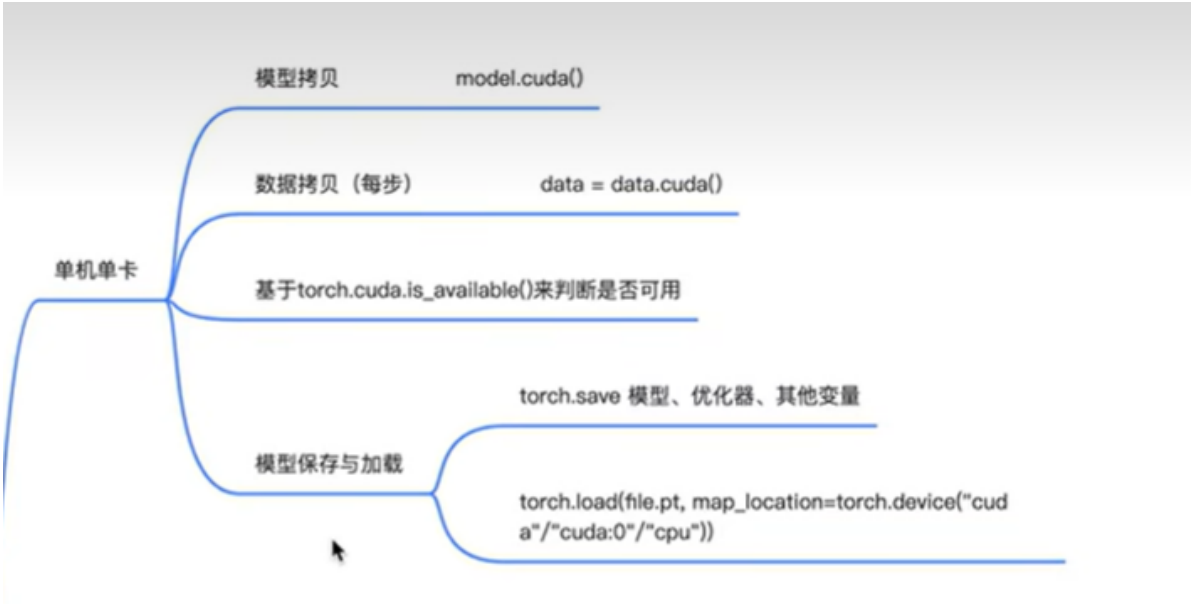
Pytorch分布式训练

2022.07.29

本文记录PyTorch的分布式训练代码编写思路，包含单机单卡、单机多卡和多机多卡，包括数据拷贝、模型拷贝以及模型的保存与加载，代码见Pycharm

1.单机单卡（最常用）



- 首先判断cuda是否可用
- 然后将模型和数据都拷贝到GPU上
- 再注意在加载模型的时候，还要传入map_location参数

2.单机多卡

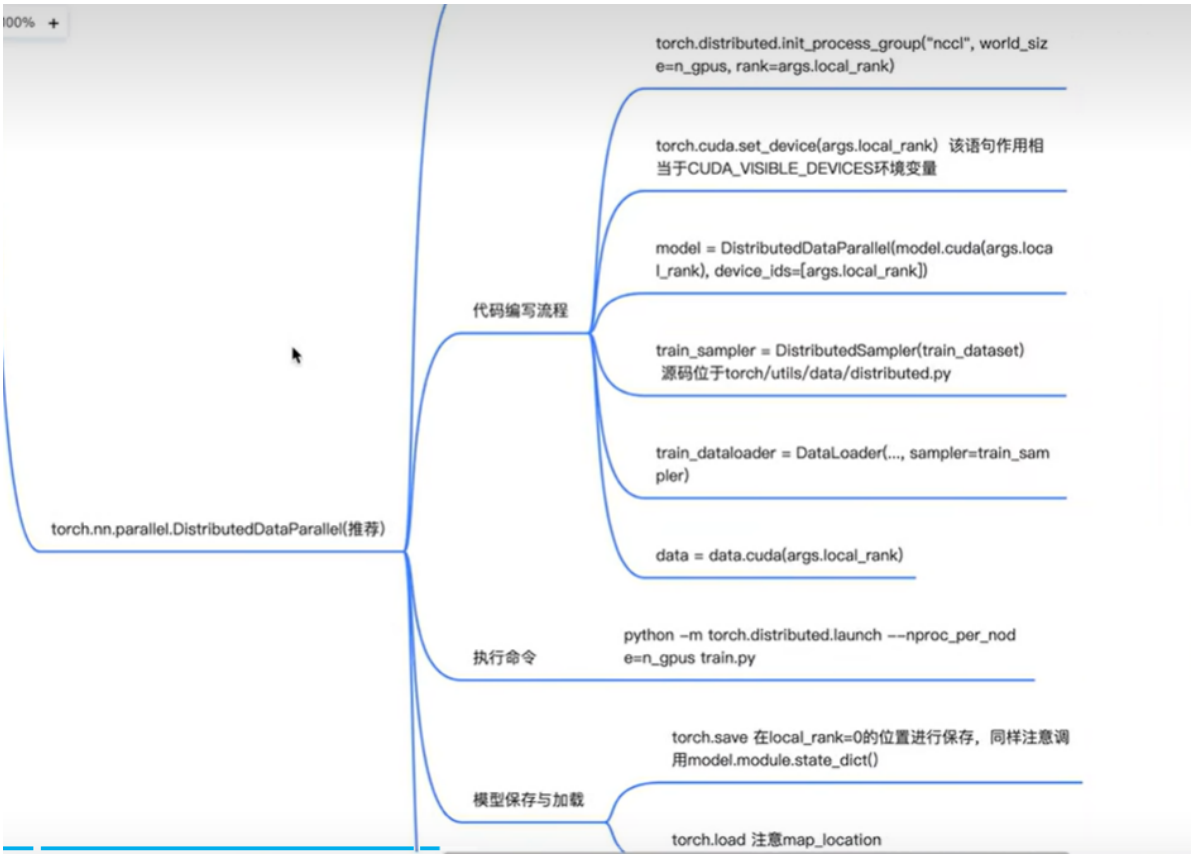
首先要检测GPU的数据，然后再指定使用多少块GPU

2.1 DataParallel方法（单进程实现多卡）

实现单机多卡有两个方法，第一个是torch.nn.DataParallel方法，但该方法几乎被淘汰了，不过它改动代码比较简单（因为是单机进程，不建议使用）

- 拷贝模型的时候注意要传入device_id，指定使用哪几块GPU
- torch.save注意模型需要调用model.module.state_dict，而不是简单的model.state_dict
- 同样注意在加载模型的时候，还要传入map_location参数
- 注意batch_size应该是每个GPU的batch_size的总和

2.2 DistributedDataPallel（多进程实现多卡）



多进程执行多卡训练，效率高

2.2.1 代码编写要点

- 首先初始化一个进程组，用torch.distributed.init_process_group，第一个参数指定GPU的通信方式（一般是nccl），world_size指定使用多少张GPU卡，rank指定当前进程是在第几个GPU上
- 采用torch.cuda.set_device指定当前进程能用到的GPU卡的名称，相当于CUDA_VISLBE_DEVICES
- 然后进行模型拷贝，device_ids中传入一张卡即可，因为是一张卡跑一个进程
- train_sampler是为了将train_dataset上的数据分配到每张卡上来进行训练（随机分配）
- 然后在DataLoader中sampler就是传入train_sampler，不需要设置shuffle，因为在DataLoder中sampler和shulffle是互斥的
- 最后是数据拷贝，args.local_rank表示的是当前节点的第几张卡

2.2.2 执行命令

命令：python -m torch.distributed.launch --nproc_per_node=n_gpus train.py

2.2.3 模型保存与加载

注意是在local_rank=0（GPU0）上保存，调用的是model.module.state_dict()（因为模型已经被DistributedDataParallel包裹起来了）

torch.load注意指定map_location，指定要加载到哪块GPU上

2.2.4 注意事项

- train.py中要有接受local_rank的参数选项，launch会传入这个参数
- 每个进程的batch_size应该是一个GPU所需要的batch_size大小
- 在每个周期开始处，调用train_sampler.set_epoch(epoch)可以使得数据充分打乱（每个epoch获得索引是不一样的）
- 有了sampler，就不需要在DataLoder中设置shuffle=True

3. 多机多卡

同样也是在DistributedDataPallel方法中实现