# 图解分布式训练（八）—— ZeRO 学习

来自： AiGC面试宝典

宁静致远

2023年09月29日 12:03



**一、什么是 3D 并行?**

3D 并行可以让大型模型以非常有效的方式进行训练

**二、3D 并行 策略有哪些?**

- DataParallel (DP)
- TensorParallel (TP)
- PipelineParallel (PP)

**2.1 DataParallel (DP)**

- 介绍：假设有N张卡，**每张卡都保存一个模型，每一次迭代（iteration/step）都将batch数据分割成N个等大小的micro-batch，每张卡根据拿到的micro-batch数据独立计算梯度，然后调用AllReduce计算梯度均值，每张卡再独立进行参数更新。**
- 举例说明：

```
# 假设模型有三层：L0, L1, L2
# 每层有两个神经元   # 两张卡
GPU0:    L0 | L1 | L2   ---|----|---   a0 | b0 | c0   a1 | b1 | c1
GPU1:   L0 | L1 | L2   ---|----|---   a0 | b0 | c0   a1 | b1 | c1
```

**2.2 TensorParallel (TP)**

- 介绍：**每个张量都被分成多个块，因此不是让整个张量驻留在单个 GPU 上，而是张量的每个分片都驻留在其指定的 GPU 上。在处理过程中，每个分片在不同的 GPU 上分别并行处理，最终结果在步骤结束时同步。**这也被称作横向并行。
- 举例说明：

```
# 假设模型有三层：L0, L1, L2
# 每层有两个神经元   # 两张卡
GPU0:    L0 | L1 | L2   ---|----|---   a0 | b0 | c0   a1 | b1 | c1
GPU1:   L0 | L1 | L2   ---|----|---   a0 | b0 | c0   a1 | b1 | c1
```

**2.3 PipelineParallel (PP)**

- 介绍：模型在多个 GPU 上垂直（层级）拆分，因此只有模型的一个或多个层放置在单个 GPU 上。每个 GPU 并行处理管道的不同阶段，并处理一小部分批处理。
- 举例说明：

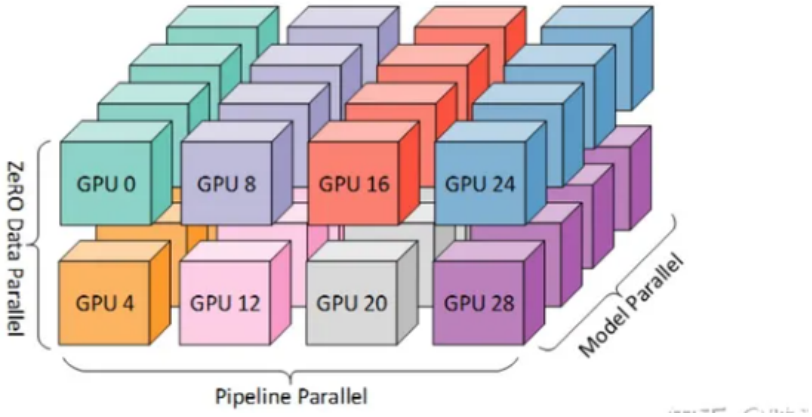
```
# 假设模型有8层
# 两张卡
|  L0 | L1 | L2 | L3 |   | L4 | L5 | L6 | L7 |
=====================  ====================
        GPU0                  GPU1
```

**三、为什么需要 ZeRO?**

虽然 DataParallel (DP) 因为简单易实现，所以目前应用相比于其他两种 广泛，但是 由于 DataParallel (DP) 需要每张卡都存储一个模型，导致 显存大小 成为 制约模型规模 的 主要因素。

既然 每张卡都存储一个模型 会 增加 模型训练过程中的显存占用，那么 是否可以 让 每行卡训练 1/N 的模型参数，然后 合并起来就是一个完整模型呢？ 这样，随着卡数的增加，每张卡 用于 模型训练的显存占用将减低，能够训练的模型也就越大。

如今训练大模型离不开各种分布式并行策略，ZeRO系列技术就是一种显存优化的数据并行方案，旨在训练超大规模的语言模型。




**四、ZeRO 的 核心思想是什么?**

去除数据并行中的冗余参数，使每张卡只存储一部分模型状态，从而减少显存占用。

**五、ZeRO 显存如何分配?**

ZeRO将模型训练阶段中每张卡的显存内容分为两类：

- 模型状态：包括参数、梯度和优化器状态，其中优化器状态占比 75% 。
- 剩余状态：除了模型状态之外的显存占用，包括激活值、各种临时缓冲区以及无法使用的显存碎片。

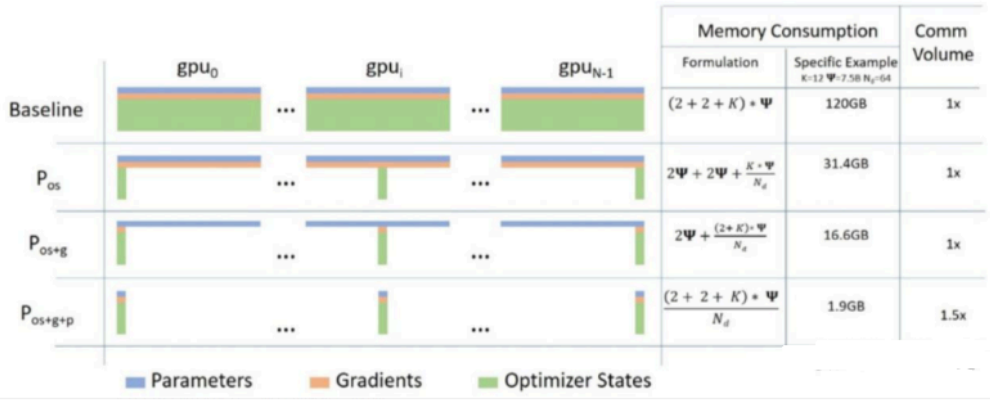
来看一个例子，GPT-2含有1.5B个参数，如果用fp16格式（混合精度），只需要3GB显存，但是模型状态实际上需要耗费24GB！所以模型状态就成了头号显存杀手，它也是ZeRO的重点优化对象。而其中优化器状态又是第一个要被优化的。

**六、ZeRO 优化策略是怎么样?**

针对模型状态的存储优化（去除冗余），ZeRO使用的方法是分片，即每张卡只存 1/N的模型状态量，这样系统内只维护一份模型状态。

ZeRO 具有三个主要的优化阶段（ZeRO-1，ZeRO-2，ZeRO-3），它们对应于优化器状态（optimizer states）、梯度（gradients）和参数（parameters）的分片。累积启用时：

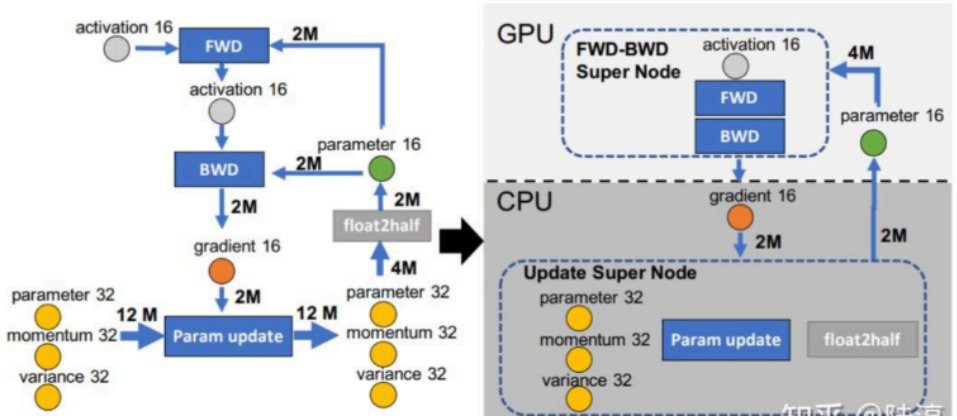
- 优化器状态分区 (*Pos*) – 内存减少 4 倍，通信量与数据并行性相同
- 添加梯度分区 (*Pos+g*) – 内存减少 8 倍，通信量与数据并行性相同
- 添加参数分区 (*Pos+g+p*) – 内存减少与数据并行度 Nd 成线性关系。例如，拆分为 64 个 GPU ( Nd =64) 内存将减少到 1/64 。GPU 通信量略有增加 50%。




> 注：图中Memory Consumption 第二列给出了一个示例： k=12,Φ=7.5B,Nd=64 ，可以看到随着 ZeRO 阶段深入，显存优化相当明显。

**一张卡训不了大模型，根因是显存不足，ZeRO-Offload则将训练阶段的某些模型状态下放（offload）到内存以及CPU计算，即显存不足，内存来补。相比于昂贵的显存，内存廉价多了。**

下图是某一层的一次迭代过程，使用了混合精度训练，前向计算（FWD）需要用到上一次的激活值（activation）和本层的参数（parameter），反向传播（BWD）也需要用到激活值和参数计算梯度，当我们用Adam优化器进行参数更新时，假设模型参数量是 M，在混合精度训练的前提下，边的权重要么是2M（fp16），要么是4M（fp32）。为了不降低计算效率，将前两个节点放在GPU，后两个节点不但计算量小还需要和Adam状态打交道，所以放在CPU上，Adam状态自然也放在内存中，为了简化数据图，将前两个节点融合成一个节点FWD-BWD Super Node，将后两个节点融合成一个节点Update Super Node，**沿着gradient 16和parameter 16把数据流图切分为两部分，分布对应GPU和CPU**




**七、ZeRO Offload后的计算流程是怎么样?**

- 在GPU上面进行前向和后向计算，将梯度传给CPU。同时为了提高效率，可以将计算和通信并行起来，GPU在反向传播阶段，可以待梯度值填满bucket后，一边计算新的梯度一边将bucket传输给CPU，当反向传播结束，CPU基本上已经有最新的梯度值了；
- cpu进行参数更新，再将更新后的参数传给GPU。