

0.Before Starting

1.登陆

FAQ

2.WebShell

3.配置环境 (非root)

miniconda

直接下载python3配置环境

非root下载pip3

venv

Git

4.安装所需包

Case 1:PyTorch

安装PyTorch

测试PyTorch

5.本地链接到服务器

本地终端ssh

得到服务器私钥

更改id_rsa权限

链接

VScode ssh

本地浏览器打开服务器Jupyter

0.Before Starting

在23年末的时候从钱光武老师那里得知我们SCUPI有自己的GPU了，作为CS专业的学生，肯定是要想办法试一试。恰好当时正好有个大创项目，构思的时候就想着用深度学习，于是就申请了一下这个名额。

虽然名额申请的很顺利，但是尝试使用服务器的过程可以说是历经坎坷，目前有的教程可能都是关于仿真计算的，和我理解的使用GPU训练模型完全不同。于是乎我就想着把整个流程从0开始写下来，也可以方便以后的CS同学。

为了方便阅读，每个地方都会分为两个部分：第一个部分是操作，只会附带少量必要的提醒；第二个部分是对第一个部分操作的解释，还是推荐大家仔细去阅读。

1. 登陆

- 通过<https://121.48.227.161:7091>，进入我们服务器账户的登陆页面，你应该可以看到下图所示的界面。
- 使用你的bb账户与密码登陆进去。你应该可以看到如下的界面。



- 进入到HPC服务。

FAQ

不熟悉浏览器的可能会受到一点阻碍，因为很可能你的浏览器会弹出以下图片或类似的内容。这时候你就看底下应该有类似“了解详情”或能展开之类的按钮，点击之后在详情的最底下应该有类似于“继续连接”或者“直接连接”之类的东西，点击你就可以进入登陆界面了。

⚠ 您与此网站之间建立的连接不安全

请勿在此网站上输入任何敏感信息（例如密码或信用卡信息），因为攻击者可能会盗取这些信息。[了解详情](#)

2. WebShell

纵观各个教程，根本没有提到WebShell，而这个确实对于要训练AI模型的同学最重要的。

在HPC服务的右上角，你会发现WebShell的图标，点进去WebShell，你就可以直接通过ssh链接到远程服务器了，但是这时候你用 `nvidia-smi` 或者类似的检查GPU状态的命令你就会发现此时你根本没有GPU。

这是因为你此时在节点一 (admin1) ，但是我们的服务器在节点gpu2上，如果你想要使用学院的GPU你需要更换节点，`ssh gpu2` 这个命令就可以让你在节点1上远程连上gpu2，同时，你还会发现两个节点共享文件，很不错。

但是同时，你也会发现学校服务器上什么都没有，在admin1上你可以使用python3，但是到了gpu2上你就用不了了。我个人猜测是因为python3在root目录里面，admin1节点有execute权限，但是gpu2没有。同时你也用不了pip去下载你需要的安装包，因为你没有root权限。你可以用 `which python` 查你使用的python的路径，也可以差别的。

3.配置环境（非root）

此时你就需要将环境配置在用户目录下，推荐直接使用miniconda，建立虚拟环境，好处不多说。

需要注意的是，所有需要安装的操作都要在admin1节点下进行，否则会出现找不到包之类的错误。

miniconda

可以直接参考这两个网址¹ ²

1. 执行一下代码，安装miniconda。

```
mkdir -p ~/miniconda3 #创建安装目录
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O
~/miniconda3/miniconda.sh #从网上下载安装包
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

可能需要耐心等待。

2. 激活conda。

```
source ~/.bashrc
```

3. 检测conda

```
conda --version
```

此时正常的话你就可以看到你的conda version了。

4. 此时你就可以利用conda来创建虚拟环境，并且可以指定相应环境的python版本了。

直接下载python3配置环境

我主要是根据这篇博客³ 进行配置的。

- 在python官网上下载相应的Linux版本：<https://www.python.org/downloads/source/>。这里首先安装Python-3.8.1(可以下载更高版本的，3.8.1有点低了现在)，下载，解压：

```
wget https://www.python.org/ftp/python/3.8.1/Python-3.8.1.tgz
tar -zxvf Python-3.8.1.tgz
```

- 创建安装目录

```
mkdir ~/Python3.8
```

- 配置安装位置，编译安装

```
cd Python-3.8.1  
.configure --prefix="/public/home/username/Python3.8/"  
make  
make install
```

- 添加用户环境变量 (echo >是覆盖写入, echo >>是追加写入)

```
echo "export PATH=/public/home/username/Python3.8/bin:$PATH">>~/.bashrc
```

- 查看python版本 (若ssh连接可能需要重连以应用新环境变量), 此时python3版本是刚刚安装的而非系统全局的版本。

```
python3
```

需要注意这个安装目录, 原帖的安装目录前后是有一定不同的, 但是我根据我的过程做了点修改, 应该没问题了。

非root下载pip3

除了python3, 你还需要在用户目录下下载pip3来安装所需的包。

这个我是完全按照网上博客⁴ 进行配置的。

需要注意的是我们要安装pip3, 因为我们用的是python3 (我也不知道python2行不行, 没试过), 所以这个博客里的第一个步骤, 安装pip, 应该是

```
wget https://bootstrap.pypa.io/get-pip.py  
python3 get-pip.py --user
```

其他的都一样, 我就不重复了。

现在, 你已经基本上把配置环境的基石整好了, 其他的就是把自己需要的包用pip3下下来。需要注意的就是下载的速度挺慢的, 耐心等待就行, 虽然慢但是是可以接受的。

需要注意的是, 安装完pip3可以更新一下, 否则可能下不了包。

更新指令⁵:

```
python3 -m pip install --upgrade pip
```

venv

如果你使用传统的方法直接下python的话, 我们也还是推荐使用虚拟环境的, 此时可以使用python传统的venv。详情参考这篇blog⁶。

1. 安装Virtualenv

```
pip install virtualenv
```

2. 其余如何使用, 请参考⁶, 我就不赘述了。

Git

服务器自带的git版本好像是1.8.几, 有的时候可能不是很合适, 这时候就需要我们重新在用户目录安装更高级的git (有需要再看就行), 参考这篇blog⁷。

需要注意一下的问题:

1. 步骤2的下载源代码压缩包, 可以从Git官网上去找最新版本的压缩包。
2. 用最新版本之后, 他后面很多命令的名字都要跟这变。比如这个 `tar -zxvf v2.33.1.tar.gz`, 如果你用 `v2.40.0` 版本的git的话就给他换掉, 主要就是因为你下的包名字变了。
3. 步骤四需要用到 `configure` 文件, 但是你会发现你解压完的文件夹里只有 `configure.ac` 文件, 这就需要你自己生成 `configure` 文件。参考blog^{8 9}。

直接在相应的目录里执行一下命令:

```
aclocal  
autoconf  
libtoolize  
autoheader  
automake --add-missing
```

需要注意的是, 我直接就是全部执行了, 但其实里面有的命令可能没啥用, 我也没关, 同时参考blog遇到的问题我都没遇到, 咱也不需要额外下载, 直接执行这写命令就行。

4. 步骤6里提到的“打开你的终端配置文件”, 其实就是打开你的 `~/.bashrc` 文件, 同时需要注意的是, 不要按照他的代码添加。

我们添加如下代码:

```
PATH="/public/home/你的学号/git/bin:$PATH"
```

如果你下了miniconda且观察仔细的话, 你会发现这和你miniconda里添加的环境变量基本是一样的, 除了miniconda换成了git。

同时这个“/public/home/你的学号”, 其实就是你的用户目录, 你可以用

```
cd ~  
pwd
```

这两行命令在shell里查看你的用户目录。

5. 除此之外, 就没什么问题了, 按照流程走, 你就会获得一个最新版本的git。

需要注意的是, 如果你在上述过程中, 不幸的改错了某个环境变量, 同时又使改变生效了, 你可能会遇到很严重的问题, 比如基本上所有的命令行命令都失效了。

此时试一下:

```
export PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
```

应该就会恢复正常，同时检查一下自己改了什么环境变量，给它改正确。

4. 安装所需包

Case 1: PyTorch

因为我是用PyTorch，所以我就写一下我下PyTorch注意事项。

安装PyTorch

其实也就一个需要注意的，就是CUDA版本要和我们gpu上支持的CUDA版本相一致，我下的是11.6版本。

```
pip3 install torch torchvision torchaudio --index-url  
https://download.pytorch.org/whl/cu116
```

PyTorch官网¹⁰会提供相应的代码，如果未来咱gpu上的驱动还有CUDA版本更新了，咱就可以下更高版本的PyTorch了。

除此之外，这个博客¹¹也可能对你有帮助。

测试PyTorch

我们安装好PyTorch，自然是要测试一下，尤其是要测试一下GPU能不能用，毕竟这是我们费这么多劲使用学校服务器的原因。

一般来将我们会用nvidia-smi来检查服务器里NVIDIA显卡的状态。

而在PyTorch里我们一般用torch.cuda.is_available()来检查有没有GPU可用。

此外，我们还可以用torch.cuda.device_count来查看系统中GPU的数量。

我在运行这两段代码的时候一点问题没有，但是等到我把需要计算的数据load到GPU的时候，也就是我要用GPU进行计算的时候，他给我来一个RuntimeError。

核心的错误长这样：

```
RuntimeError: device >= 0 && device < num_gpus INTERNAL ASSERT FAILED at  
"../aten/src/ATen/cuda/CUDAContext.cpp":50, please report a bug to PyTorch.
```

这个当时困扰我很久，但是是有几个帖子^{12 13}做了解答的，但是我当时按照他们的代码并没有成功。这两个帖子是你搜这个问题，在搜索引擎出来的排名第一的结果，但是因为这个没有解决我的问题，所以我后来就往下翻，有看到一个别的，但实际上他们都是一样的，都是对环境变量做调整，最后我用的解决办法就是在用torch前，放上一下两行代码：

```
import os  
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
```

这个CUDA_VISIBLE_DEVICES环境变量就是告诉设置系统里的GPU哪些是可见的，理论上设成“0”就是只有第0号显卡是可见的，也就是你只能用这一个显卡。你也可以改成别的，理论上0、1、2、3都可以。理论上也可以设置多块显卡。

之后我就是用了一段测试代码来检测的，如果自己有别的训练模型的代码也可以直接试一试去训练模型来测试GPU可不可用。

我的测试代码如下（主要是由GPT生成的）：

```
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "2"

import torch

print(torch.cuda.current_device())

# 定义矩阵大小
N = 20000
M = 20000
K = 20000

# 随机生成两个矩阵并将其移动到 GPU 上
a = torch.randn(N, M, device="cuda")
b = torch.randn(M, K, device="cuda")

# 执行矩阵乘法并计算运行时间
start_time = torch.cuda.Event(enable_timing=True)
end_time = torch.cuda.Event(enable_timing=True)

start_time.record()
c = torch.matmul(a, b)
end_time.record()
torch.cuda.synchronize()

# 计算运行时间并输出结果
elapsed_time_ms = start_time.elapsed_time(end_time)
print(f"Matrix multiplication took: {elapsed_time_ms:.2f} milliseconds")
```

5.本地链接到服务器

之前所有的操作我们只要使用WebShell就可以实现，但是很显然只用WebShell不是很方便，我们接下来介绍如何本地链接到服务器。

本地终端ssh

如果你对ssh有所了解，应该知道想要ssh需要私钥（id_rsa）与公钥（id_rsa.pub）。服务器有自己的私钥和公钥，我们自己的电脑也有自己的私钥和公钥，所以，理论上我们有两种方法：

1. 得到服务器的私钥，并通过这个私钥链接服务器。
2. 把我们自己电脑的公钥加到服务器上。

我最开始看别的教程采用了方法2，这个方法看着就不是很好的样子，毕竟有点像把门上的锁换了，而不是拿到相应的钥匙。

所以，我们要采用方法1。

得到服务器私钥

很简单，从我们的登陆网页上下载就行（终于用上这个网站了）。

The screenshot shows a web-based management interface with various tabs at the top: 'Gridview', 'WebShell', '写文章 - 知乎', 'Gantt diagram...', and 'RuntimeError...'. Below the tabs, there's a navigation bar with links like '工作空间', '我的应用', '我的作业', '我的账单', '我的数据', and '我的资源'. On the left, there's a sidebar with '更多' (More) and a '我的资源' section containing a pie chart showing job states: 运行 (Running), 排队 (Queued), 保留 (Reserved), 挂起 (Held), and 其他 (Other). The chart indicates 0 for all categories. To the right of the chart is a '队列作业TOP5' section which is currently empty, showing '暂无数据' (No data available). At the bottom, there's a table titled '我的作业' (My Jobs) with columns for '作业ID', '作业名', '队列', '状态', '开始时间', '已运行时长', and '操作' (Operations). A context menu is open over a user profile icon in the top right, listing options: 常用工具 (常用工具), 个人中心 (Personal Center), 关于 (About), and 注销 (Logout). The 'Download Private Key' option is highlighted.

下载成功之后，你会得到一个id_rsa文件，给它存到~目录下的.ssh文件夹里就行，最好改一下名，毕竟里面有理论上有你自己的私钥。

更改id_rsa权限

如果现在你直接用这个私钥去链接，你会发现它回报错，`linux Permission 0644 for are too open` 或者类似的报错，总之就是类似的错误，可以参考这两篇博客^{14 15}，把权限改成只有管理员可读可写就行。

链接

有了私钥文件，就可以ssh链接了，直接打开终端，输入

```
ssh -i id_rsa地址 学号@121.48.227.161 -p 7092
```

这里边id_rsa地址，就是你们下载的服务器的id_rsa的地址，因为我们的学号就是我们用户账户的名字，所以使用我们的学号@，121.48.227.161是我们服务器的网址，7092是我们服务器远程连接的端口号。

这块我主要看的是这个博客¹⁶。

VScode ssh

毕竟我们基本还是不太会vim，也没怎么配置过vim，vim用起来还是不方便，而VScode可以说已经发展的非常成熟了，大家可以直接按照这篇博客¹⁷去配置VScode的ssh功能，配置完之后他基本上就相当于一个编辑器，但还是方便多了。

需要提醒的就是，配置完了之后可能需要重启一下VScode，或者需要更新的话更新一下VScode。VScode经常有这样的毛病，代码一点问题没有，他运行不了，但是只要重启一下立马好了。

本地浏览器打开服务器Jupyter

如果你看过李沐老师的AI相关的课程，你就会发现他是在服务器上打开Jupyter Notebook，然后镜像到本地服务器进行操作的。这个也是很不错的一种方法，但是会很麻烦，因为应该是需要两次跳转，我现在可以本地打开admin1节点的Jupyter。

大家也可以自己尝试一下，update一下这个教程。

Version: 1.0.0

Stanley Zheng (郑博文)

Github: <https://github.com/S-tanley/Using-SCUPI-s-GPU>

2024.04.4

-
1. <https://docs.anaconda.com/free/miniconda/>
 2. <https://zhuanlan.zhihu.com/p/685496400>
 3. <https://www.cnblogs.com/esctrionsit/p/13415058.html>
 4. https://blog.csdn.net/qq_38486203/article/details/88840759
 5. <https://zhuanlan.zhihu.com/p/469155815>
 6. <https://www.freecodecamp.org/chinese/news/how-to-setup-virtual-environments-in-python/>
 7. https://geek-docs.com/git/git-questions/889_git_installing_git_in_home_directory_centos_5_no_root.html
 8. <https://blog.csdn.net/mao834099514/article/details/79544467>
 9. <https://www.jianshu.com/p/b97db7c9c915>
 10. <https://pytorch.org>
 11. https://blog.csdn.net/qq_35831906/article/details/134349866
 12. <https://github.com/pytorch/pytorch/issues/110000>
 13. <https://discuss.pytorch.org/t/runtimeerror-device-0-device-num-gpus-internal-assert-failed/178118/5>
 14. <https://blog.csdn.net/pansanday/article/details/80776079>
 15. <https://blog.csdn.net/Axela30W/article/details/78981749>
 16. <https://blog.csdn.net/tyustli/article/details/122222605>
 17. <https://zhuanlan.zhihu.com/p/412736012>