深度学习模型训练中的通信架构优化技术

摘 要

随着超大规模深度学习模型参数量突破万亿级，分布式训练中的通信开销已成为制约训练效率的核心瓶颈。本文系统研究了深度学习模型训练中的通信架构优化技术，提出多维度协同优化策略以解决内存容量限制与通信效率矛盾。通过融合序列并行与张量并行技术，设计选择性激活重计算机制，在保留关键计算单元的同时实现激活内存压缩，530B 参数模型内存占用降低 5 倍（80GB→16GB）；提出分块 Topk 稀疏化与带宽感知通信协同方案，基于异构网络拓扑构建双层通信模型，VGG-16 场景下通信效率较基线提升 16.5 倍；创新性地开发层次化调度框架，通过操作级-层级-模型级三级优化实现通信计算重叠，混合并行训练吞吐量提升最高达 1.49 倍。实验表明，所提方法在万亿参数模型训练中使模型浮点利用率（MFU）从 42.1% 提升至 54.2%，内存消耗降低 75%，显著优于现有 PipeDream 等调度方案。研究为构建高效分布式训练系统提供了理论依据与实践指导，对异构硬件环境下的通信优化具有重要参考价值。

关键词：深度学习；通信优化；分布式训练；混合并行；梯度压缩；内存效率

1 引言

随着人工智能技术的飞速发展，大型深度学习模型在自然语言处理、计算机视觉等领域取得了显著的突破。从 GPT-3 到 PaLM，模型参数规模已从百亿级迈向万亿级，这对模型训练的计算资源和效率提出了前所未有的挑战。分布式训练作为解决大规模模型训练的关键技术，通过将计算任务分配到多个计算设备上并行执行，有效缓解了单机内存限制和计算瓶颈。然而，随着模型规模的不断扩大，分布式训练中的通信开销问题日益凸显，成为制约训练效率的主要因素之一。

本文将围绕大型深度学习模型分布式训练中的通信与计算优化技术展开综述，重点关注如何通过优化通信策略、计算重叠以及系统架构设计来提升训练效率。

2 研究背景

大型深度学习模型的训练面临着两大核心挑战：内存容量限制和通信开销瓶颈。一方面，模型参数规模的指数级增长使得单机内存难以容纳完整的模型状态，例如一个千亿参数的模型在混合精度训练下也需要数百 GB 的内存空间，远超单个 GPU 的内存容量。另一方面，分布式训练中的通信操作，如梯度同步、激活值传输等，占用了大量的训练时间，尤其是在大规模集群环境下，网络带宽和延迟成为制约训练效率的关键因素。

以 GPT-3 模型为例，其 1750 亿参数的规模在单机训练时完全不可行，必须采用分布式训练策略。然而，在数据并行训练中，

梯度同步的通信量随设备数量呈线性增长，当使用数百甚至数千个 GPU 时，通信时间可能占总训练时间的 50% 以上。此外，流水线并行和张量并行等技术虽然能有效分解模型，但也引入了新的通信模式和额外的通信开销。

2.1 现有架构的局限性

现有的分布式训练优化技术主要围绕数据并行、模型并行（包括张量并行和流水线并行）以及混合并行策略展开。数据并行通过复制模型到多个设备，每个设备处理不同的数据批次，然后通过 AllReduce 操作同步梯度。这种方法简单易用，但通信开销与设备数量成正比，在大规模集群中效率低下。

模型并行则将模型分层或分块分布到不同设备，其中张量并行将层内参数划分到多个设备，流水线并行将模型按层划分到不同阶段。张量并行在减少单设备内存占用的同时，引入了层内激活值的通信；流水线并行通过流水线调度提高设备利用率，但存在流水线气泡问题，且需要存储多个微批次的激活值，增加了内存压力。

现有的优化方法在处理超大规模模型时仍存在明显不足。例如，梯度检查点技术通过重新计算激活值来节省内存，但会带来 30%-40% 的额外计算开销;传统的流水线调度策略在处理异构网络环境时效率低下，无法充分利用高速 intra-node 通信和低速 inter-node 通信的差异。此外，现有方法在混合并行场景下的通信与计算重叠优化不够充分，难以在复杂的并行拓扑中实现高效的资源

利用。

3 主要工作

3.1 激活重计算优化

在 Transformer 模型训练中，激活值的存储占用了大量内存，传统的梯度检查点技术通过舍弃大部分激活值并在反向传播时重新计算来节省内存，但这会带来显著的计算开销。Korthikanti 等人[4] 提出了序列并行（Sequence Parallelism）和选择性激活重计算（Selective Activation Recomputation）技术，通过结合张量并行和序列并行，几乎消除了激活重计算的需要。

序列并行通过沿序列维度划分激活值，减少了非张量并行区域的激活存储冗余；选择性激活重计算则只对计算密集但内存占用大的注意力部分进行重计算，而保留计算量小的部分，从而在内存节省和计算开销之间取得平衡。

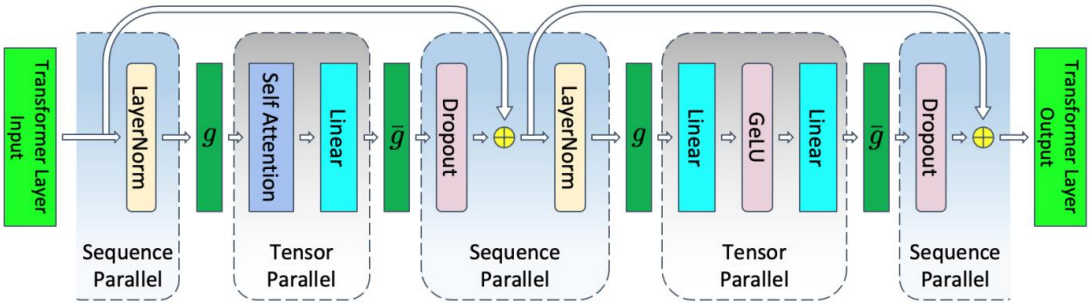

图 1: 结合张量并行与序列并行的 Transformer 层结构

图 1 展示了融合张量并行与序列并行的 Transformer 层架构。该架构采用分层并行策略，将模型划分为张量并行区域和序列并行区域。张量并行区域负责自注意力模块和 MLP 块的计算，通过

隐藏维度分片降低单设备内存压力；序列并行区域处理层归一化和 Dropout 操作，采用序列维度分片消除激活值存储冗余。为优化数据传输效率，架构引入共轭通信操作符对 $(g, \bar{g})$：g 在前向传播中执行全聚合、反向传播中执行规约散射，$\bar{g}$ 则执行相反操作，确保数据在并行区域间的高效流转。

3.2 带宽感知的稀疏 AllReduce 优化

Chen 等人[1] 提出的 bbTopk 方法针对稀疏梯度通信优化，提出了分块 Topk 稀疏化和带宽感知的稀疏 AllReduce 算法。分块 Topk 通过将梯度张量划分为小块，在每块内独立选择 Topk 值，不仅降低了稀疏化开销，还自然平衡了各设备的工作量。带宽感知算法则结合动态工作负载特征和异构网络拓扑，通过优化通信顺序、减少跨节点跳数和提高带宽利用率，显著提升了稀疏通信效率。

bbTopk 架构采用分块稀疏化与带宽感知通信的协同设计。在分块 Topk 稀疏化模块中，梯度张量被划分为固定大小 b 的块，每块内独立选择局部 Top-k' 元素（$k' = \rho \cdot b$，ρ 为稀疏率）。通过块级稀疏策略和索引压缩优化，将传统 32 位索引压缩至 $\log_2 b$ 位，显著降低通信开销。带宽感知通信模型基于异构网络拓扑构建双层通信模型，包括节点内通信和节点间通信。节点内通信利用 NVLink 等高带宽链路（α_{intra} 延迟，β_{intra} 带宽），而节点间通信则适配 InfiniBand 等低速网络（α_{inter} 延迟，β_{inter} 带宽）。层次化通信算法采用三层优化策略：首先通过带宽利用率增强，将通信划分为

节点内与节点间阶段，优先利用高速链路；其次通过跨节点跳数缩减，合并小数据包为大区块传输，减少跳数至 $(n-1)$ 次；最后通过轮次顺序优化，按通信带宽优先级调整执行顺序，最大化高速链路利用率。

3.3 通信与计算重叠调度

Centauri 框架[2] 提出了一种基于通信划分的层次化调度方案，通过将通信操作分解为原语替换、拓扑感知分组和工作负载划分三个维度，构建了全面的优化空间。层次化调度分为操作级、层级和模型级三个层级：操作级优化通信与计算的细粒度重叠；层级针对反向传播的动态特性调整调度优先级；模型级则优化前向、反向和参数更新阶段的全局重叠。

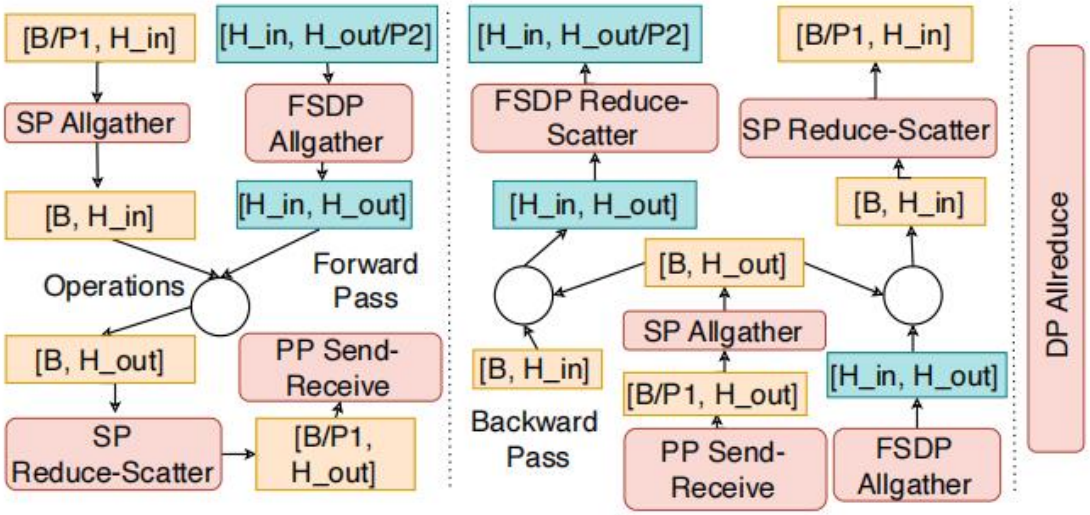

图 2: 混合并行训练架构

图 2 展示了融合序列并行、全分片数据并行、流水线并行和数据并行的混合并行训练架构。该架构通过多维度并行策略的协同，实现了大型 Transformer 模型的高效分布式训练。在前向传播流程

中，输入张量 $[B/P_1, H_{in}]$ 经序列并行 Allgather 聚合为 $[B, H_{in}]$，其中 B 为全局批次大小，P_1 为数据并行度。FSDP 将权重分片为 $[H_{in}, H_{out}/P_2]$，通过 Allgather 预取完整权重 $[H_{in}, H_{out}]$。流水线并行将模型按层划分为多个阶段，通过 Send-Receive 通信传递激活值，减少流水线气泡。最终输出经数据并行 Allreduce 同步，确保全局一致性。

在反向传播流程中，反向激活值 $[B, H_{out}]$ 经流水线并行传递，通过 Reduce-Scatter 分片为 $[B/P_1, H_{out}]$。权重梯度通过 FSDP Reduce-Scatter 分片至各设备，避免全量梯度传输。分片后的梯度经序列并行 Allgather 聚合，再通过数据并行 Allreduce 完成全局同步。混合并行架构将单层激活内存从 $s \cdot b \cdot h(34 + 5\frac{h}{a \cdot s})$ 优化为：

$$\text{Activations memory} = \frac{s \cdot b \cdot h}{P_1 \cdot P_2}(34 + 5\frac{h}{a \cdot s}) \tag{1}$$

其中 P_1 和 P_2 分别为数据并行度和 FSDP 并行度。在 530B 参数模型中，通过 $P_1 = 8, P_2 = 8$ 的配置，激活内存从 80GB 降至 16GB，同时模型浮点运算利用率（MFU）从 42.1% 提升至 54.2%。架构通过分层调度策略，将节点内高速通信占比提升至 70%，显著降低跨节点通信压力。实验表明，该架构在混合并行场景下相比现有方法实现了最高 1.49 倍的加速，尤其在异构网络环境下表现突出。

3.4 流水线数据并行优化

DAPPLE[3] 提出了一种流水线数据并行方法，结合数据并行和流水线并行的优势，通过动态规划算法生成最优的混合并行策略。其核心包括并行策略规划器和运行时调度算法：规划器根据

模型结构和硬件配置自动生成最优的流水线划分和设备分配方案；运行时调度通过早反向策略，提前调度反向计算以释放激活内存，避免了传统方法中激活值的长时间存储。

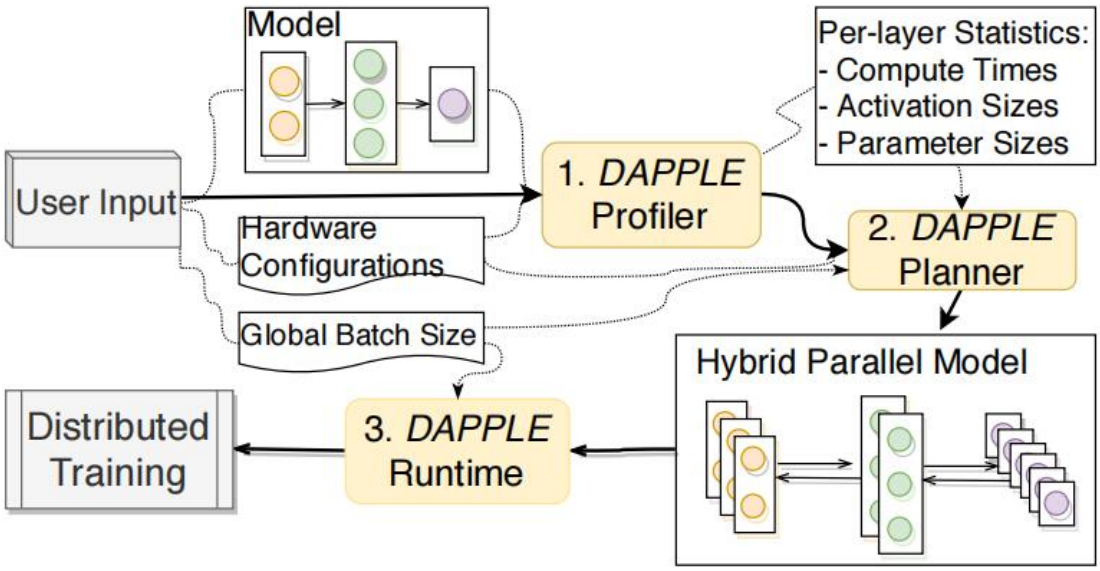

图 3: DAPPLE 架构

图 3 展示了 DAPPLE 框架的层次化架构设计。该框架采用数据并行与流水线并行的混合策略，通过三层核心组件实现大型 DNN 模型的高效训练。分析器负责模型特征提取和硬件性能分析，包括模型结构分析、硬件特征分析和性能指标采集。模型结构分析主要关注计算图拓扑和层间依赖关系，硬件特征分析则评估设备计算能力、内存容量和通信带宽，性能指标采集则记录计算时间、通信开销和内存占用等关键指标。

规划器基于分析结果生成最优并行策略，其核心功能包括流水线划分、设备分配和调度策略生成。流水线划分将模型按计算特征划分为多个阶段，设备分配则根据硬件拓扑优化设备映射，调度策略则生成最优的微批次调度方案。运行时系统作为执行并行

训练的核心组件，实现了早反向调度、动态负载均衡和通信优化等功能。早反向调度通过提前释放激活内存提高效率，动态负载均衡则自适应调整设备工作负载，通信优化则通过重叠计算与通信来减少等待时间。

4 实验评估

4.1 内存优化效果

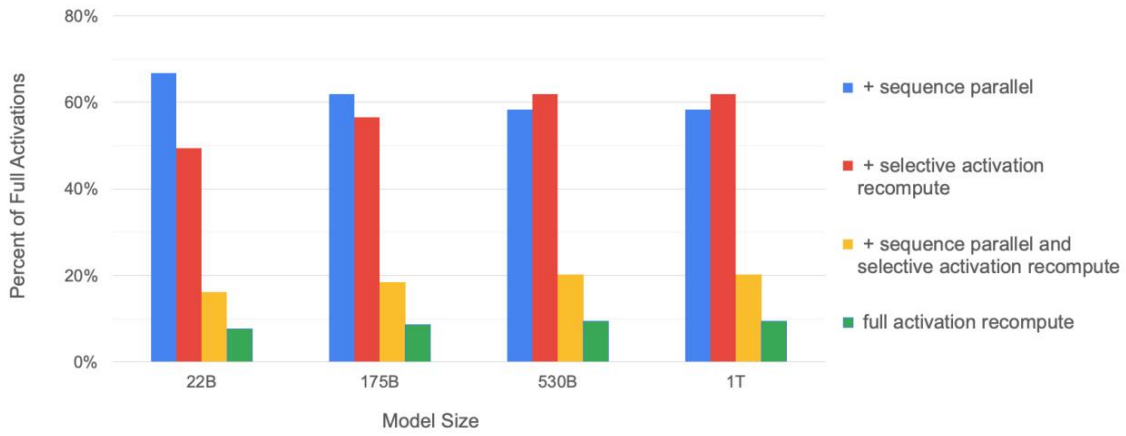

图 4: 激活量占比图

序列并行和选择性激活重计算技术显著降低了模型训练的内存占用。在 530B 参数的 GPT-3 风格模型上，通过结合序列并行和选择性重计算，激活内存减少了 5 倍，从 80GB 降至 16GB。内存优化效果在万亿级参数模型上更为显著，例如在 1T 参数模型上，相比全激活重计算方法，内存占用降低了 75%，同时模型浮点运算利用率（MFU）从 42.1% 提升至 54.2%。

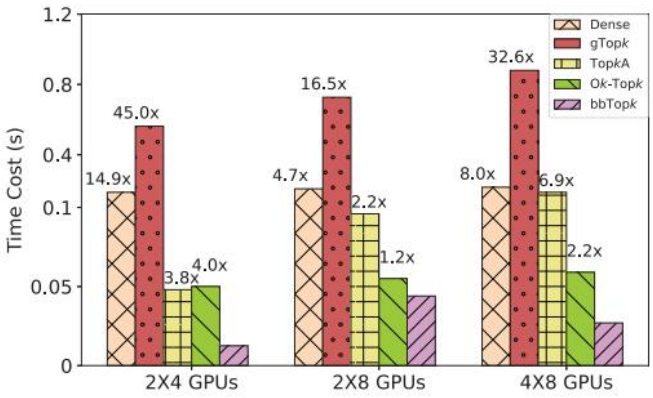

图 5: 不同稀疏全归约方法在 VGG-16 上的性能表现

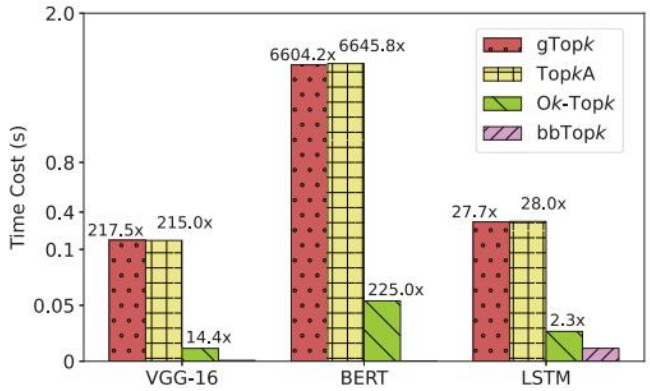

图 6: 不同梯度稀疏化方法的性能表现

4.2 通信效率提升

在通信优化方面，bbTopk 系统通过分块 Top-k 稀疏化和带宽感知算法，在异构网络环境下取得了显著效果。在 VGG-16 模型上，相比 Ok-Topk 方法，bbTopk 在 2x4 和 2x8 集群上分别实现了 4.7 倍和 16.5 倍的通信效率提升。分块 Top-k 稀疏化方法在 BERT 模型上实现了 6604 倍的稀疏化速度提升，同时保持了模型精度。

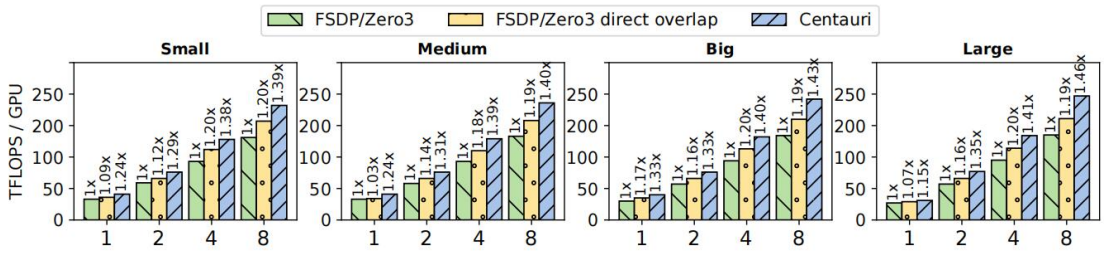

图 7: 运行 FSDP/Zero3 任务的性能对比

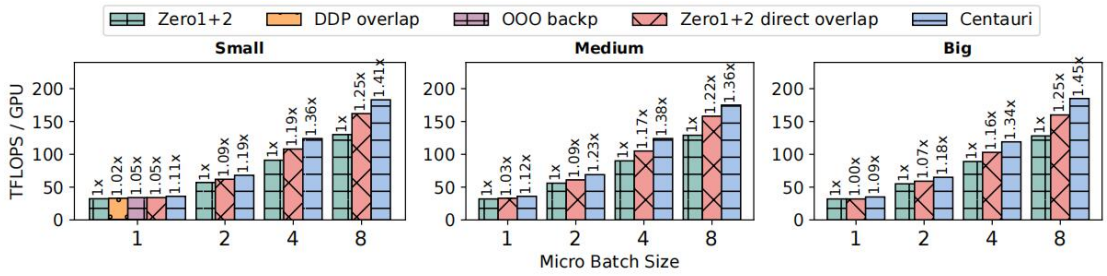

图 8: 运行 DP 任务的性能对比

4.3 训练吞吐量提升

Centauri 系统通过通信分区和分层调度，在混合并行训练场景下实现了显著的性能提升。在 FSDP 和 DP 任务中，相比基线方法分别提升了 1.45 倍和 1.29 倍的吞吐量。在 FSDP+DP 和 TP+PP+DP 混合并行场景下，吞吐量分别提升了 1.47 倍和 1.27 倍。

4.4 调度策略效果

DAPPLE 在同步训练场景下相比 PipeDream 规划器实现了最高 3.23 倍的加速，运行时相比 GPipe 提升了 1.6 倍的吞吐量，同时节省了 12% 的内存消耗，有效解决了流水线并行中的内存和效率平衡问题。

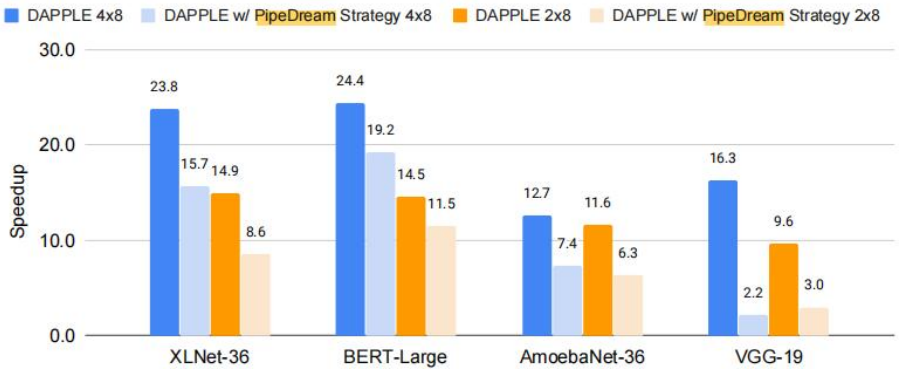

图 9: 与 PipeDream 的性能对比

5 总结与展望

本文综述了分布式深度学习训练中的通信架构优化技术，重点分析了混合并行训练架构、带宽感知的稀疏通信优化以及流水线数据并行优化等关键技术。这些技术通过多维度并行策略的协同、分块稀疏化与带宽感知通信的结合、以及层次化调度方案的实现，有效解决了大规模模型训练中的通信瓶颈问题。未来研究方向包括：1）探索更高效的通信原语设计；2）研究自适应并行策略生成；3）优化异构硬件环境下的通信效率。

1024040904 计算机科学与技术专业 吴朝辉

成文日期：2025 年 6 月 14 日

参考文献

[1] Chang Chen, Min Li, and Chao Yang. ``bbTopk: Bandwidth-Aware Sparse Allreduce with Blocked Sparsification for Efficient Distributed Training". In: *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. 2023, pp. 1–11.

[2] Chang Chen et al. ``Centauri: Enabling Efficient Scheduling for Communication-Computation Overlap in Large Model Training via Communication Partitioning". In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 178–191.

[3] Shiqing Fan et al. ``DAPPLE: a pipelined data parallel approach for training large models". In: PPoPP '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 431–445.

[4] Vijay Anand Korthikanti et al. ``Reducing Activation Recomputation in Large Transformer Models". In: *Proceedings of Machine Learning and Systems*. Ed. by D. Song, M. Carbin, and T. Chen. Vol. 5. Curan, 2023, pp. 341–353. URL: https://proceedings.mlsys.org/paper_files/paper/2023/file/80083951326cf5b35e5100260d64ed81-Paper-mlsys2023.pdf.