pytorch 分布式计算 坑/bug 梳理篇

来自： AiGC面试宝典

宁静致远

2024年01月27日 19:44

动机

pytorch用的人越来越多，大的模型都需要用gpu或者多张gpu甚至多节点多卡进行分布式计算，但是坑也很多，本文主要 介绍 读者 在 进行 pytorch 分布式计算 所遇到的 坑/bug 的 梳理 及 填坑记录。

一、使用 DistributedDataParallel（分布式并行）时，显存分布不均衡问题

- 问题描述：

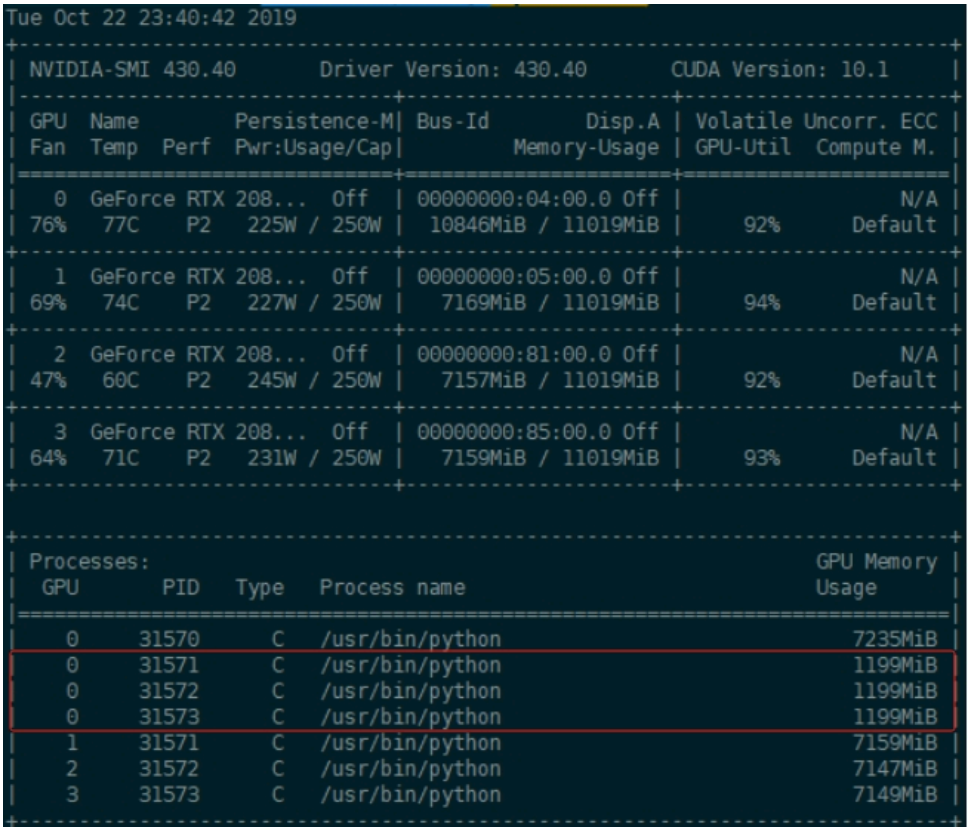
如果用 DistributedDataParallel （分布式并行）的时候，每个进程单独跑在一个 GPU 上，多个卡的显存占用用该是均匀的，比如像这样的：

```
Tue Oct 22 23:34:40 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 430.40       Driver Version: 430.40       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce RTX 208...  Off  | 00000000:04:00.0 Off |                  N/A |
| 56%   66C    P2   225W / 250W |   7171MiB / 11019MiB |     95%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce RTX 208...  Off  | 00000000:05:00.0 Off |                  N/A |
| 59%   68C    P2   234W / 250W |   7167MiB / 11019MiB |     94%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce RTX 208...  Off  | 00000000:81:00.0 Off |                  N/A |
| 45%   59C    P2   240W / 250W |   7153MiB / 11019MiB |     94%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GeForce RTX 208...  Off  | 00000000:85:00.0 Off |                  N/A |
| 62%   70C    P2   233W / 250W |   7161MiB / 11019MiB |     94%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     30917      C   /usr/bin/python                             7161MiB |
|    1     30918      C   /usr/bin/python                             7157MiB |
|    2     30919      C   /usr/bin/python                             7143MiB |
|    3     30920      C   /usr/bin/python                             7151MiB |
+-----------------------------------------------------------------------------+
```

> 注：在 Distributed 模式下，相当于你的代码分别在多个 GPU 上独立的运行，代码都是设备无关的。比如你写 t = torch.zeros(100, 100).cuda()，在4个进程上运行的程序会分别在4个 GPUs 上初始化 t。所以显存的占用会是均匀的。

然而，有时会发现另外几个进程会在0卡上占一部分显存，导致0卡显存出现瓶颈，可能会导致cuda-out-of-memory 错误。比如这样的：



- 问题定位：

该问题主要由 以下代码导致：

```
checkpoint = torch.load("checkpoint.pth")
model.load_state_dict(checkpoint["state_dict"])
```

注：上述代码运行后，程序 load 一个 pretrained model 的时候，torch.load() 会默认**把load进来的数据放到0卡上，这样4个进程全部会在0卡占用一部分显存。**

- 解决方法：

把load进来的数据map到cpu上：

```
checkpoint = torch.load("checkpoint.pth", map_location=torch.device('cpu'))
model.load_state_dict(checkpoint["state_dict"])
```

二、如果是用pytorch实现同步梯度更新，自研 数据接口，出现 第一个epoch结尾处程序卡死问题

如果是用pytorch实现同步梯度更新，然后数据接口是自己写的话一定要注意保证每张卡分配的batch数是一样的。因为如果某张卡少了一个batch的话，其他卡就会等待，从而程序卡在torch.all_reduce()上。最后的情况就会出现在第一个epoch结尾处程序卡住，而且没有报错信息。

三、在微调大模型的时候，单机2卡的时候正常训练，但是采用4卡及其以上，就会卡住，卡在读完数据和开始训练之间?

先确认几张卡都能正常使用和通信，然后看看是不是batchsize分配之类的问题导致无限等待某一张卡了。再就是只留4条数据，每张卡只跑一条数据试试看。