ZOMI 大模型：分布式训练

张量并行

Tensor Parallel

大模型业务全流程

数据 & 模型算法

2. 数据处理

- 开源数据
- 数据预处理
- 向量数据库

3. 模型算法

- LLM模型架构
- 多模态一切皆Tokens

模型训练 & 微调

4. 模型训练

- 混合精度
- 梯度检查
- 梯度累积
- …
- 5. 分布式并行
- 训练集群稳定性

6. 模型微调

- 全参微调
- 低参微调
- 指令微调

模型验证 & 推理部署

7. 模型验证

- 下游任务
- 测评标准
- bench mark

8. 推理与智能体

- 量化压缩
- 推理加速
- 9. Agent 智能体

集群算力准备

1. AI 集群建设

- 计算、存储、网络
- AI 集群机房建设
- AI 集群上线与运维

大模型系列 – 分布式训练加速

- **具体内容**

1. **分布式加速库：**
   - 业界常用分布式加速库 & 作用
2. **DeepSpeed 特性：**
   - 基本概念 - 整体框架 – Zero-1/2/3 – ZeRO-Offload – ZeRO-Infinity
3. **Megatron 特性：**
   1. 总体介绍 – 整体流程 – 并行配置 – DP – TP – PP

Bilibili – ZOMI酱

我的合集和视频列表 › 合集·【AI框架】分布式并行策略

播放全部

合集 | 7个视频 | 2022-12-15更新

随着深度学习中的数据规模和网络规模越来越大，训练神经网络会耗费越来越多的时间，势必需要从单 GPU 训练向多 GPU 训练甚至多机训练进行扩展。比如在大规模人脸识别中，训练上千万人脸 ID 需要对最后的全连接层做模型并行，而 GPT-3 为代表的大...

默认排序 升序排序 编辑

去创作中心添加视频

分布式并行策略基础介绍！【分布式并行】系列第01篇

6424 2022-12-15

PyTorch数据并行怎么实现？DP、DDP、FSDP数据并行原理？【分布

9300 2022-11-1

什么是张量并行？张量并行的数学原理是什么？【分布式并行】系列

5353 2022-11-5

张量还能自动并行？MindSpore张量自动并行啥原理？【大模型与分

2895 2022-11-5

模型并行的流水线并行来啦！了解下GPipe和PipeDream？【分布式

4723 2022-11-5

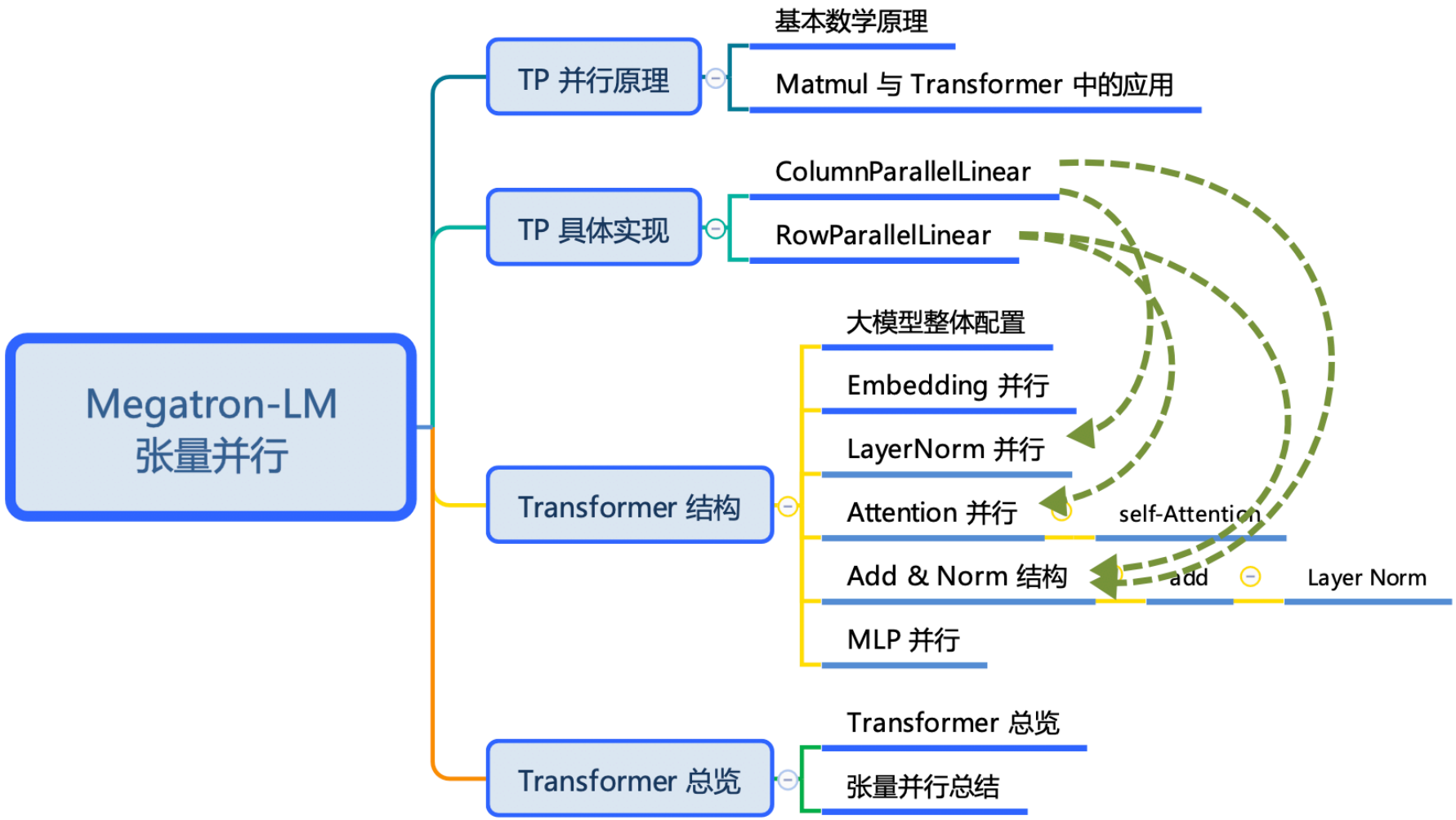
混合并行？多维并行？有多维度混合在一起并行吗？【分布式并行】

2774 2022-11-6

分布式训练总结！【大模型与分布式训练】系列第07篇

4707 2022-11-11

思维导图


Megatron-LM
张量并行
TP 并行原理
基本数学原理
Matmul 与 Transformer 中的应用
TP 具体实现
ColumnParallelLinear
RowParallelLinear
Transformer 结构
大模型整体配置
Embedding 并行
LayerNorm 并行
Attention 并行
self-Attention
Add & Norm 结构
add
Layer Norm
MLP 并行
Transformer 总览
Transformer 总览
张量并行总结

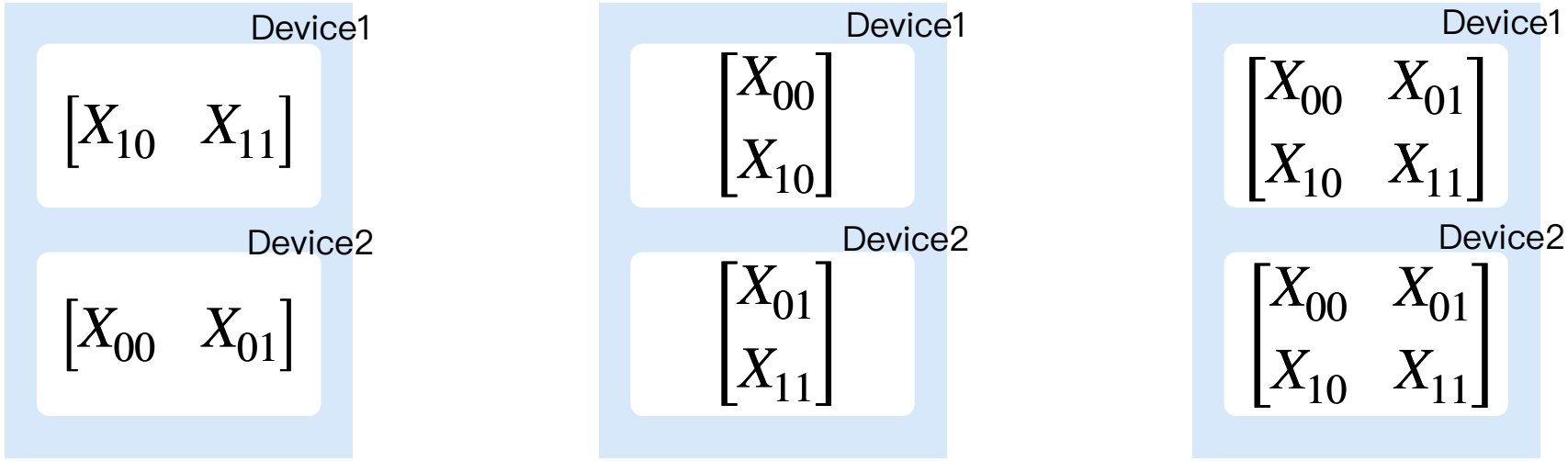
Megatron-LM 01

TP 并行原理

Mathematical Principles 数学原理

- 张量切分方式

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



行切分　　列切分　　复制

Mathematical Principles 数学原理

- 利用分块矩阵计算法则，将 A 分别做矩阵按列切分和按行切分

$$XA = Y$$

A 列切分

$$X \times [A_1 \quad A_2] = [XA_1 \quad XA_2] = Y$$

A 行切分

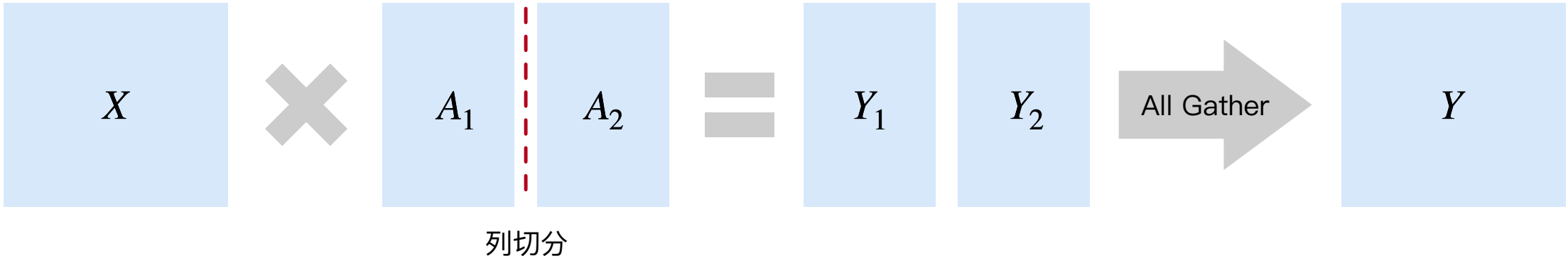
$$[X_1 \quad X_2] \times \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = [X_1A_1 + X_2A_2] = Y$$

MatMul 矩阵乘算子并行

- X作为激活输入， A作为算子权重，将 A 分按列切分

$$XA = Y$$

$$X \times [A_1 \quad A_2] = [XA_1 \quad XA_2] = Y$$

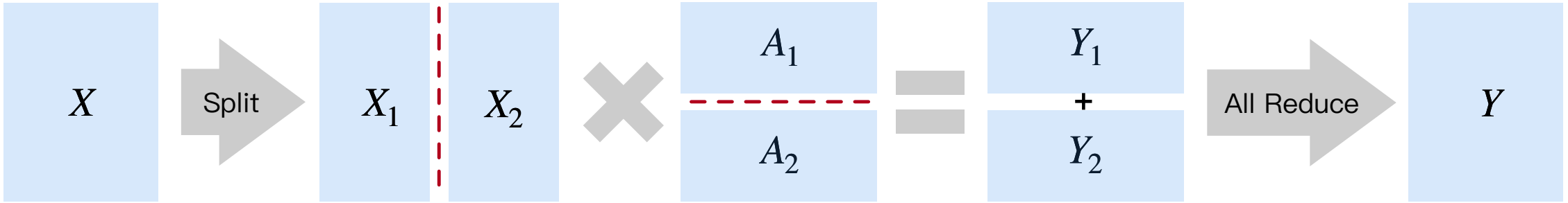


MatMul 矩阵乘算子并行

- X作为激活输入， A作为算子权重，将 A 分按行切分

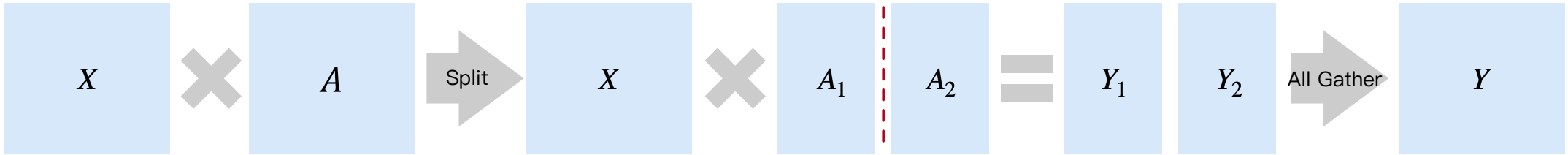
$$XA = Y$$

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

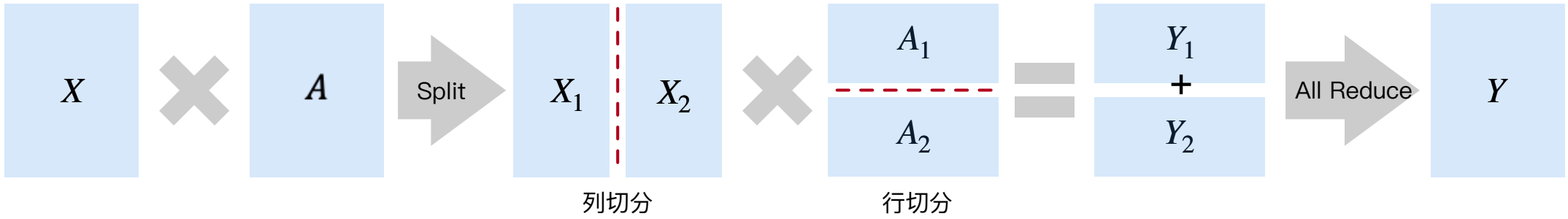


TP 并行原理

1. 把参数 A 按照列 Col 切分 → 把结果按照列拼接起来，与不使用并行数学计算等价结果；



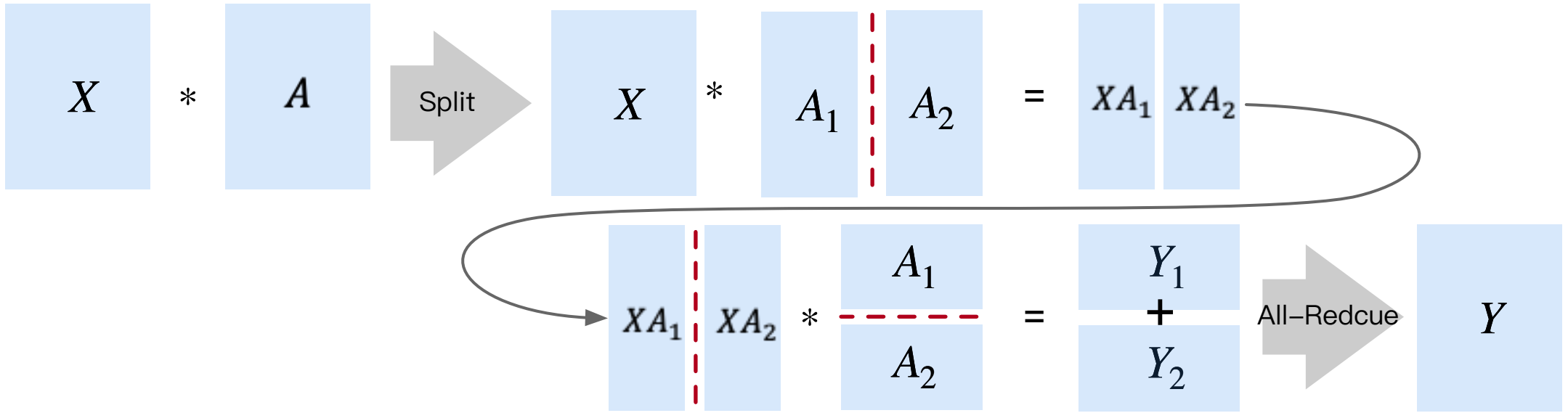
1. 把激活 X 按照列 Col 切分，参数 A 按照行 Row 切分 → 对输出执行加法，得到结果；



1. 每次 Split 引起两次额外通信（前向传播和后向传播各一次）

TP 并行原理

- 如果多次连续矩阵乘，利用数学上的传递性和结合律。可以把数据通信延后到后续计算步骤，即把参数进行列切 All Gather 和 Split 省略掉，从而降低通信量。



Megatron-LM 02

Transformer MLP

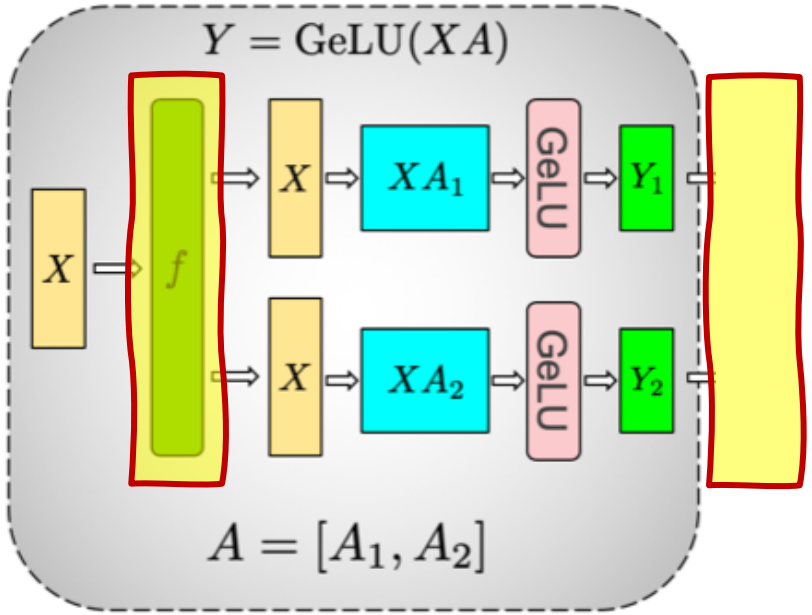
Transformer: MLP 列切分

$$Y = \mathrm{GeLU}(XA)$$

option I

$$\mathrm{A} = [A_1 \quad A_2]$$

$$[Y_1, Y_2] = [\mathrm{GeLU}(XA_1), \mathrm{GeLU}(XA_2)]$$



Copy

All Gather

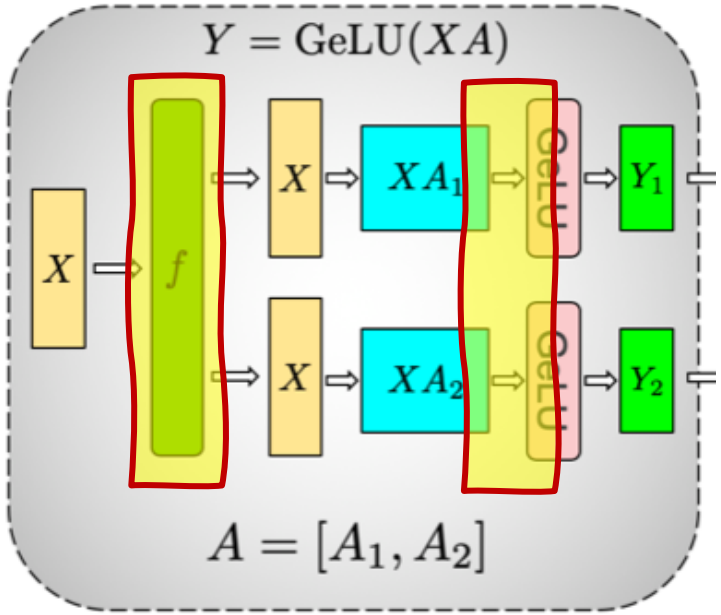
Transformer: MLP 行切分

$$Y = \mathrm{GeLU}(XA)$$

option I

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

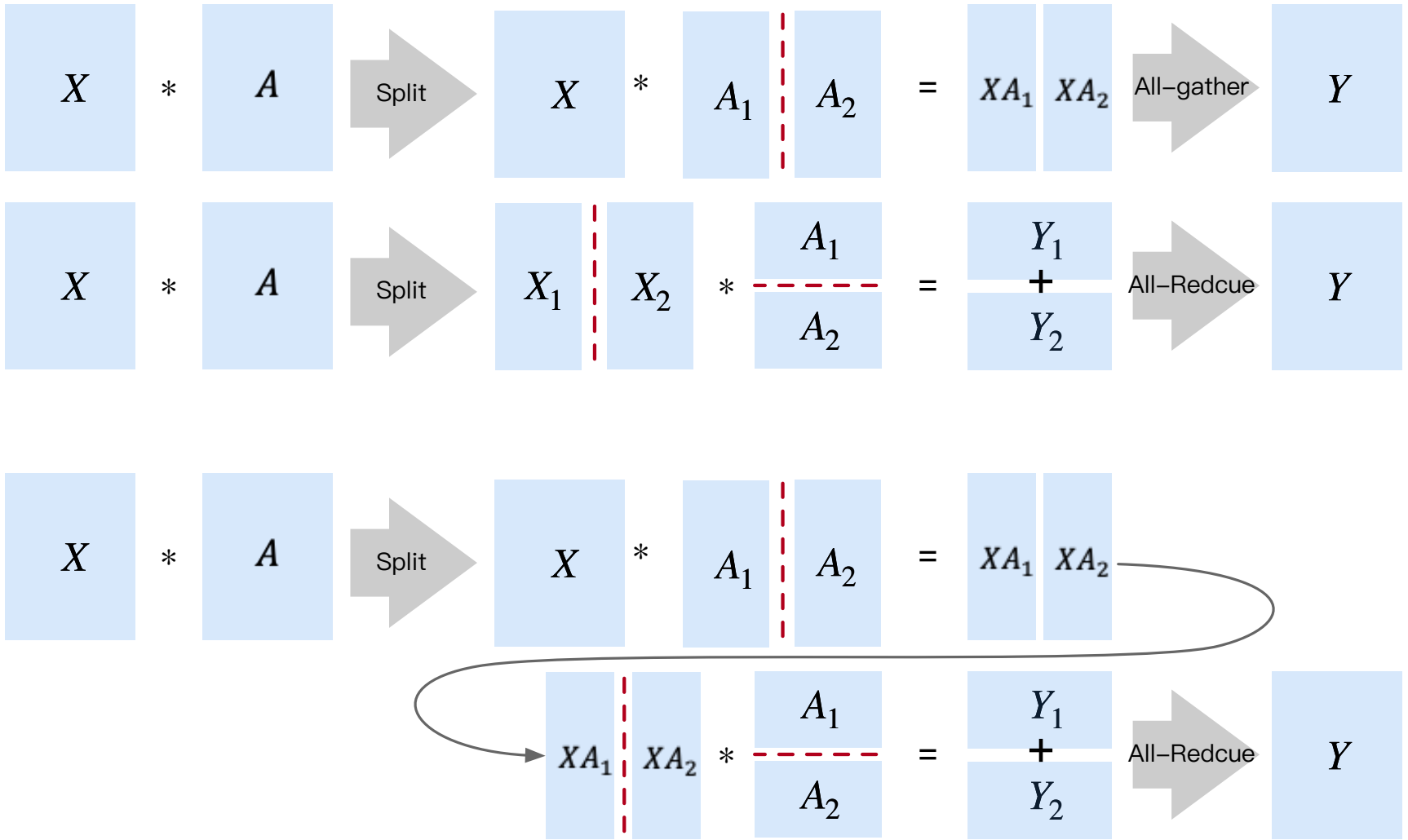
$$Y = \mathrm{GeLU}(X_1A_1 + X_2A_2)$$



Split

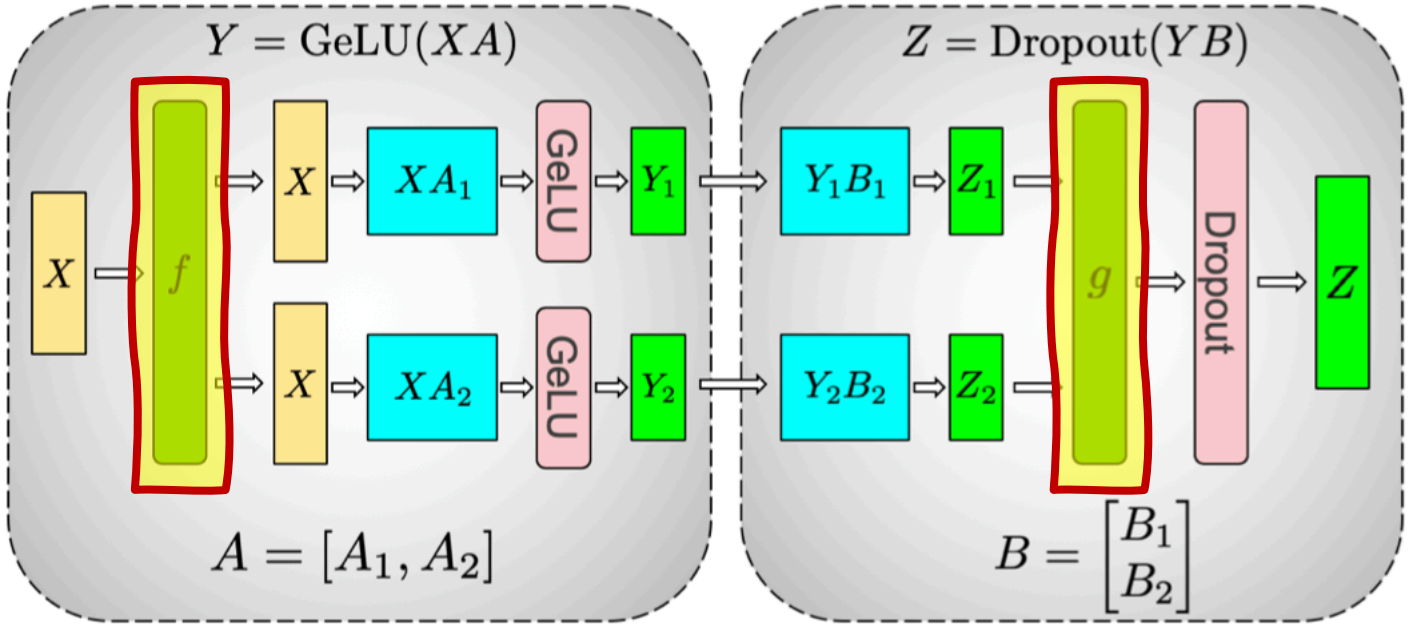
All Reduce

TP 并行原理


X * A
Split
X * A1 A2
= XA1 XA2
All-gather
Y
X * A
Split
X1 X2 * A1 / A2
= Y1 + Y2
All-Redcue
Y
X * A
Split
X * A1 A2
= XA1 XA2
XA1 XA2 * A1 / A2
= Y1 + Y2
All-Redcue
Y

Transformer: MLP

$$Z = \text{Dropout}(\text{GeLU}(XA), B)$$



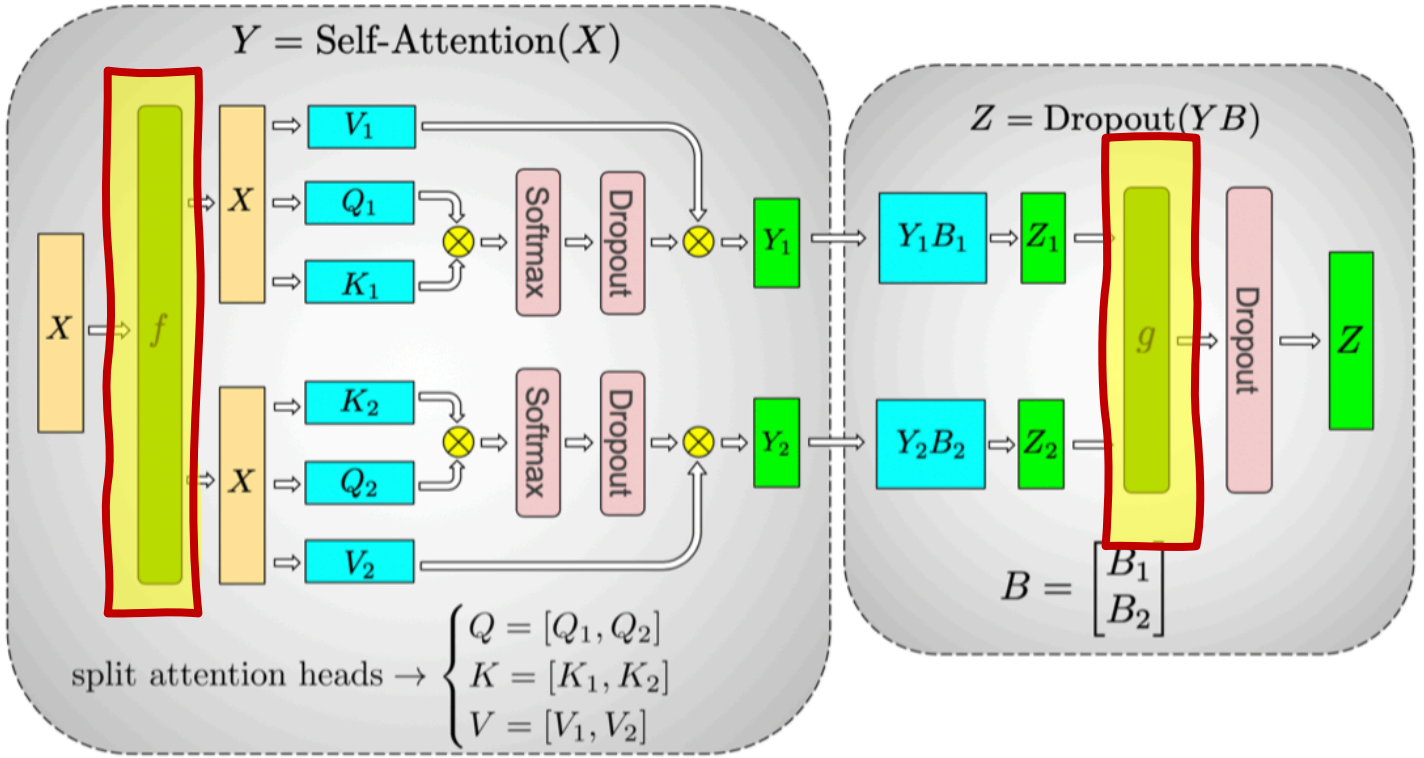
Copy

(a) MLP

All Reduce

Transformer: Self-Attention

$$Z = \text{Dropout}(\text{Self-Attention}(XA), B)$$



Copy

(b) Self-Attention

All Reduce

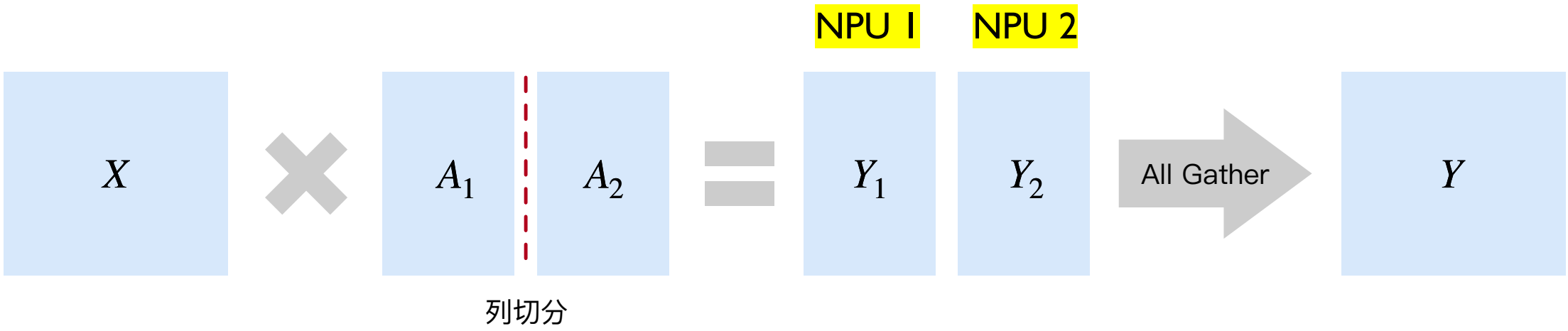
Megatron-LM 03

ColParallelLinear

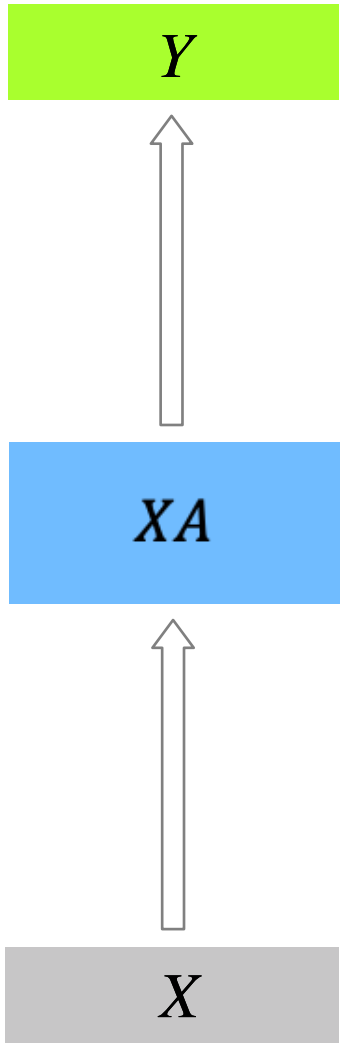
ColumnParallelLinear 列切分

- 是对权重进行列切分：

$$Y = XA = X[A_1, A_2] = [XA_1, XA_2]$$



ColumnParallelLinear 列切分



g

Forward: $Y = [Y_1, Y_2]$

$(\text{all} - \text{gather})$

Backward: $\delta L / \delta Y_1 (\text{split})$

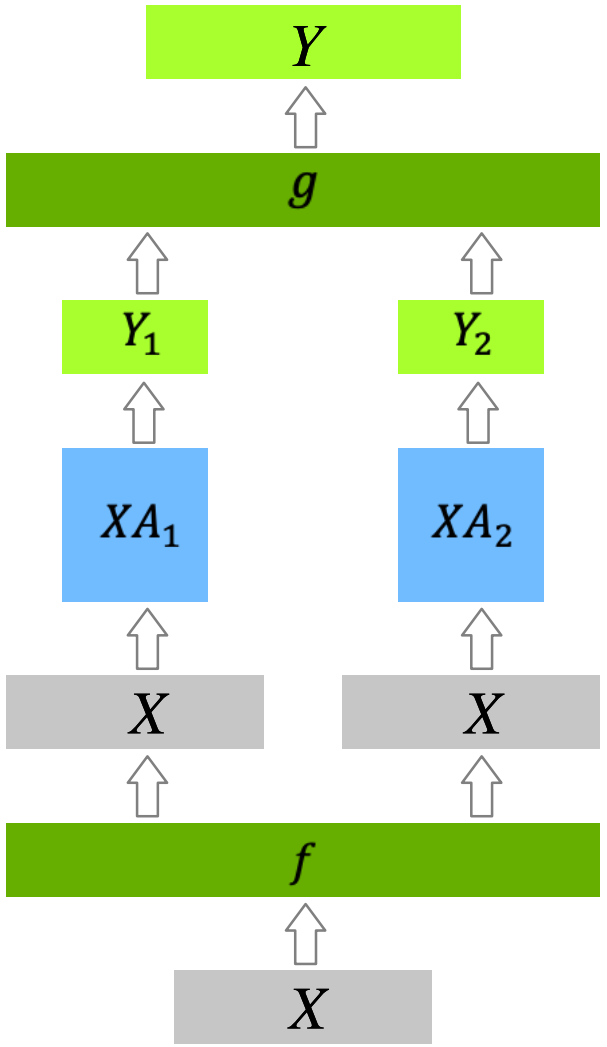
$A = [A_1, A_2]$

f

Forward: $X (\text{identity})$

Backward: $\frac{\delta L}{\delta X} = \frac{\delta L}{\delta X_1} + \frac{\delta L}{\delta X_2}$

$(\text{all} - \text{reduce})$



ColumnParallelLinear 前向传播

$$Y = XA = X[A_1, A_2] = [XA_1, XA_2]$$

- Input：每个 NPU 得到一份完整输入 X，identity 即前行过程中将 X 分发到每个 NPU；
- computer：输出 Y_1, Y_2 按列被切分过，每个 PU 只有对应的一份；
- output：通过 All-Gather 来进行聚合 Y_1, Y_2 ，即使 $Y = [Y_1, Y_2]$。

g

Forward: $Y = [Y_1, Y_2]$ $(\text{all} - \text{gather})$

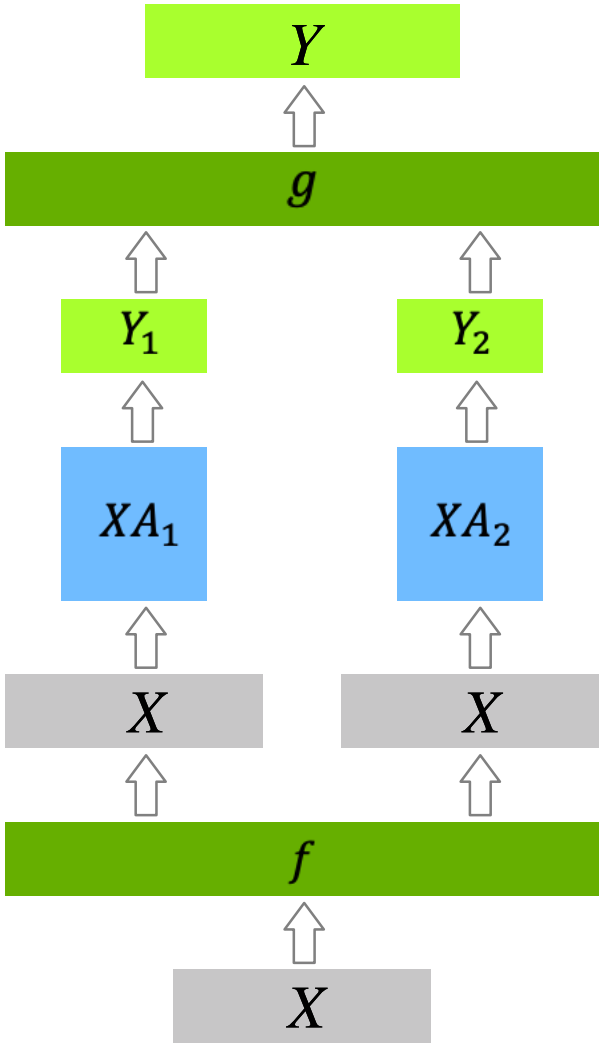
Backward: $\delta L / \delta Y_1 (\text{split})$

$$A = [A_1, A_2]$$

f

Forward: $X(\text{identity})$

Backward: $\frac{\delta L}{\delta X} = \frac{\delta L}{\delta X_1} + \frac{\delta L}{\delta X_2}$ $(\text{all} - \text{reduce})$



ColumnParallelLinear 反向传播

$$Y = XA = X[A_1, A_2] = [XA_1, XA_2]$$

- Input：对上层输入梯度 $\delta L/\delta Y$ 进行切分，保证每个 NPU 有一份梯度$\delta L/\delta Y_i$
- computer：每个 NPU 都进行关于 X 的梯度计算，每个 NPU 得到一份对 X 的梯度；
- output：通 All Reduce 将每个 NPU 关于 X 的梯度进行相加得到完整梯度，即 $\frac{\delta L}{\delta X} = \frac{\delta L}{\delta X_1} + \frac{\delta L}{\delta X_2}$；

g

Forward: $Y = [Y_1, Y_2]$ $(\text{all} - \text{gather})$

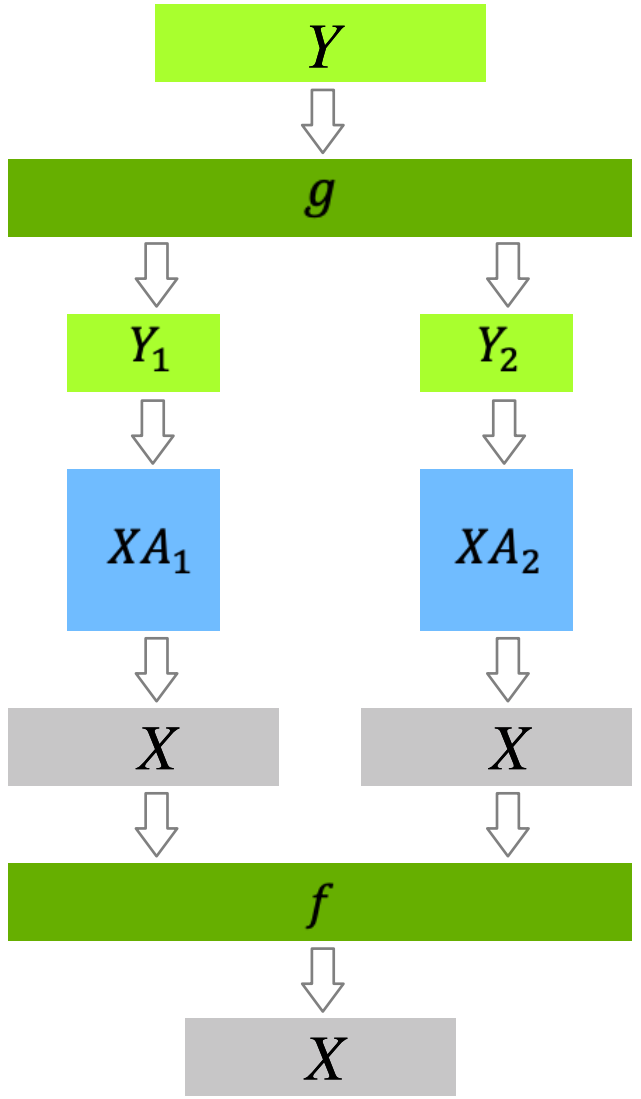
Backward: $\delta L/\delta Y_1 (\text{split})$

$$A = [A_1, A_2]$$

f

Forward: $X(\text{identity})$

Backward: $\frac{\delta L}{\delta X} = \frac{\delta L}{\delta X_1} + \frac{\delta L}{\delta X_2}$ $(\text{all} - \text{reduce})$



f 操作

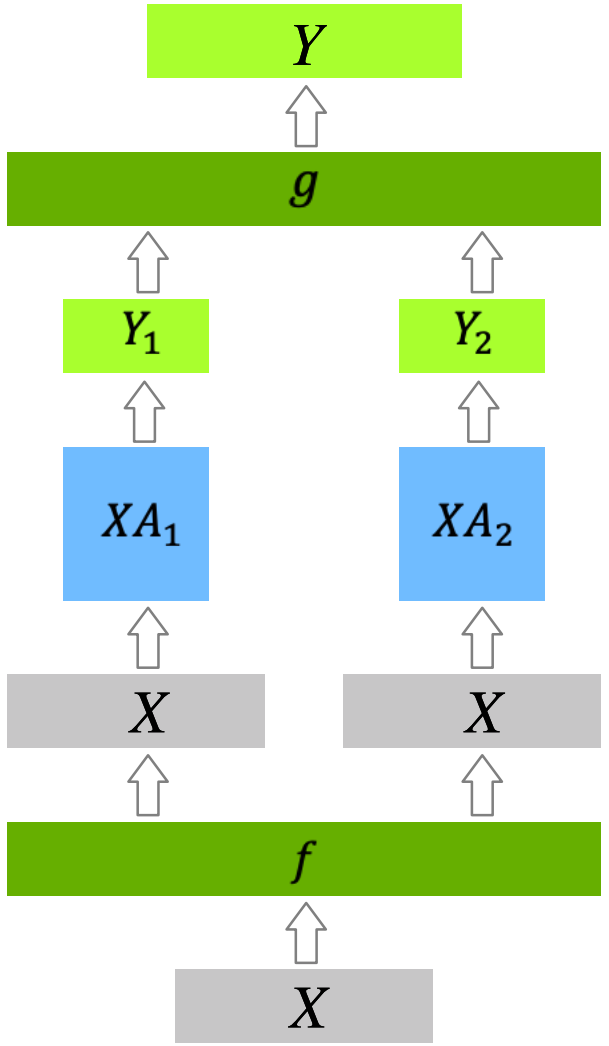
- f 操作对输入激活 X 进行处理：
  - forward 时候每个 NPU 拷贝玩这个的备份 X；
  - backward 时候对每个 NPU 执行 All-Reduce 获取关于 X 的梯度；

f

Forward: $X(\text{identity})$

Backward: $\frac{\delta L}{\delta X} = \frac{\delta L}{\delta X_1} + \frac{\delta L}{\delta X_2}$

$(\text{all} - \text{reduce})$



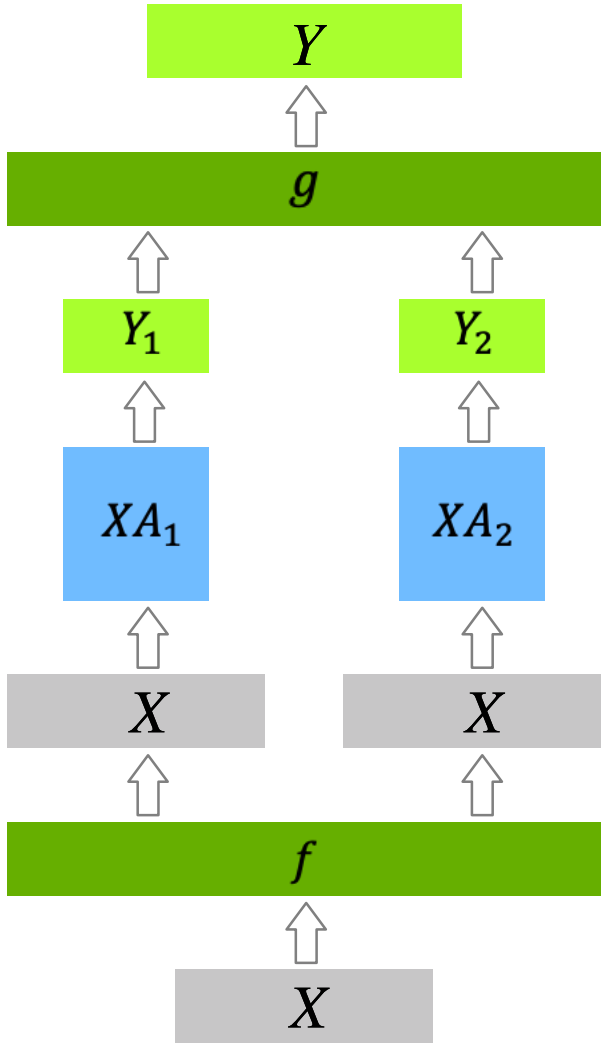
g 操作

- g 操作处理完并行内容后输出 Y：
  - forward 时候执行 all-gather 把每个 NPU 中的 Y 合并；
  - backward 时候执行 split，将梯度 scatter 到不同的 NPU 上；

g

Forward: $Y = [Y_1, Y_2]$ $(\text{all} - \text{gather})$

Backward: $\delta L / \delta Y_1 (\text{split})$



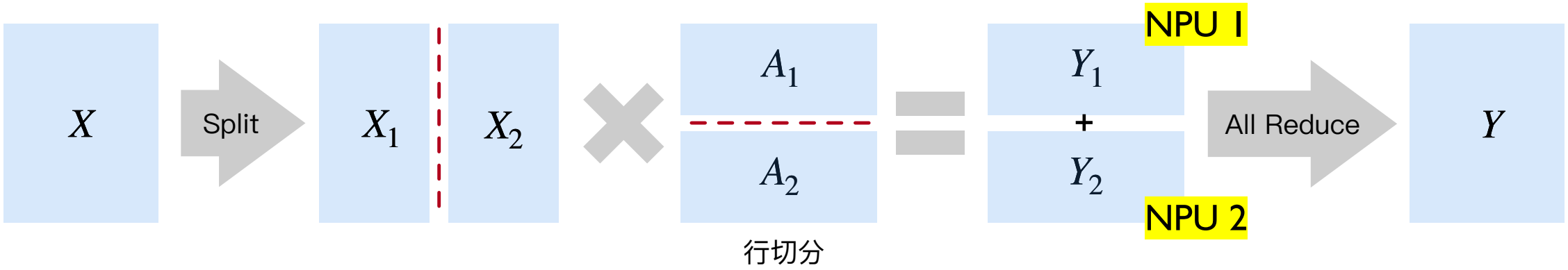
Megatron-LM 04

RowParallelLinear

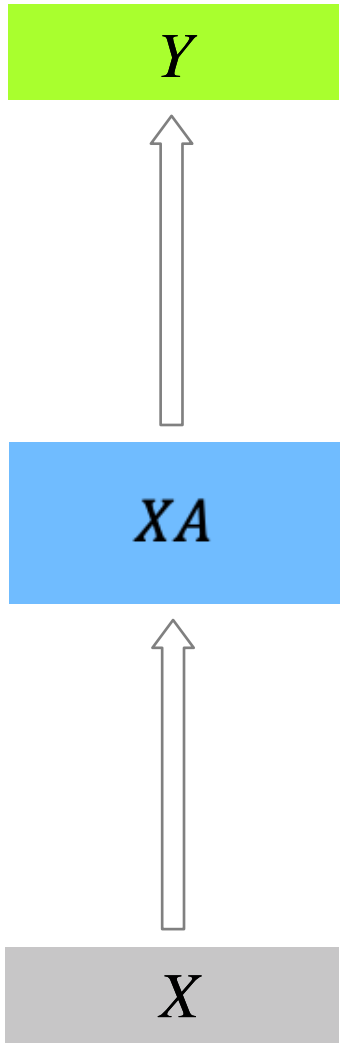
RowParallelLinear 行切分

- 对权重 A 进行行切分，公式为 Y = XA，X 是输入，A 是权重，Y 是输出，行切分针对 A 第一个维度进行切分：

$$XA = [X_1, X_2][A_1 A_2] = X_1 A_1 + X_2 A_2 = Y_1 + Y_2 = Y$$



RowParallelLinear 列切分



g

Forward: $Y = Y_1 + Y_2$

$(\text{all} - \text{reduce})$

Backward: $\delta L / \delta Y_i (\text{identity})$

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

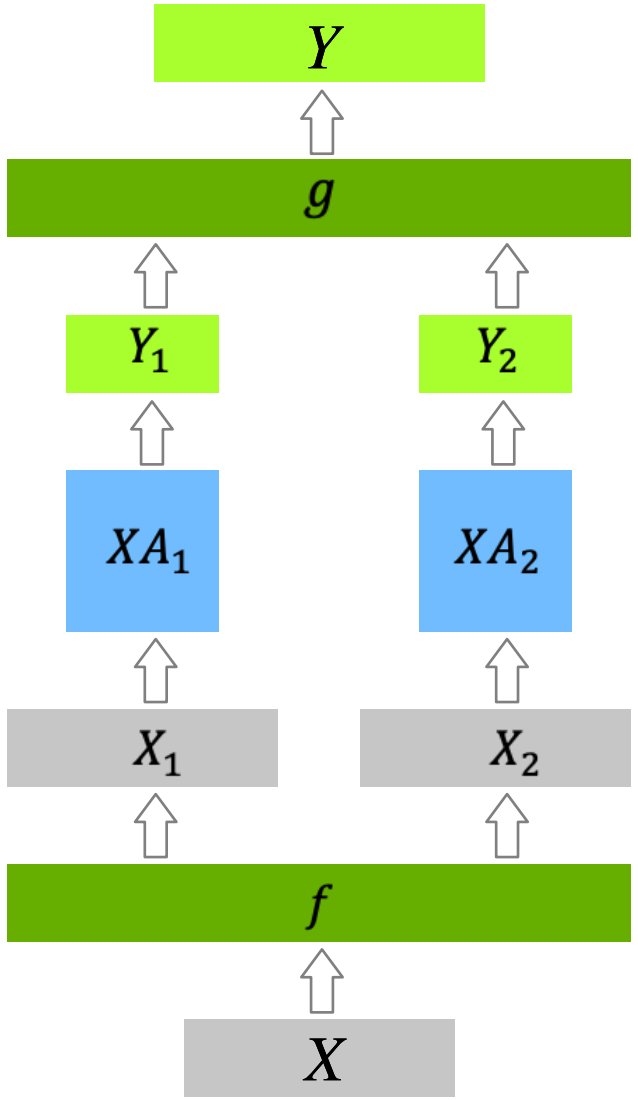
$$X = [X_1, X_2]$$

f

Forward: X_i (split)

Backward: $\frac{\delta L}{\delta X} = \left[\frac{\delta L}{\delta X_1}, \frac{\delta L}{\delta X_2}\right]$

$(\text{all} - \text{gather})$



RowParallelLinear 前向传播

$$Y = XA = [X_1, X_2][A_1 A_2]$$
$$= X_1 A_1 + X_2 A_2$$

- Input：对输入激活 X 进行列切分 Split，每张 NPU 得到独立的一份 X_i；
- computer：计算 $Y_i = X_i A_i$ ，即每个 NPU 有其对应一份数据 Y_i；
- output：由于每个 NPU 的 Y_i Shape 相同，合并前需要通过 All-Reduce 进行通信同步；

g

Forward: $Y = Y_1 + Y_2$ $(\text{all} - \text{reduce})$

Backward: $\delta L / \delta Y_i (\text{identity})$

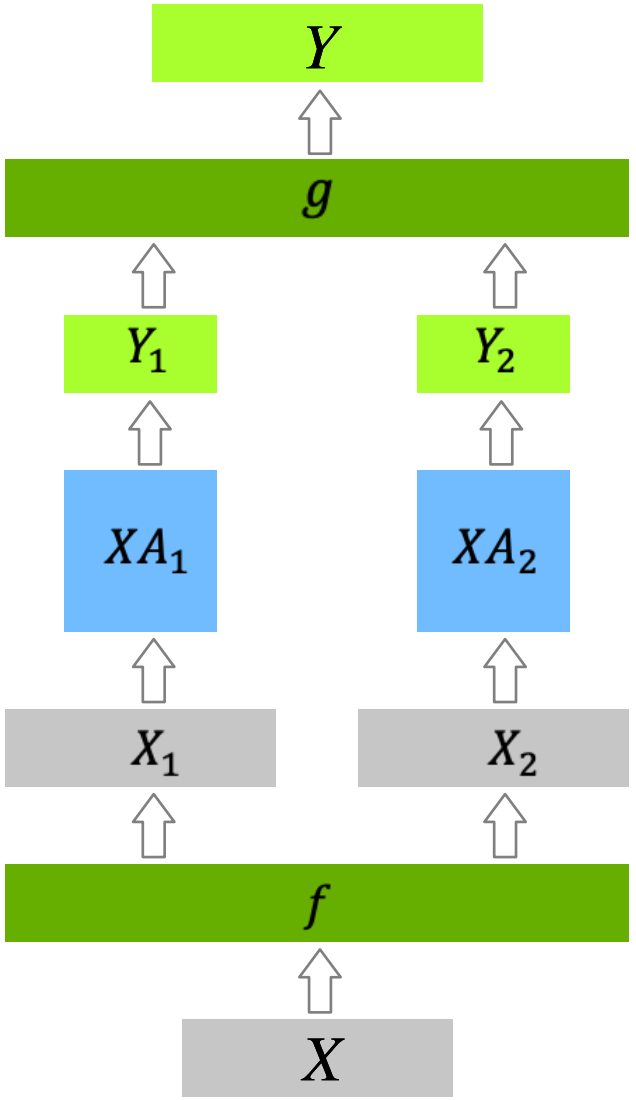
$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$X = [X_1, X_2]$$

f

Forward: X_i (split)

Backward: $\frac{\delta L}{\delta X} = \left[\frac{\delta L}{\delta X_1}, \frac{\delta L}{\delta X_2}\right]$ $(\text{all} - \text{gather})$



RowParallelLinear 反向传播

$$Y = XA = [X_1, X_2][A_1 A_2]$$
$$= X_1 A_1 + X_2 A_2$$

- Input：对上层输入梯度 $\delta L/\delta Y$ 进传播，因为 Y_i Shape 相同，因此需要直接 scatter到每个 NPU
- computer：每个 NPU 都进行关于 X 的梯度计算，每个 NPU 得到一份对 X 的梯度；
- output：通 All Gather 将每个 NPU 关于 X_i 的梯度进行聚合，得到完整的 X；

g

Forward: $Y = Y_1 + Y_2$ $(\text{all} - \text{reduce})$

Backward: $\delta L/\delta Y_i (\text{identity})$

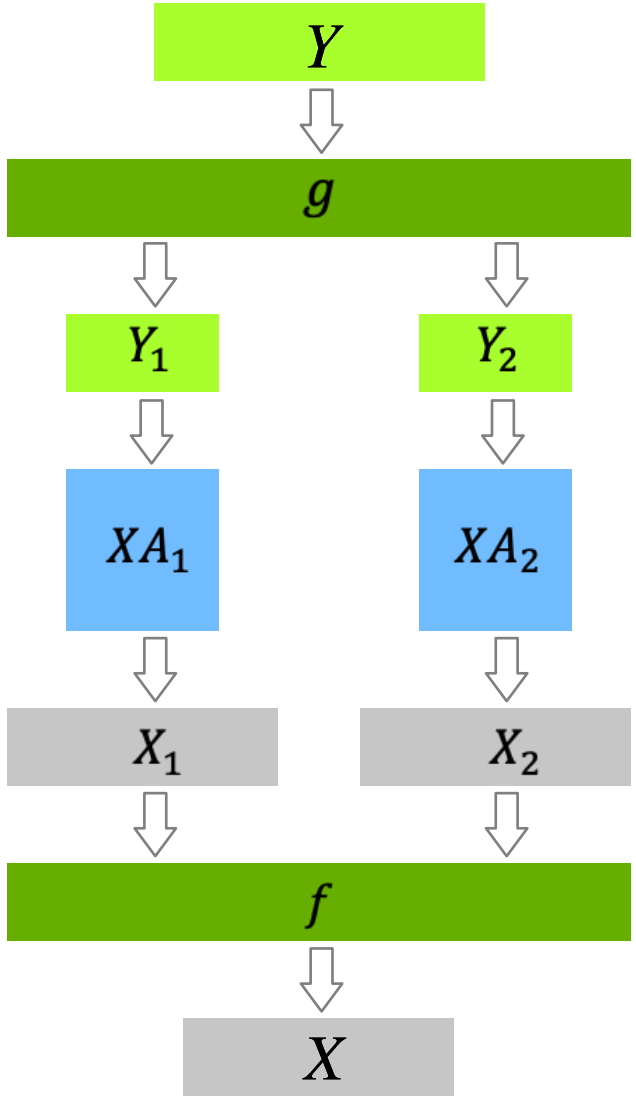
$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$X = [X_1, X_2]$$

f

Forward: X_i (split)

Backward: $\frac{\delta L}{\delta X} = \left[\frac{\delta L}{\delta X_1}, \frac{\delta L}{\delta X_2}\right]$ $(\text{all} - \text{gather})$



f 操作

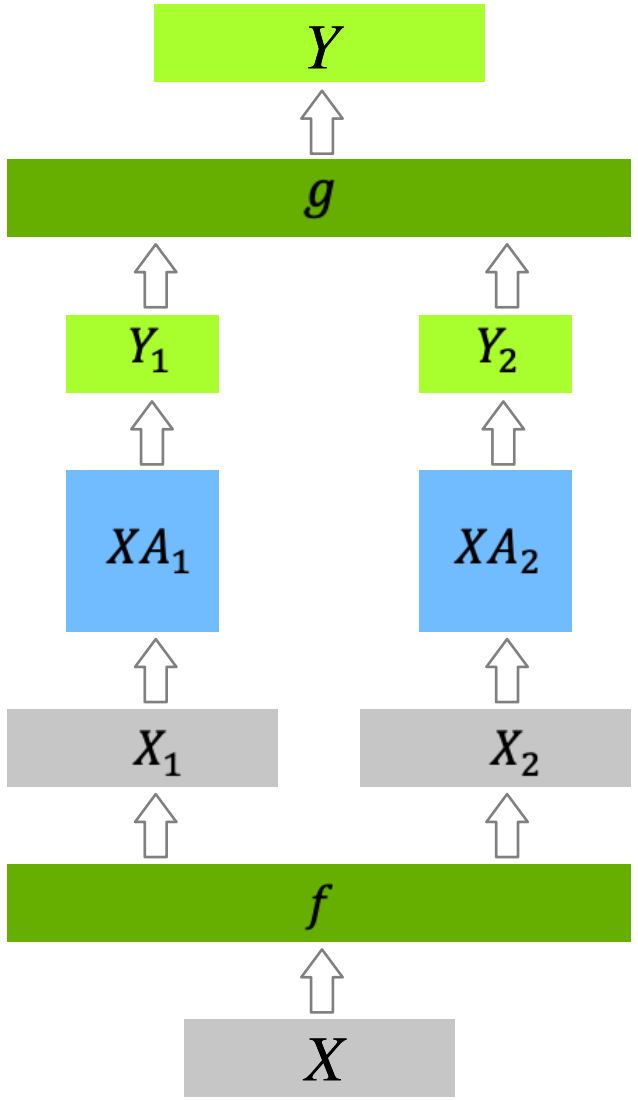
- f 操作通过 scatter_to_tensor_model_parallel_region：
  - forward 时候把 X 切分 X_i 到每个 NPU 上；
  - backward 时候通过 All Gather 将 X_i 聚合成完整的 X；

f

Forward: X_i (split)

Backward: $\frac{\delta L}{\delta X} = \left[\frac{\delta L}{\delta X_1}, \frac{\delta L}{\delta X_2}\right]$

$(\text{all} - \text{gather})$



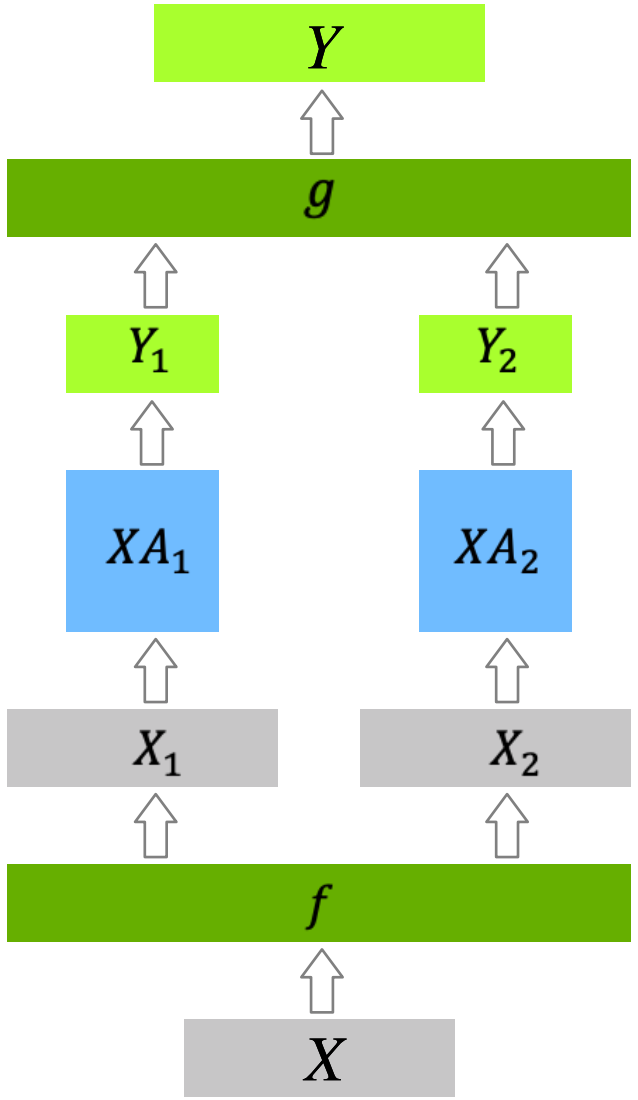
g 操作

- g 操作通过 reduce_from_tensor_model_parallel_region：
  - forward 时候每个 NPU 的 Y_i Shape 相同，通过 All-Reduce 进行同步；
  - backward 事后直接 identity 拷贝；

g

Forward: $Y = Y_1 + Y_2$ $(\text{all} - \text{reduce})$

Backward: $\delta L / \delta Y_i (\text{identity})$



Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem