Training

多 GPU + 并行式计算是大模型训练的标配，这是因为
- 模型参数 + forward/backward 运算所需的内存通常会大于单个 GPU 上所提供的
- 增加运算效率, 降低训练时间

并行式计算的核心：计算 (compute), 通信 (commute), 读写 (read/write).

大模型的训练是 计算 (compute-bound) 和 带宽 (memory-bound) 之间的 tradeoff.

并行计算框架: DeepSpeed, Megatron-LM.

I. Prerequisites

collective communication

目标：通过调用 NCCL, 实现并行计算时 GPU 之间的高速数据传输 (commute).
1. 同节点 (node) 内 GPU 的通过 NVLink 传输
2. 跨节点的 GPU 之间通过 InfiniBand 传输

基本通信方式: AllReduce, Reduce, AllGather, Broadcast, ReduceScatter.

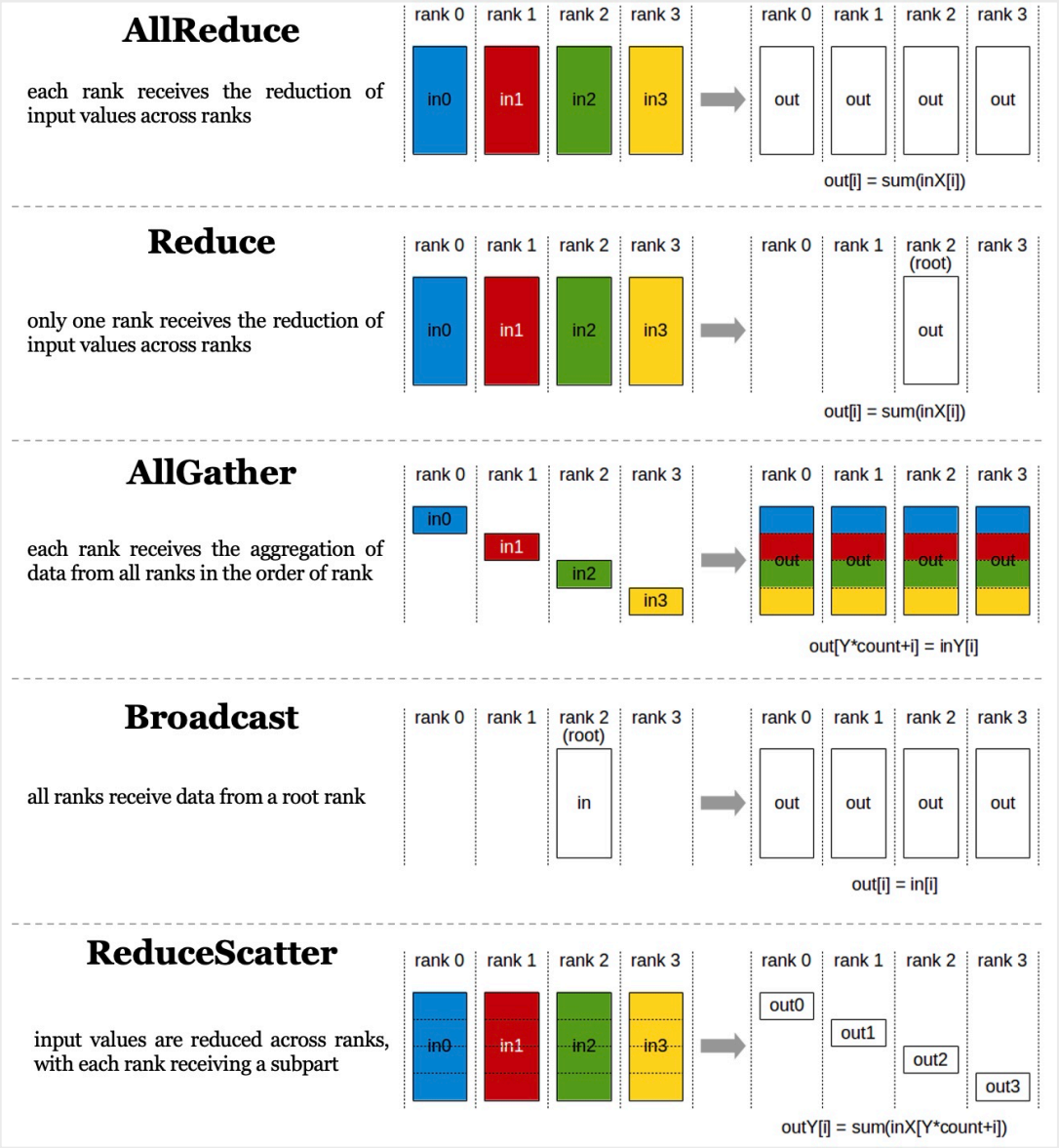

Figure 1: collective communication primitives

II. Parallelism

目标：

1. 将训练数据 (batch) 或模型模块 (tensor) 切分到不同的GPU上运算
2. 通过 collective communication 组合将运算结果汇总

notations

- (batch) data : X # dim: (B x T) x C
- (layer) model : A, B # dim: C x C

X, A, B均可以抽象为二维矩阵.

causal self-attention layers: A -> self-attention, B -> MLP

position-wise feed-forward layers: A -> MLP, B -> MLP

a. 基础并行计算方式

- (fully sharded) data parallelism (DP)
  - naive: 将 X row-wise 切分到不同的GPU，A, B 不切分 (copy to each GPU).
  - fully sharded: X row-wise 切分，同时对 A, B 以及gradient, optimizer state 切分.

- tensor parallelism (TP):
  - A column-wise 切分, B row-wise 切分, 通过AllReduce 收集计算结果.

- pipeline parallelism (PP)
  - A, B 分别 copy 到不同的GPU, 顺序计算.

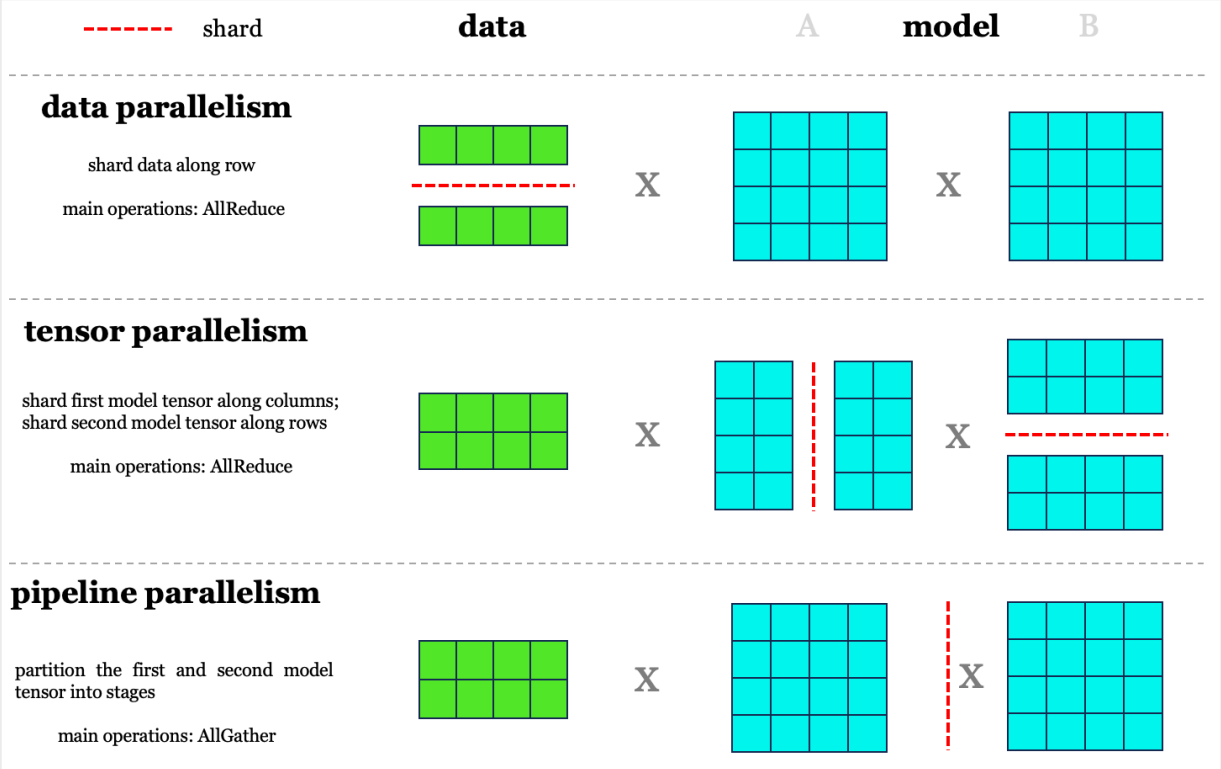

Figure 2: basic parallelisms

b. 进阶并行计算方式

- mixed
  - 节点内TP, 节点间PP.

- sequence parallelism
  - X column-wise (token-wise) 拆分, 用于 (layer) normalization.

- expert parallelism (EP)
  - 在 MoE 架构中, causal self-attention 中的 MLP -> expert, copy to each GPU.

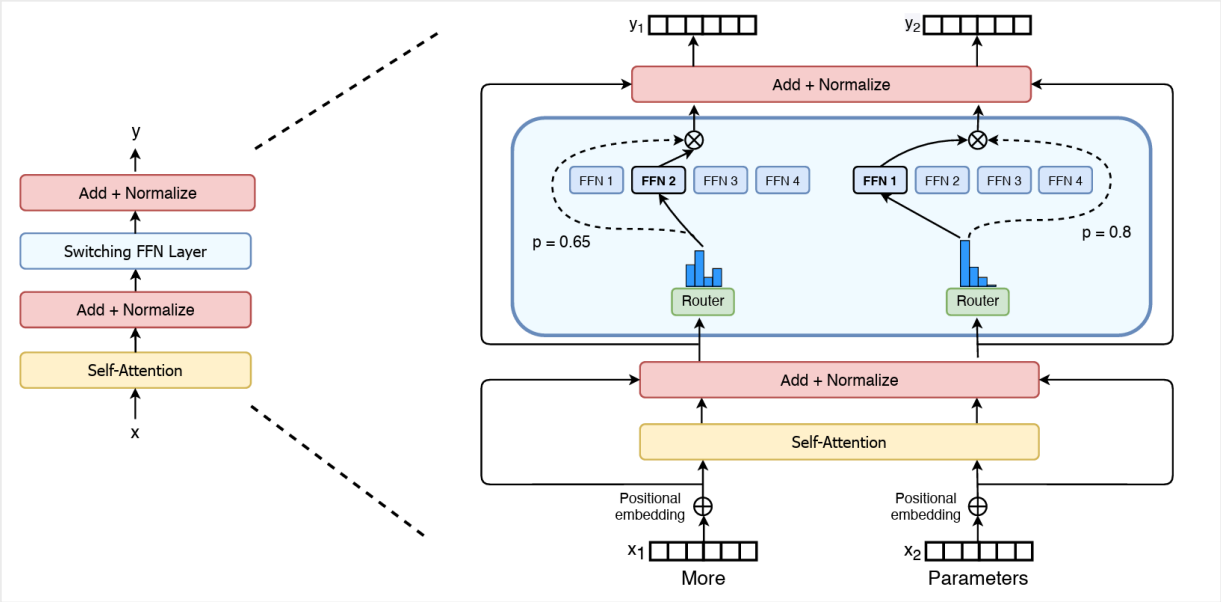

Figure 3: Mixture of Expert (from switch transformer)

III. Training Tasks

notations

token id: [X] # (dim: T x 1)
logits: [X_log] # (dim: T x V)

a. pre-train

forward: [X] -> model -> [X_log]
loss function: cross entropy loss on [1:T] tokens

b. supervised fine-tune (SFT)

forward: [X, Y] -> model -> [X_log, Y_log]
loss function: cross entropy loss on the Y tokens

c. RL with verifiable reward (RLVR)

forward: [X] -> model -> [Y_log] -> [top_1 (Y_log)] # (tokens are generated 1 by 1)
loss function: r * [top_1 (Y_log)], where r = R(top_1 (Y_log), Y_true) # R -> reward fn

* Reference

parallelism

- Stanford cs336, Lecture 7 & 8
- https://zhuanlan.zhihu.com/p/623746805
- https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html
- MoE：月球大叔, EP07 (YouTube)

code

- pre-train: lit-llama
- fine-tune & RLVR: VeRL