AI 大模型开发工程师
之大模型微调核心之算力

讲师：李希沅

目录

1 预训练流程

掌握大模型核心三要素

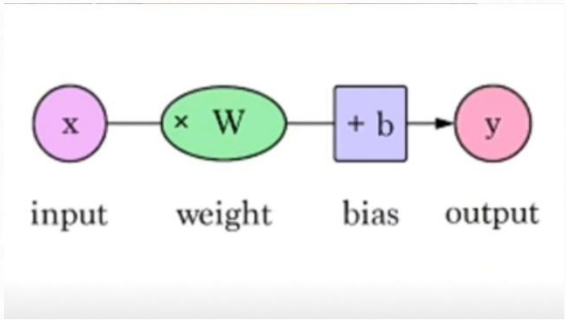
LLM 大模型预训练是什么?

1、**大语言模型预训练**：给定一个**已知输入（X）**和**已知结果（Y）**,不断修改/更新这个**大模型的参数**，让这个模型的**输出无限逼近**这个已知结果（Y）的过程。

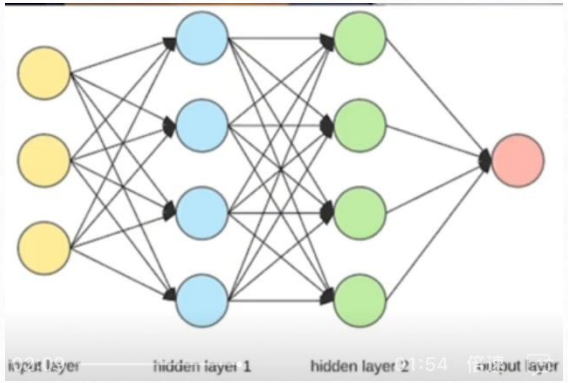
- 当这个差值（大模型的输出值和实际值之间的差异）足够小（损失函数），变成我们可以接受的状态，预训练完成。

2、神经网络的最基础的一个神经元计算单元

- X和Y的线性关系



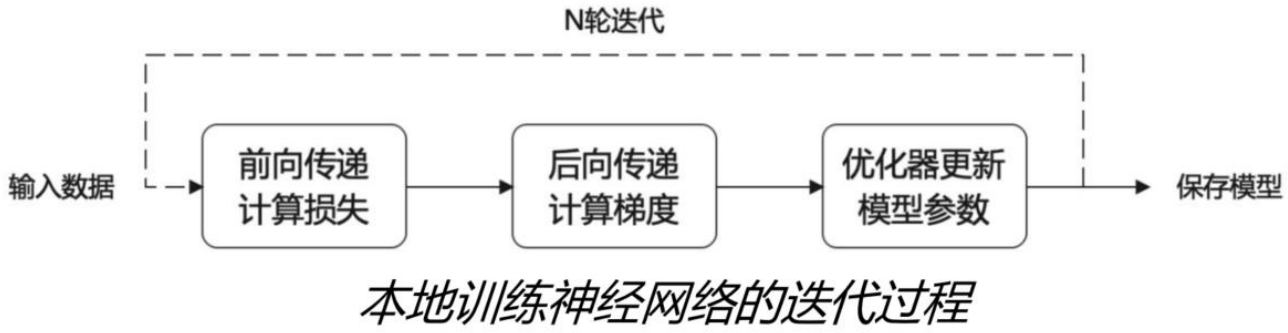
- 最基础的一个神经网络



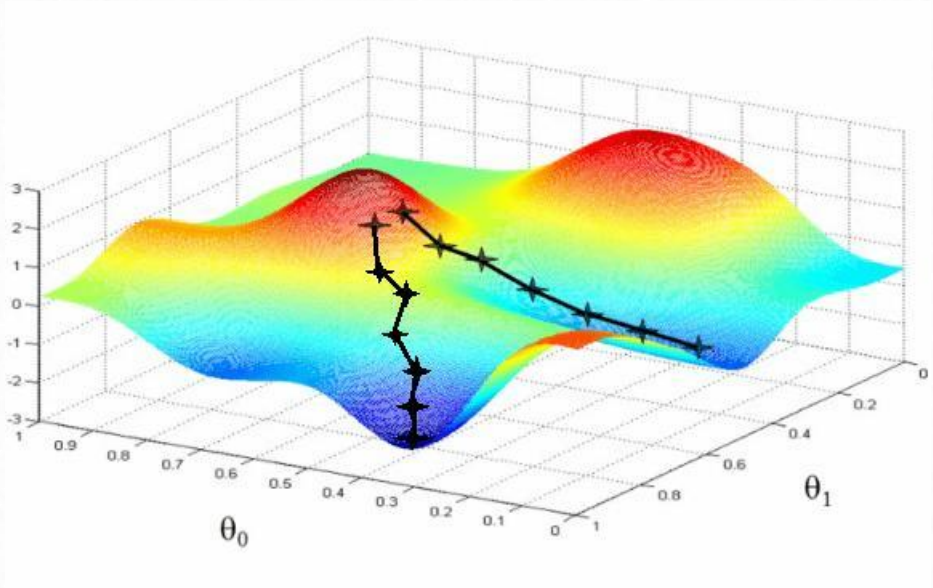
LLM 大模型本地预训练过程

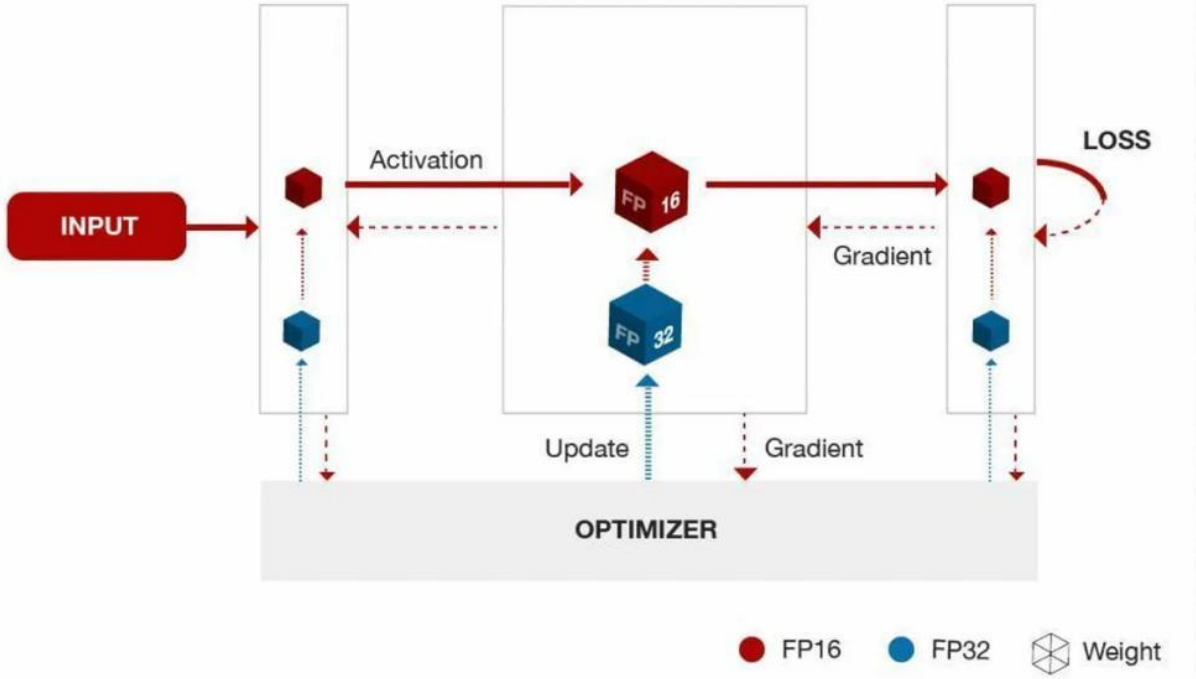
1、大语言模型预训练过程：

- ...
- 输入 Batch 数据（batch_size）
- 前向传播计算损失
- 后向传播计算梯度
- 优化器更新大模型参数
- ...



本地训练神经网络的迭代过程





2 预训练挑战

大语言模型预训练的两大挑战

1、显存效率：模型参数量太大，显存不够用
- 即使目前显存最大的 GPU 也放不下这些大模型的模型参数。
- 例如：175B 参数量的 GPT-3模型参数需要占用 700 GB（175B * 4bytes）的显存。参数梯度是700GB，Adam 优化器状态需要 1400 GB，共计需要 2.8 TB 的显存。

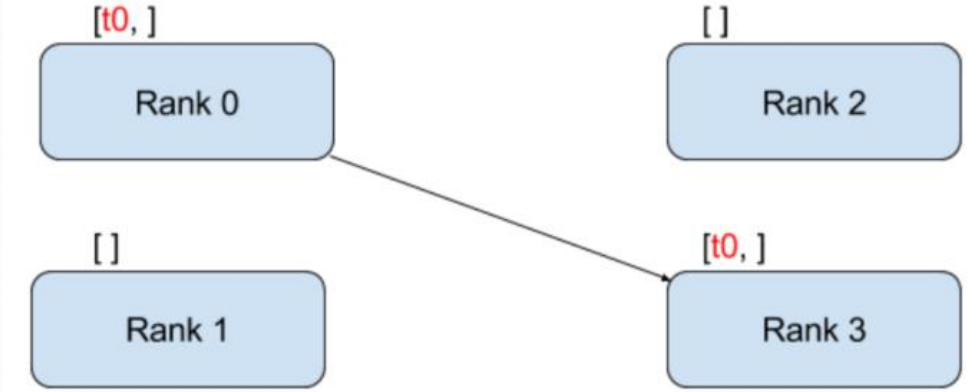
2、计算效率：训练数据量多，模型参数量大，计算量大，单机训练时间久
- 即使我们能将大模型放在一张 GPU 上，训练大模型需要的海量计算操作需要耗费很长时间。例如：用英伟达 A100 显卡训练 175B 参数量的 GPT-3 模型大约需要 288 年。

怎么办？

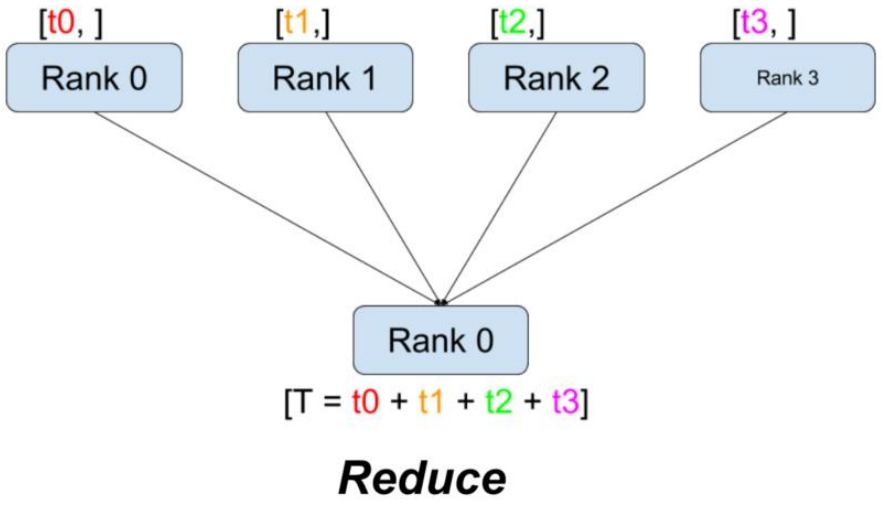
3 预训练网络通信

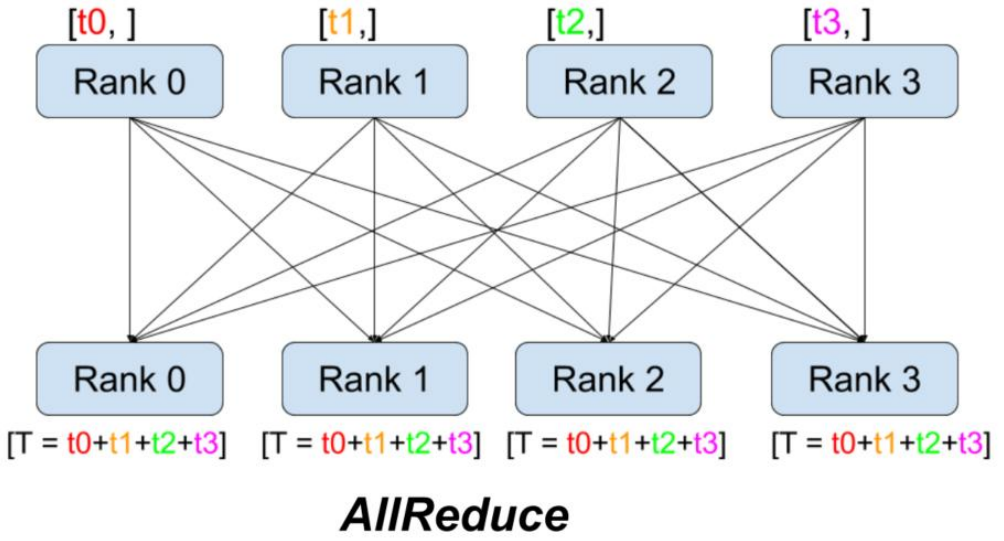
大语言模型预训练的点对点通信与集体通信

1、**点对点通信**：一个进程发送数据，一个进程接收数据，速度慢，成本低。



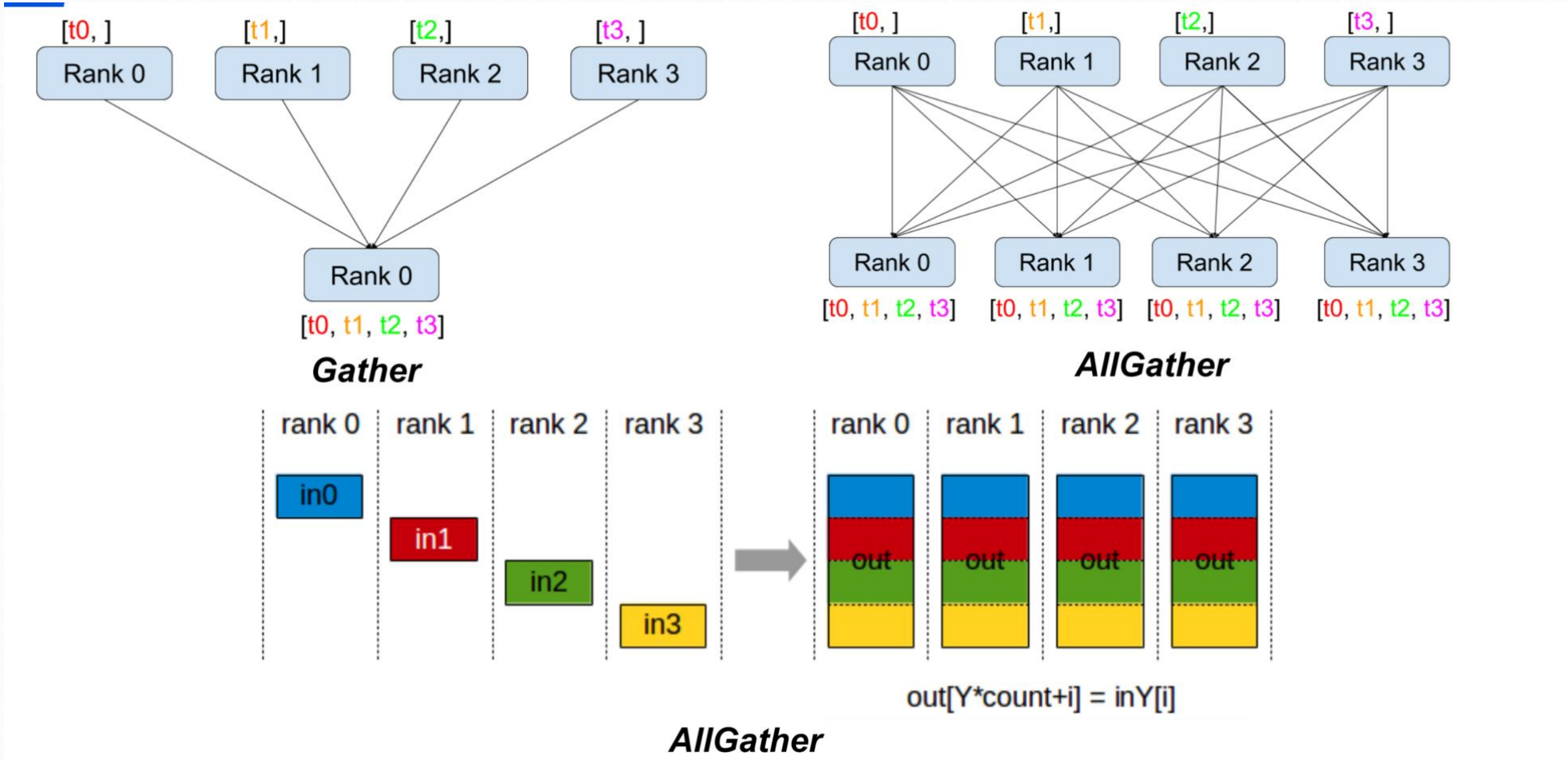
2、**集体通信**：多个进程发送数据，多个进程接收数据，速度快，成本高。





大语言模型预训练的点对点通信与集体通信

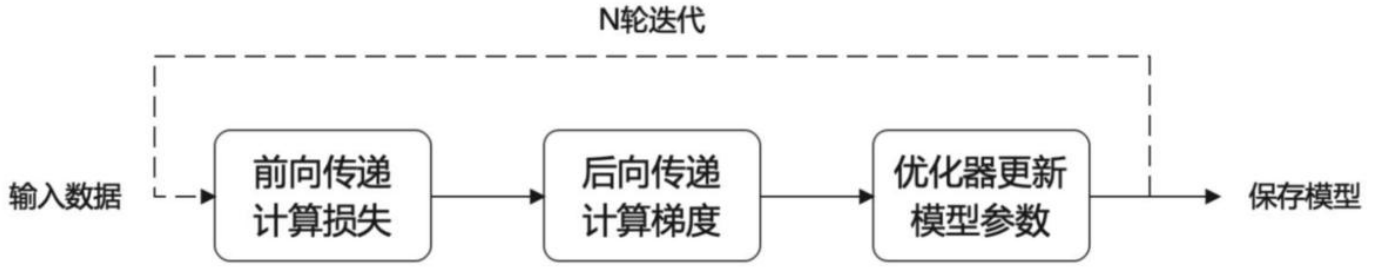
1、集体通信：多个进程发送数据，多个进程接收数据，速度慢，成本高。



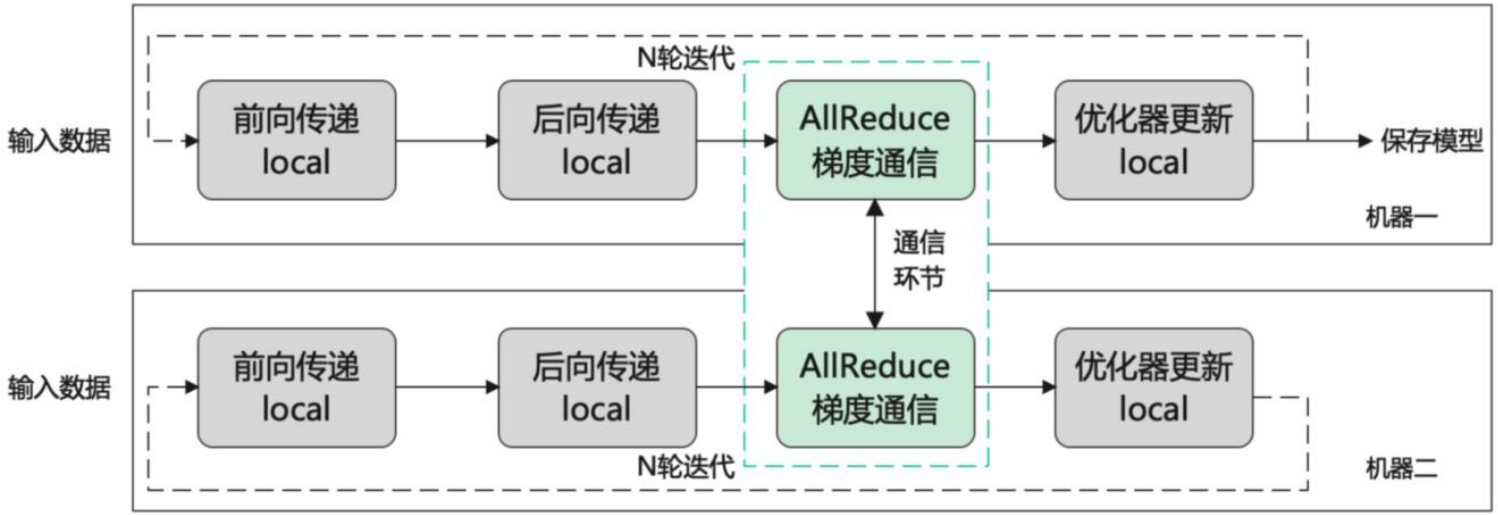
4 预训练数据并行

大语言模型预训练之数据并行

1、数据并行：指的是将整个数据集切分为多份，每张 GPU 分配到不同的数据进行训练，每个进程都有一个完整的模型副本。



本地训练神经网络的迭代过程

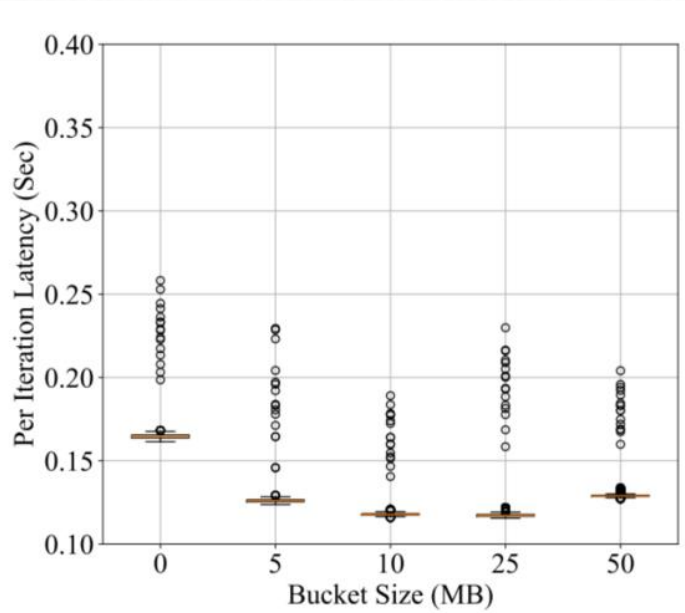


数据并行的迭代过程

大语言模型预训练之数据并行优化

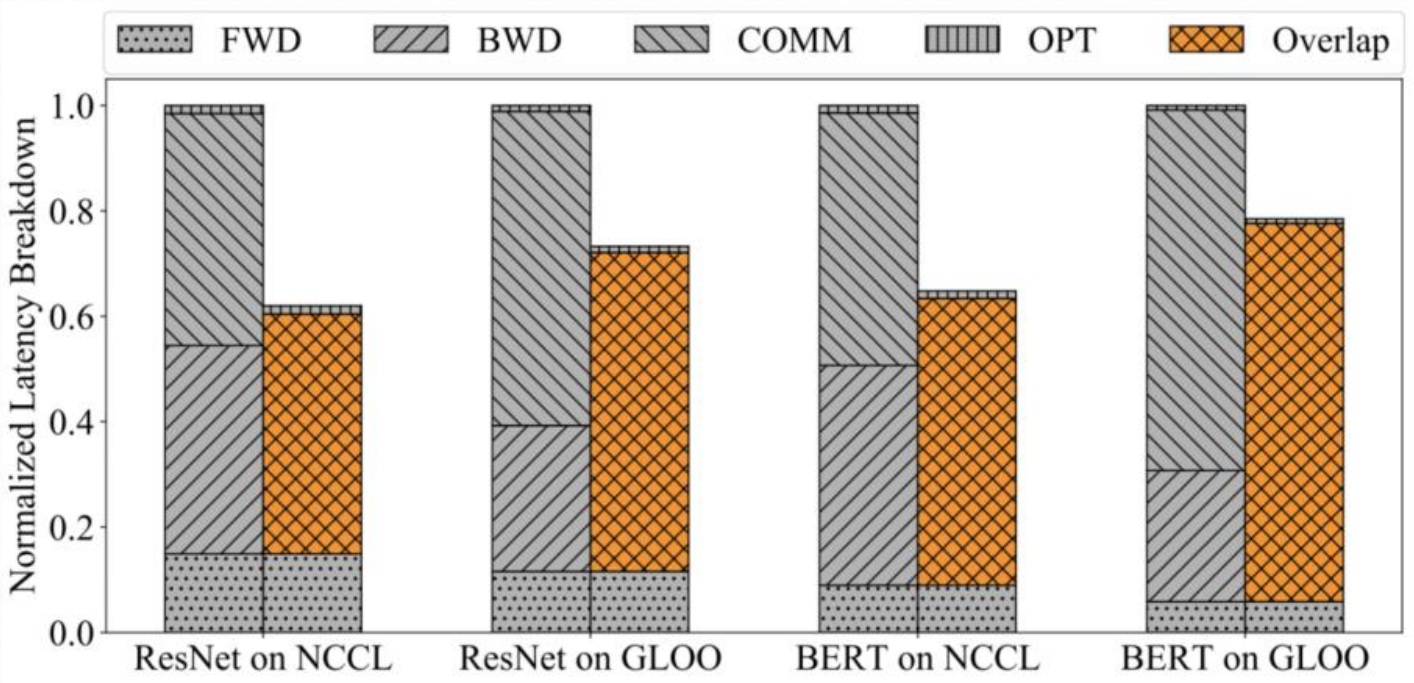
1、数据并行三个提高效率的技巧

- 梯度分桶：动机是集体通信在大张量上比在小张量上效率更高。
- 计算与通信重叠：有了梯度分桶之后，在等待同一个桶内的梯度计算完后，就可以进行通信操作。
- 跳过梯度同步：梯度累加，减少梯度通信的频次。



(a) ResNet50 on NCCL

梯度分桶bucket_size的影响

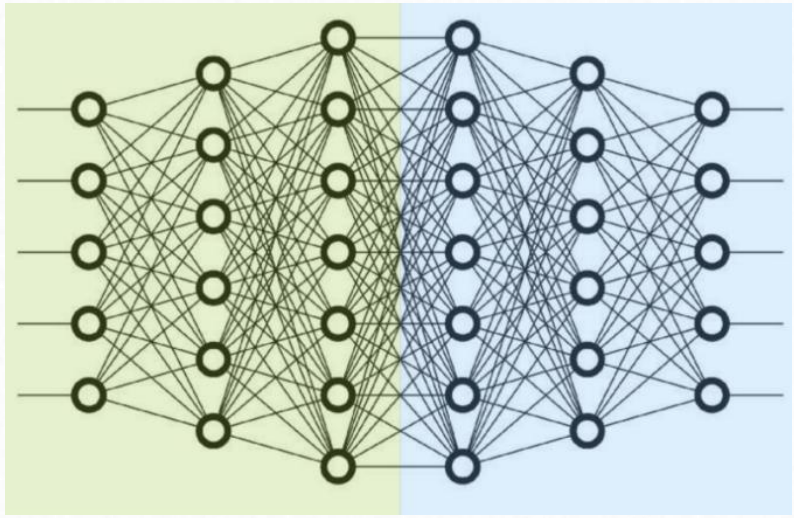


5 预训练模型并行

大语言模型预训练之模型并行

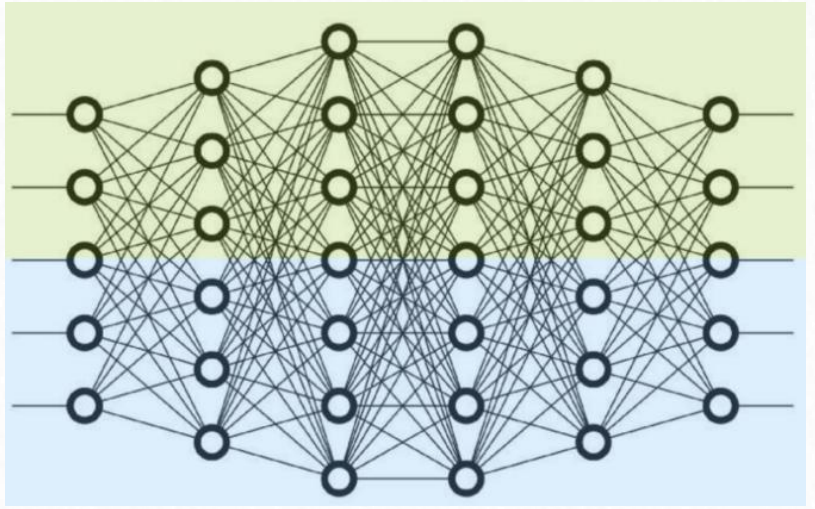
1、流水线并行（inter-layer）

- 层间划分，将不同的层划分到不同的 GPU 上
- 前 3 层在 0 号卡上，后 3 层在 1 号卡上



2、张量并行（intra-layer）

- 层内划分，切分一个独立的层划分到不同的 GPU 上
- 0 号卡和 1 号卡分别计算某个层的不同部分



大语言模型预训练之模型并行

对于一个简单的矩阵乘法GEMMs$Y = XA$.

按照对权重矩阵A的分块方式，张量并行分为行并行和列并行。

行并行

权重矩阵A按行分为两块，

同时将输入X按列分为两块。

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}; X = [X_1 \quad X_2]$$

$$XA = [X_1 \quad X_2] \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = X_1A_1 + X_2A_2 = Y_1 + Y_2 = Y$$

列并行

将权重矩阵A按列来分成两块，

不用分割输入X。

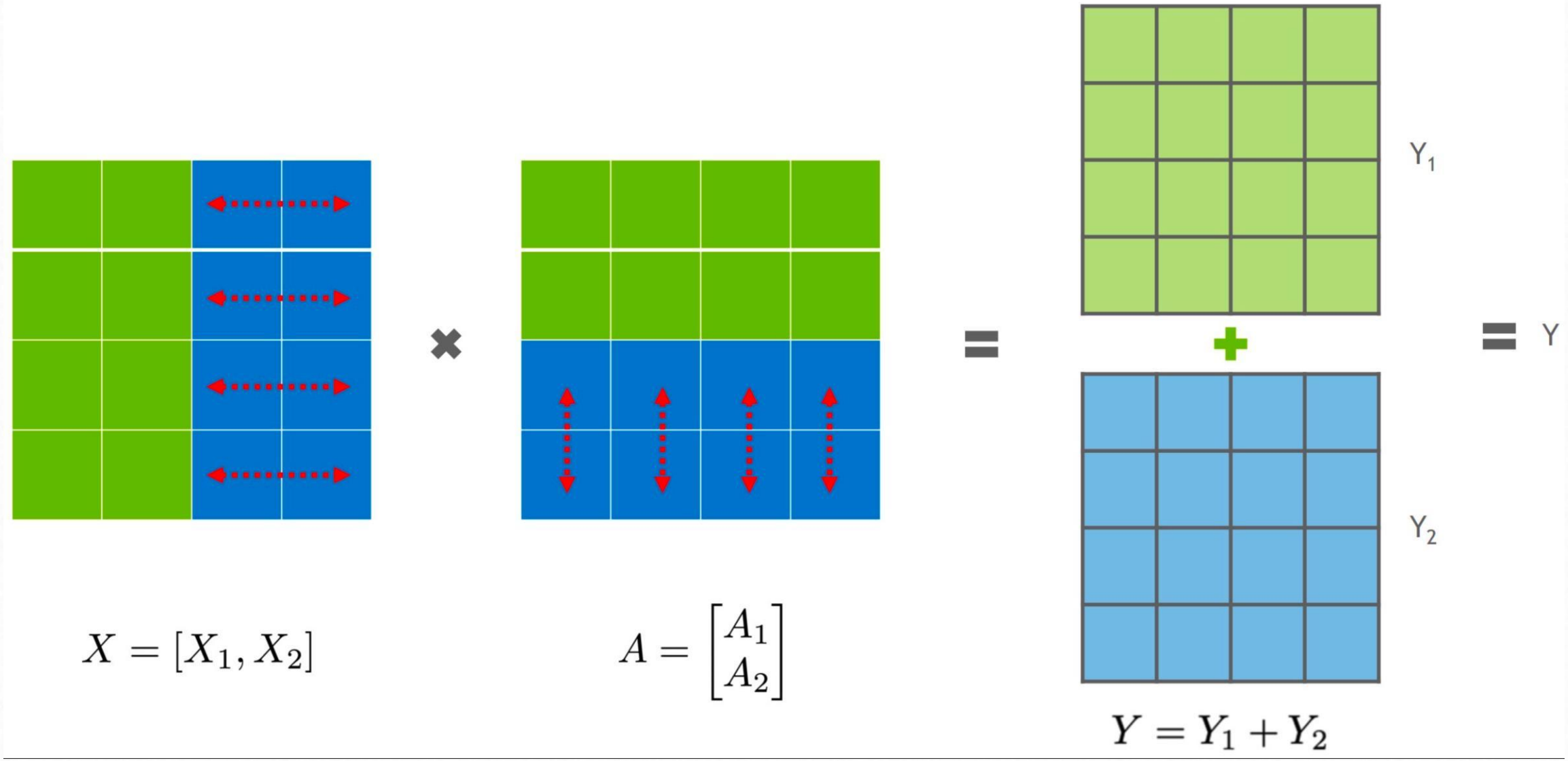
$$A = [A_1 \quad A_2]$$

$$XA_1 = Y_1$$

$$XA_2 = Y_2$$

$$XA = Y = [Y_1, Y_2]$$

大语言模型预训练之行并行

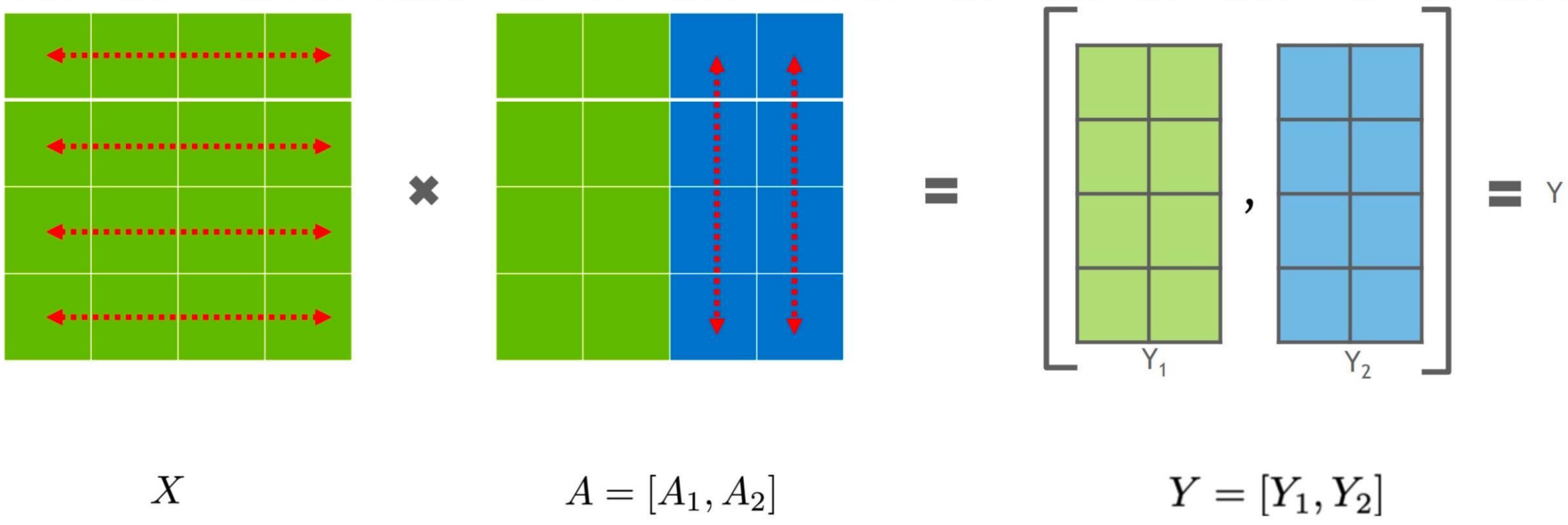


$$X = [X_1, X_2]$$

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

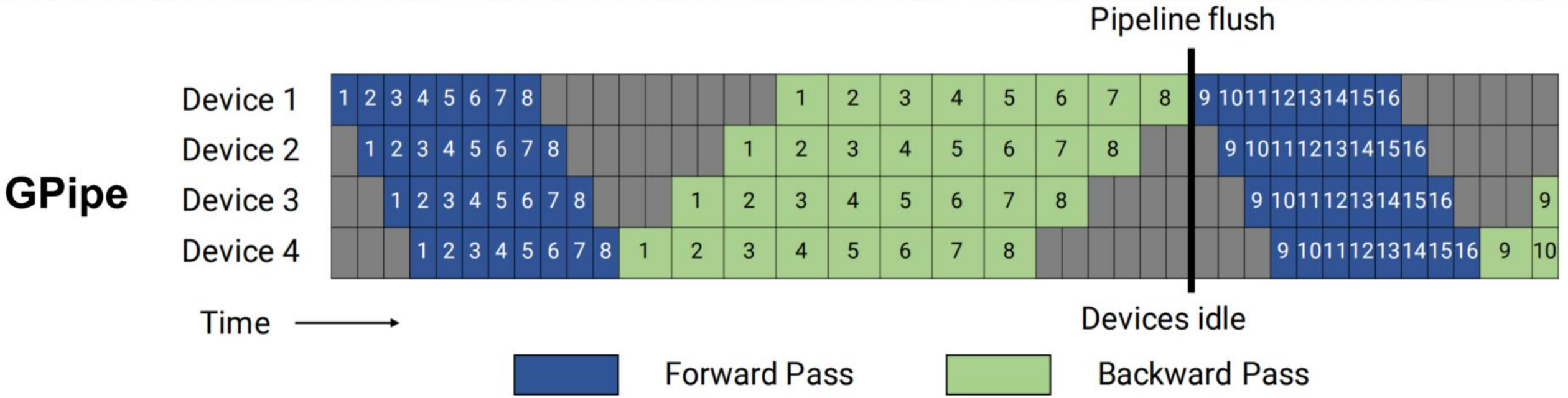
$$Y = Y_1 + Y_2$$

大语言模型预训练之列并行



X $A = [A_1, A_2]$ $Y = [Y_1, Y_2]$

大语言模型预训练之流水线并行



6 预训练3D并行

大语言模型预训练之3D并行

1、数据并行：计算效率高、实现简单。
- 显存效率：每张卡上都保存了完整的模型、梯度、优化器状态，因此显存效率不高。
- 计算效率：当增加并行度时，单卡的计算量是保持恒定的，可以实现近乎完美的线性扩展。但规约梯度的通信开销，与模型大小成正相关。

2、张量并行：因模型结构而异，实现难度大。
- 显存效率：随着并行度增加，成比例地减少显存占用。是减少单层神经网络中间激活的唯一方法。
- 计算效率：频繁的通信，限制了两个通信阶段之间的计算量，影响了计算效率，计算效率很低。

3、流水线并行：通信成本最低
- 显存效率：减少的显存与流水线并行度成正比。但流水线并行不会减少每层中间激活的显存占用。
- 计算效率：成本更低的点对点（P2P）通信。通信量与流水线各个阶段边界的激活值大小成正比。

总结：3D并行是由数据并行(DP)、张量并行(TP)和流水线并行(PP)组成.
DP: Data Parallelism　TP:Tensor Parallelism　PP:Pipeline Parallelism

大语言模型预训练之3D并行实例

Bloom-176B 进行预训练时，在 384 张 NVIDIA A100 80GB GPU (48 个节点) 上使用了 3D 并行（数据并行、流水线并行、张量并行 ）策略，针对 350B 个Token 训练了大约 3.5 个月

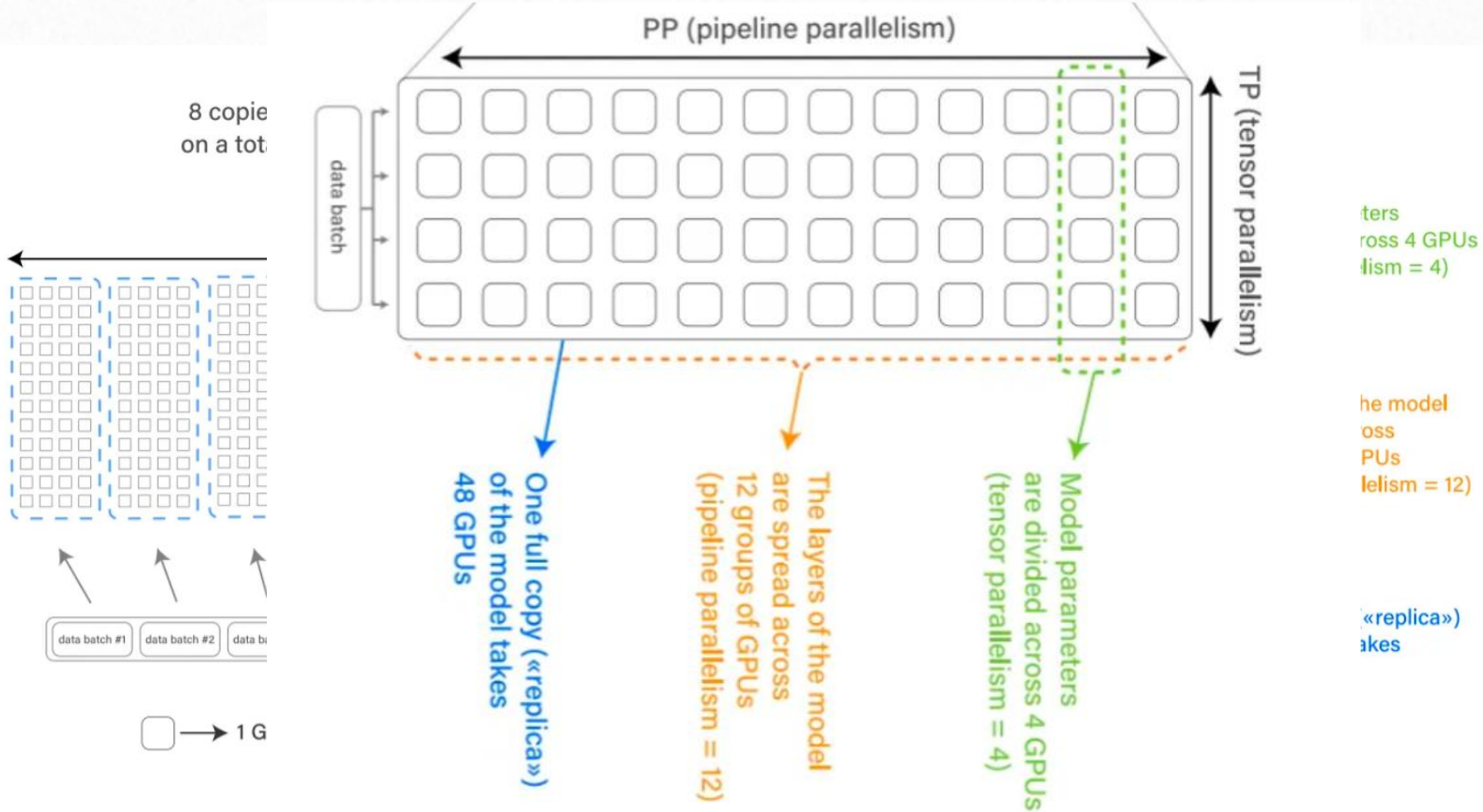


Figure 6: DP+PP+TP combination leads to 3D parallelism.

大语言模型预训练之3D并行实例

1、数据并行 + 流水线并行 + 数据并行 3D 并行训练框架

- Microsoft DeepSpeed
- NVIDIA Megatron

谢谢观看