



先进编译实验室
Advanced Compiler

自动并行-并行划分

嘉宾： 王俊祥



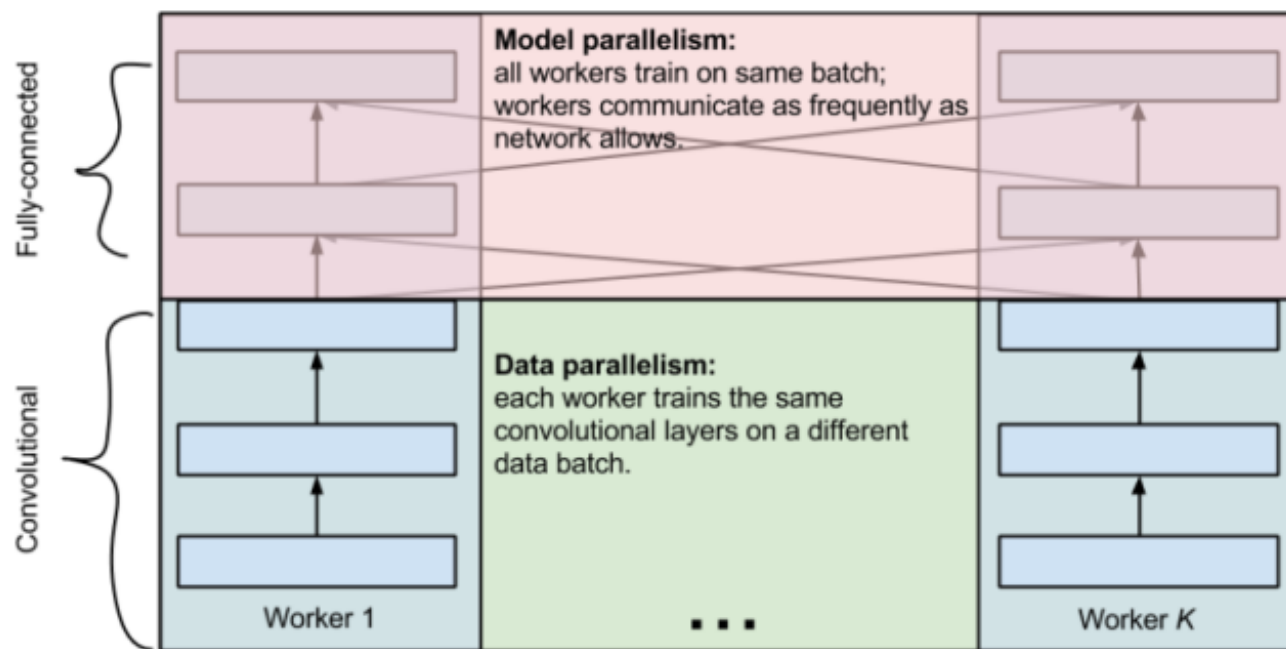
先进编译实验室
Advanced Compiler





并行划分

- 1、背景
- 2、数据并行
- 3、模型并行
- 4、混合并行
- 5、总结

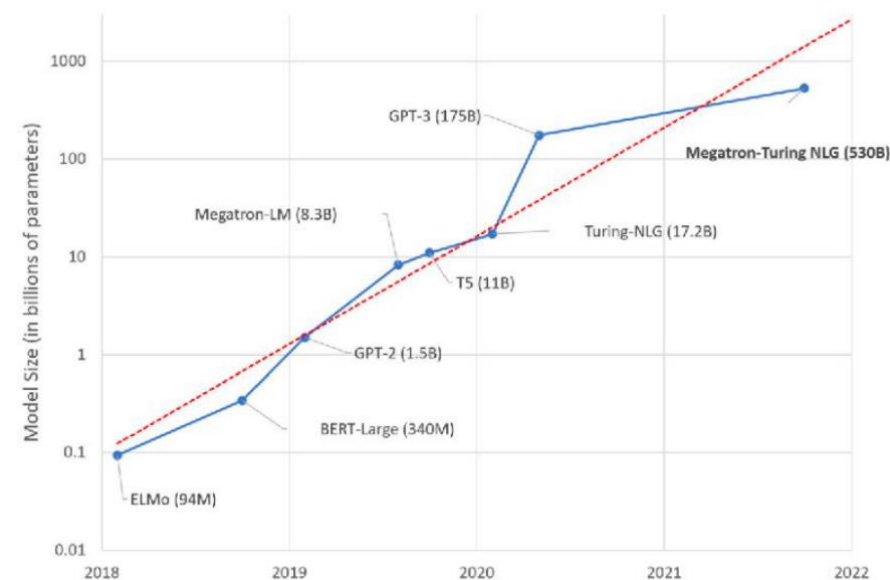




1、背景

数据与模型规模的扩大

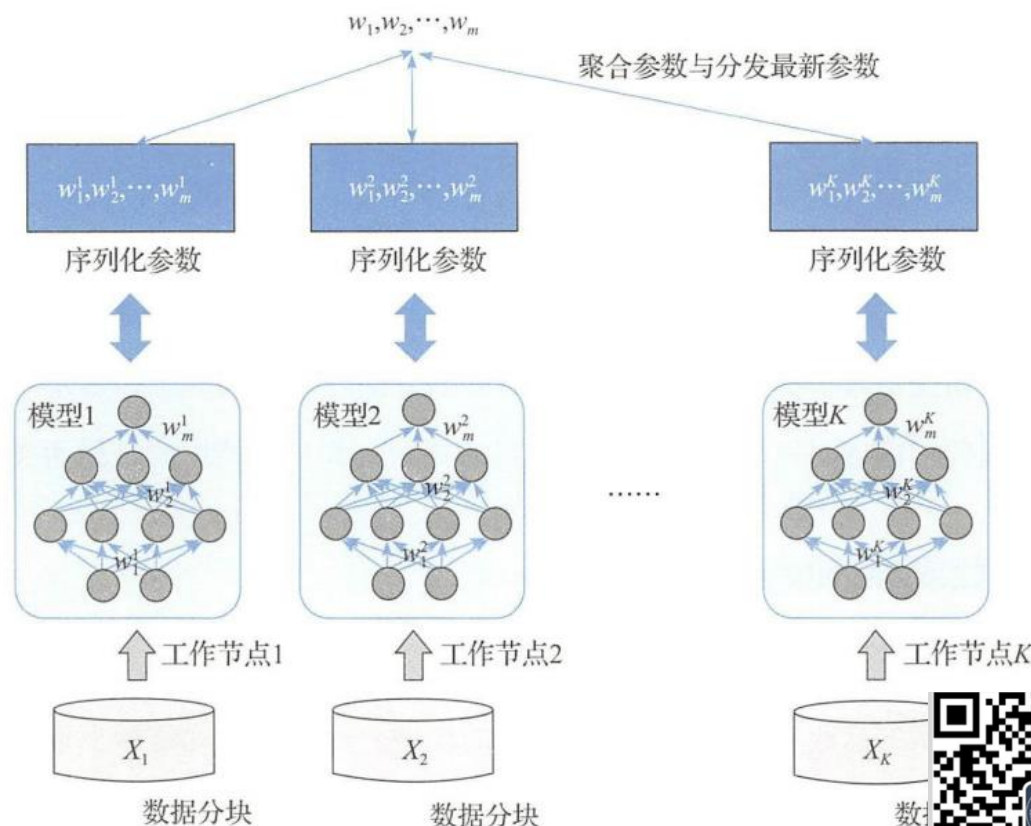
单机瓶颈



2、数据并行

提升训练速度

数据并行划分的是数据，把整个样本空间划分为多个子集，然后分发给不同的工作节点。每个工作节点拥有完整的神经网络模型，每次训练仅将一批数据输入模型，进行前向传播、计算误差、反向传播，最后进行参数更





2、数据并行

1、DP

2、DDP

3、FSDP





2.1、DP

- 单进程，多线程，只能适用于1台机器的情况（单机多卡）
- 不能结合模型并行的方法
- 使用Parameter Server (PS, 参数服务器) 架构，默认GPU 0 为主GPU。

- 负载不均衡
- 通信开销随着 GPU 数量线性增长

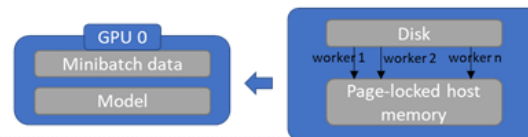


Data Parallel

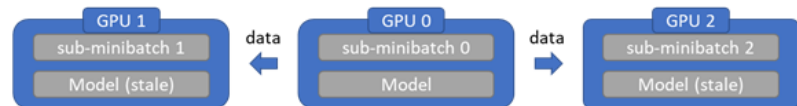
One GPU (0) acts as the master GPU and coordinates data transfer.

Implemented in PyTorch `data_parallel` module

1. Transfer minibatch data from page-locked memory to GPU 0 (master). Master GPU also holds the model. Other GPUs have a stale copy of the model



2. Scatter minibatch data across GPUs



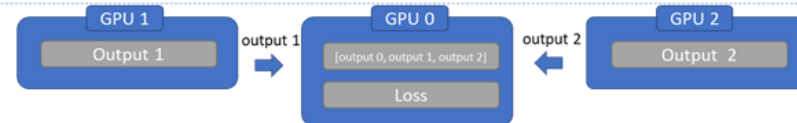
3. Replicate model across GPUs



4. Run forward pass on each GPU, compute output. Pytorch implementation spins up separate threads to parallelize forward pass



5. Gather output on master GPU, compute loss



6. Scatter loss to GPUs and run backward pass to calculate parameter gradients



7. Reduce gradients on GPU 0



8. Update Model parameters





2.2、DDP

- 多进程，可以适用于多台机器（多机多卡，也可用于单机多卡）
- 可以结合模型并行
- All-Reduce模式，只传输梯度
- 占用GPU内存比实际需要的多

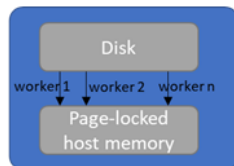
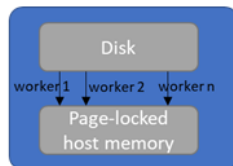
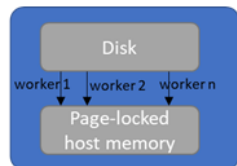


Distributed Data Parallel

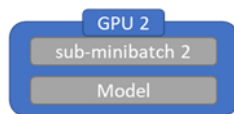
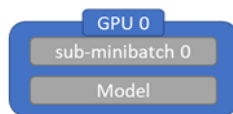
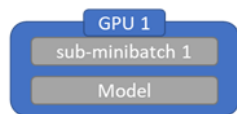
No master GPUs

Implemented in PyTorch
DistributedDataParallel
module

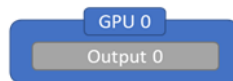
1. Load data from disk into page-locked memory on the host. Use multiple worker processes to parallelize data load. Distributed minibatch sampler ensures that each process loads non-overlapping data



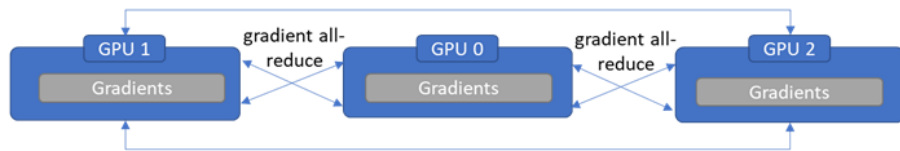
2. Transfer minibatch data from page-locked memory to each GPU concurrently. No data broadcast is needed. Each GPU has an identical copy of the model and no model broadcast is needed either



3. Run forward pass on each GPU, compute output



4. Compute loss, run backward pass to compute gradients. Perform gradient all-reduce in parallel with gradient computation



5. Update Model parameters. Because each GPU started with an identical copy of the model and gradients were all-reduced, weights updates on all GPUs are identical. Thus no model sync is required



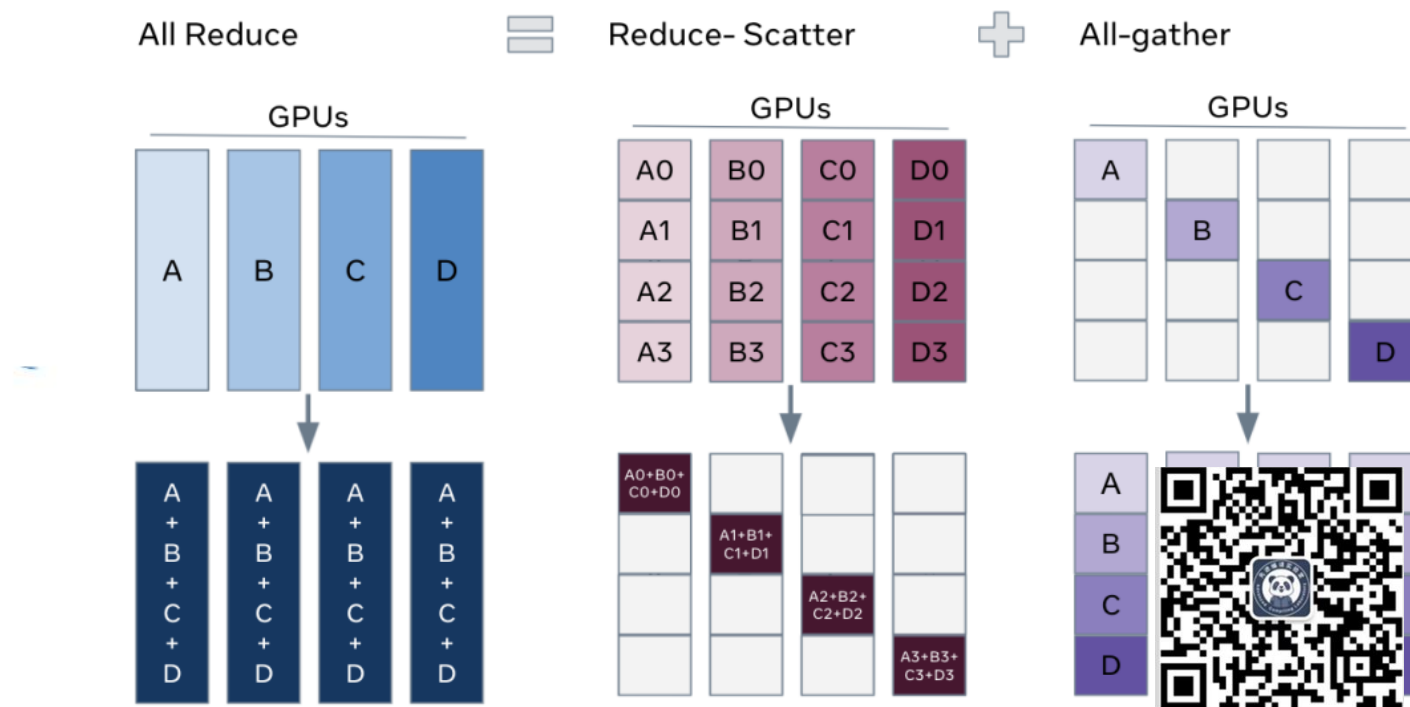


2.3、FSDP

Fully Sharded Data Parallel

FSDP shards all of **model's parameters, gradients and optimizer states** across data-parallel workers and can optionally **offload the sharded model parameters to CPUs**.

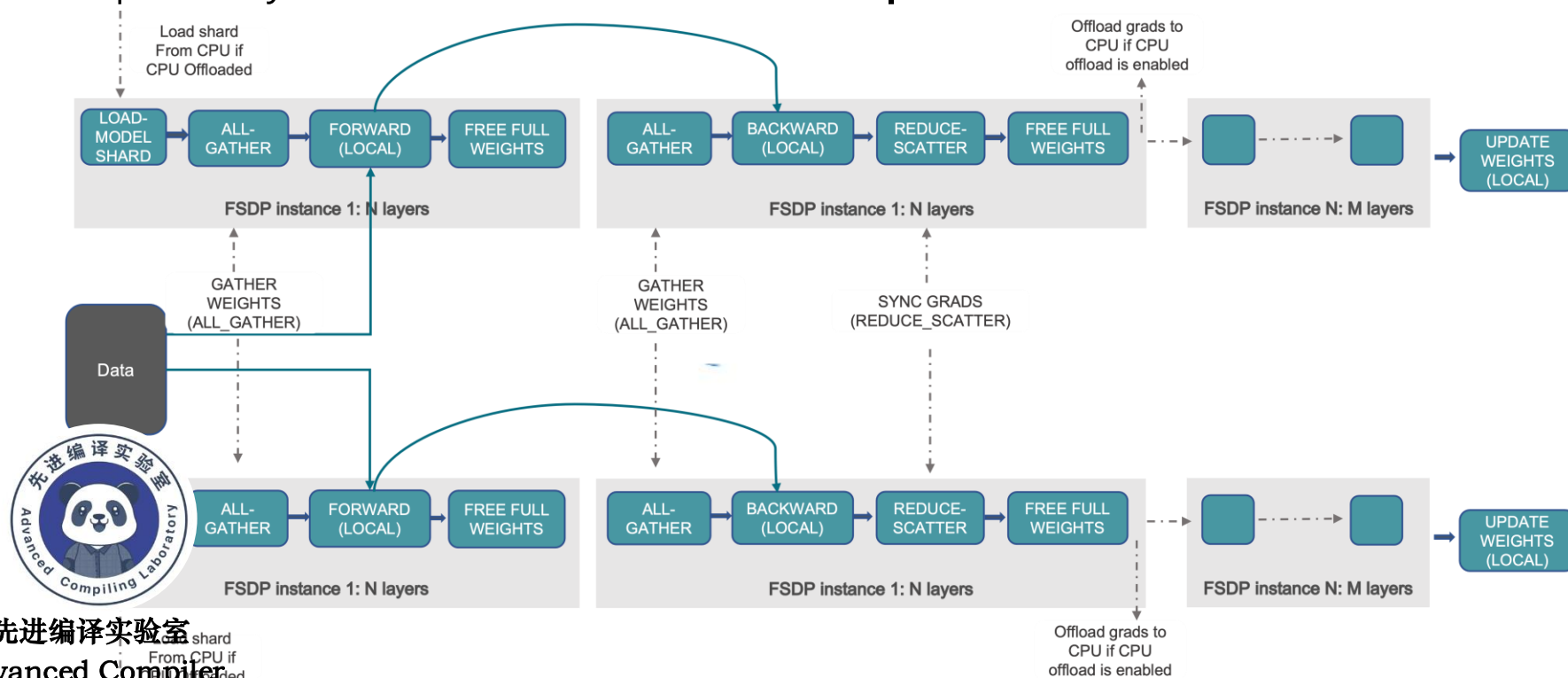
把DDP之中的All Reduce操作分解为独立的 Reduce-Scatter 和 All-gather 操作。



2.3、FSDP

Fully Sharded Data Parallel

FSDP shards all of **model's parameters, gradients and optimizer states** across data-parallel workers and can optionally **offload the sharded model parameters to CPUs**.

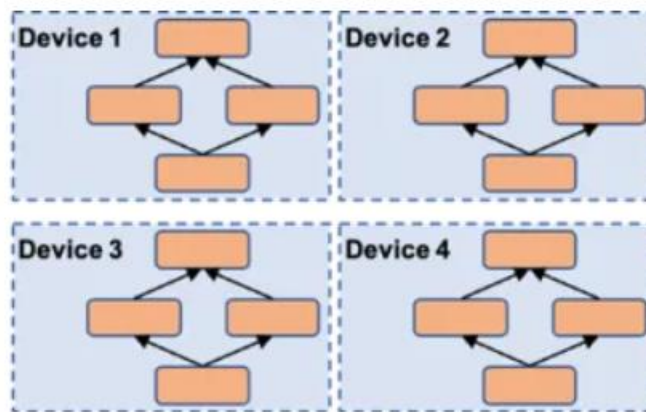




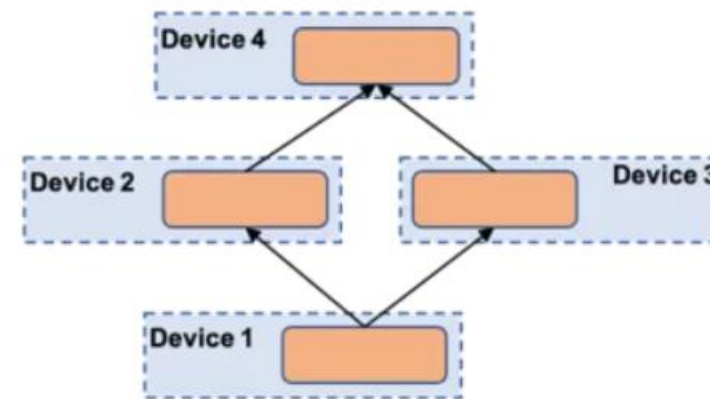
3、模型并行

解决单个训练设备的内存不足以容纳整个模型问题

将模型拆分成多个模型分片，将多个模型分片放置在不同的训练设备上



数据并行



模型并行

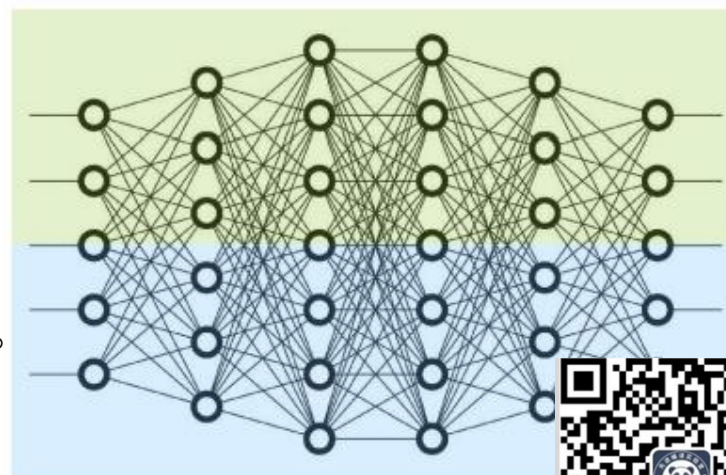
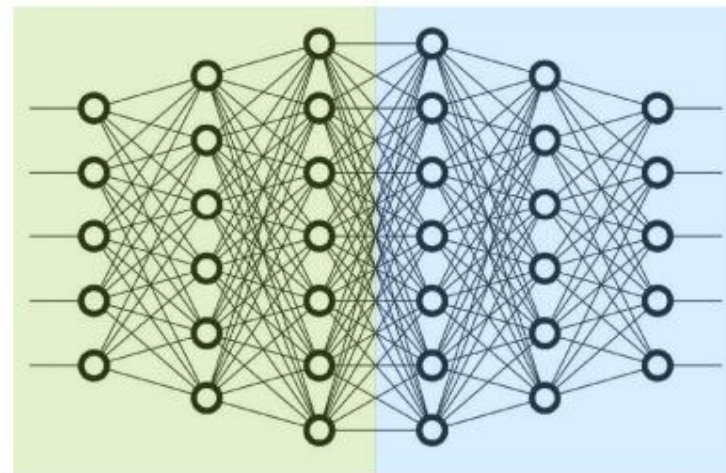




3、模型并行

流水线并行（层间）

张量并行（层内）

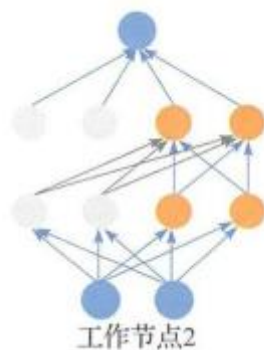
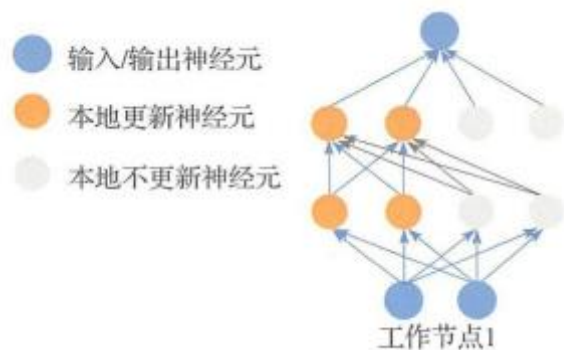


- 流水线并行是把模型不同的层放到不同设备之上。
- 张量并行则是层内分割，把某一个层做切分，放置到不同设备之上。





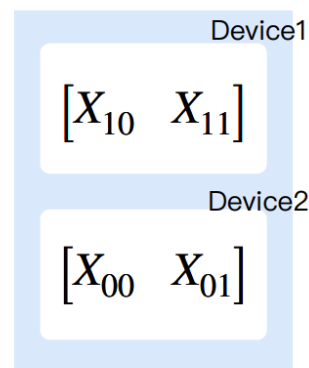
3.1、张量并行



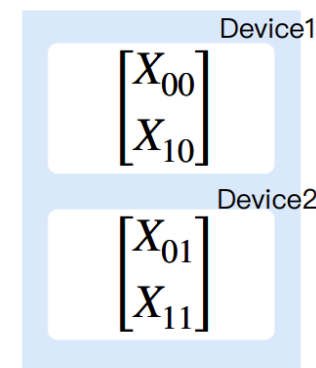
$$[X] = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$

$$[X] = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$

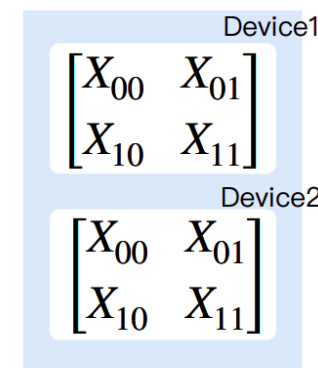
$$[X] = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$$



行切分



列切分



复制

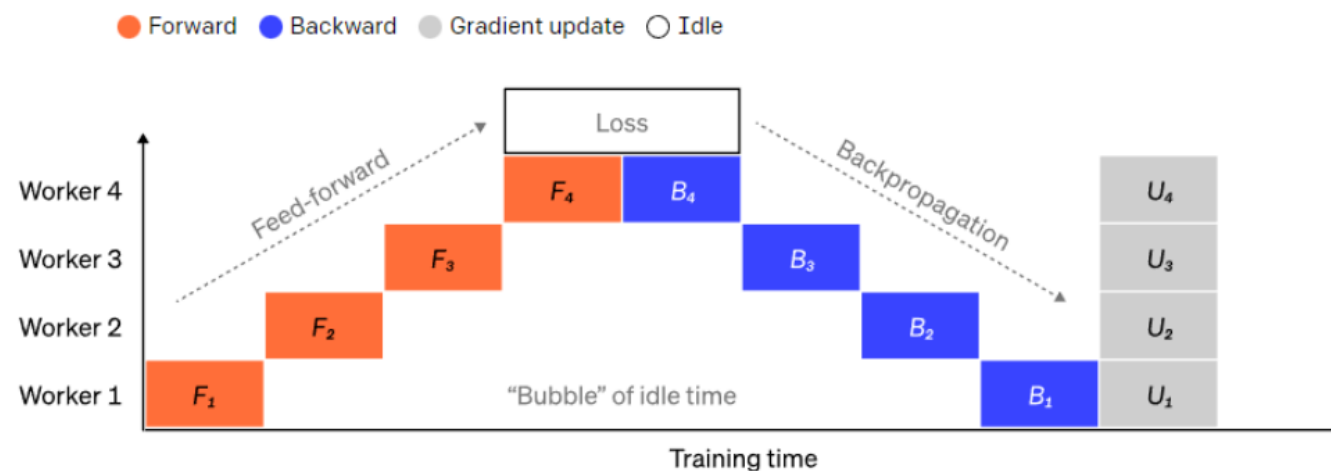
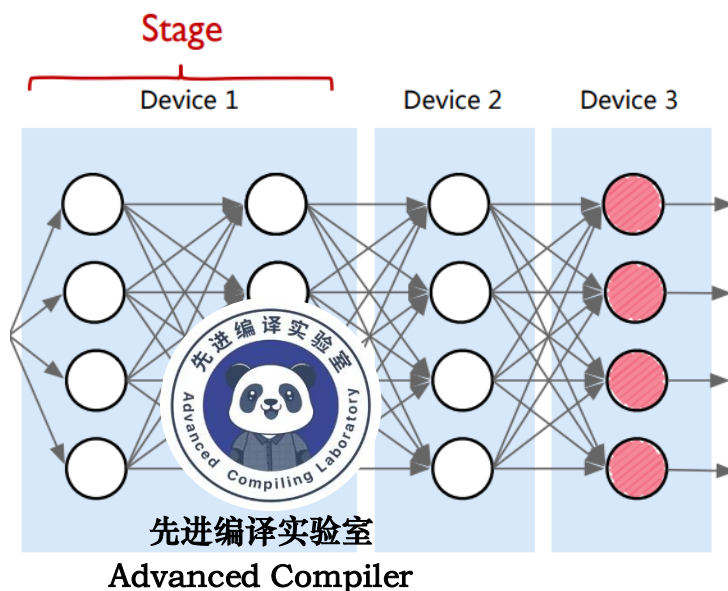
存在无法切分的层
单次通信量大，通信频繁



3.2、流水线并行

单设备只负责网络中部分层的计算

把通信和计算重叠起来以便“掩盖”通信时间



“F”、“B”和“U”分别代表前向、反向和更新操作。下标指示数据在哪个节点上运行。由于顺序依赖性，数据一次只能在一个节点上运行，从而会导致大量空闲时间。

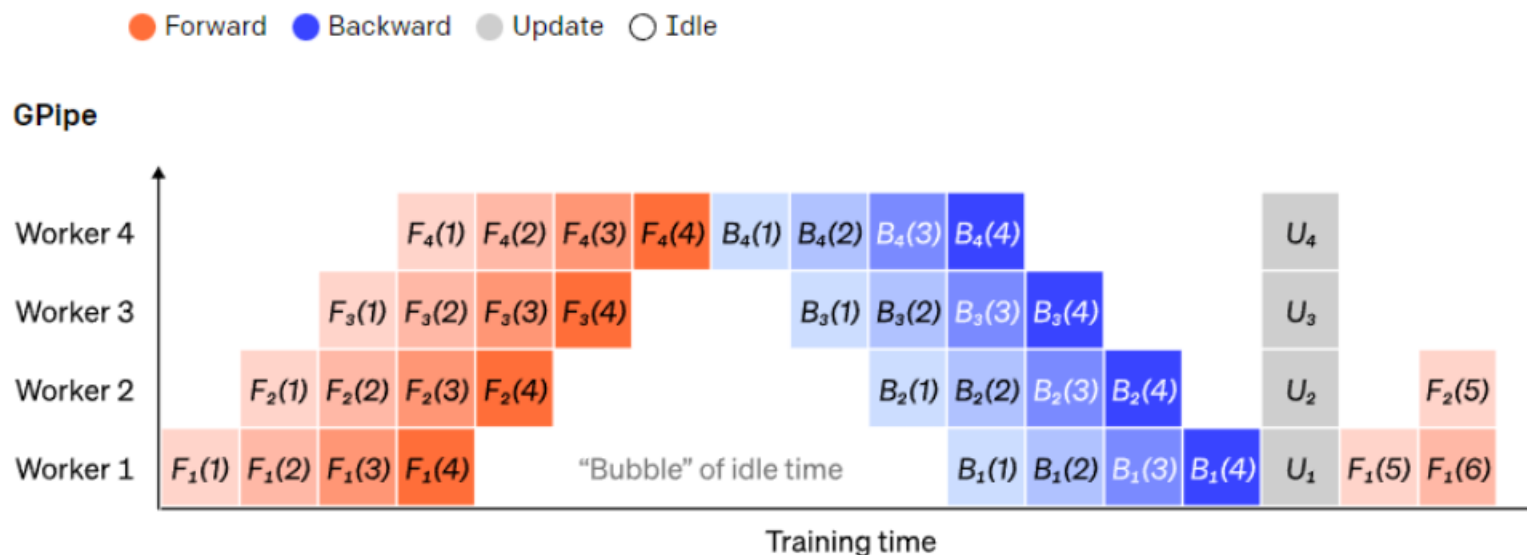




3.2、流水线并行

GPipe

- 微批法
 - 梯度累积
 - 重计算
-
- 硬件利用率低
 - 内存占用大

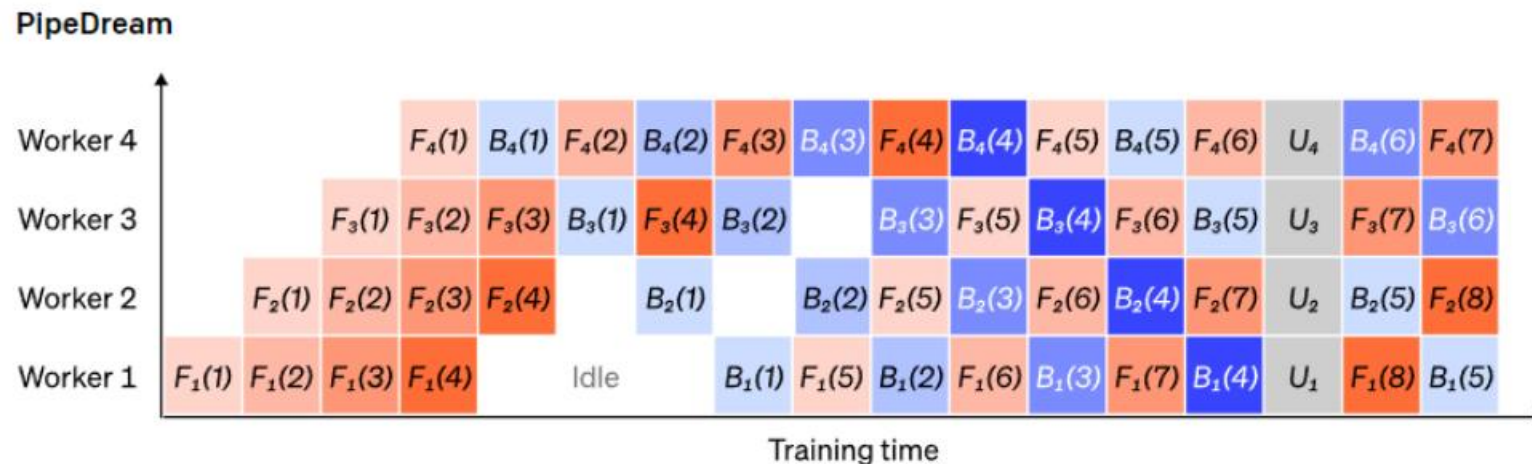




3.2、流水线并行

PipeDream

- 1F1B策略

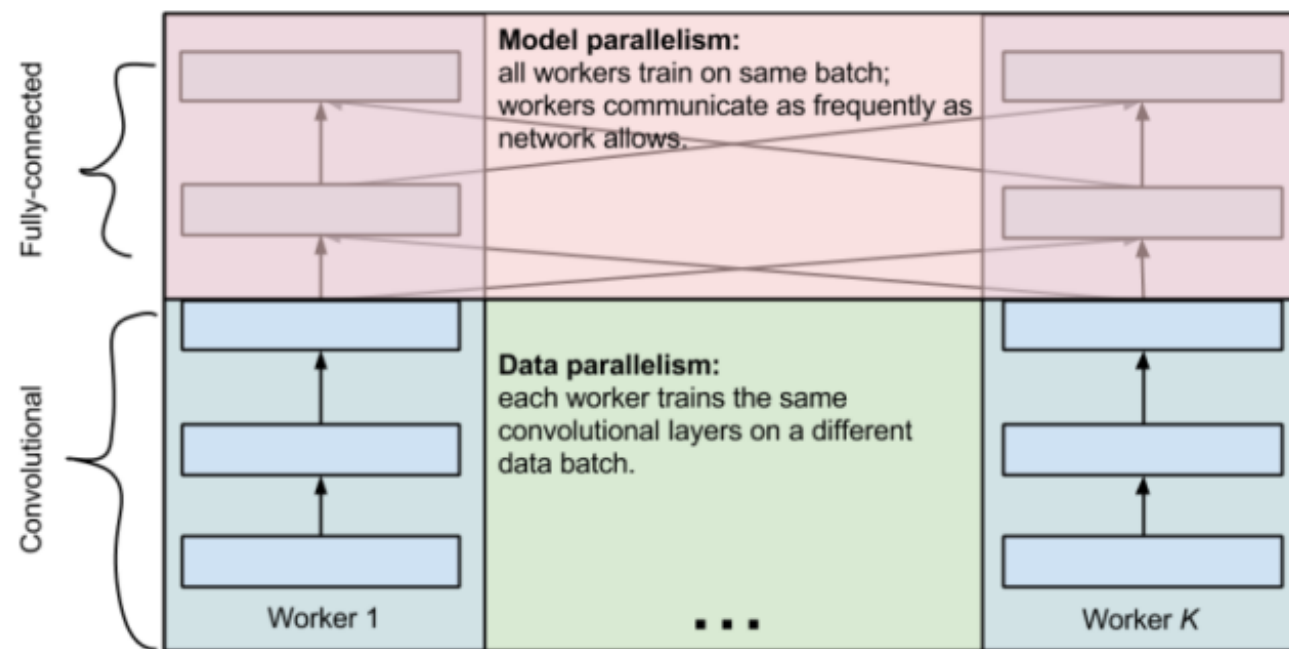




4、混合并行

简单的理解就是将数据并行和模型并行进行混合，从而达到一个更好的训练效果。

卷积层数据比参数大，适合数据并行，全连接层参数比数据大，适合模型并行。

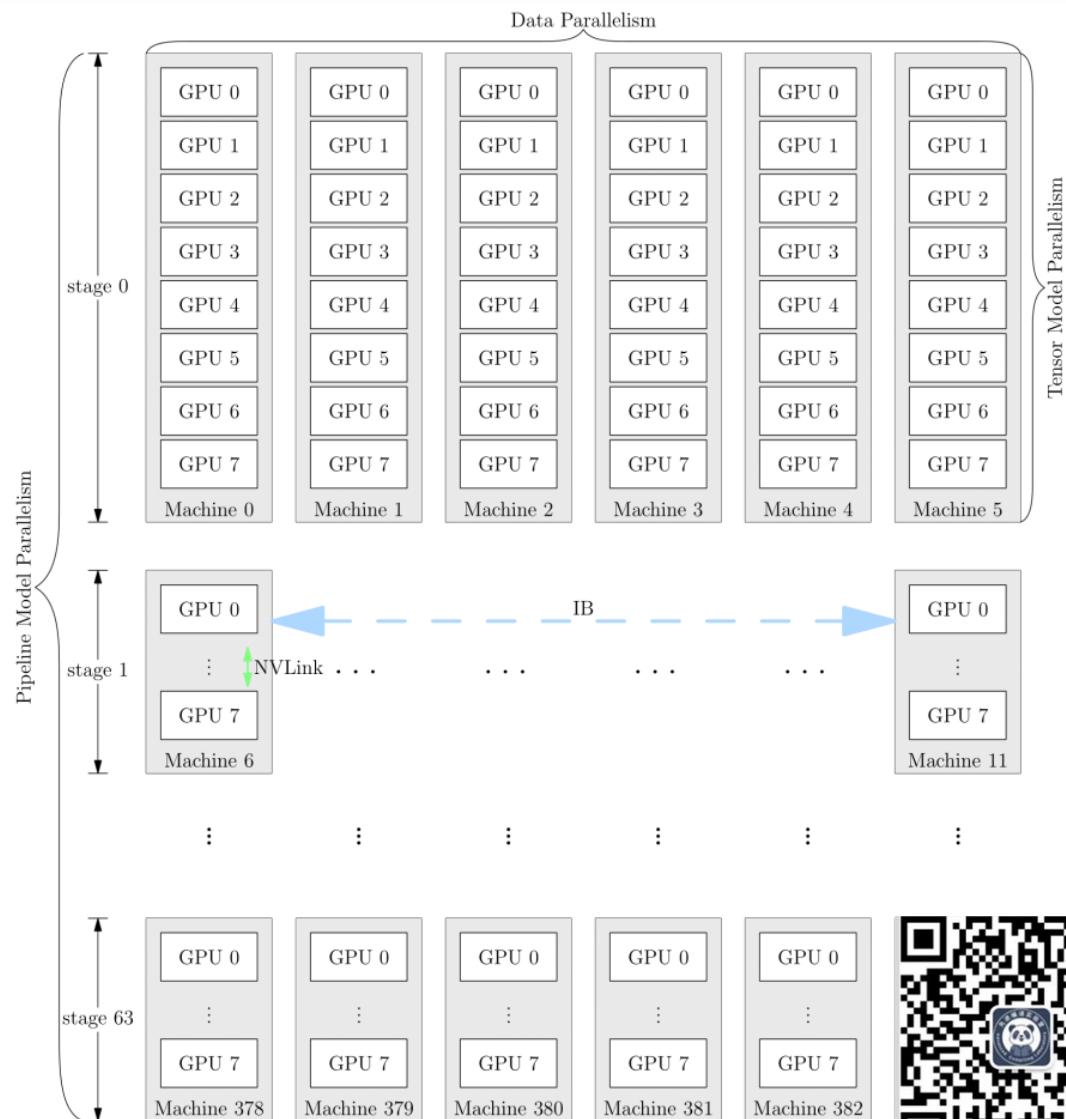




4、混合并行

以GPT-3超大模型为例

综合考虑模型特点以及硬件条件，
结合数据并行和模型并行的特点，能够
灵活满足训练对时间开销的需要





5、总结

数据并行

将数据划分到不同的设备并行训练，每个设备都有完整的网络模型。
提升训练速度，减轻单设备上计算和存储压力。

模型并行

张量并行：将模型层内划分到不同的设备并行训练。
流水线并行：将模型层间划分到不同的设备并行训练。
解决单设备无法容纳完整模型问题。



行+模型并行





感谢大家聆听

参考资料

分布式机器学习 算法、理论与实践

https://pytorch.org/tutorials/intermediate/FSDP_tutorial.html?highlight=fsdp

<http://hihu.com/zvideo/1564591472208547841>

<http://blogs.com/rossiXYZ/>

