            up 7-00:00:00      5   idle cpu56c-[2-6]
cpu56c256g        up 7-00:00:00      9   idle cpu56c256g-[1-9]
gpu-a100-8card    up 7-00:00:00      1  alloc gpu-a100-8card-1
gpu-l20           up 7-00:00:00      3   drng gpu-l20-[1-2,9]
gpu-l20           up 7-00:00:00      4  drain gpu-l20-[4-6,8]
gpu-l20           up 7-00:00:00      1    mix gpu-l20-3
gpu-a30           up 7-00:00:00      2    mix gpu-a30-[1-2]
gpu-a10           up 7-00:00:00     11    mix gpu-a10-[1-8,10-11,13]
gpu-a10           up 7-00:00:00      9   idle gpu-a10-[9,12,14-20]
gpu-a40           up 7-00:00:00     22    mix gpu-a40-[1-18,22-25]
gpu-a40           up 7-00:00:00      3   idle gpu-a40-[19-21]
gpu-a100          up 7-00:00:00      2    mix gpu-a100-[1,3]
gpu-a100          up 7-00:00:00      3  alloc gpu-a100-[2,4-5]
gpu-l20-cjchen    up 7-00:00:00      2  alloc gpu-l20-[7,10]
```

2 交互式作业·基础

最简salloc

仅在命令行中执行`salloc`将会申请到单核CPU节点，常用于配置环境和拷贝数据等简单任务。执行`salloc`会获得像这样的输出：

```
salloc: Granted job allocation 251
salloc: Waiting for resource configuration
salloc: Nodes cpu48c-1 are ready for job
```

最后一行中的“cpu48c-1”就是集群分配的资源，执行`ssh cpu48c-1`即可登录该节点进行作业。

常规salloc

多数深度学习任务需要用到不同的GPU，申请资源时需要用户指定(指定分区、显卡数量等信息)，一般需要指定或建议指定以下信息：

- 分区`--partition=XX`
- 显卡数量`--gres=gpu:X`
- CPU核心数量`-c X`　　可以一张gpu配置8个cpu核心

例如`salloc --partition=gpu-l20 --gres=gpu:8 -c 48`将会申请有8张L20显卡、48核心的节点，会看见类似的数输出：

```
salloc: Granted job allocation 252
salloc: Waiting for resource configuration
salloc: Nodes gpu-l20-1 are ready for job
```

最后一行中的“gpu-l20-1”就是集群分配的资源，执行`ssh gpu-l20-1`即可登录该节点进行作业。

更多salloc参数

直接查看系统手册：

```
salloc --help
```

3 环境配置及其工具

先看“交互式作业·基础”，再申请到的资源上进行操作！！！

conda

conda是最常用的虚拟环境、python包管理和底层依赖管理工具，其最大优势在于**管理底层依赖**。对于python包管理(下载/安装/升级)有更加出色的工具，如：uv。

建议安装 Miniconda，集群上有安装包，直接执行以下命令安装(后根据提示进行安装)：

```
sh /home/share/ Miniconda3-latest-Linux-x86_64.sh
```

新建环境

```
# new_env是新环境的名称，python=3.9指定python版本（可选）
conda create -n new_env python=3.9
```

增加依赖

```
conda install XXX
```

uv(强力推荐)

conda适合管理底层依赖，而uv擅长管理python包(PYPI)和python项目，比起其他工具uv更快、更强、更友好，uv和conda可以搭配使用。

和conda一样，安装依赖只会安装在激活的env中。优点是比conda快，包更全面。缺点是主要是用来python包和依赖。

一句话：能uv解决的优先uv解决，缺什么底层依赖就用conda补齐全。

uv官网

用户自行安装uv若遇到网络问题，可以使用我们编写的安装脚本：

```
ssh /home/share/Official/tools/uv/install.sh
```

4 脚本式作业

交互式作业模式常用于debug和配置环境，存在一个明显缺陷“即申即用”，得在终端等着资源分配。

在只需要运行代码的场景下，脚本式作业更合适，只需提交任务脚本，任务自动排队。

脚本式作业的资源申请和代码运行目录都写在`slurm`文件中，一下介绍如何写slurm脚本。

```
#!/bin/bash
# 作业配置
#SBATCH --job-name=test           # 作业名称（可选，但建议设置该项）
#SBATCH --output=output_%j.log    # 标准输出日志（%j 会被替换为作业ID）（可选）
#SBATCH --error=error_%j.log      # 标准错误日志（可选）

# 资源配置
#SBATCH --nodes=1                 # 申请 1 个节点
#SBATCH --ntasks=1                # 总共 1 个任务（MPI 进程数）
#SBATCH --cpus-per-task=48        # 每个任务使用48个CPU核
#SBATCH --partition=gpu-l20       # 指定分区
#SBATCH --gres=gpu:8              # 指定显卡数量

# 跳转目录(替换自己的工作目录)
cd ~/demo

# 激活虚拟环境(替换自己的虚拟环境路径)
source ~/minionda3/envs/[your_venv]bin/activate
# 启动作业
python main.py

# 或者一步到位（替换前两步）
~/miniconda3/envs/[your_venv]/bin/python main.py
```

写好作业脚本后，用户只需在命令行中执行`sbatch xxx.slurm`(xxx.slurm为作业脚本)即可提交作业，将自动排队执行。

如果有其他需求(如指定节点等)，可以通过`sbatch --help`命令查看手册。

5 交互式作业·进阶

申请到的资源都可以通过IDE(如vscode)连接，进行交互。

在`C:\Users\[username]\.ssh\config`中添加配置：

```
Host cluster
  HostName 210.40.56.85
```

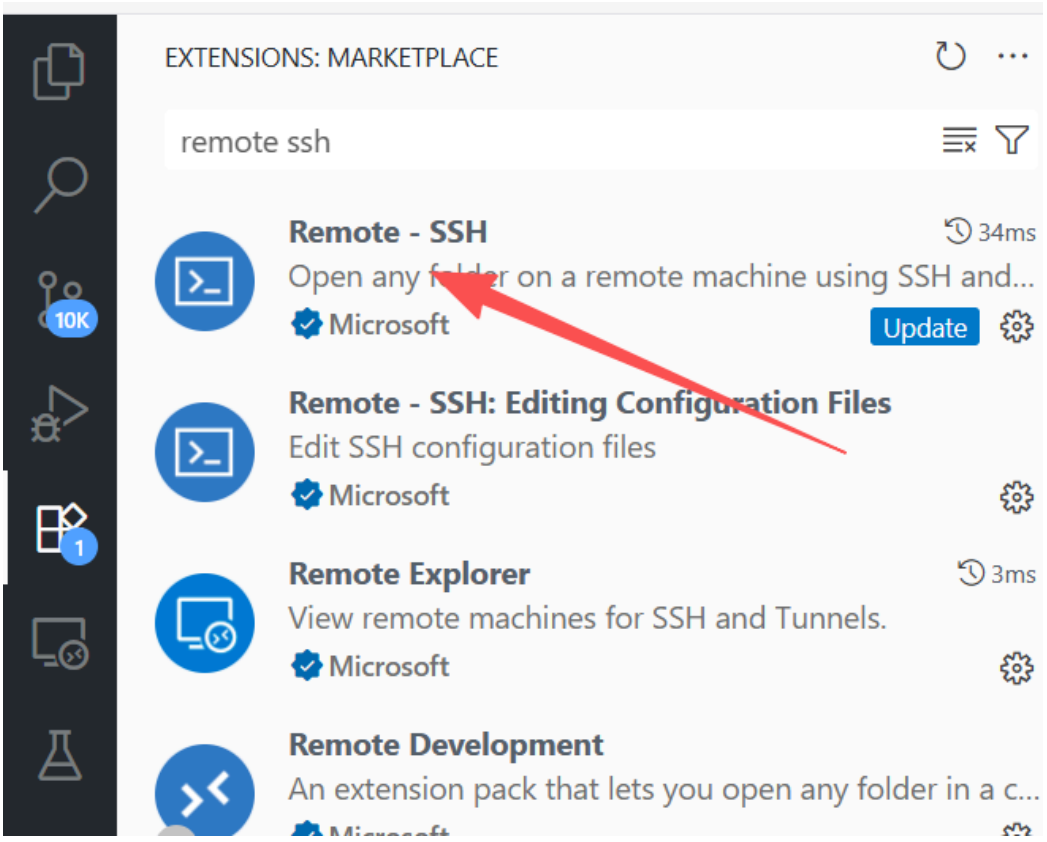
```
    User [your_username]
    Port 21563

Host node
    HostName [your_node_name]
    User [your_username]
    Port 22
    ProxyJump cluster
```

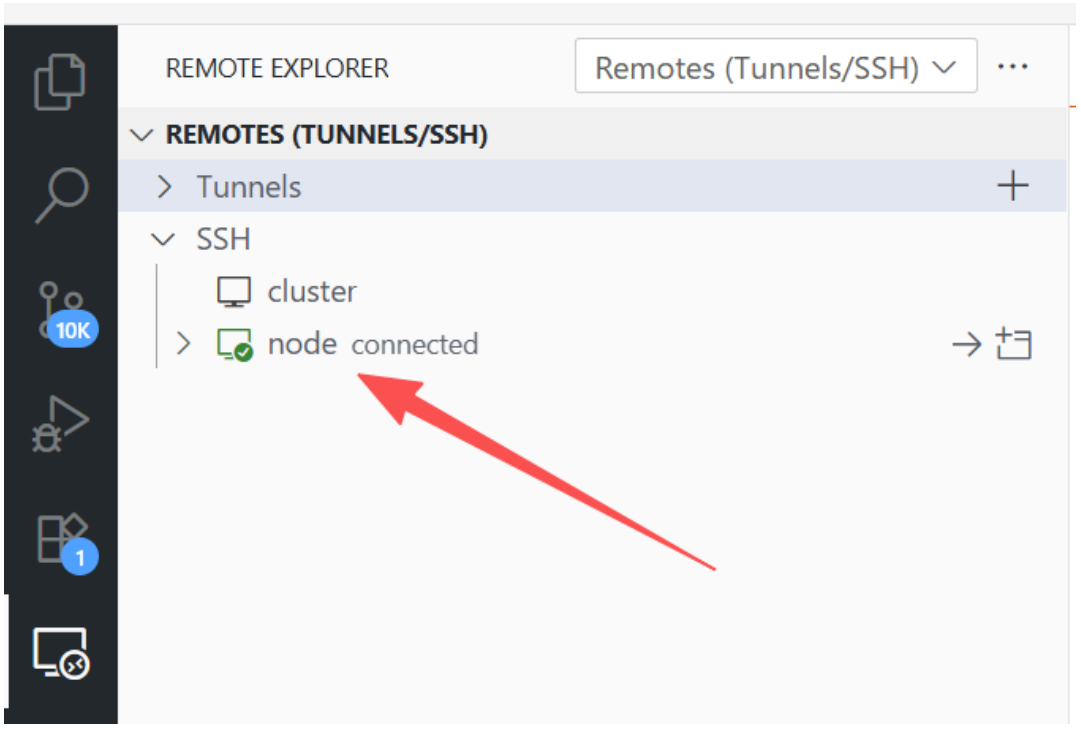
其中`cluster`为集群的登陆节点，`node`为用户申请到的节点。**记得修改`[your_username]`和`[your_node_name]`(申请到的节点)!!!**

vscode交互使用资源

step 1 安装remote ssh插件(可同时安装python语言相关插件)



step 2 选择远程资源


REMOTE EXPLORER
Remotes (Tunnels/SSH)
REMOTES (TUNNELS/SSH)
Tunnels
SSH
cluster
node connected
10K
1