LeNet-5

实验思路

根据LeNet的论文，构造网络（见conv.py），使用NMIST数据集，直接用torchvision提高的函数下载导入（见utils.py），最后在main.py中实现训练的过程以及测试的步骤（见main.py）。

实验环境

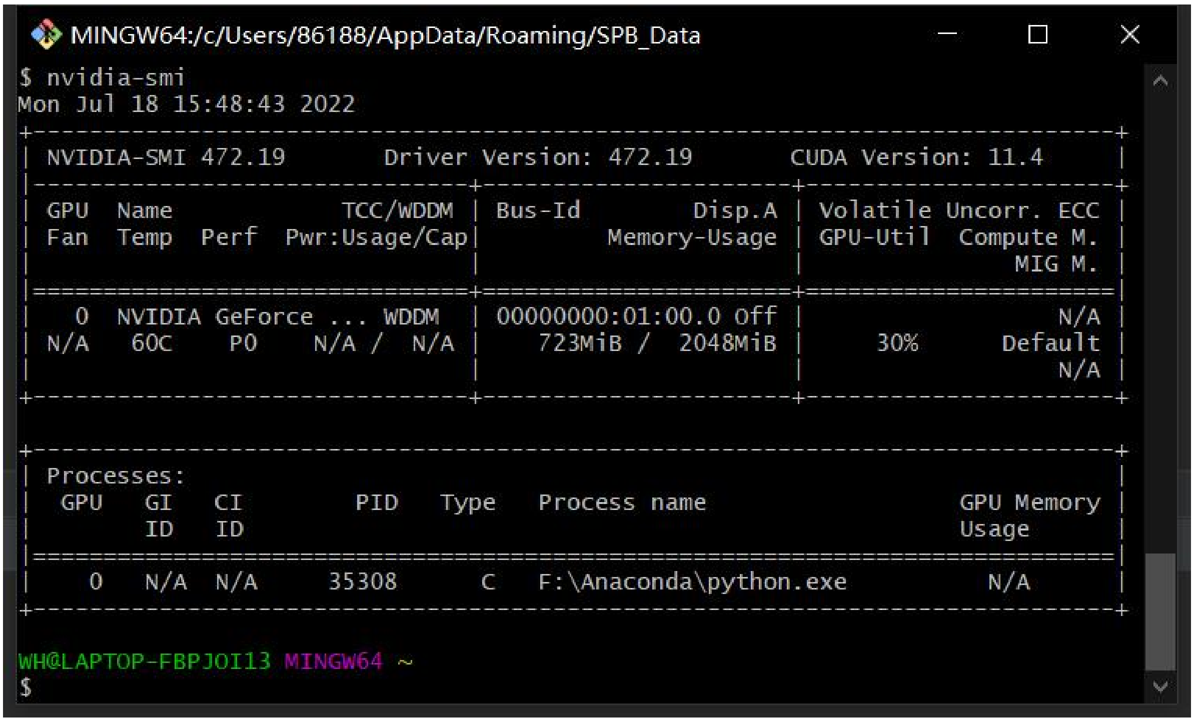
硬件环境：

GPU: Nvidia GeForce MX350

CPU: intel core i5 10th Gen

实验截图

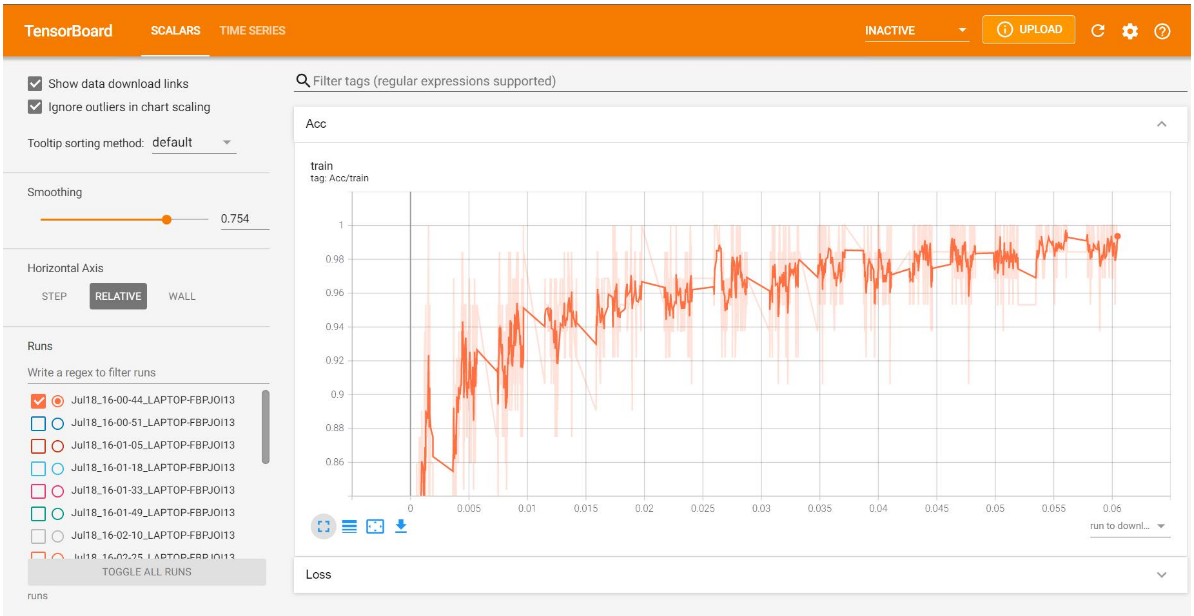
GPU使用率截图



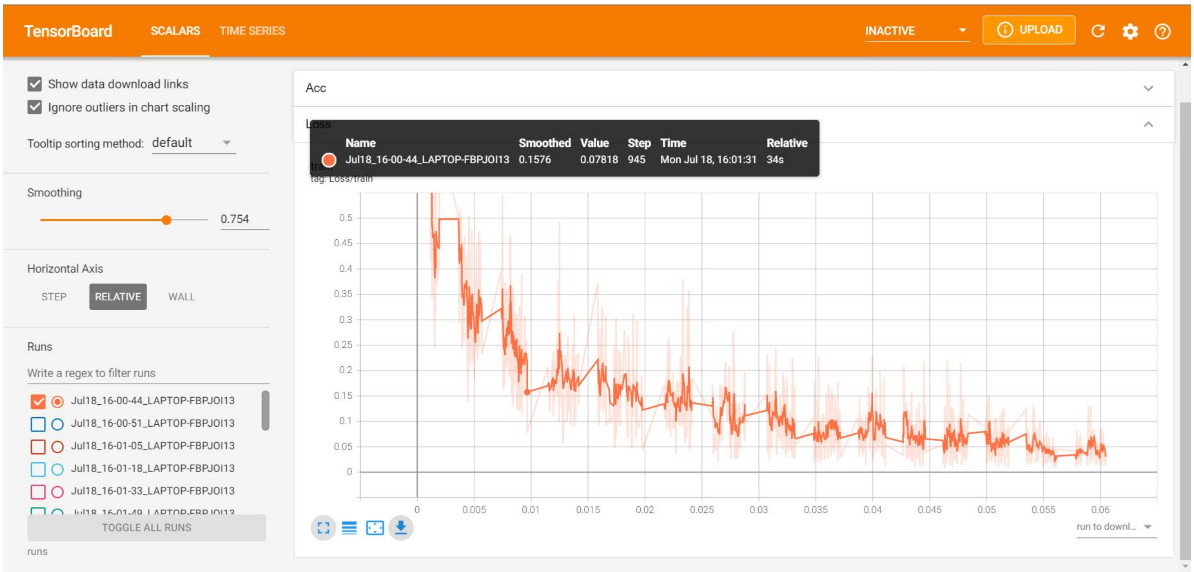
```
MINGW64:/c/Users/86188/AppData/Roaming/SPB_Data
$ nvidia-smi
Mon Jul 18 15:48:43 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 472.19       Driver Version: 472.19       CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   60C    P0    N/A /  N/A |    723MiB /  2048MiB |     30%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     35308      C   F:\Anaconda\python.exe            N/A      |
+-----------------------------------------------------------------------------+

WH@LAPTOP-FBPJOI13 MINGW64 ~
$
```

LeNet损失函数、精确度截图

LeNet训练阶段精确度曲线



LeNet训练阶段损失函数曲线



LeNet测试集准确度

```
F:\Anaconda\python.exe "E:/Grade Three/ShortSemester/HPC101/lab5/LeNet-5/main.py"
Now start training
Already load the data
size of training set is:  938 | size of test set is: 157
Training Time: Epoch[1/15], Total loss 802.4512
Training Time: Epoch[2/15], Total loss 312.9650
Training Time: Epoch[3/15], Total loss 230.6375
Training Time: Epoch[4/15], Total loss 181.9661
Training Time: Epoch[5/15], Total loss 149.4177
Training Time: Epoch[6/15], Total loss 125.9481
Training Time: Epoch[7/15], Total loss 108.0310
Training Time: Epoch[8/15], Total loss 93.9804
Training Time: Epoch[9/15], Total loss 82.6292
Training Time: Epoch[10/15], Total loss 73.3027
Training Time: Epoch[11/15], Total loss 66.4058
Training Time: Epoch[12/15], Total loss 60.2132
Training Time: Epoch[13/15], Total loss 54.6760
Training Time: Epoch[14/15], Total loss 50.3322
Training Time: Epoch[15/15], Total loss 46.2792
Save model successfully and start to test the model
Test accuracy : 0.9845

Process finished with exit code 0
```

经过15个epoch的训练，lenet在测试集上准确度达到了98.45%

GPT2

本实验模型基于minGPT（https://github.com/karpathy/minGPT）改写而来

实验思路

通过对transformer和gpt2的论文学习以及对minGPT开源代码的学习，改写了模型代码；其中trainer.py的作用为提供开始训练及测试的函数，model.py为构造模型的代码，utils.py为一些调用的工具类型函数，train.py为启动代码；训练时，直接调用sh run_*.sh即可

实验理解

tokenizer选择了hugging face提供的API，直接导入了训练完成的BPE tokenizer，然后自己构建了数据集，每次取原始数据集的1025长度的tokens，使用bpe将前1024个tokens和后1024个tokens分别分词，输入模型；gpt2的实验目的是通过观察前面的tokens，预测出下一个token；Trainer类提供了初始化的接口和训练的接口，同时也有实现数据并行的DDP，数据转到gpu上面实验的代码等。

model.py是gpt2的模型代码，从GPT类进入后，根据config中gpt2的类型确定超参数的数值，之后根据参数构建网络组件，初始化网络参数等部分均在__init__中实现；forward是模型的主要代码，wte是token embedding，wpe是position embedding，两个相加之后得到同时包含位置信息和token信息的embedding信息，将其输入到层层堆叠的block当中，最后再通过全连接层映射到vocab维度，和target进行cross entropy得到loss

Block类是实现每个block的代码，其中self.atten使用的是CausalSelfAttention的代码，另外transformer中采用了residual network。

最后，若干个模块拼接在一起，组成了gpt2模型全部的代码。

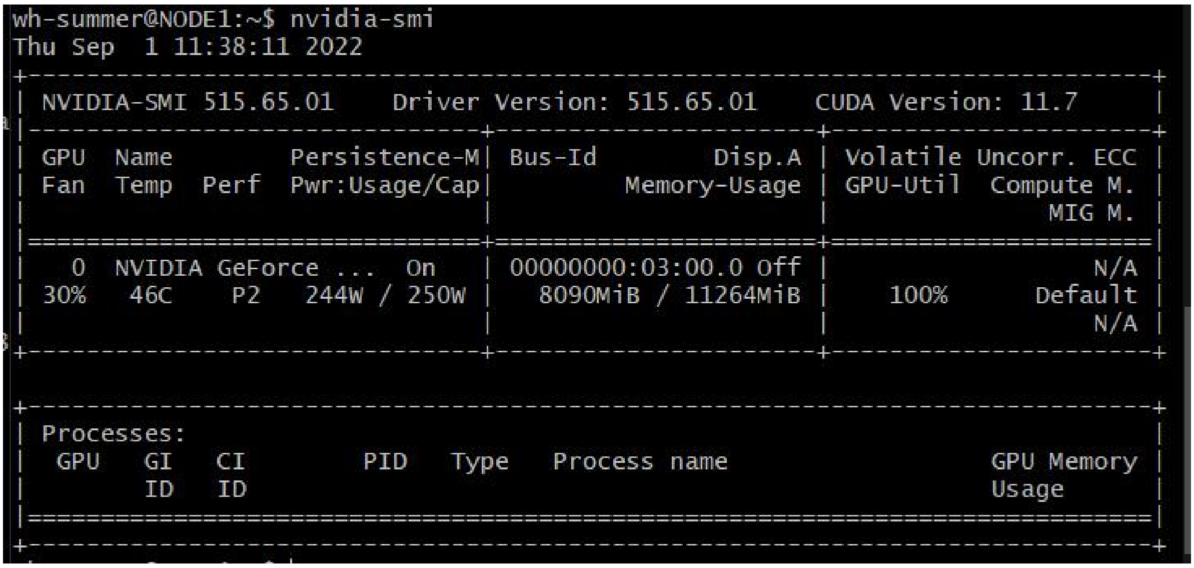
实验环境

超算队提供的平台

并行策略

采用了数据并行的方式，使用pytorch提供的DistributedDataParallel函数将数据集平分给不同的节点，因为模型本身参数量较大，所以设置batch_size为1，因此不同节点每次并行计算一个batch的数据后独立计算梯度，每个进程将梯度依次传给下一个进程，直到所有的进程得到全部的梯度，最后整体做梯度下降，之后再各自计算分配到的batch的数据，周而复始

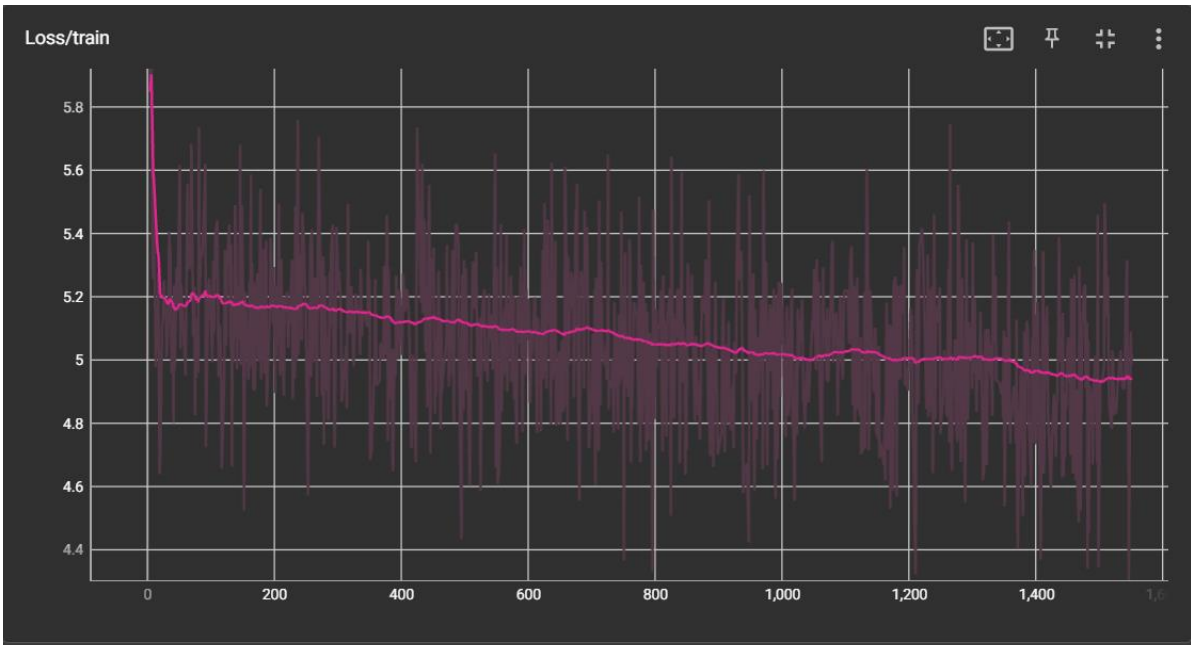
实验截图



```
wh-summer@NODE1:~$ nvidia-smi
Thu Sep  1 11:38:11 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:03:00.0 Off |                  N/A |
| 30%   46C    P2   244W / 250W |   8090MiB / 11264MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

实验结果

tensorboard损失函数曲线



2节点

```
====== TRAIN TIME
====== TRAINING LEN IS 1541
====== EPOCH: [0/4], STEP: [0/1541], TRAIN LOSS 10.90811
====== EPOCH: [0/4], STEP: [500/1541], TRAIN LOSS 6.19105
====== EPOCH: [0/4], STEP: [1000/1541], TRAIN LOSS 6.61747
====== EPOCH: [0/4], STEP: [1500/1541], TRAIN LOSS 6.30675
====== EPOCH: [1/4], STEP: [0/1541], TRAIN LOSS 6.56909
====== EPOCH: [1/4], STEP: [500/1541], TRAIN LOSS 5.26274
====== EPOCH: [1/4], STEP: [1000/1541], TRAIN LOSS 6.13393
====== EPOCH: [1/4], STEP: [1500/1541], TRAIN LOSS 5.81166
====== EPOCH: [2/4], STEP: [0/1541], TRAIN LOSS 6.08854
====== EPOCH: [2/4], STEP: [500/1541], TRAIN LOSS 4.74014
====== EPOCH: [2/4], STEP: [1000/1541], TRAIN LOSS 5.68105
====== EPOCH: [2/4], STEP: [1500/1541], TRAIN LOSS 5.45014
====== EPOCH: [3/4], STEP: [0/1541], TRAIN LOSS 5.77131
====== EPOCH: [3/4], STEP: [500/1541], TRAIN LOSS 4.40659
====== EPOCH: [3/4], STEP: [1000/1541], TRAIN LOSS 5.41740
====== EPOCH: [3/4], STEP: [1500/1541], TRAIN LOSS 5.13750
Time is: 46.45446827809016 min
save the model
```

```
====== TRAIN TIME
====== TRAINING LEN IS 1541
====== EPOCH: [0/4], STEP: [0/1541], TRAIN LOSS 10.98853
====== EPOCH: [0/4], STEP: [500/1541], TRAIN LOSS 6.48694
====== EPOCH: [0/4], STEP: [1000/1541], TRAIN LOSS 5.86231
====== EPOCH: [0/4], STEP: [1500/1541], TRAIN LOSS 5.71837
====== EPOCH: [1/4], STEP: [0/1541], TRAIN LOSS 5.84731
====== EPOCH: [1/4], STEP: [500/1541], TRAIN LOSS 5.68601
====== EPOCH: [1/4], STEP: [1000/1541], TRAIN LOSS 5.48873
====== EPOCH: [1/4], STEP: [1500/1541], TRAIN LOSS 5.04098
====== EPOCH: [2/4], STEP: [0/1541], TRAIN LOSS 5.30913
====== EPOCH: [2/4], STEP: [500/1541], TRAIN LOSS 5.31240
====== EPOCH: [2/4], STEP: [1000/1541], TRAIN LOSS 5.14490
====== EPOCH: [2/4], STEP: [1500/1541], TRAIN LOSS 4.62831
====== EPOCH: [3/4], STEP: [0/1541], TRAIN LOSS 4.93268
====== EPOCH: [3/4], STEP: [500/1541], TRAIN LOSS 4.98226
====== EPOCH: [3/4], STEP: [1000/1541], TRAIN LOSS 4.86429
====== EPOCH: [3/4], STEP: [1500/1541], TRAIN LOSS 4.34862
Time is: 46.454470455646515 min
save the model
```

使用了2节点，每个节点1块GPU的数据并行，因测试发现中等大小的X.txt的token数量在3.15M左右，所以总共设定的4个epoch完成12M的token训练任务；batch_size设为1；训练全部时长为46.45min，训练完12M个token时长为44.24min，损失也成功收敛，低于7；

1节点

```
The device is  cuda
====== TRAIN TIME
====== TRAINING LEN IS 3081
====== EPOCH: [0/4], STEP: [0/3081], TRAIN LOSS 10.98853
====== EPOCH: [0/4], STEP: [500/3081], TRAIN LOSS 6.72221
====== EPOCH: [0/4], STEP: [1000/3081], TRAIN LOSS 6.41928
====== EPOCH: [0/4], STEP: [1500/3081], TRAIN LOSS 6.09135
====== EPOCH: [0/4], STEP: [2000/3081], TRAIN LOSS 5.87746
====== EPOCH: [0/4], STEP: [2500/3081], TRAIN LOSS 5.95929
====== EPOCH: [0/4], STEP: [3000/3081], TRAIN LOSS 5.76737
====== EPOCH: [1/4], STEP: [0/3081], TRAIN LOSS 5.91194
====== EPOCH: [1/4], STEP: [500/3081], TRAIN LOSS 5.93509
====== EPOCH: [1/4], STEP: [1000/3081], TRAIN LOSS 5.77170
====== EPOCH: [1/4], STEP: [1500/3081], TRAIN LOSS 5.47760
====== EPOCH: [1/4], STEP: [2000/3081], TRAIN LOSS 5.53770
====== EPOCH: [1/4], STEP: [2500/3081], TRAIN LOSS 5.53647
====== EPOCH: [1/4], STEP: [3000/3081], TRAIN LOSS 5.19615
====== EPOCH: [2/4], STEP: [0/3081], TRAIN LOSS 5.52452
====== EPOCH: [2/4], STEP: [500/3081], TRAIN LOSS 5.48708
====== EPOCH: [2/4], STEP: [1000/3081], TRAIN LOSS 5.44500
====== EPOCH: [2/4], STEP: [1500/3081], TRAIN LOSS 5.16286
====== EPOCH: [2/4], STEP: [2000/3081], TRAIN LOSS 5.24477
====== EPOCH: [2/4], STEP: [2500/3081], TRAIN LOSS 5.15223
====== EPOCH: [2/4], STEP: [3000/3081], TRAIN LOSS 4.80683
====== EPOCH: [3/4], STEP: [0/3081], TRAIN LOSS 5.24304
====== EPOCH: [3/4], STEP: [500/3081], TRAIN LOSS 5.16729
====== EPOCH: [3/4], STEP: [1000/3081], TRAIN LOSS 5.17569
====== EPOCH: [3/4], STEP: [1500/3081], TRAIN LOSS 4.87085
====== EPOCH: [3/4], STEP: [2000/3081], TRAIN LOSS 5.07560
====== EPOCH: [3/4], STEP: [2500/3081], TRAIN LOSS 4.87081
====== EPOCH: [3/4], STEP: [3000/3081], TRAIN LOSS 4.54390
Time is: 35.7539 min
save the model
```

使用1节点，避免了节点通讯的时间成本，训练全部时长为35.75min，训练完12Mtokens时长为34.05min，损失也成功收敛，低于7；

复现

```
#如果使用2节点的话需要修改master_addr和world_size，将master_addr改成主节点的地址，端口随便选一个，world_size设置为2，sh脚本中也要将--nnodes改成2
# 主节点
sh run_0.sh
# 子节点
sh run_1.sh

#如果使用1节点，则修改master_addr，其他不变
sh run_0.sh
```