预训练大模型并行加速技术分享

张 艳
鹏城实验室网络智能研究部开源所

大规模模型并行训练

数据并行（DP）:相同的模型分布在不同的GPU卡上，在不同的GPU上使用不同的数据

模型并行（MP）:将模型切分到不同的GPU上，减少单卡参数量

流水线并行（PP）:基于模型并行，一个batch结束前开始下一个batch,以充分利用计算资源

混合并行（HP）:混合使用上述的两种或三种

数据并行

同步训练：每个前向、反向结束后显式同步

- 适合同构场景
- 传统中心化PS：存在性能瓶颈
- All-Reduce：目前最广泛采用，几乎所有框架都支持

异步训练：只进行部分同步或不显式同步

- 适合异步训练，可能导致潜在的收敛性问题
- 传统异步方法：ASGD等
- 其他：把其中部分的计算节点组成一个组，每次在这个组之内进行梯度的汇总和更新

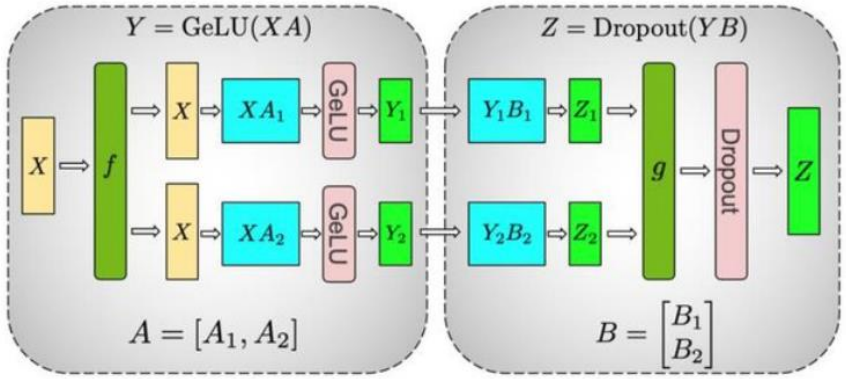
新型Parameter Server：BytePS

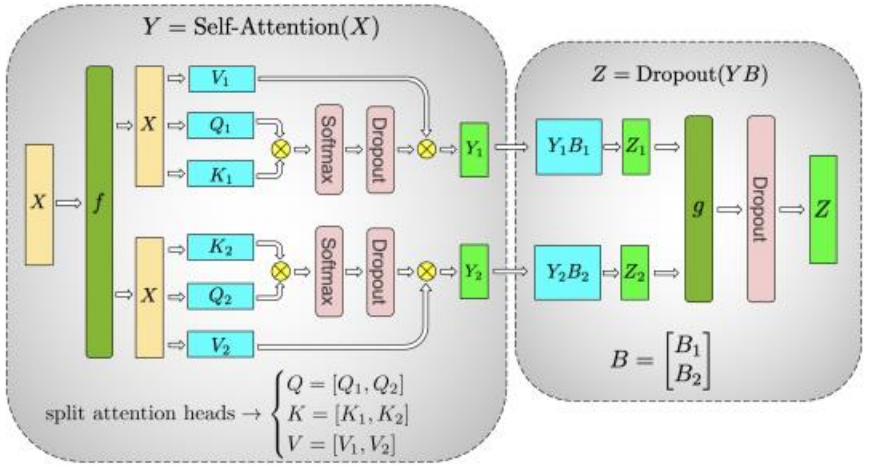
模型并行（megatron-LM）

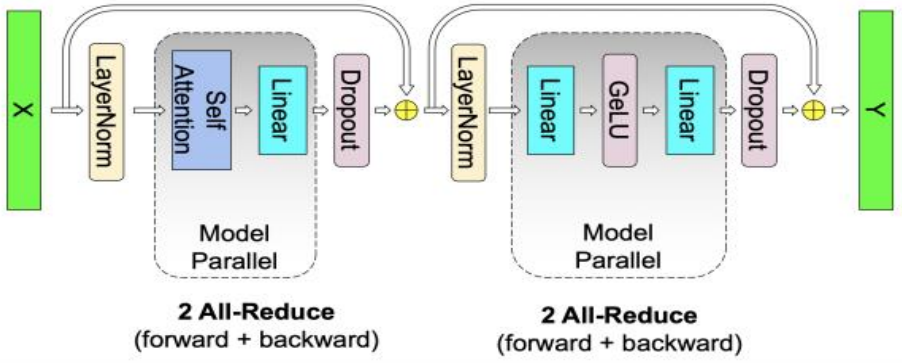
手动并行：提出transformer模型并行原语

两种切分方式：行切分、列切分

实际的transformer中，一个self attention+两层mlp，前向总共做两次all reduce的通信，反向也只要做两层all reduce的通信



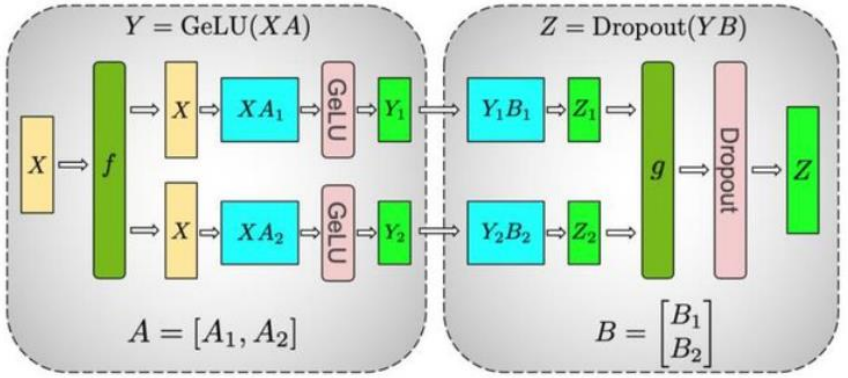




模型并行

目标：找到最优的切分方式，减少训练过程中的通信次数

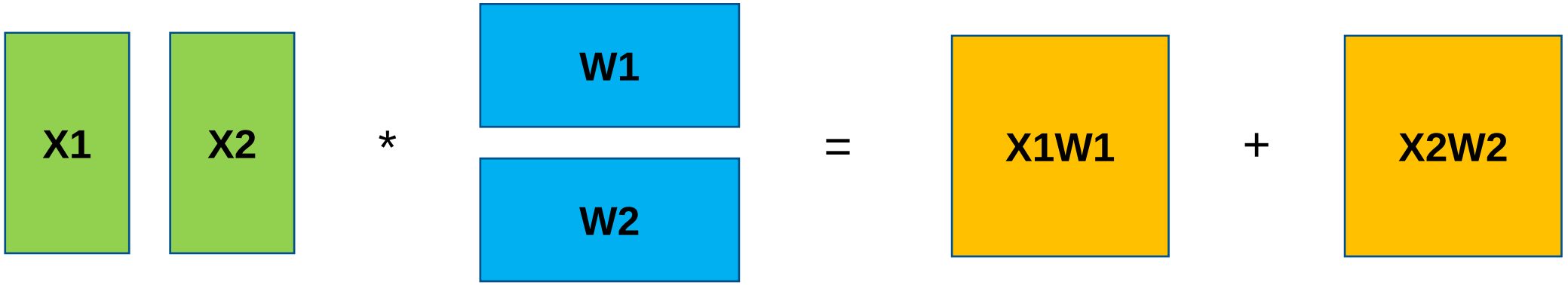
例子：多GPU计算两个张量的乘积?
Y=XW（X:输入 W:权重 Y:输出）



W的切分	切分后	倒退X的切法	要求
横切	$\begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$	[X1,X2]	X1的最后一个维度=W1的最前一个维度
纵切	[W1,W2]	X	X2的最后一个维度=W2的最前一个维度
$Y=XW=[X_1, X_2]\begin{bmatrix} W_1 \\ W_2 \end{bmatrix}=X_1W_1+X_2W_2$			

模型并行 矩阵W横向切

$$Y=XW=[X_1,X_2]\begin{bmatrix}W_1\\W_2\end{bmatrix}=X_1W_1+X_2W_2$$



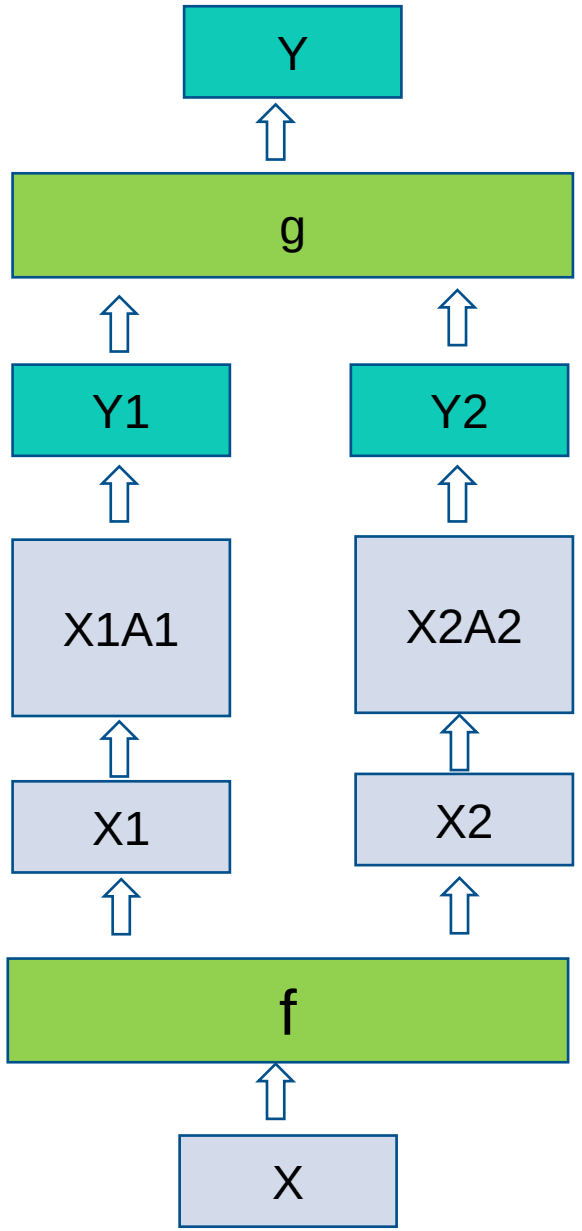
2）X的切分
X=[bs, seq_len,hidden_size] =[32,1024,1024]，则
X1=[32,1024,512]
X2=[32,1024,512]

1）对W横向平均切
W=[1024,1024],切完后W1=[512,1024];
W2=[512,1024];

3）X1W1的维度是：
X1=[32,1024,512]
W1=[512,1024]
X1W1=[32,1024,1024]

3）X2W2的维度是：
X2=[32,1024,512]
W2=[512,1024]
X2W2=[32,1024,1024]

模型并行 矩阵W横向切



forward: $Y = Y_1 + Y_2$ (all-reduce)

backward: $\frac{\partial L}{\partial \mathrm{Y}_i} = \frac{\partial L}{\partial \mathrm{Y}}$

$$A = \begin{bmatrix} \mathrm{A}_1 \\ \mathrm{A}_2 \end{bmatrix}$$

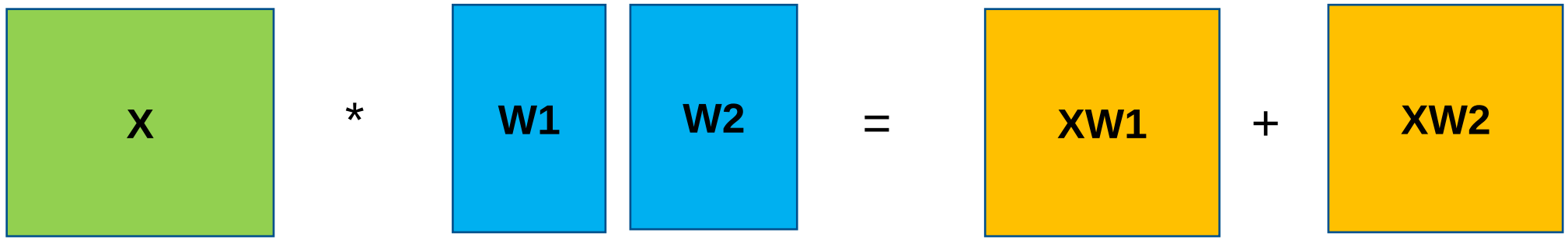
$$\mathrm{X} = [X_1, X_2]$$

forward: $X_i(split)$

backward: $\frac{\partial L}{\partial x} = [\frac{\partial L}{\partial X_1}, \frac{\partial L}{\partial X_2}]$ (all-gather)

模型并行 矩阵W纵向切

$$Y = XW = X[W_1, W_2] = [XW_1, XW_2]$$

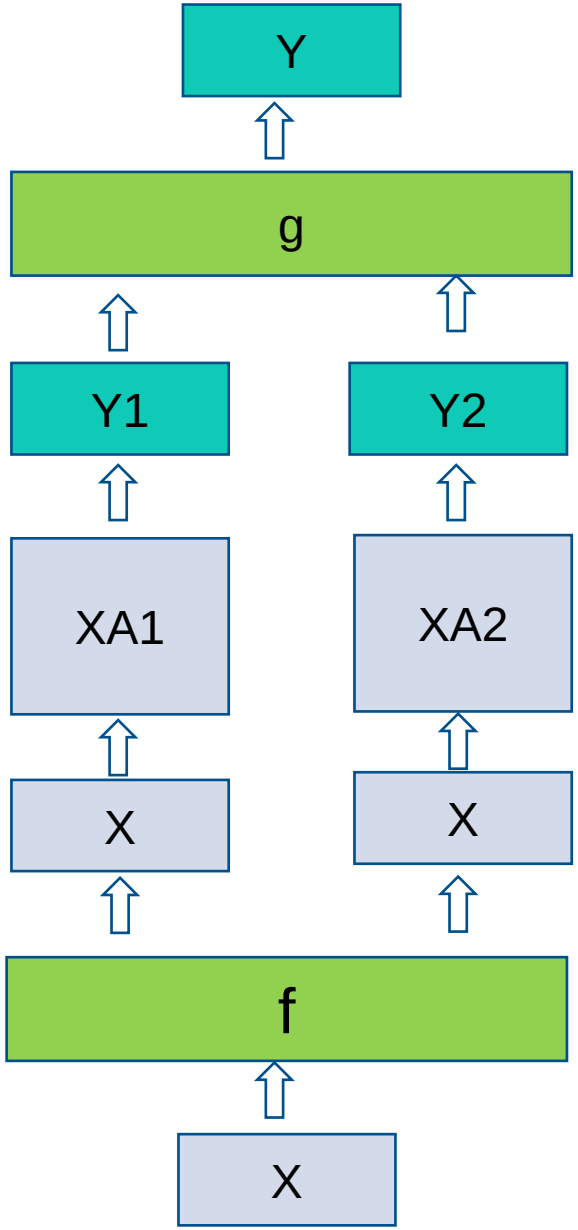


2）因为W1和W2的行数不变，为了保证X和W1、W2相乘，则X的列数不变
X=[bs, seq_len,hidden_size]
=[32,1024,1024]

1）对W对列平均切
W=[1024,1024],切完后W1=[1024,512];
W2=[1024,512];

3）XW1的维度是：
[32,1024,512]
XW2的维度也是：
[32,1024,512]
两者最后一个维度串联起来得到[32,1024,1024]

模型并行 矩阵W纵向切



forward: $Y = [Y_1,\ Y_2]$ (all-gather)

backward: $\frac{\partial L}{\partial Y_i}\ (split)$

$$A = [A_1, A_2]$$

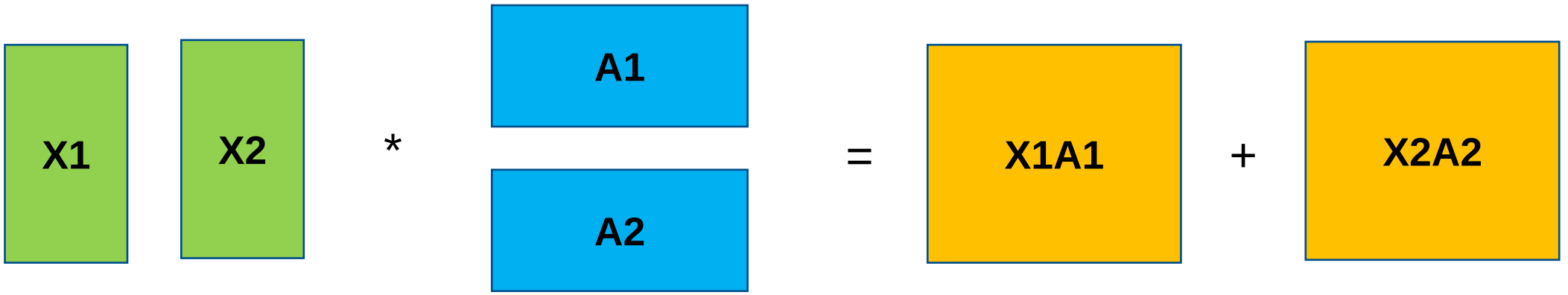
forward: X

backward: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial X_1} + \frac{\partial L}{\partial X_2}$ (all-reduce)

Megatron：MLP的多GPU实现

Transformer中MLP，两个线性层，实现了hidden_size -> 4*hidden_size -> hidden_size的变换
XA → RELU → XB → Dropout

XA、XB这两个线性层的并行性？ 横横、横纵、纵横、纵纵



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

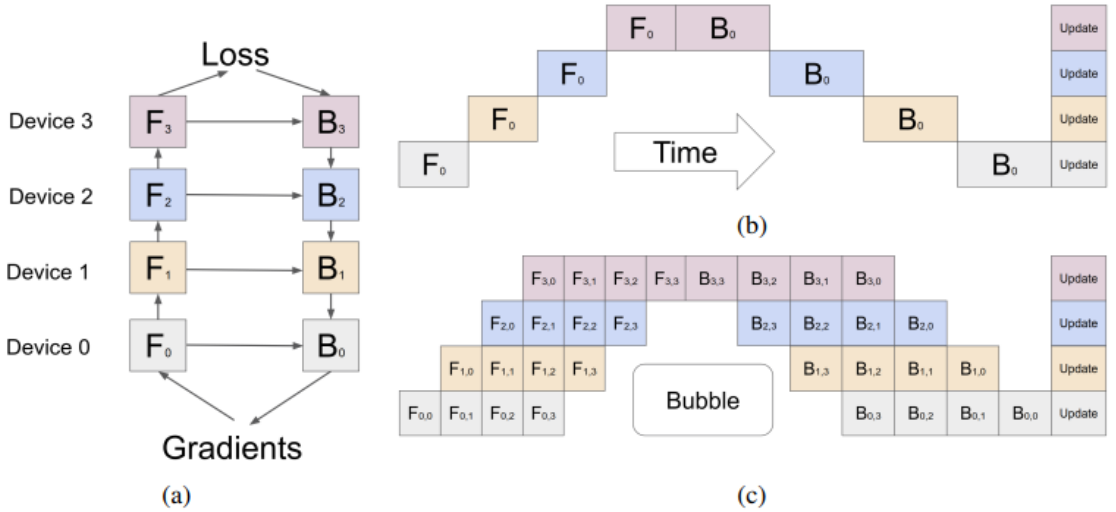
Y=ax：可以在X1A1和X2A2相加之前，先执行y=ax，分别得到aX1A1和aX2A2,再相加

gelu(X1A1)+gelu(X2A2) != gelu(X1A1+X2A2),需要在gelu之前设置一个同步点

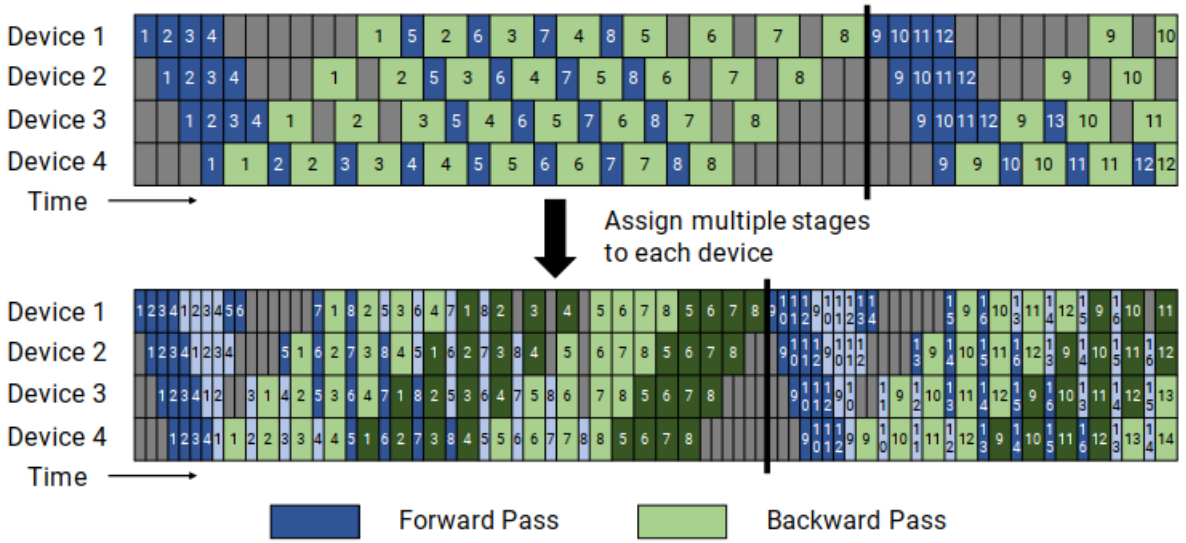
$Y = XA = X[A_1, A_2] = [XA_1, XA_2]$ → XA1和XA2通过最后一个维度拼接的，可以先计算Gelu(XA1)和gelu(XA2)，然后再拼接

Pipeline流水线并行

切分方式：按层切分（流水线并行）
层内切分（模型并行）



进一步减少pipeline的气泡，每个设备执行层的多个子集的计算。



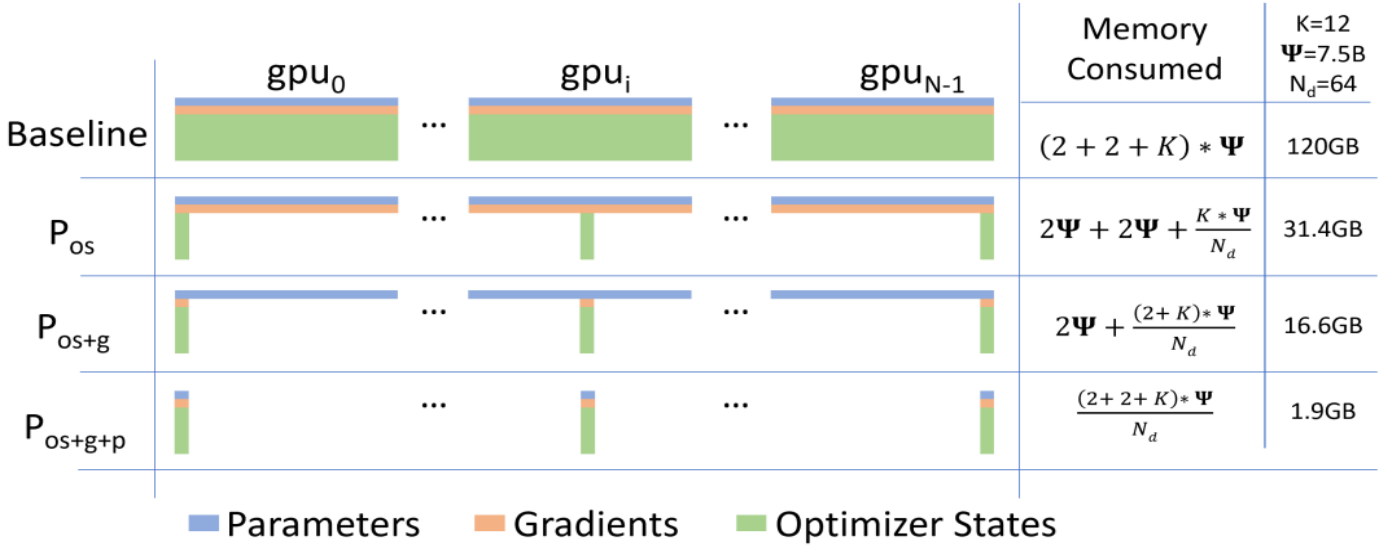
Zero Redundancy Optimizer (Zero)

问题：

- 存储构成：参数、梯度、优化器状态（巨大）
- 现象：每个worker都进行了相同的optimizer运算，存在冗余

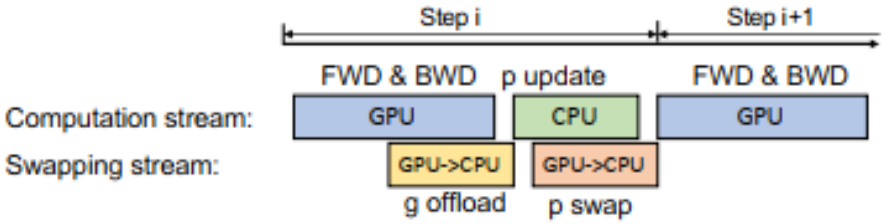
思路：

- 将optimizer state 分布到不同的worker上
- 在每个worker分别运行不同的optimizer计算，并all-gather参数更新

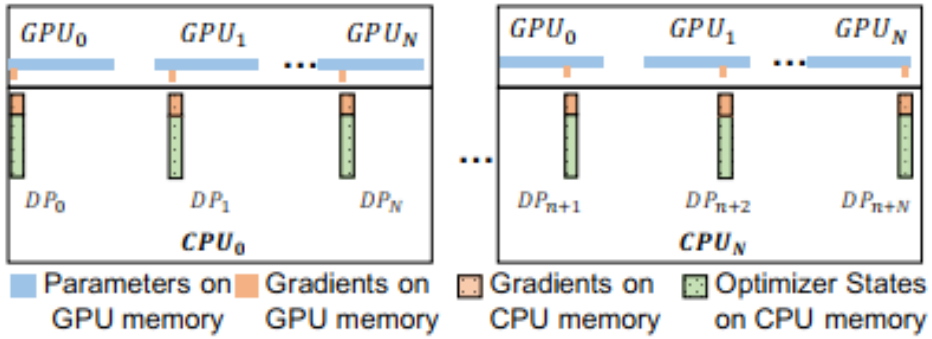


Zero offload

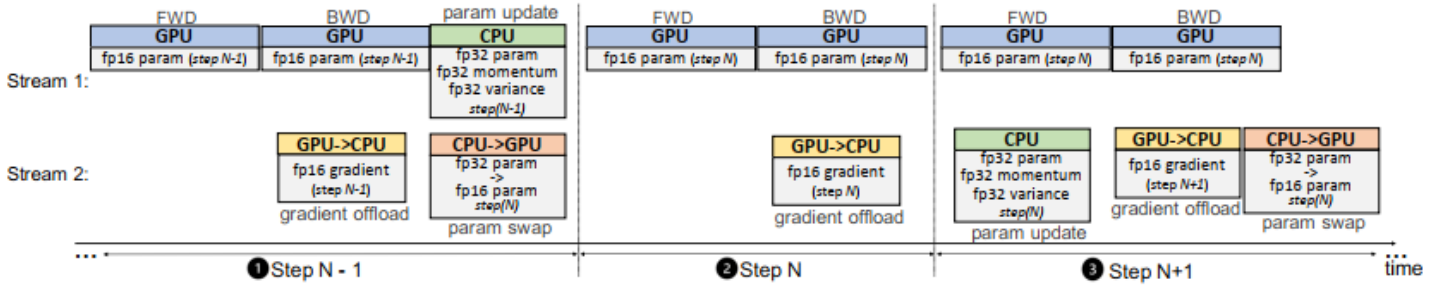
- 将部分数据和计算转移到CPU上



- 将优化器参数同样分散到多个CPU上



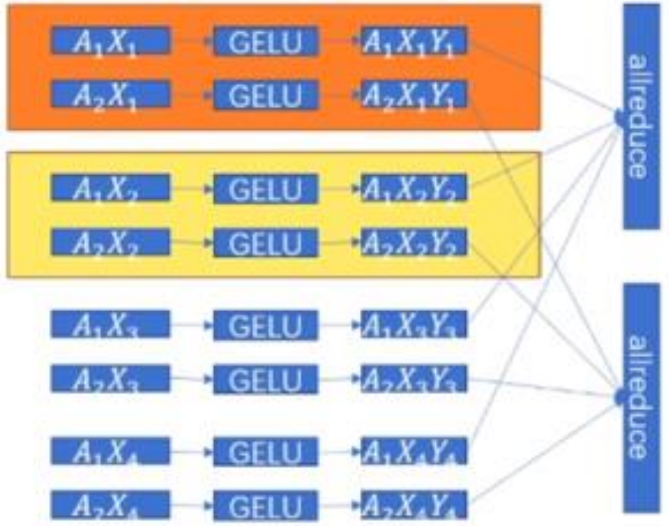
- 将CPU上状态更新与Step N+1的GPU上的计算重叠起来
- 40TFLOPs/GPU on V100 for 10B model



计算与通信重叠&分布式矩阵乘法

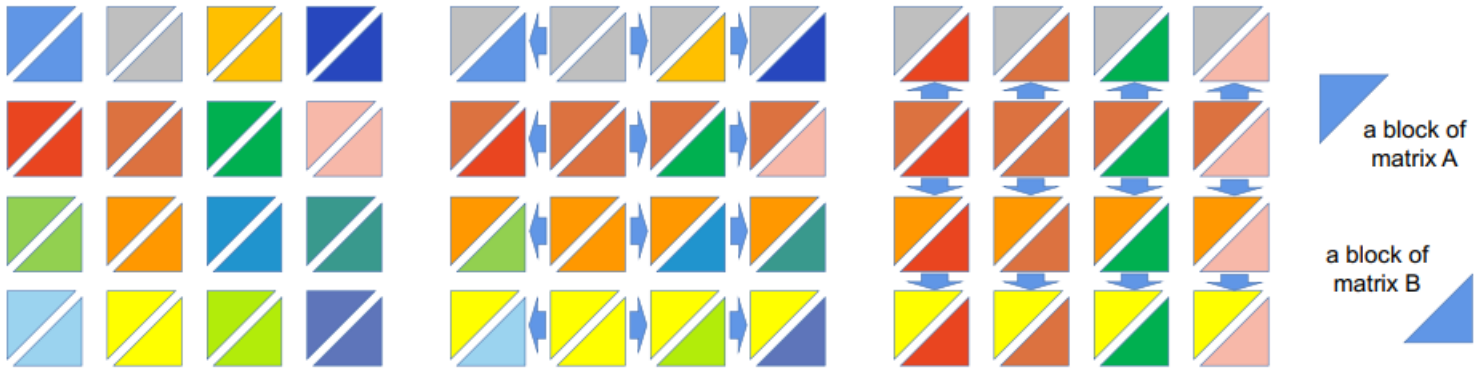
计算与通信重叠：

- 数据传输与计算重叠
- 节点之间模型计算结果传输与计算重叠



分布式矩阵乘法：

- 二维切分
- 通信：$\frac{1}{\sqrt{p}}$，　存储：$\frac{1}{p}$



谢谢大家！