SUSTCSC竞赛培训-4

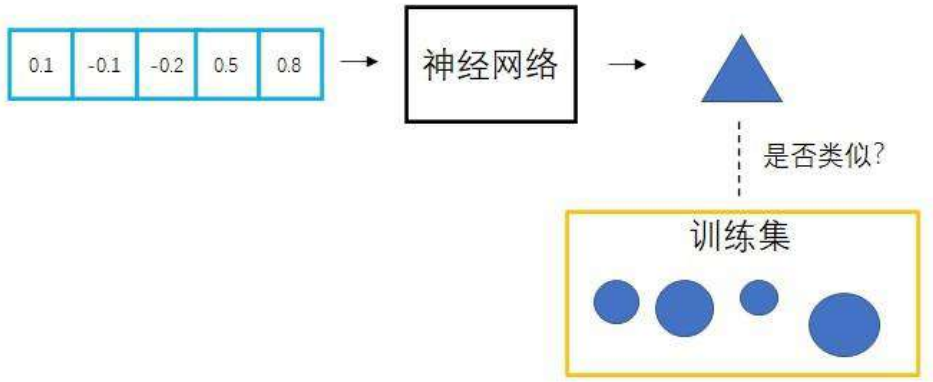
DiT图像生成挑战

肖翊成

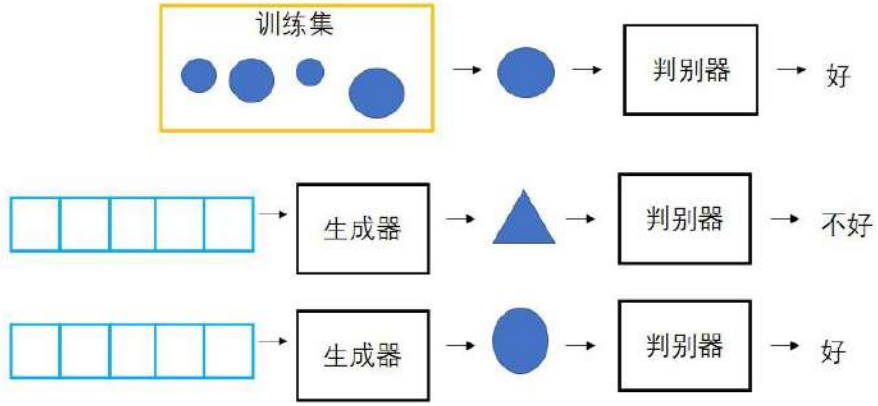
目录

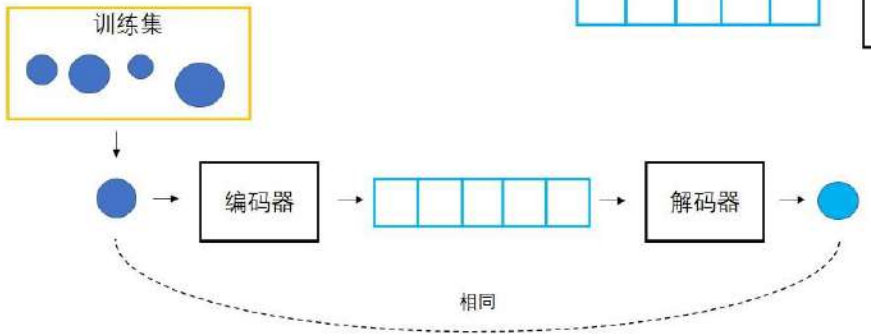
- Diffusion模型
- DiT（Diffusion Transformer）
- 并行推理加速
- 任务要求

图像生成模型



- 神经网络图像生成
  - 学习如何把一个向量映射成一张图片
  - 只有同类型图片，却没有「指导」，高速模型“怎么”生成更好的
- GAN神经网络图像生成
  - 生成器 + 判别器
- VAE变分自编码器
  - 把图像编码成向量，再利用神经网络
  - 编码器 + 解码器
  - 解码器负责图像生成





扩散模型

- 扩散模型是一类近年在图像生成任务中表现非常出色的生成模型
- 一种特殊的VAE
- 其核心思想是通过「正向加噪」（编码）和「反向去噪」（解码）来构建出复杂的额样本分布
  - 正向过程 $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$
  - 反向过程 $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$
  - 采样过程 $x_T \to x_{T-1} \to \cdots \to x_0$

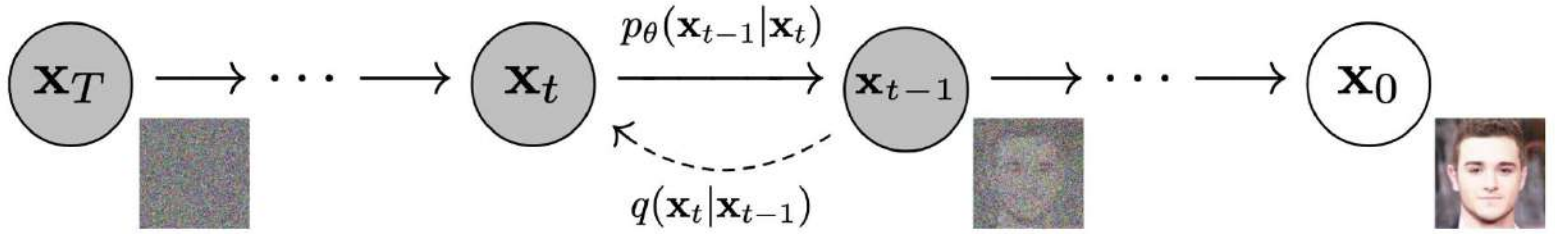


Figure 2: The directed graphical model considered in this work.

Ho, J., Jain, A., & Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models* (No. arXiv:2006.11239). arXiv. http://arxiv.org/abs/2006.11239

扩散模型

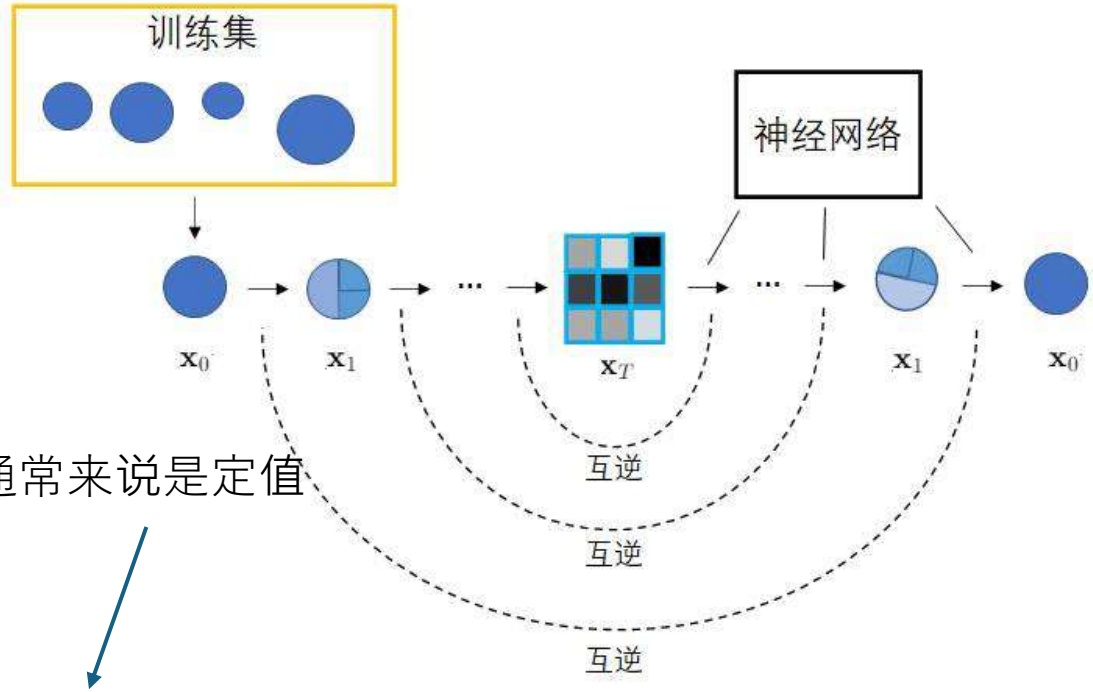
Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

- 扩散模型的灵感来自热力学
  - 一个分布可以通过不断地添加噪声变成另一个分布
- **编码过程**固定成添加噪声的过程
- **解码器**是一个可学习的神经网络
- 训练解码器，使其是**正向过程的逆过程**



$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

通常来说是定值

$$\mathbf{x}_{t-1} \sim \mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t \mathbf{I}) \qquad \tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t) \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

损失函数： $L = ||\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)||^2$

只需要拟合均值即可

DiT(Diffusion Transformer)

- 在DiT以前，Diffusion模型主要使用U-Net结构作为主干网络
- U-Net包含一个下采样路径和一个对应的上采样路径
  - 捕捉上下文信息
  - 有效地传递特征
  - 改善梯度传递的速度
- 最初的扩散模型都是针对图像的像素空间进行操作，像素空间具有非常强的空间局部性特征，卷积网络天然针对该任务有天然的适配性（卷积核）

Peebles, W., & Xie, S. (2023). Scalable Diffusion Models with Transformers (No. arXiv:2212.09748). arXiv. https://doi.org/10.48550/arXiv.2212.09748

DiT(Diffusion Transformer)

- DiT做了以下的改进：
  - Patchify 输入（ViT）
  - 位置编码与时间步嵌入
  - 基于潜空间的建模
    （低维 Latent Space）
- 本质在解决：
  - 减少Token数量保证运算效率
  - 需要让去噪过程和Step相关
  - 保留空间相关性

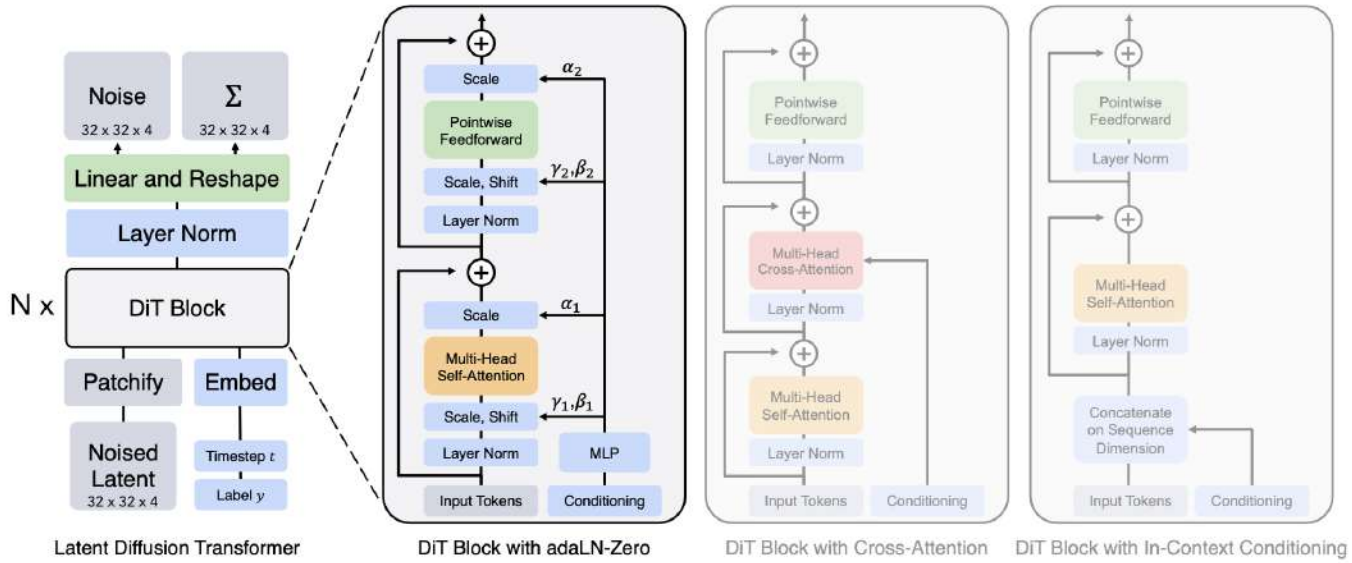


Figure 3. **The Diffusion Transformer (DiT) architecture.** *Left:* We train conditional latent DiT models. The input latent is decomposed into patches and processed by several DiT blocks. *Right:* Details of our DiT blocks. We experiment with variants of standard transformer blocks that incorporate conditioning via adaptive layer norm, cross-attention and extra input tokens. Adaptive layer norm works best.

Peebles, W., & Xie, S. (2023). Scalable Diffusion Models with Transformers (No. arXiv:2212.09748). arXiv. https://doi.org/10.48550/arXiv.2212.09748

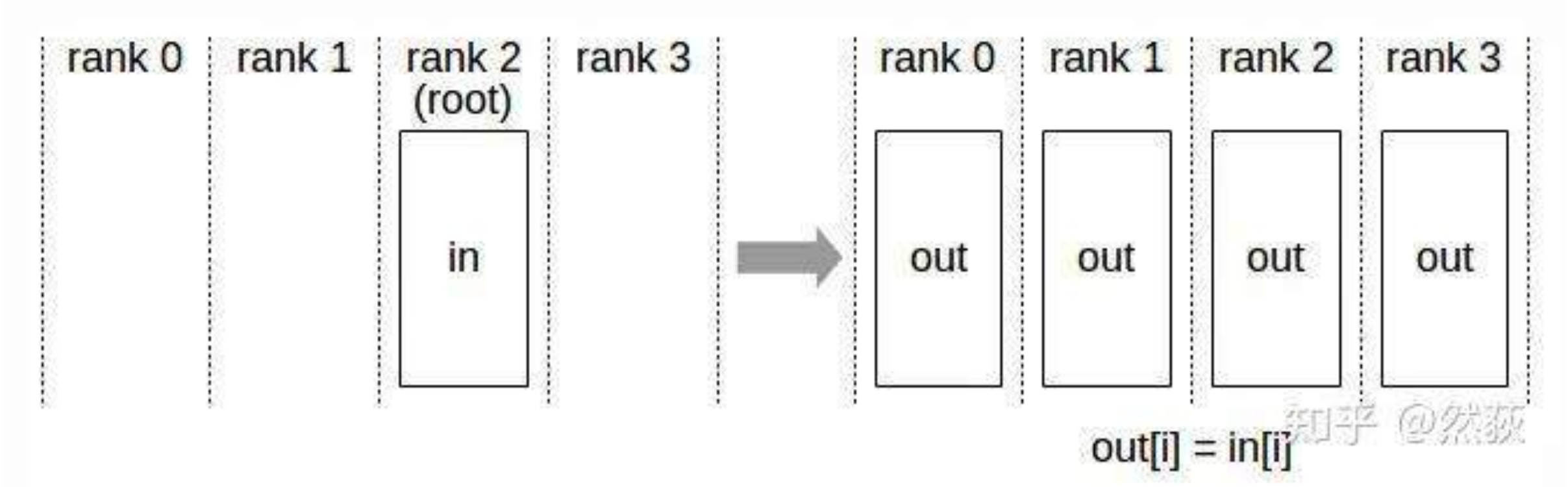
并行推理加速

- NCCL NVIDIA Collective Communications Library
- 后端类型
  - PCIe 同一主机内GPU通过PCIe总线直接通信
  - NVLink 通过NVLink高速互连总线通信
  - Ethernet/IB 通过以太网/InfiniBand网络跨节点通信 （单机无关）
- 集合通信类型（Collective Communication）

	Broadcast	Gather	Scatter	Reduce	AllGather	AllReduce	Reduce-Scatter
对象	一对多	多对一	一对多	多对一	多对多	多对多	多对多
描述	整的发	合一起	反Gather	加一起	Gather + Broadcast	Reduce + Broadcast	Reduce +Scatter
应用场景	加载模型, 模型初始化	收集数据	模型Scatter到不同卡上	多节点运算	模型并行里前向计算里的参数全同步	数据并行	模型并行里在前向allgather后的反向计算

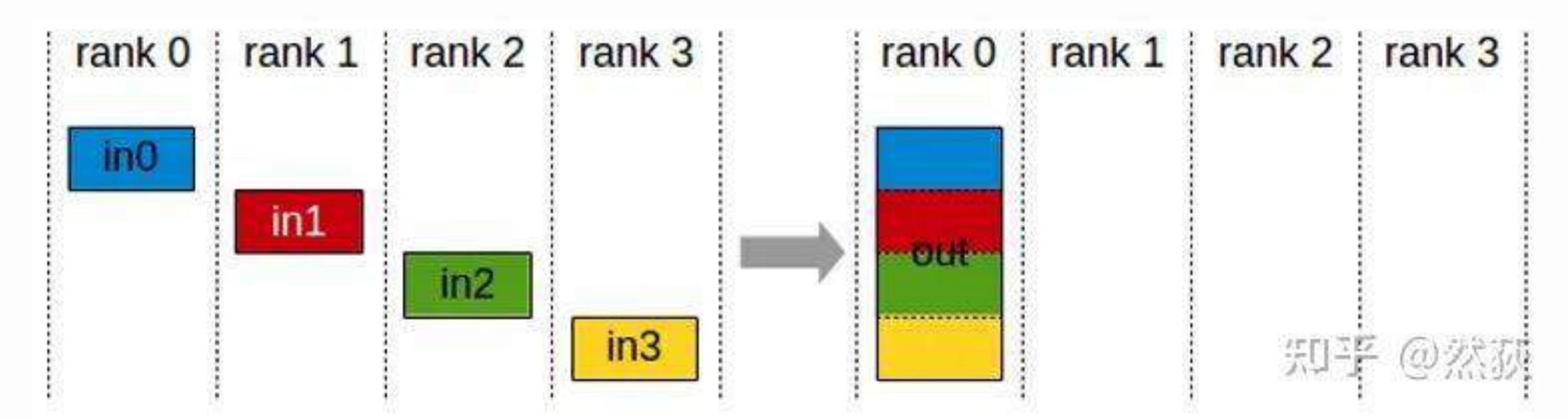
BroadCast

一对多



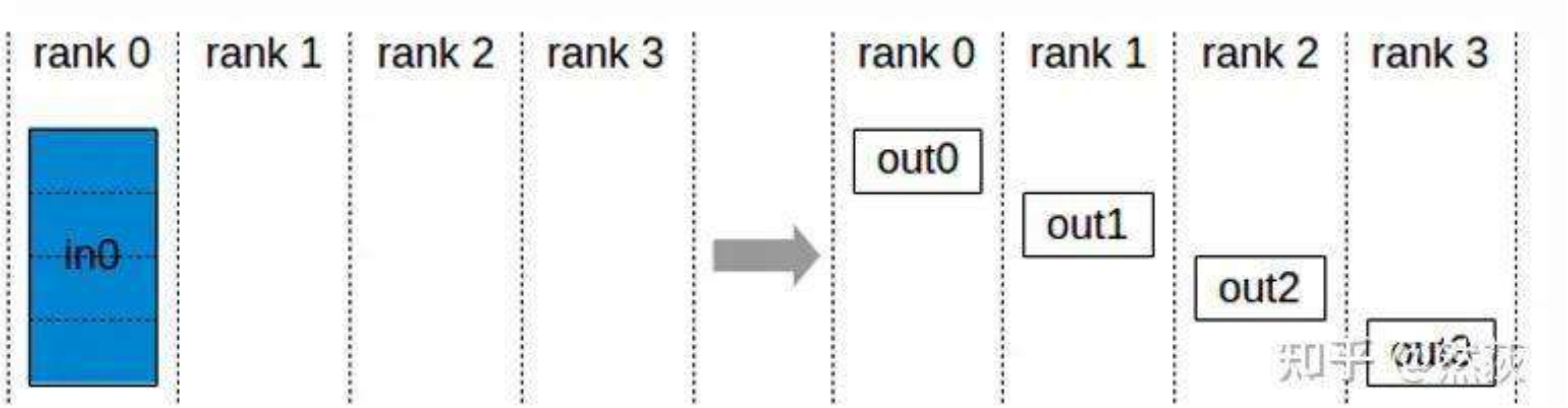
Gather

多对一


rank 0
rank 1
rank 2
rank 3
in0
in1
in2
in3
rank 0
rank 1
rank 2
rank 3
out
知乎 @然获

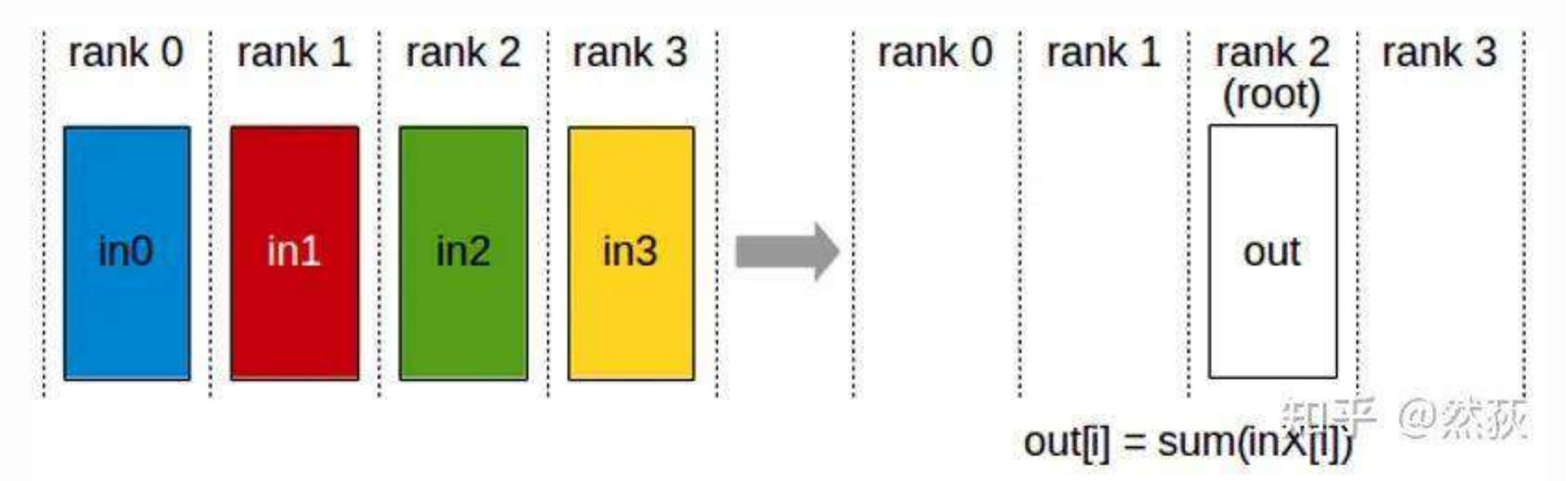
Scatter

一对多 Gather反向过程



Reduce

多对一

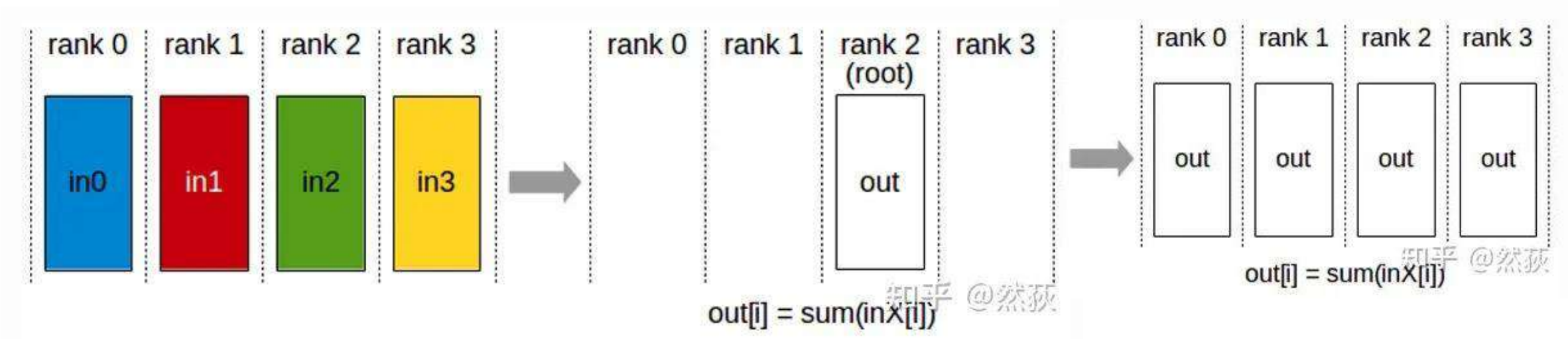

rank 0
rank 1
rank 2
rank 3
in0
in1
in2
in3
rank 0
rank 1
rank 2
(root)
rank 3
out
out[i] = sum(in[i])
知乎 @焚荻

AllGather

多对多 Gather + BroadCast

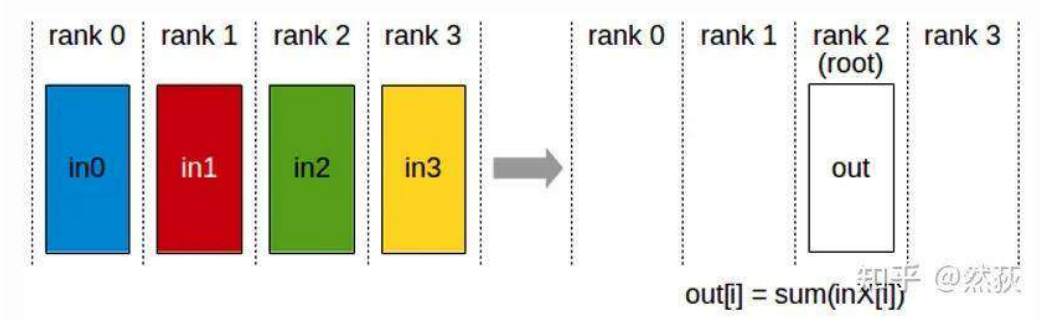
AllReduce

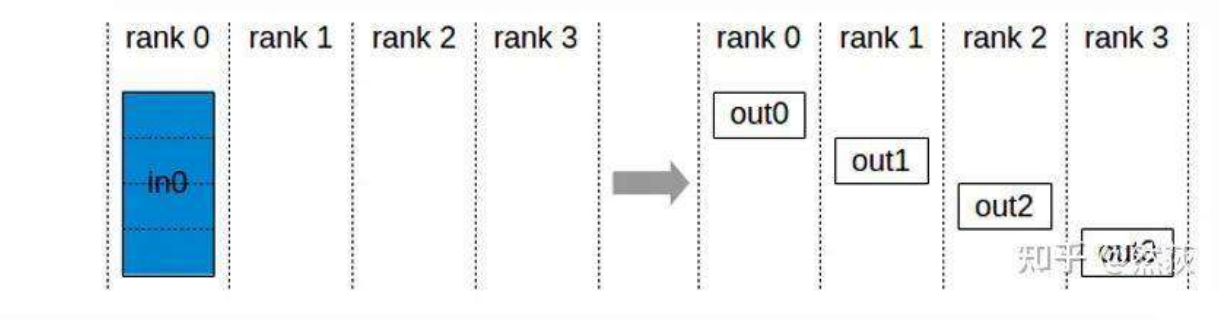
多对多 （Reduce + BroadCast）

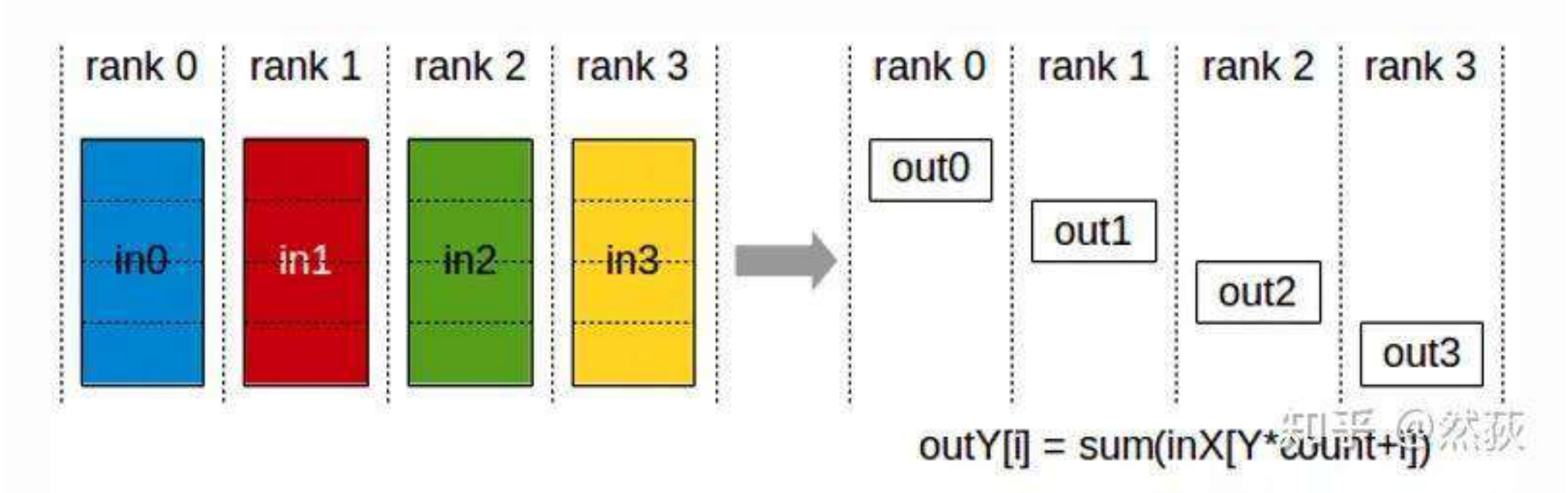


ReduceScatter

多对多 （Reduce + Scatter） AllGather反过程

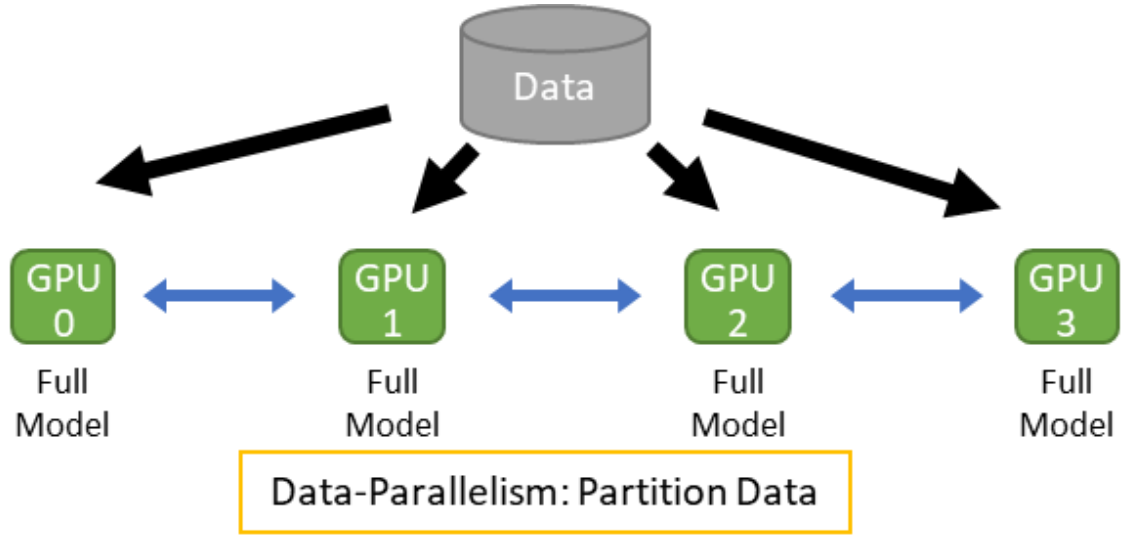




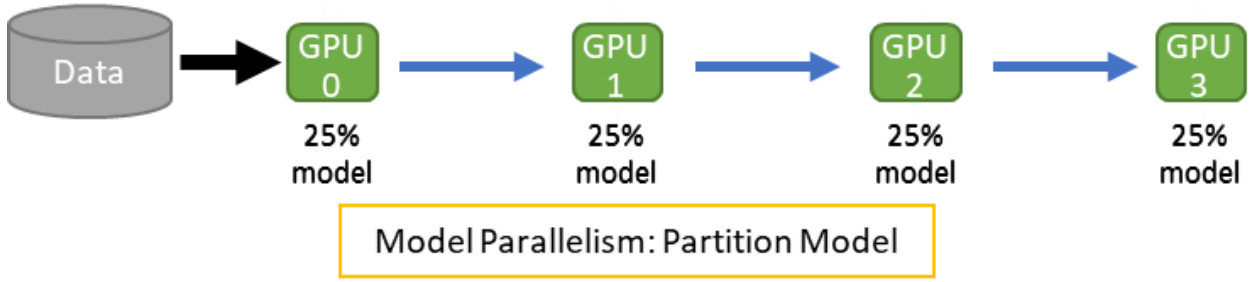


并行推理加速

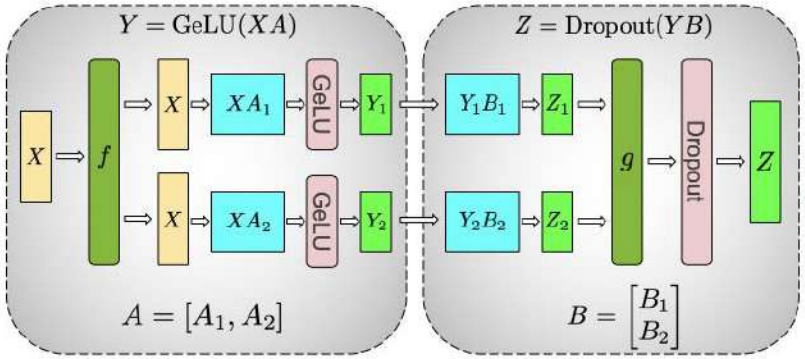
- 数据并行
  - 训练：前向无通信，反向通过AllReduce更新梯度
  - 推理：前向无通信



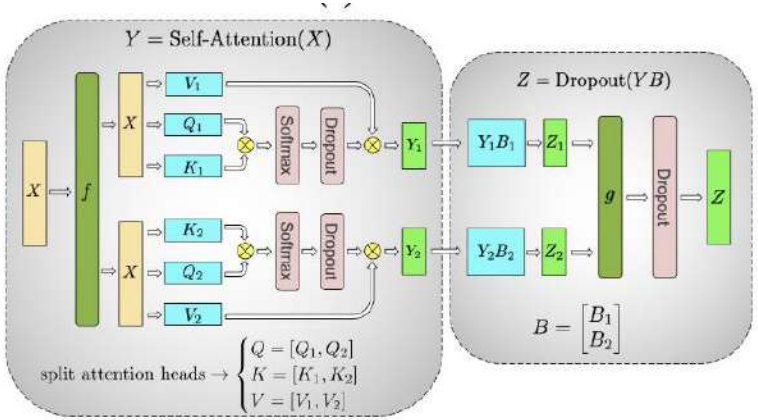
并行推理加速



- 模型并行（流水线并行）
  - 训练：前向点对点通信Send/Recv，反向点对点通信Send/Recv
  - 推理：前向点对点通信Send/Recv
- 模型并行（模型并行）
  - 通常用于模型较大的场景中
  - 训练：前向g进行AllReduce算激活值，反向f进行AllReduce算梯度
  - 推理：前向g进行AllReduce算激活值



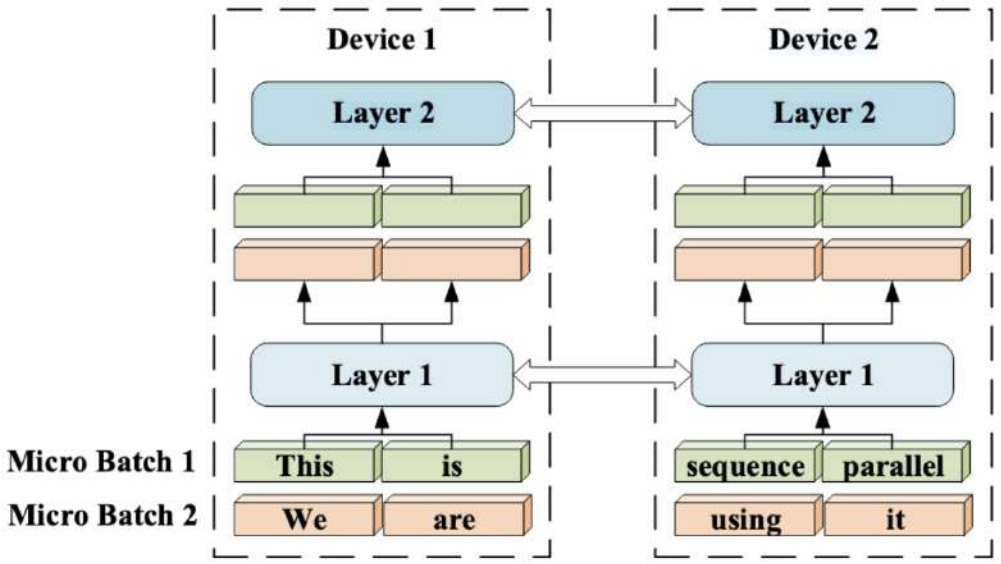
(a) MLP

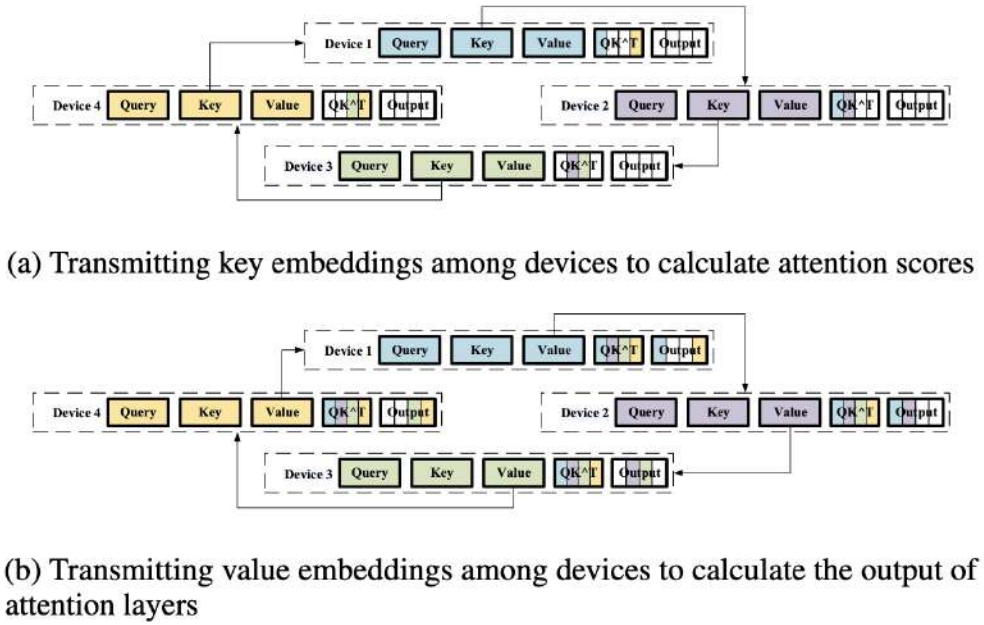


(b) Self-Attention

并行推理加速

- 序列并行：Attention计算
  - 适合长序列场景
  - 推理：前向AllGather + ReduceScatter 全局注意力分数, RingSelfAttention等方式





(a) Transmitting key embeddings among devices to calculate attention scores

(b) Transmitting value embeddings among devices to calculate the output of attention layers

- 提醒：不要盲目使用并行策略，根据模型自身特性调整，记得Profile

任务要求

- 本任务的核心并不在加速比，更多的核心在于如何分析DiT可以加速的方向
- 这需要结合Profile的结果来找到性能瓶颈
- 推荐使用的Profiler是Torch自带的Profiler
- 可以使用chrome自带的网站查看Profile的结果
  - chrome://tracing/
- 最多可以使用4卡，Baseline为单卡结果

```
with torch.profiler.profile(
    activities=[
        torch.profiler.ProfilerActivity.CPU,
        torch.profiler.ProfilerActivity.CUDA,
    ],
    schedule=torch.profiler.schedule(wait=1, warmup=1, active=3),
    on_trace_ready=torch.profiler.tensorboard_trace_handler('./log'),
    record_shapes=True,
    profile_memory=True,
    with_stack=True
) as prof:
    for step in range(10):
        output = model(input)
        loss = output.sum()
        loss.backward()
        prof.step()
```