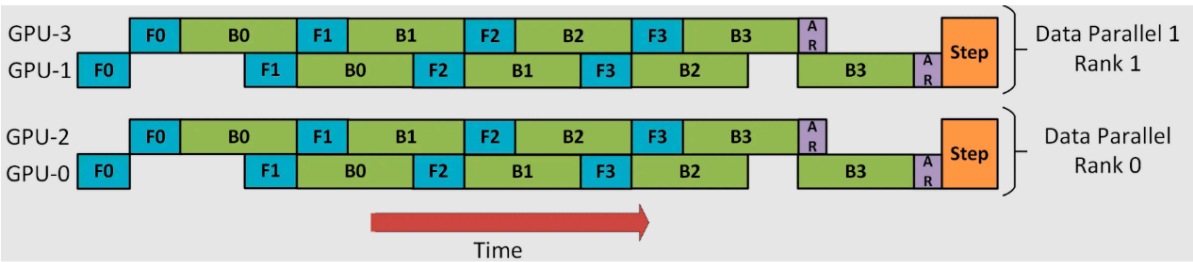


6.多维度混合并行

1.常见的分布式并行技术组合

1.1 DP + PP

下图演示了如何将 DP 与 PP 结合起来使用。



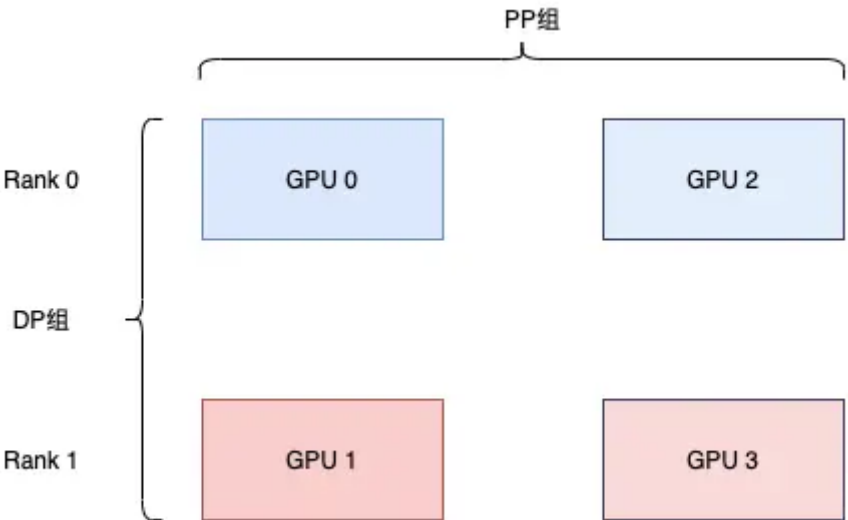
图中展示了 **4 台 GPU** (GPU-0, GPU-1, GPU-2, GPU-3) 用于训练一个模型。每个 GPU 在训练过程中负责不同的数据部分，按照 **数据并行** 的方式分配计算任务。

每个步骤 (Step) 由多个计算块组成，其中：

- **F0, F1, F2, F3** 代表 **前向传播 (Forward Pass)** 过程中的计算。
- **B0, B1, B2, B3** 代表 **反向传播 (Backward Pass)** 过程中的计算。
- **A** 和 **R** 表示 **通信** 操作：数据并行中的通信通常在 **前向传播和反向传播之间** 发生，以保证各 GPU 之间数据的同步。
 - **A**：表示数据的 **All-Gather** 操作，通常在前向传播时进行，以汇总每个 GPU 的数据。
 - **R**：表示数据的 **Reduce-Scatter** 操作，通常在反向传播时进行，以将梯度分散到每个 GPU。

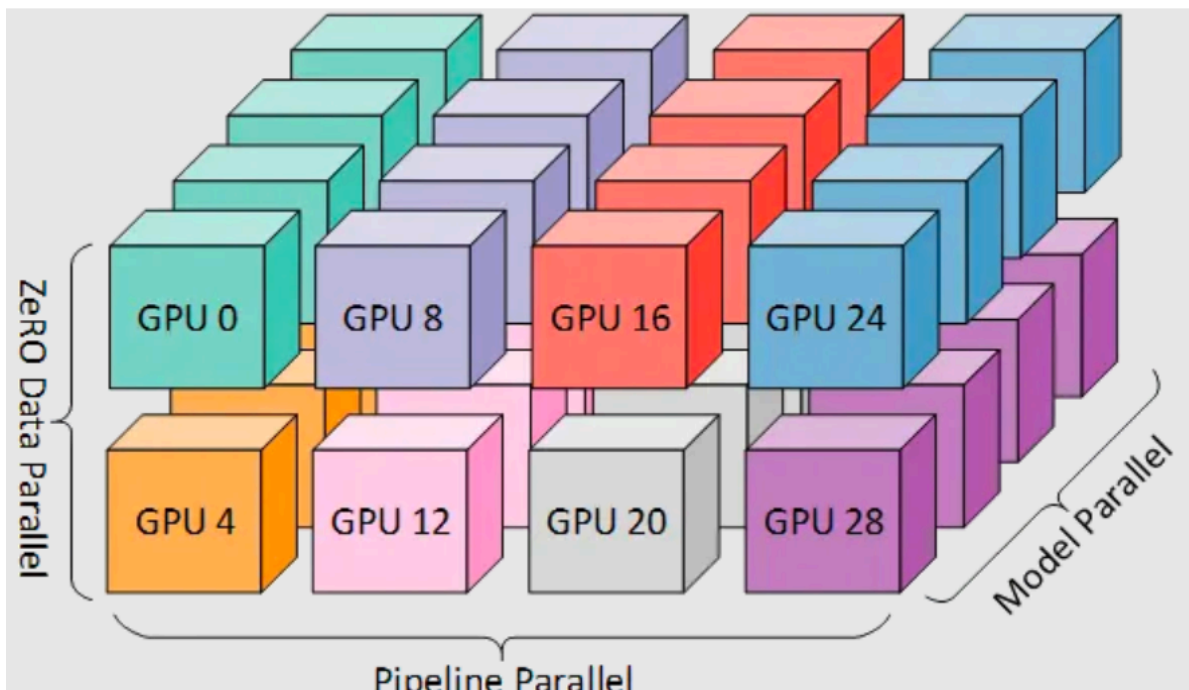
这里重要的是要了解 DP rank 0 是看不见 GPU2 的，同理，DP rank 1 是看不到 GPU3 的。对于 DP 而言，只有 GPU 0 和 1，并向它们供给数据。GPU0 使用 PP 将它的一些负载转移到 GPU2。同样地，GPU1 也会将它的一些负载转移到 GPU3。

由于每个维度至少需要 2 个 GPU；因此，这儿至少需要 4 个 GPU。



1.2 3D并行 (DP + PP + TP)

由于每个维度至少需要 2 个 GPU，因此在这里你至少需要 8 个 GPU 才能实现完整的 3D 并行。



PP (Pipeline Parallelism)：将模型分成多个阶段，流水线方式依次处理；

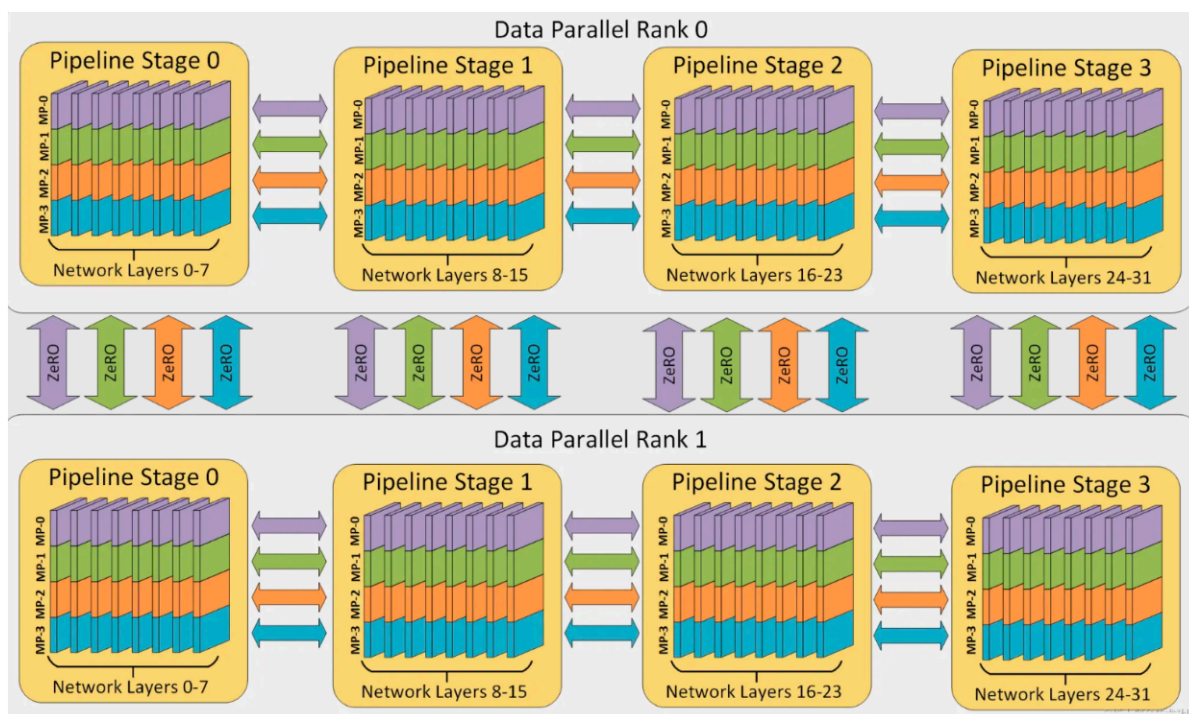
DP (Data Parallelism)：数据分块并行处理；

TP (Tensor Parallelism)：在多个计算单元上并行处理张量，适用于非常大的数据结构和深度学习模型。

1.3 ZeRO-DP + PP + TP

ZeRO，作为 DeepSpeed 的主要功能之一，它是 DP 的超级可伸缩增强版，并启发了 PyTorch FSDP 的诞生。通常它是一个独立的功能，不需要 PP 或 TP。但它也可以与 PP、TP 结合使用。

当 ZeRO-DP 与 PP 和 TP 结合使用时，通常只启用 ZeRO 阶段 1（**只对优化器状态进行分片**）。



而 ZeRO 阶段 2 还会**对梯度进行分片**，ZeRO 阶段 3 还会**对模型权重进行分片**。虽然理论上可以将 ZeRO 阶段 2 与 流水线并行一起使用，但它会对性能产生不良影响。**每个 micro batch 都需要一个额外的 reduce-scatter 通信来在分片之前聚合梯度**，这会增加潜在的显著通信开销。根据流水线并行的性质，我们会使用小的 micro batch，并把重点放在算术强度 (micro batch size) 与最小化流水线气泡 (micro batch 的数量) 两者间折衷。因此，增加的通信开销会损害流水线并行。

在使用流水线并行时，我们希望找到 **micro batch size** 和 **流水线气泡** 之间的平衡。若 **micro batch size 太大**，可能会导致每个阶段的计算量过大，进而导致 **较大的通信开销**，因为每次处理的批量数据较大，可能需要额外的通信步骤来保证数据的一致性（例如梯度的 reduce-scatter 操作）。

如果 **micro batch size 太小**，虽然通信开销可能较小，但会导致 **流水线气泡过多**，即每个阶段的计算没有完全利用。这样，计算资源的使用效率会降低。为了实现性能的最优化，需要平衡这两者

Note

Micro batch size：表示每次训练时输入模型的数据块大小。在流水线并行中，数据会被划分成多个 micro batch（小批量）。每个 micro batch 会通过流水线的不同阶段进行处理，每个阶段完成后会将结果传递给下一个阶段。

算术强度 (Arithmetic Intensity)：指的是每次处理的计算量（例如每个 micro batch 中包含的计算操作数）。算术强度越大，计算量就越大，因此可以更有效地利用硬件资源。

流水线气泡 (Pipeline Bubble)：流水线气泡指的是由于不同阶段的计算速度不一致所造成的等待时间。在流水线并行中，每个阶段的处理时间可能不同，导致有些阶段在等待其他阶段完成时处于空闲状态，产生空闲期或“气泡”。

此外，由于 PP，层数已经比正常情况下少，因此并不会节省很多内存。PP 已经将梯度大小减少了 $1/PP$ ，因此在此基础之上的梯度分片和纯 DP 相比节省不了多少内存。

Note

PP (流水线并行) 会将模型分成多个阶段，每个阶段处理不同的层。由于这种方式，每个阶段只需要存储模型的一部分（而不是整个模型），因此可以减少内存的消耗。**但是**，由于每个阶段只负责一部分层数，相比正常单机训练，模型层数已经较少了。

PP 已经将每个阶段的梯度大小减少了，因为每个阶段只负责模型的一部分。假设你有 N 个层，如果你用 PP 将模型分成 P 个阶段，那么每个阶段只处理 N/P 层的模型。这样就减少了每个阶段需要存储的梯度信息。

然而，这并不能与纯 **数据并行 (DP)** 相比节省很多内存。因为在 **数据并行** 中，虽然每个设备也需要存储一份完整的模型，但数据并行的优点是每个设备可以并行处理不同的训练数据。这使得在 **PP + DP** 的组合中，内存开销通常会比单纯的 **PP** 更有效地分摊。

除此之外，我们也可以采用 DP + TP 进行组合、也可以使用 PP + TP 进行组合，还可以使用 ZeRO3 代替 DP + PP + TP，ZeRO3 本质上是 DP+MP 的组合，并且无需对模型进行过多改造，使用更方便。

Note

DP + TP（数据并行 + 张量并行）组合：

- **数据并行 (DP)**：将数据集分割成多个部分，每个部分在不同的处理单元（如 GPU）上并行计算。每个 GPU 都有一份模型的副本，处理不同的数据。
- **张量并行 (TP)**：将大型张量（例如大矩阵或多维数组）拆分到不同的处理单元上，每个处理单元只负责处理张量的一部分。
- **组合**：在这种组合中，数据并行负责分配不同的数据部分，张量并行则将每个张量拆分到不同的计算单元上，从而加速大模型的训练。

这种组合的优势在于，它可以同时利用 **数据并行** 提高计算速度，同时使用 **张量并行** 处理模型的巨大规模，特别是在非常大的模型上，可以将模型分成多个部分进行并行计算。

PP + TP（流水线并行 + 张量并行）组合：

- **流水线并行 (PP)**：将模型分成多个阶段，每个阶段由不同的GPU进行计算，依次传递数据。每个GPU只负责模型的一部分（例如某一层或几层）。
- **张量并行 (TP)**：将张量分割到不同的处理单元上，利用多个GPU并行计算。
- **组合**：在这个组合中，流水线并行负责将模型拆分成多个阶段进行处理，而张量并行则进一步细分每个阶段的计算任务。

这种组合适用于需要细分模型计算的场景，尤其是当模型非常大且需要跨多个设备进行处理时。通过组合 **PP** 和 **TP**，可以将计算任务更细粒度地分配到多个GPU上。

ZeRO3 (Zero Redundancy Optimizer 3) 替代 DP + PP + TP：

- **ZeRO3** 是 **ZeRO (Zero Redundancy Optimizer)** 的第三个版本，设计目标是优化大规模模型训练中的内存使用。ZeRO3 是 **数据并行 (DP)** 和 **模型并行 (MP)** 的结合体：
 - **数据并行 (DP)**：每个设备处理不同的数据。
 - **模型并行 (MP)**：将模型的不同部分分配到不同的设备上。
- **优势**：ZeRO3 通过减少冗余的存储和计算需求，显著提高了内存使用效率，它通过智能地分配内存和计算负载，避免了传统方法中的大量数据冗余。
- **不需要过多改造**：使用 ZeRO3 时，用户无需对模型进行大量的修改。它可以直接在现有的训练框架上实现，不像传统的 **PP + TP** 等方法那样需要对模型结构进行大规模调整。

2.业界大模型混合并行策略

2.1 CodeGeeX (13B)

CodeGeeX 是一个具有 130 亿参数的多编程语言代码生成预训练模型。CodeGeeX 采用华为 MindSpore 框架实现，在鹏城实验室“鹏城云脑II”中的192个节点（共1536个国产昇腾910 AI处理器）上训练而成。CodeGeeX 历时两个月在20多种编程语言的代码语料库（> 8500 亿 Token）上预训练得到。

CodeGeeX 使用纯解码器的GPT架构，并使用自回归语言建模。CodeGeeX 的核心架构是39层的Transformer解码器。在每个Transformer层包含：多头自注意力模块、MLP模块、LayerNorm和残差连接。使用类GELU的FastGELU激活，其在Ascend 910 AI处理器上更加高效，整个模型架构如下图所示：

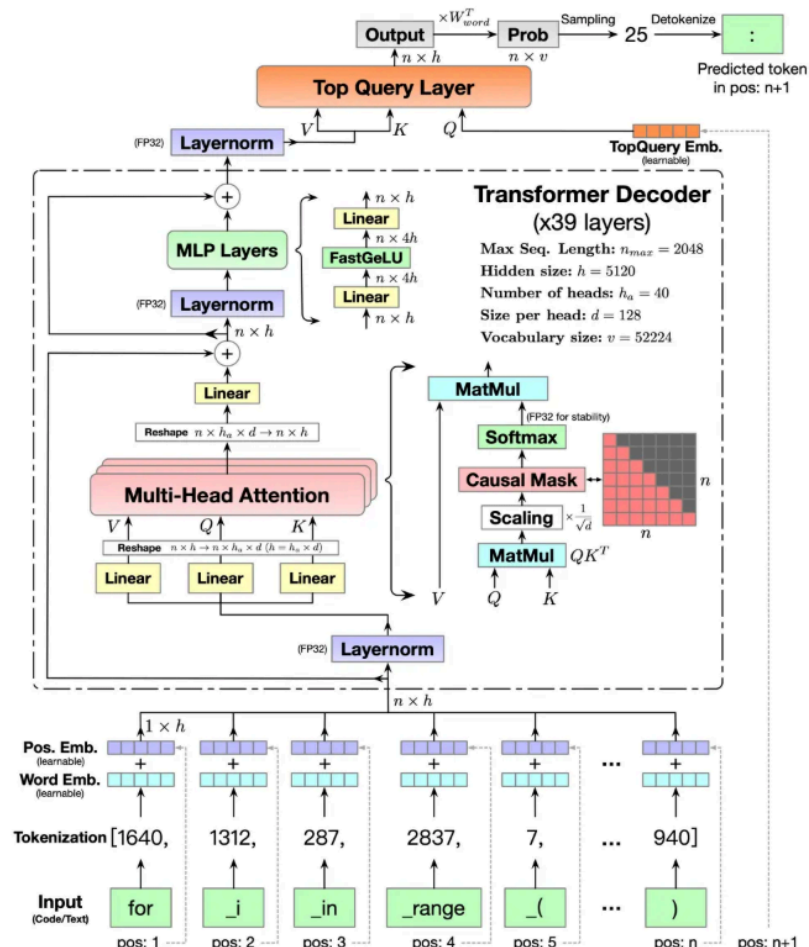


Figure 2: CodeGeeX's model architecture. CodeGeeX is a code generation model with **13B** parameters, **consisting of 39-layer left-to-right transformer decoders** and **a top query layer**. It takes text/code tokens as input and outputs the probability of the next token autoregressively.

Note

GELU (Gaussian Error Linear Unit, 高斯误差线性单元) 最早由 Google 在 2016 年提出 (Hendrycks & Gimpel), 是 Transformer 系列模型 (包括 BERT、GPT 等) 常用的激活函数。

其定义公式为:

$$\text{GELU}(x) = x \cdot \Phi(x)$$

其中 $\Phi(x)$ 是标准高斯分布的累积分布函数 (CDF)。展开近似形式为:

$$\text{GELU}(x) \approx 0.5x \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right) \right)$$

特点:

- **优点:** 平滑、连续、带概率性门控效果 (相比 ReLU 更柔和)。
- **缺点:** 涉及到 \tanh 和多项式运算, 计算开销较大, 在一些硬件 (尤其是国产芯片如昇腾 910) 上可能效率不高。

FastGELU 是对 GELU 的 **近似优化版本**, 主要目标是 **减少计算开销**, 同时保持和 GELU 相似的性能表现。

常见的 FastGELU 近似公式是:

$$\text{FastGELU}(x) = x \cdot \sigma(1.702x)$$

其中 σ 是 sigmoid 函数。

也就是说，FastGELU 把复杂的 tanh 和多项式近似，替换成了更便宜的 sigmoid 运算。

为了提高训练效率，CodeGeeX采用8路模型并行组和192路数据并行组进行混合并行训练；同时，启用ZeRO-2来进一步减少优化器状态的内存消耗。

2.2 GPT-NeoX (20B)

GPT-NeoX-20B 是一个具有 200 亿参数通用的自回归密集型预训练语言模型。在12 台 Supermicro AS-4124GO-NART 服务器上训练；其中，每台服务器配备 8 个 NVIDIA A100-SXM4-40GB GPU，并配置了两个 AMD EPYC 7532 CPU。所有 GPU 都可以通过用于 GPUDirect RDMA 的四个 ConnectX-6 HCA 之一直接访问 InfiniBand 交换结构 (switched fabric)。两台 NVIDIA MQM8700-HS2R 交换机 (通过 16 个链路连接) 构成了该 InfiniBand 网络的主干，每个节点的 CPU 插槽有一个链路连接到每个交换机。每个训练节点的架构图如下所示：

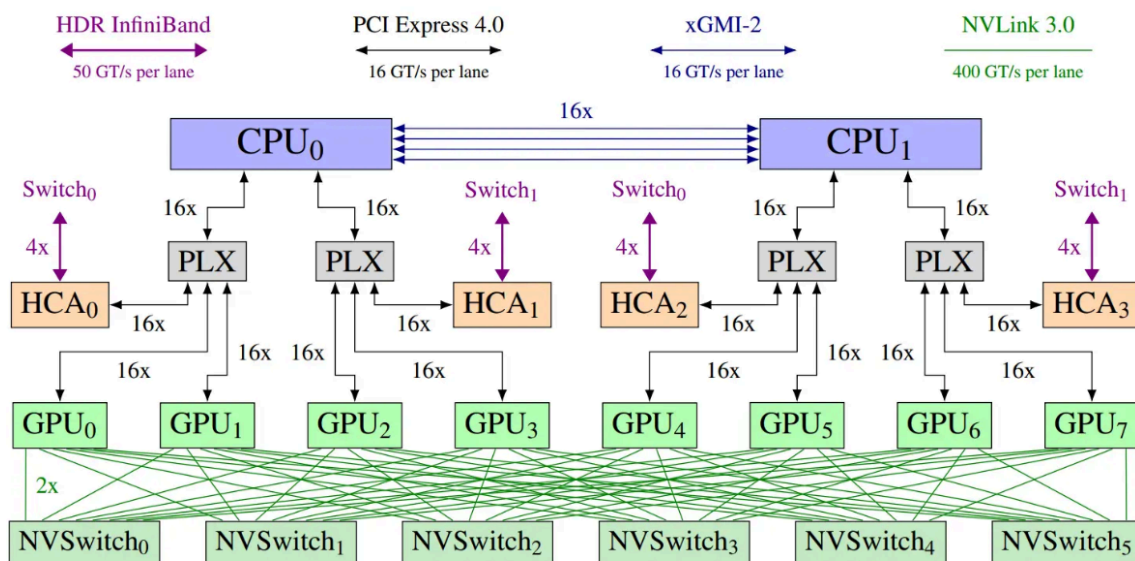


Figure 2: Architecture diagram of a single training node.

Note

在你提到的架构中，PLX 是一个常见的硬件组件名称，指的是一种 **PCIe (Peripheral Component Interconnect Express) Switch**。PLX 通常用于在数据中心和高性能计算环境中扩展 PCIe 总线，以支持多个设备之间的高带宽连接。

PLX (PCIe Switch) 解释：

- **功能：**PLX 是一种 **PCIe 交换机**，通常用于将多个设备（如 GPU、网络卡等）连接到主机系统上。它的作用是将一个或多个 PCIe 主机端口与多个 PCIe 从端口相连，从而允许多个设备共享 PCIe 带宽。
- **作用：**在这类系统中，**PLX Switch** 通常用于扩展 **PCIe 总线**，使得多个 GPU（或者其他硬件）能够通过一个或多个连接与主机进行高带宽通信。
- **应用场景：**特别是在数据中心，或者涉及多个 GPU 的高性能计算任务（如深度学习、AI 模型训练等）中，PLX 交换机可以有效地管理多个设备的 PCIe 带宽，避免带宽瓶颈。

HCA (Host Channel Adapter) 解释：

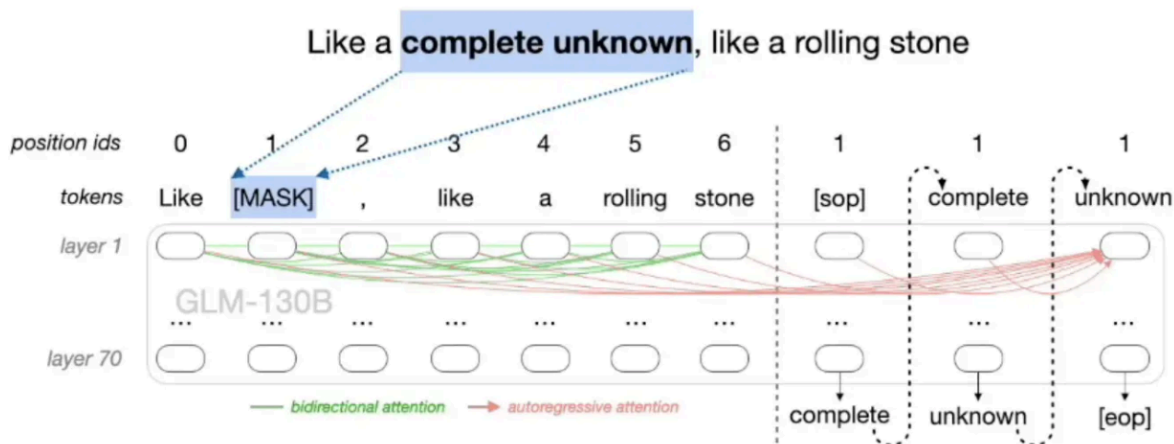
- HCA 是一种 **网络适配器**，用于将主机（如服务器、工作站）连接到高速网络（例如 InfiniBand 或以太网）。它负责 **管理数据传输、处理网络通信**，并提供与主机内存或 GPU 之间的高效数据传输通道。
- HCA 允许 **远程直接内存访问 (RDMA)**，意味着数据可以直接在内存之间传输，而不需要通过 CPU 处理。这对于提高性能和减少延迟至关重要，尤其在大规模训练模型时。

GPT-NeoX-20B 采用了数据并行、流水线并行和张量并行相结合的方式训练。

同时，作者发现，在给定硬件设置的情况下，最有效方法是将张量并行大小设置为 2，将流水线并行大小设置为 4。这允许最通信密集的进程，张量和流水线并行发生在节点内，数据并行通信发生在节点边界之间。

2.3 GLM (130B)

GLM-130B 是一个由清华开源的双语（中文和英文）双向稠密模型，拥有 1300 亿参数，模型架构采用通用语言模型（GLM）。在超过 4000 亿个文本标识符上预训练完成。GLM-130B 利用自回归空白填充作为其主要的预训练目标，下图中的句子为例，它掩盖了随机的连续文本区间（例如，“complete unknown”），并对其进行自回归预测。



在实际训练中，GLM-130B 使用两种不同的掩码标识符（[MASK] 和 [gMASK]），分别用于短文和长文的生成。此外，它还采用了最近提出的旋转位置编码（RoPE）、DeepNorm 层规范化和高斯误差 GLU（GeGLU）技术。所有这些设计和技术都对 GLM-130B 大规模语言模型的稳定训练和高精度性能有所帮助。具体来说，GLM-130B 模型含有 70 层 Transformer，隐层维度 12,288，最大序列长度 2,048，以及一个基于 [icetk](#) 的 150,000 个标识符的双语分词器。

它的预训练目标由两部分组成：第一部分（95%）是自监督的预训练，即在公开的大规模语料库以及其他一些较小的中文语料库上的自回归空白填充。第二部分（5%）是在 T0++ 和 DeepStruct 中 70 个不同数据集的抽样子集上进行多任务指令预训练，格式为基于指令的多任务多提示序列到序列的生成。这种设计使 GLM-130B 可以在其他数据集上进行了零样本学习，以及从英文到中文的零样本迁移。

GLM-130B 的预训练持续了 60 天，使用 96 个 DGX-A100 (40G) 节点，共 768 张 GPU 卡。采用了**流水线模型并行与张量并行、数据并行策略相结合的方式**，形成 3D 并行策略。

为了进一步减少流水线引入的气泡，利用 DeepSpeed 的 PipeDream-Flush 实现来训练具有相对较大的全局批量大小 (4,224) 的 GLM-130B，以减少时间和 GPU 内存浪费。通过数值和实证检验，采用 4 路张量并行组和 8 路流水线并行组，达到每张 GPU (40G) 135 TFLOP/s。

2.4 OPT (175B)

OPT-175B 是 Meta AI 开源的一个拥有 1750 亿参数的语言模型，利用**完全分片数据并行（FSDP）与 Megatron-LM 张量并行（8路组）**在 992 个 80GB A100 GPU 上训练了 OPT-175B。训练数据包含 180B 个 token，对应 800GB 的数据，持续训练了约 33 天。

每个 GPU 的利用率高达 147 TFLOP/s。OPT-175B 将 Adam 状态使用 FP32，并将其分片到所有主机上；而模型权重则使用 FP16。为了避免下溢，使用了动态损失缩放。

① Note

FSDP 的核心概念：

1. **完全分片**：FSDP 将 **模型参数、优化器状态和梯度** 分割到不同的计算节点上，甚至在每个 GPU 上进行分片。每个 GPU 只存储模型的一部分参数，而不是整个模型的副本。这样能够极大地减少每个设备的内存消耗。
2. **数据并行**：每个 GPU 上都进行 **数据并行** 计算，每个 GPU 执行相同的计算任务，但每个 GPU 处理不同的数据批次 (batches)。FSDP 与传统的数据并行方法相同，但它通过分片模型和优化器状态来减轻内存压力。
3. **内存优化**：FSDP 的关键优势是通过分片，减少了每个 GPU 上需要存储的内存量。这对于 **大模型** (如 OPT-175B) 尤其重要，因为它们的参数量非常庞大，单个 GPU 无法容纳整个模型。
4. **动态损失缩放**：为了避免在 **半精度计算** 中可能出现的下溢 (即值过小无法表示)，FSDP 使用了 **动态损失缩放** 技术，确保计算稳定性并最大化数值精度。

当执行训练步骤时，FSDP 会根据需要 **调用不同部分**：

1. **前向传播**：
 - 当模型进行前向传播时，只有 **必要的模型参数** 被从 **相应的 GPU 群** 中 **加载**。每个 GPU 只加载它负责的 **模型部分**。
 - 如果模型分为多个部分 (例如多层或分布式张量)，每个 GPU 群会根据输入数据 **计算自己的部分**，然后通过网络将 **结果传递到下一个阶段** (如流水线并行中的“下一个阶段”)。
2. **反向传播**：
 - 在反向传播中，每个 GPU 群只计算 **自己负责的参数的梯度**。
 - 在更新阶段，**梯度分片** 会被 **同步** (通常使用 **all-reduce 操作**)。每个 GPU 群会更新 **它所负责的模型参数部分**，并在每次更新时通过同步获得 **全局梯度**。
3. **优化器更新**：
 - 优化器状态会被分配到多个 GPU 群，每个 GPU 群更新自己分配的 **优化器状态部分**。
 - 由于优化器状态也是分片存储，FSDP 会在 **训练的每个步骤** 确保所有 GPU 群的优化器状态同步。每个 GPU 群只会更新它负责的 **优化器部分**。

FSDP 在 OPT-175B 训练中的应用：

- **模型权重使用 FP16**：OPT-175B 的模型权重被存储为 **半精度浮点数 (FP16)**，这能够节省内存并加速训练过程。
- **Adam 优化器状态使用 FP32**：在训练中，使用 **全精度 (FP32)** 存储 Adam 优化器状态，以确保优化过程的稳定性和准确性，避免精度损失。
- **模型和优化器状态分片**：FSDP 将 **Adam 状态和模型权重** 分片到所有 **992 个 GPU** 上。每个 GPU 只负责模型的一部分，减少了每个 GPU 的内存负担。

2.5 Bloom (176B)

Bloom-176B 是一个拥有 1760 亿参数自回归大语言模型 (LLM)，它是迄今为止开源的最大的多语言 (含 46 种自然语言和 13 种编程语言) 大模型，整个模型架构如下图所示：

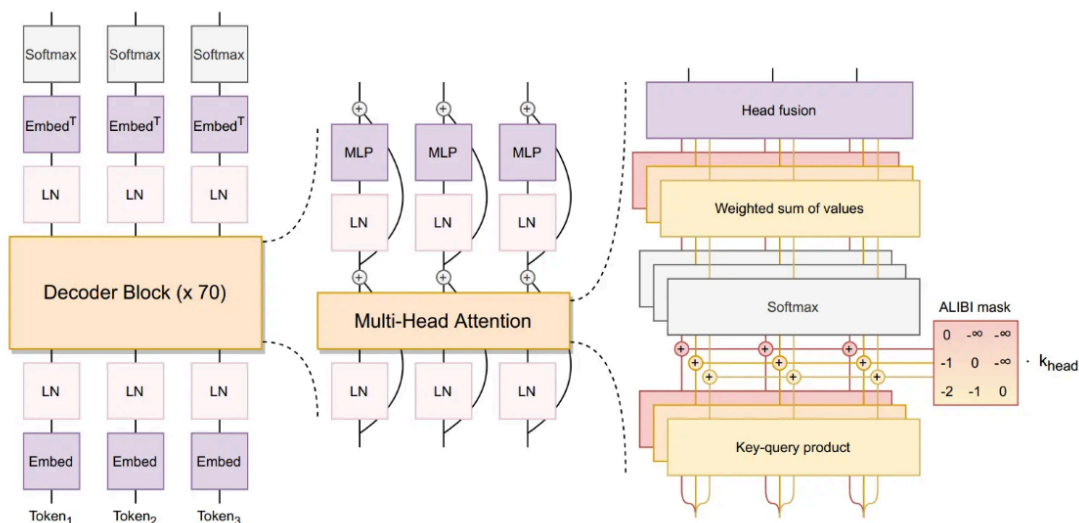


Figure 5: The BLOOM architecture. The k_{head} slope parameters for ALIBI are taken as $2 \frac{-8i}{n}$ with n the number of heads and $i \in 1, 2, \dots, n$.

Note

ALIBI Mask: ALIBI (Attention with Linear Biases) 掩码是一种通过给注意力权重添加线性偏差 (bias) 来优化模型的机制。它通过掩码机制来防止模型看到不应该看到的未来信息 (例如, 在自回归生成任务中, 当前 token 不应看到未来的 token)。

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。

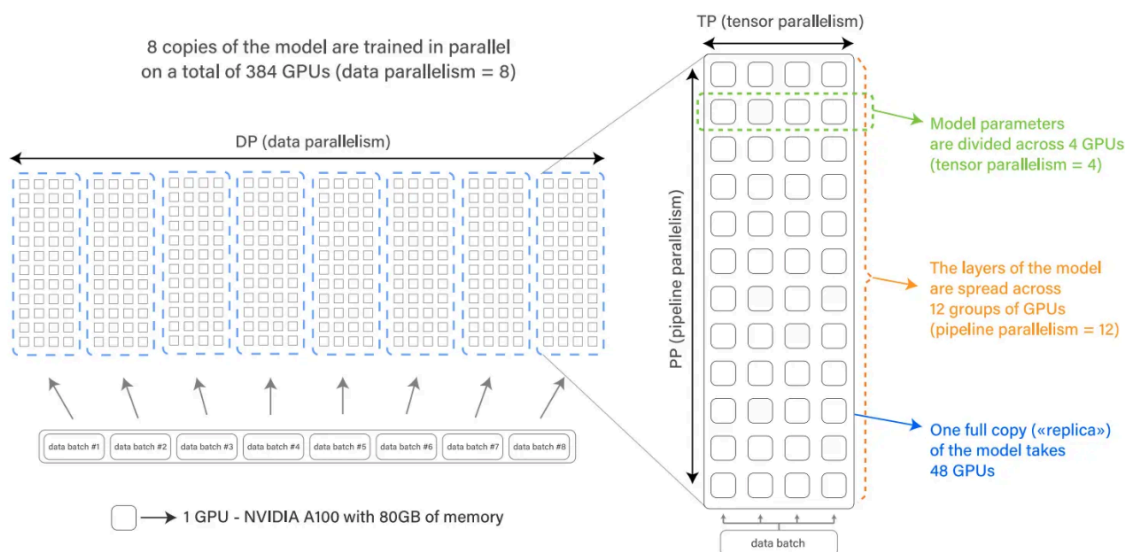


Figure 6: DP+PP+TP combination leads to 3D parallelism.

左边: 每个数据批次 (data batch 1, 2, ..., 8) 通过数据并行分配给不同的 GPU, 每个 GPU 处理不同的输入数据批次。每个数据批次的计算结果将会通过流水线和张量并行进行分配和计算。

右边: 在 张量并行 (TP) 的区域, 模型的参数分布在多个 GPU 上。每个模型参数会分配到不同的 GPU 上进行计算。

模型副本: 每个 数据并行副本 会占用 48 个 GPU, 其中包括张量并行和流水线并行的计算资源。


2.6 Megatron-Turing NLG (530B)

Megatron-Turing NLG-530B 是微软和英伟达联合推出的一个包含 5300 亿参数的自回归大语言模型。使用了 Transformer 解码器的架构，其中：Transformer层数、隐藏层维度、注意力头分别为 105、20480 和 128。序列长度为2048，全局批量大小为1920。

在训练时，每个模型副本跨越 280 个 NVIDIA A100 GPU，节点内采用Megatron-LM 的 8 路张量并行组，节点间采用 35 路流水线并行组。整个训练过程一共使用了 4480 块英伟达 A100 GPU，在 2700 亿个 Token 上面训练

3.总结

本文主要讲解了常见的大模型分布式并行技术的组合策略，同时，也讲述了目前业界的一些大模型所使用的并行策略，具体如下表所示。

模型	DP	TP	PP	ZeRO Stage	FSDP (ZeRO Stage 3)	GPUs	FP16/BF16
Bloom-176B	8	4	12	ZeRO-1	-	384 张 A100 80GB	BF16
CodeGeeX-13B	192	8	-	ZeRO-2	-	1,536 张 Ascend 910 32GB	FP16
GLM-130B	24	4	8	ZeRO-1	-	768 张 A100 40G	FP16
OPT-175B	124	8	-	-		992 张 80GB A100	FP16
Megatron-Turing NLG-530B	16	8	35	N/A	-	4480 张 A100 80G	BF16
GPT-NeoX-20B	12	2	4	ZeRO-1	-	96 张 A100 40G	FP16