

DRoPipe: 基于双维数据并行和轮询多层流水线并行的分布式训练策略

周光耀¹⁾ 田文洪²⁾ 张家树¹⁾ 薛瑞尼³⁾ 李天瑞¹⁾

¹⁾(西南交通大学计算机与人工智能学院 成都 611756)

²⁾(电子科技大学信息与软件工程学院 成都 610054)

³⁾(电子科技大学计算机科学与工程学院 成都 610054)

摘 要 分布式并行训练是建立大规模神经网络模型的基础技术。现有的分布式并行训练和推理主要基于单层流水线并行模式,即每个流水线阶段需独立占用一个或一组算力节点。然而,在实际工程应用和理论研究中,单层流水线并行模式需要在一个节点中布置多个连续的神经网络层,导致流水线气泡时间多,限制了并行训练速度的改进。针对上述问题,本文提出了基于双维度数据划分和轮询多层流水线并行模式的分布式并行训练策略DRoPipe。在DRoPipe中,新型轮询多层流水线并行模式允许在单节点中部署多个离散流水线阶段,可降低每个算力节点单次计算或通信需要处理的神经网络层数和数据量,从而减小了算力节点之间的等待时间,加速了整个并行训练过程。为了实现DRoPipe中数据和流水线阶段的优化划分,本文推导了轮询流水线并行策略DRoPipe的时间损失理论模型,提出了基于隔板并行邻域搜索的多维度并行模式交叉优化方法。本文在多种真实算力设备集群中对典型大模型GPT和Llama系列的并行训练任务进行了实验。实验结果表明,相比于现有先进的单层流水线单维数据划分并行策略、基于1F1B的单层流水线并行策略、单层流水线非均等数据并行策略,DRoPipe分别平均提升98.0%、74.9%、51.3%并行训练速度;相比于基于本文新型并行模式的消融并行策略-单层流水线双维数据并行策略和轮询流水线单维数据并行策略,DRoPipe分别提速60.0%和7.5%;验证了DRoPipe的有效性和先进性。

关键词 分布式并行训练;大规模神经网络;轮询流水线并行;双维度数据划分;交叉优化;隔板邻域搜索

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2026.01770

DRoPipe: Dual Dimensional Data Parallelism and Round-Robin Multi-Layer Pipeline Parallelism Based Distributed Training Strategy

ZHOU Guang-Yao¹⁾ TIAN Wen-Hong²⁾ ZHANG Jia-Shu¹⁾ XUE Rui-Ni³⁾ LI Tian-Rui¹⁾

¹⁾(School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756)

²⁾(School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054)

³⁾(School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054)

Abstract Distributed parallel training is the fundamental technology to build large-scale neural network models. The existing distributed parallel training and reasoning are mainly based on the single-layer pipeline parallelism, i. e., each pipeline stage needs to independently occupy one or a group of computing-power-nodes. However, in practical engineering applications and theoretical research, the single-layer pipeline parallelism requires the arrangement of multiple consecutive

收稿日期:2025-10-07;在线发布日期:2026-04-13。本课题得到国家自然科学基金项目(面上项目No. 62471411、外国青年学者基金No. W2433163、铁路联合基金重点支持项目No. U2468207)、四川省科技厅国际创新合作项目(No. 2024YFH0317)、中央高校基本科研业务费专项资金项目(Nos. 2682025CX127, 2682026ZT007)和华为技术合作项目(No. R111626H01006)资助。周光耀(通信作者),博士,讲师,中国计算机学会(CCF)会员,主要研究领域为高性能分布式并行计算、优化算法、深度学习理论。E-mail: guangyao_zhou@swjtu.edu.cn。田文洪,博士,教授,中国计算机学会(CCF)杰出会员,主要研究领域为云计算、大模型训练推理优化与调度。张家树,博士,教授,主要研究领域为分布式与并行信号处理、人工智能及应用、通信安全。薛瑞尼,博士,副教授,主要研究领域为分布式系统、高性能计算、图计算。李天瑞,博士,教授,中国计算机学会(CCF)杰出会员,主要研究领域为云计算、大数据、人工智能、粒计算。

neural network layers in one node, resulting in a large amount of pipeline bubble time restricting the improvement of training speed. Focusing on the above issues, this paper proposes DRoPipe, a novel distributed parallel training strategy based on dual-dimensional data partitioning and round robin multi-layer pipeline parallelism. In DRoPipe, the novel round robin multi-layer pipeline parallelism enables the deployment of multiple discrete pipeline stages within a single node, which reduces the number of neural network layers and data volume that each computing node must process in once computation or communication, hence reducing the waiting times between computational nodes and accelerating the entire parallel training process. To achieve the optimal partitioning of data and pipeline stages in DRoPipe, this paper further derives the theoretical model of time cost and proposes the bulkhead parallel neighborhood search based cross-optimization method for the multi-dimensional parallelism. This paper conducts experiments on the parallel training tasks of typical large models GPT and Llama series in multiple real computing-power-device clusters. The experimental results showcase that: compared with the existing advanced single-layer pipeline parallel strategy with single-dimensional data partitioning, 1F1B-based single-layer pipeline parallel strategy and unequal data partitioning-based single-layer pipeline parallel strategy, DRoPipe increases parallel training speed by 98.0%, 74.9% and 51.3% respectively; compared with other ablation parallel strategies based on our proposed novel parallelism, including single-layer pipeline parallel strategy with dual-dimensional data parallelism and round robin pipeline strategy with single-dimensional data parallelism, DRoPipe increases speed respectively by 60.0% and 7.5%, demonstrating DRoPipe's effectiveness and advancement.

Keywords distributed parallel training; large-scale neural networks; round robin pipeline parallelism; dual-dimensional data partitioning; cross optimization; bulkhead neighborhood search

1 引言

随着人工智能技术的发展, 神经网络已广泛应用于计算机视觉处理^[1-2]、自然语言处理^[3]等应用场景。现有神经网络层类型包括卷积神经网络^[4]、Transformer 网络^[5-6]、图神经网络^[7-8]等, 其中以 Transformer 网络层为基础网络层的大规模语言模型 (large-scale language model, LLM) 是当前人工智能领域中最受关注的技术之一。由于参数规模巨大, 大规模深度神经网络模型的训练依赖于如图形处理单元 (graphics processing unit, GPU) 等大量算力节点的集群分布式并行计算。与此同时, 算力集群中并行训练 workflow 优化亦成为分布式并行计算和人工智能领域的研究热点^[9-10]。

由于集群总能耗等于单个节点平均功率、节点数量、总运行时间三者之积, 因此大量节点数量和时间将导致大量的能耗。另外, 随着电子元器件持续工作时间增加将带来整体温度的提升, 导致热损耗增加, 从而降低电能的有效转换率、也降低了算力资

源的有效利用率。为了提升分布式并行训练效率, 现有研究建立了多种并行策略。文献[11]在流水线模型并行的基础上引入基于微批次的数据并行, 提出了 GPipe, 改进了原有单数据批次流水线并行的效率。在 GPipe 的基础上, 文献[6]提出针对 LLM 的词条级数据并行模式及其并行策略 TeraPipe, 有效提升了以词条数据作为主要输入数据的 LLM 的并行训练效率。随着针对深度学习模型的分布式并行计算的研究深入, 相关研究领域逐步形成以数据并行、张量模型并行、流水线模型并行为基础并行模式的多维度并行策略: HIPPIE^[12]、NASPIPE^[13]、FOLD3D^[14]、DAPPLE^[15]、UMPIPE^[16]等。这些并行策略在很大程度上改进了大规模深度学习模型的并行执行效率、优化了显存占用、降低了 GPU 集群能耗, 推进了大规模模型的发展。

大规模深度学习模型智能策略和分布式并行技术的研究中, 研究人员需要对智能策略、分布式并行技术、优化调度方法等进行多次实验验证^[17]。在实际工程应用和理论研究中, 普遍存在可供长时间连续运行的高性能算力集群节点数不足、研发测试周

期紧张等情况。现有分布式并行训练融合了数据并行、张量模型并行和流水线并行,在实际应用中取得了很好的并行效率,大大推进了大模型的发展^[9,11,13]。但仍存在以下问题亟须解决:(1)在现有流水线并行模式中,所有的流水线阶段(virtual pipeline stage)需要独立占用一个或一组节点;流水线阶段的划分受限于算力设备节点数量,一个算力节点上需要部署连续的神经网络层,导致更多的流水线气泡时间,甚至显存不足。例如,当有100层神经网络、10个算力节点时,若按照均衡分配,现有流水线并行模式要求每个算力节点至少包含10层连续排序的神经网络层,此时,每个算力节点都必须等待前序节点完成10层连续神经网络层的计算和通信过程后,才可开启后序过程;且每个算力节点中需要同时保存10层神经网络的中间数据,对显存容量提出了更高要求。(2)现有数据并行模式仅在微批次维度或词条维度进行单维度数据划分,未考虑同时进行多维度数据划分,这限制了单次计算或通信过程的显存和带宽占用下限。例如,当一个小批次包含128条数据序列,每个序列包含128条词条数据时,若采用单维度数据划分,则在微批次维度数据或在微词条序列维度最小数据颗粒度都仅能划分为 1×128 ,此时各层流水线阶段的计算和通信过程之间重叠(overlap)颗粒度最小为 1×128 ,且每次运算或通信过程中最小需要处理 1×128 对应的中间数据。

鉴于此,结合Megatron-LM^[5]中多虚拟流水线(multiple pipeline stages or multiple chunks)思路,本文针对全正全反(all forward all backward, AFAB)时序模式下的流水线训练提出基于轮询多层流水线并行模式的算力节点复用;并进一步结合多维度数据划分,构建新型分布式训练策略,提出了相关理论模型和优化算法。本文主要贡献如下:

(1)轮询多层流水线模型并行(round robin multi-layer pipeline model parallelism, RPMP):针对现有先进单层流水线并行模式中一个流水线阶段必须独占一个算力节点的局限性,本文提出了轮询多层流水线并行模式,即按照固定的算力节点顺序,依次循环多轮地将流水线阶段分配给各个算力节点的并行模式。轮询多层流水线并行模式允许同一个算力节点可以处理多个不同的、非连续的流水线阶段的计算和通信过程,因此增加了流水线划分的灵活性,使得流水线阶段数可以超过算力节点数,在降低峰值显存的同时提升了并行速度。

(2)DRoPipe并行训练策略(dual-dimensional data partitioning and round robin multi-layer pipeline parallelism-based distributed training strategy):本文提出了基于双维度数据并行和轮询多层流水线并行的新型并行策略DRoPipe,通过理论推导建立了DRoPipe的时间损失的解析表达式。为了进一步加速理论时间损失计算过程,本文推导并建立了多方案组流水线层间递推公式的快速并行计算模型,允许并行计算单元同时计算多个并行方案对应的时间损失,显著加速了DRoPipe并行策略中多维度并行方案的评估和优化过程。

(3)基于隔板并行邻域搜索的多维度并行策略交叉优化方法:在DRoPipe的理论损失模型的基础上,本文进一步设计了隔板并行邻域搜索以求解轮询多层流水线并行的优化方案。结合隔板并行邻域搜索和基于基础时间函数拐点的快速数据划分、张量划分方法,本文建立了交叉优化方法,实现了对DRoPipe中多维度并行策略的自动优化搜索。

(4)本文从多个角度对DRoPipe并行训练策略及其理论损失计算模型和优化方法的性能进行了实验验证。结果表明了本文DRoPipe并行策略在相同算力集群配置下可以提升分布式并行速度,验证了DRoPipe并行策略的有效性和先进性。

本文后续结构组织如下:本文第2节回顾了相关工作现状;第3节详细介绍了新型并行训练策略DRoPipe的基础并行模式和结构建模;第4节呈现了DRoPipe并行策略的理论时间损失的解析式和快速递推计算模型;第5节提出了针对DRoPipe多维度并行模式自动搜索的优化算法,即基于隔板并行邻域搜索的多维度交叉优化方法;第6节开展了仿真数据驱动和真实算力集群环境驱动的实验评估,并对DRoPipe并行策略及其理论损失计算模型和优化算法进行综合验证和分析;第7节对本文进行总结并梳理未来研究计划。

2 研究现状

2.1 分布式并行模式

数据并行(分割数据样本)、张量模型并行(分割模型参数、需复制数据,神经网络层内并行)、流水线模型并行(分割模型层,神经网络层间并行)是目前深度学习模型的三种基础并行模式^[5,9,18],其并行过程示意图如图1所示。

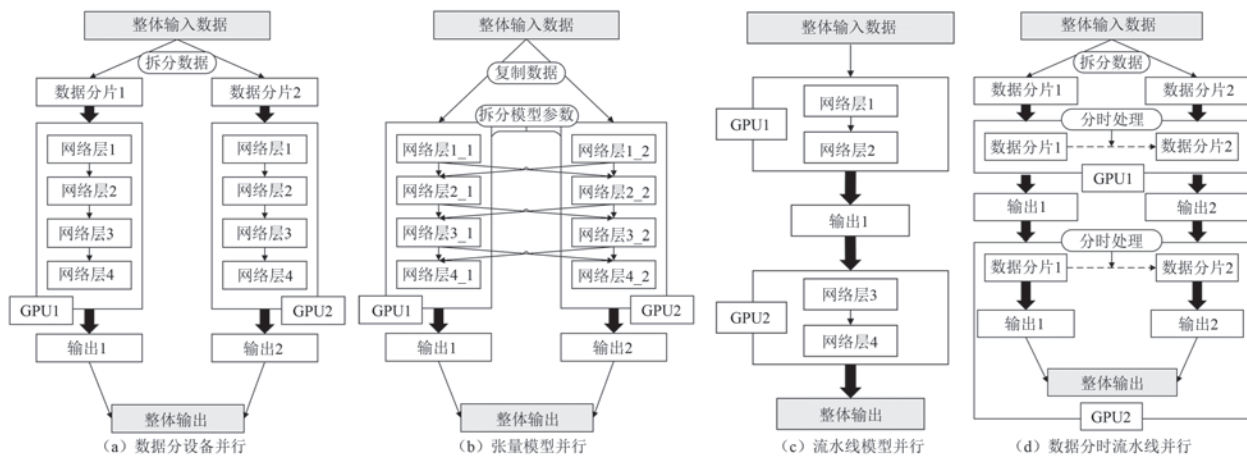


图1 现有基础并行模式的过程示意图

随着分布式并行训练的发展,分布式并行策略也逐渐从包含一个或者两个并行模式的策略发展为包含三个甚至更多并行模式的混合多维度并行训练策略。在此基础上,亦衍生出多种其他并行模式,如算子并行、优化器并行、非均等数据并行、双流水线并行、动态端到端流水线并行^[16,19-21]等。

2.2 分布式并行训练策略

针对分布式并行训练策略,Yanping Huang等人将基于微批次处理的数据并行模式与流水线并行模式结合,提出了GPipe并行训练策略^[11]。在GPipe的基础上,Deepak Narayanan等人引入了1F1B(one forward one backward)训练模式并提出了PipeDream并行策略^[22]。DeepSpeed^[23]和Megatron-LM^[5]等经典并行训练策略融合了张量模型并行和流水线并行,同样受到了广泛的关注和应用。Zhuohan Li等学者针对大语言Transformer神经网络提出了基于词级别(Token-level)的数据并行模式的TeraPipe并行策略^[6]。Zixuan Wang等学者针对利用缓存相干分解存储系统实现高效的大规模深度学习训练,并提出了COARSE,以允许低延迟和并行访问工作GPU间共享的训练数据和模型参数^[24]。Lianmin Zheng等学者将多维度并行策略按照内部并行和外部并行进行分级划分,并提出了Alpa策略以对分布式并行训练模式进行自动划分^[25]。Guangyao Zhou等改善了分布式训练中数据划分的颗粒度,提出了基于非均等微批次数据并行的流水线并行策略UMPIPE^[16]。Shiqing Fan等学者将1F1B与数据并行和流水线模型并行进行结合,提出了DAPPLE,以提升并行训练速度^[15]。Dongsheng Li等学者结合数据并行、张量模型并行、流水线模型并行模式和1F1B更新模式,提出了一种显存高效的混合并行策略HIPPIE,其中提出一种新型的混

合并行调度器来优化模型训练过程^[12]。Zhiquan Lai等使用自动模型分割器自动部署3D并行模式并提出了一种具有高资源利用率的自动化训练框架Merak^[26]。DeepSeek-AI团队设计了DualPipe策略,大幅加速了训练过程并降低了训练成本^[20]。

2.3 分布式并行训练策略

对于并行训练方案的优化方法主要包含数学规划方法和智能优化算法^[18,27-29]。

现有研究中用于求解并行训练方案的数学规划方法主要包括动态规划和线性规划。动态规划是一种可行且广泛使用的可求解数据并行或模型并行最优划分方案的方法^[27]。针对多维混合并行训练,Zhenbo Sun等学者提出了AdaPipe,其中提出一种结合自适应重构和划分的两级动态规划算法对流水线阶段方案进行搜索^[28]。针对异构分布式设备节点中的多维混合并行策略,Yuhao Chen等学者采用了基于动态规划的动态重划分并提出FTPipeHD分布式训练方法。其他一些使用动态规划的研究实例包括:PipeDream^[22]、Dapple^[15]、Terapipe^[6]、EffTra^[30]、PaSE^[31]、Pipepar^[32]等。线性规划也是求解并行训练优化训练方案的常用方法,其应用实例包括:NetPlacer^[33]、HGP4CNN^[4]、DPDA^[34]、IceFrog^[35]等。

分布式并行训练策略的复杂度的增加对智能优化算法提出了需求。徐金龙等学者结合约束关系搜索算法、计算图分片算法提出了同步优化算法,从而对考虑计算和通信的3D混合并行训练策略的优化方案进行搜索^[18]。Changyao Lin等学者针对异步深度学习多任务推理系统提出了基于任务的算子并行模式TOP,利用多智能体强化学习算法MADDPG协同优化任务启动器和模型调度器,并提出一种融合图神经网络的策略梯度算法对流水线阶段进行划

分^[36]。Guangyao Zhou 等针对同时考虑计算和通信时间的混合并行训练策略提出了基于改进多维度二分法的交叉搜索算法以同时求解数据划分和流水线划分的并行方案^[37]。Yushen Bi 等学者针对分布式异构系统中的深度学习并行训练 workflow 提出了一种自适应冲突感知的调度算法 AISAW, 降低了整个异构系统的完工时间^[38]。Xiaofeng Zou 等学者考虑了深度神经网络(deep neural network, DNN)模型的执行时间、高可扩展性和内存限制, 将 DNN 的计算图转化为有向无环图(directed acyclic graph, DAG), 并提出异构调度方法 PSHP 和多层级 DAG 划分算法 LMDP 来确定并行训练策略^[29]。其他算法还包括: 图划分算法^[7-8]、近最优层划分局部搜索^[39]等。

根据研究现状, 现有并行训练策略推进了大模型的发展。然而, 相应的并行策略中流水线并行模式均为单层流水线并行, 即每个算力节点上仅有一个流水线阶段, 这要求同一个节点上所部署的神经网络层一定是连续的网络层。另外, 现有研究中数据划分维度仅有一个维度, 微批次维度或微词条序列维度。与现有研究不同, 本文结合轮询调度原则提出了轮询流水线模型并行模式(亦可称为多层流水线并行模式)和双维度数据并行模式。轮询流水线模型并行模式中同一个节点上可以包含多个流水线阶段, 实现了神经网络层的离散流水线部署。双维度数据并行模式允许同时对微批次维度和微词条序列维度进行数据划分, 进一步改进了数据划分的颗粒度。结合轮询流水线并行和双维度数据并行, 本文构建了完善的并行策略 DRoPipe 及其理论损失计算模型和优化方法, 在相同算力节点下, 实现对深度神经网络分布式并行训练性能的改进。

3 DRoPipe 的基础并行模式

为了提升深度学习模型的分布式并行训练性能, 本文提出基于轮询多层流水线并行和双维度数据并行的新型并行策略 DRoPipe。针对 DRoPipe 并行策略, 本文理论推导了其时间损失的解析表达式和快速计算模型, 在此基础上进一步设计了基于隔板并行邻域搜索的多维度并行模式交叉优化方法, 以实现 DRoPipe 中优化并行方案的自动搜索。

本节主要介绍组成 DRoPipe 并行策略的轮询多层流水线并行模式和双维度数据并行模式。

为了便于呈现并行训练策略的结构建模和理论推导, 表 1 列出了本文使用的主要符号及其说明。

表 1 符号及其说明。

符号	说明与描述
n	算力节点数
σ	神经网络层数
v	轮询流水线的轮询层数
l	流水线阶段数
P	一个小批次中数据序列的数量
p	微批次数
Q	一个序列中词条数量
q	微词条序列数
L_i	第 i 层神经网络
C_i	第 i 个算力节点中所有神经网络层组成的集合
S_i	第 i 个流水线阶段中所有神经网络层组成的集合
α_i	S_i 中神经网络层序号最大值
β_i	S_i 中神经网络层序号最小值
λ	所有流水线阶段网络层最大值组成的向量
$d_{j,k}$	小批次中第 j 个序列的第 k 个数据元素或词条
D	小批次数据矩阵 $D = \{d_{j,k}\}_{P \times Q}$
ρ	子数据块数量 $\rho = pq$
ψ_{ik}	第 i 个流水线过程对于第 k 个子数据块开始时间
H_i	第 i 个流水线过程处理一个子数据块的耗时
X_i	第 i 个流水线过程中最后一个子数据块结束时间

3.1 轮询多层流水线并行模式

为了便于阐述轮询多层流水线模型并行模式(RPMP)及其与现有单层流水线并行的区别, 以下将对一个示例进行说明。图 2 所示为现有先进的单层流水线并行模式和本文所提轮询流水线并行模式的正向传播过程结构对比图(反向传播过程结构对比亦类似), 该示例中, 含有 12 层同构神经网络层, 4 个同构算力节点, 子数据块划分数为 4。

如图 2(a)所示, 现有流水线并行模式属于单层连续流水线结构, 其中每个算力节点仅可容纳一个流水线阶段, 这意味着分配给同一个算力节点中的神经网络层一定是连续的。此时, 在图 2(a)所示场景中, 最多只能划分为 4 个流水线阶段, 每个流水线阶段包含 3 层神经网络(每层神经网络处理一个子数据块的耗时设为 f)。则此时, 现有先进的单层流水线并行模式下的总耗时(完工时间)为 $21f$ 。

轮询多层流水线并行模式的示意图如图 2(b)所示。轮询流水线并行模式允许算力设备复用, 算力集群依次循环分配流水线阶段, 提升了给定算力节点数的情况下流水线阶段数的上限。引入轮询流水线并行模式后, 允许非连续的神经网络层分配于同一算力节点, 因此轮询流水线又属于多层离散流水线结构。如图 2(b)所示, 虽然只有 4 个算力节点,

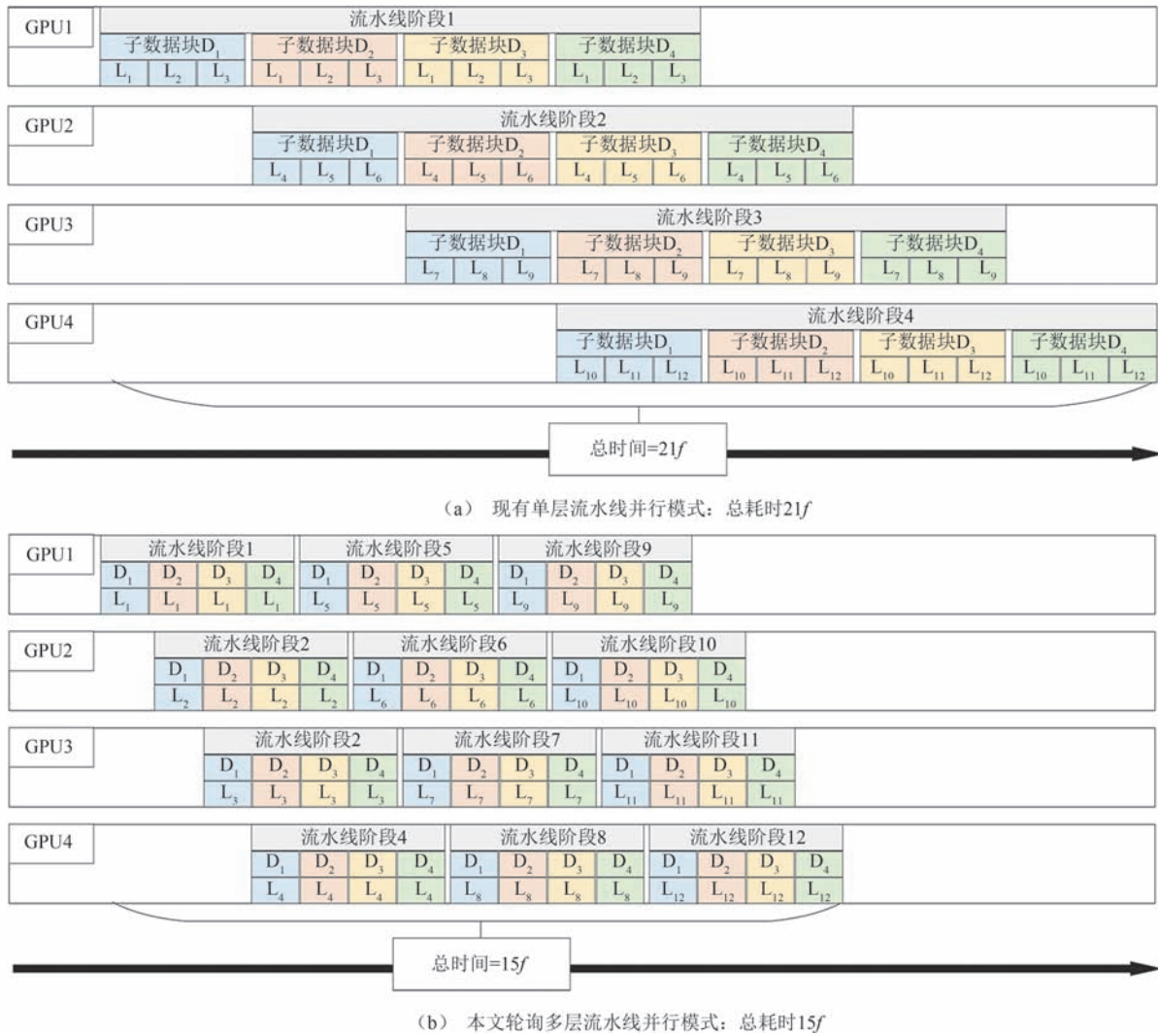


图2 两种流水线并行模式正向传播过程示例对比图(包含12层同构神经网络层,4个同构算力节点,子数据块划分数为4,其中 D_i 表示第 i 个子数据块, L_i 表示第 i 层神经网络,每个算力节点处理一层神经网络中一个子数据块的正向传播过程耗时记为 f ,同一子数据块在不同网络层的正向传播过程用同样颜色表示)

但轮询流水线并行模式允许将神经网络按照网络层数划分为12个流水线阶段,其中第 i 层、 $i+4$ 层、 $i+8$ 层神经网络处于同一节点($i \in \{1, 2, 3, 4\}$)。此时,轮询流水线并行模式下的总耗时降低为 $15f$,相比于单层流水线并行模式提升了40%并行处理速度;在有效计算总时长固定的前提下,平均每个算力设备的利用率为 $4/5$,相比于单层流水线并行模式的 $4/7$,提升幅度为40%。

图2所示实例说明了轮询多层流水线模式可以提升并行处理速度。此时,单层流水线并行模式和轮询多层流水线并行模式中各个算力设备累积显存相同,均为3层神经网络层的中间数据所需显存;单流水线并行模式的峰值显存需求亦为3层神经网络,而轮询流水线并行模式的瞬时峰值显存得以改

进,仅为1层神经网络层的中间数据所需显存。

为了进一步从理论上分析轮询流水线并行的优势,令正向传播并行处理过程中包含 n 个算力节点、 m 层神经网络,其中 $m \gg n$ 。为了简化理论分析,同样假设数据划分为等于节点数 n 。由于 $m \gg n$,可近似看作 m 整除 n 。单层流水线并行模式下仅能划分为 n 个流水线阶段,每个流水线阶段包含 m/n 层连续的神经网络层,此时各流水线阶段处理一个子数据块耗时为 mf/n (其中每层神经网络处理一个子数据块耗时记为 f),其正向传播过程总耗时为 $(2n-1)mf/n$,各节点平均气泡时间为 $(n-1)mf/n$ 。轮询流水线并行模式允许将神经网络划分为 m 个流水线阶段,其中第 i 层与第 $i+n$ 层(每间隔 n 层)处于同一算力节点,其中 $1 \leq i \leq m-n$,此时每个节点

上包含 $v = m/n$ 个非连续的流水线阶段, 每个流水线阶段处理一个子数据块的时间为 f , 其正向传播过程总耗时为 $(m + n - 1)f$, 各节点平均气泡时间为 $(n - 1)f$, 相比于单层流水线并行在处理速度方面提升了 $\frac{nm - m - n^2 + 1}{nm + n^2 - 1} \rightarrow \frac{n - 1}{n}$ 倍、气泡时间减小为单层流水线并行的 $n/m = 1/v$ 。此时, 轮询流水线并行的峰值显存仅为 1 层神经网络中间数据量, 即说明了轮询流水线并行可降低峰值显存占用, 同时亦可提升并行速度, 降低气泡时间。

在相同并行处理速度下对单层流水线并行和轮询流水线并行进行对比。同样针对 m 层神经网络, 数据划分为 n 。假设算力节点数为 k , 则根据文献[37]中的理论推导公式可得单层流水线并行模式的处理耗时为 $(n + k - 1)mf/k$ 。

针对上述场景(其中 $m \gg n$), 单层流水线并行模式需要 m 个算力节点才能达到轮询流水线并行模式在 n 个算力节点下的并行处理速度。因此, 在相同并行处理速度的需求下, 轮询流水线并行模式大大降低了算力节点数量的需求, 即通过少量算力节点(n 个)完成了单层流水线并行模式在大量算力节点(m 个, 其中 $m \gg n$)中相同速度的并行处理过程。这是因为: 单层流水线并行中, 各个算力设备的空闲时间(气泡时间, bubbles)未被充分利用, 每个算力设备的平均利用率仅为 $n/(m + n - 1)$; 而轮询流水线并行中, 将每个设备的平均利用率提升到 $m/(m + n - 1) \rightarrow 1$, 是单层流水线并行的 m/n 倍。

设深度神经网络含有 σ 层神经网络, 依次排序组成序列 $\langle L_1, L_2, \dots, L_\sigma \rangle$; 算力节点数为 n , 第 i 个算力节点中所有神经网络层组成的集合记为 C_i ; 流水线阶段数为 l , 第 i 个流水线阶段所有神经网络层组成的集合记为 S_i , 其中网络层最大序号记为 α_i , 最小序号记为 β_i 。由于流水线阶段包含多个连续网络层, 因此有 $S_i = \{L_{\alpha_{i-1}+1}, \dots, L_{\alpha_i}\}$ 。

(1) 单层流水线并行模式: 对于任意流水线阶段 $S_i (\forall i \in \{1, 2, \dots, l\})$, 必存在至少一个算力节点 $C_j (\exists j \in \{1, 2, \dots, n\})$ 使得 $C_j = S_i$; 且对于 $\forall i \in \{1, 2, \dots, l\}$ 和 $\forall j \in \{1, 2, \dots, n\}$, 要么 $C_j \cap S_i = \emptyset$, 要么 $C_j = S_i$, 其中 $\emptyset$ 表示空集。

(2) 轮询多层流水线并行模式: 存在一个固定的整数 N 使得对于任意流水线阶段 S_i , 若 S 所在算力节点为 C_j 即 $S_i \subset C_j$, 则对于任意 $k \in \mathbb{N}$ 有 $S_{i+kN} \subset C_j$, 其中 $i + kN \leq l$ 。

其中 $\lambda = \langle \alpha_1, \alpha_2, \dots, \alpha_l \rangle$ 可表示流水线阶段划分决策变量。上述定义严格规定了单层流水线并行模式中每个算力节点上只能有一个流水线阶段; 轮询流水线并行模式中每个算力节点可以有多个流水线阶段。而引入张量模型并行(tensor model parallelism, TMP)后, 一个流水线阶段可以同时部署于多个算力设备上, 直到同一流水线阶段中所有张量分量处理过程结束后, 才可开启后序流水线阶段。其并行结构如图 3 所示。

将每个算力节点组看成一个算力整体集合, 此时单算力节点的轮询多层流水线并行所满足的理论损失模型将同样适用于算力节点组的轮询流水线并行。轮询流水线并行模式, 可以与数据并行和张量模型并行相结合, 构建多维并行策略。

在轮询流水线并行中, 允许前序轮询层计算完毕后结束相关进程以释放对于显存的占用。而对于并行训练, 虽然需要保留正向传播过程中的中间数据(checkpoints 数据等)以支撑梯度计算, 但仍然可以通过将显存数据暂存于内存或磁盘存储的方式临时释放显存占用, 即激活检查点重计算技术(activation re-materialization)。然而, 在现有单层流水线并行中, 同一算力节点需要处理连续多层神经网络, 其连续性的要求没有为显存卸载留下足够时间, 使得算力设备需要推迟后序过程来释放前序神经网络层的显存占用。因此, 在并行推理和并行训练中, 轮询流水线并行均可有效降低峰值显存占用量, 从而降低对于算力节点数的需求。

3.2 双维度数据并行模式

在轮询多层流水线并行的基础上, 本节进一步探讨数据并行模式的改进方式, 即双维度数据并行模式(dual-dimensional data parallelism, DDP)。

现有先进并行策略中, 包含两种维度的数据划分: 微批次级别的数据划分、词条级别(微序列级别)的数据划分。为了进一步改进多维度分布式并行策略, 增加并行策略的优化颗粒度, 本文融合上述两种数据并行维度, 提出了双维度数据并行模式。为了更直观地呈现不同数据并行模式的差异, 图 4 绘制了单维度微批次数据划分、单维度词条数据划分和双维度数据划分的子数据块划分示意图。

在深度神经网络(尤其是 LLM)的分布式并行训练中, 由小批次输入数据组成的矩阵可表示为 $D = \{d_{j,k}\}_{P \times Q}$, 其中 $d_{j,k}$ 表示在一个小批次数据中第 j 个数据(或序列)的第 k 个数据元素(或词条数据),

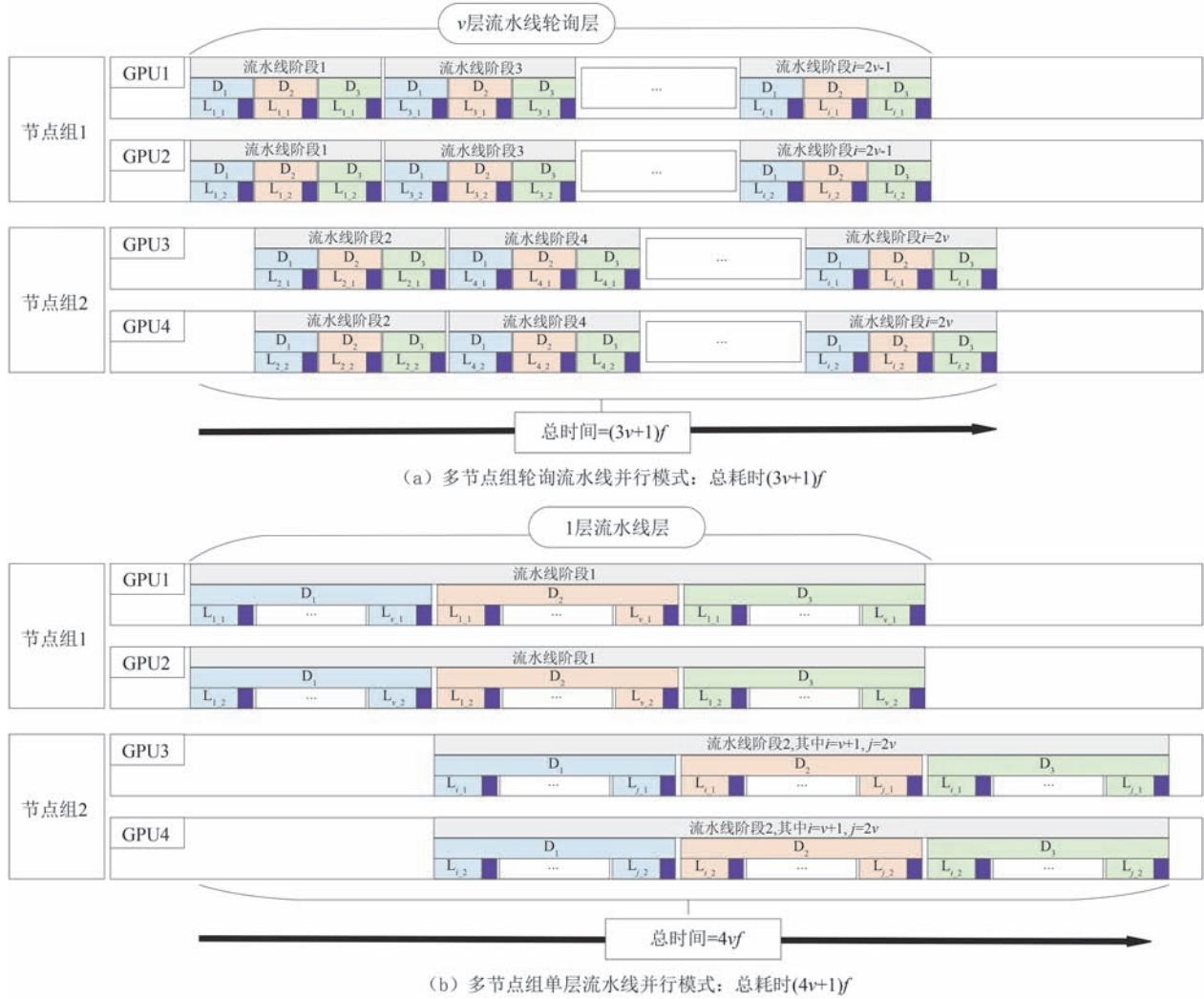


图3 多节点组轮询流水线并行模式及其对比模式(包含 $2v$ 层同构神经网络层, 4 个同构算力节点, 子数据块数为 3, 每个算力节点处理一层神经网络中一个子数据块的正向传播过程的计算加通信耗时记为 $2f$, $L_{1,2}$ 表示第一层神经网络层的第二个张量模型分量, 紫色方块表示输出数据通信时间示意图)

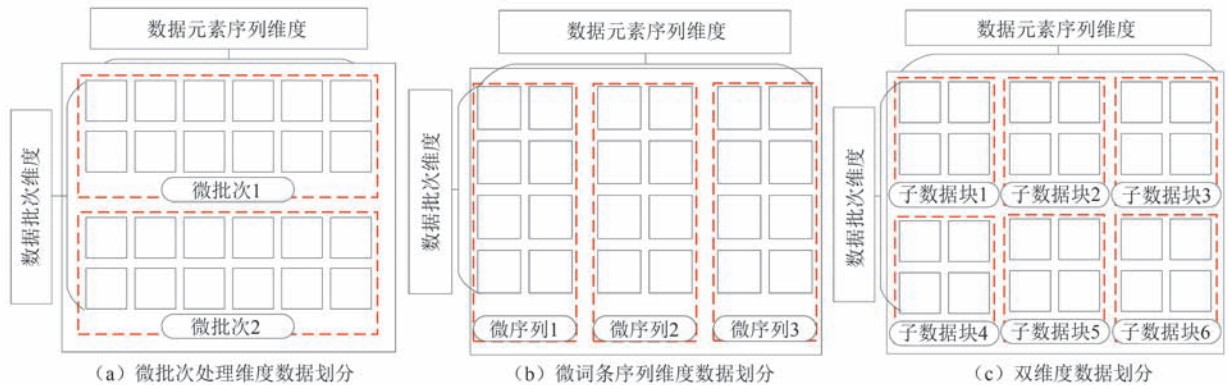


图4 不同数据划分形式示意图

P 表示一个小批次数据中词条序列的数量, Q 表示序列的长度(即一个序列中词条数量)。图4所示三种不同的数据划分模式定义描述如下。

(1) 微批次处理维度数据划分 (microbatch-

dimensional data partitioning, MBD): 在小批次数据矩阵 $D = \{d_{j,k}\}_{P \times Q}$ 的第一个维度(即数据批次维度)将数据划分为多个微批次。如将数据 $D = \{d_{j,k}\}_{P \times Q}$ 划分为 p 个微批次, 将第 i 个微批次数据矩阵可记为

$\{u_{j,k}^{(t)}\}_{\frac{P}{p} \times Q}$, 则 $u_{j,k}^{(i)} = d_{i\frac{P}{p} + j, k}^\circ$ 。

(2) 微词条系列维度或数据元素维度数据划分 (micro-token-dimensional data partitioning, MTD): 将小批次数据矩阵 $D = \{d_{j,k}\}_{P \times Q}$ 的第二个维度 (即词条维度或数据元素维度) 划分为多个微词条序列。如将数据划分为 q 个微序列, 将第 i 个微序列数据可记为 $\{u_{j,k}^{(i)}\}_{P \times \frac{Q}{q}}$, 则有 $u_{j,k}^{(i)} = d_{j, i\frac{Q}{q} + k}^\circ$ 。

(3) 双维度数据划分 (dual-dimensional data partitioning, DD): 在数据矩阵第一维度和第二个维度同时进行划分, 将数据矩阵划分为多个矩阵子块 (称为子数据块)。如将数据划分为 p 个微批次、 q 个微序列, 第 h 个微批次中第 i 个微序列数据可表示为 $\{u_{j,k}^{(h,i)}\}_{\frac{P}{p} \times \frac{Q}{q}}$, 则有 $u_{j,k}^{(h,i)} = d_{h\frac{P}{p} + j, i\frac{Q}{q} + k}^\circ$ 。

双维度数据并行通过扩展数据划分可行集引入了更好的方案, 从而提高并行训练的性能。而双维度数据并行模式的引入, 也允许: 轮询流水线并行模式中, 各个算力节点可根据实际负载均衡需求灵活配置双维度数据块划分。结合轮询流水线并行模式和双维度数据并行模式, 即形成了本文所提出 DRoPipe 并行策略 (其中亦可代入张量模型并行)。

然而, DRoPipe 并行训练策略亦存在一定不足: 轮询多层流水线并行模式通过在同一节点中部署多个非连续的流水线阶段实现了流水线阶段数的扩充, 由此也引入了更多的流水线阶段间通信损失, 在通信拥堵或者带宽较低的场景中反而可能拖慢整个过程。另外, 由于轮询流水线并行模式是本文提出的新型并行模式, 目前尚缺少相应的理论损失模型和优化方法, 需要进一步建立完善的基于轮询流水线并行的多维度并行体系; 需要为每个流水线阶段分配特定的处理进程, 增加了多进程多线程管理复杂度, 同时轮询流水线并行对于显存的优化效果也依赖于进程对显存的释放卸载技术和框架。

4 DRoPipe 的时间损失理论模型

4.1 理论解析表达式

在 DRoPipe 并行策略和理论定义的基础上, 实施该并行策略需要对具体的深度学习实例的并行方案进行优化。其中两个关键的操作是: 对具体并行模式划分方案的评估和对优化方案的自动搜索。建立理论损失模型是实现并行训练方案评估和后续优化的关键条件。为了适应广泛的应用场景, 本节对

DRoPipe 并行策略的理论损失模型的通用表达式进行推导, 其中不仅需要考虑不同流水线阶段的计算过程和通信过程之间的耦合关系, 还需要考虑流水线阶段不同轮询层之间的递推关系。

在 DRoPipe 中, 正向传播和反向传播的流程结构具有相似性和独立性, 其理论损失形式相似。因此, 本节主要呈现正向传播过程的理论公式。

由于轮询流水线并行允许多个流水线阶段放置在同一个算力设备节点上, 因此在理论损失模型的推导过程中, 可将同一个流水线阶段的时间损失作为一个整体。不妨设轮询流水线的轮询层数为 v , 根据节点组数每层轮询的阶段数为 n (算力节点数或算力节点分组数), 则总流水线阶段数为 vn ; 设一个小批次多序列数据按照双维度数据划分为 $\rho = pq$ 个子数据块, 其中微批次数为 p 、微序列数为 q 。将第 i 层轮询层中第 j 个流水线阶段 (对应于第 $n(i-1) + j$ 个流水线阶段) 处理一个子数据块所需计算时间记为 F_{ij}^P , 输出数据传输通信所需时间记为 F_{ij}^M , 其中, 一个流水线阶段的处理时间包含多个张量分量在该流水线节点组上所有算力节点的计算时间和通信时间的完工时间。将流水线阶段计算过程和通信过程合并计数, 则可将第 i 层轮询层第 j 个流水线过程处理第 k 个子数据块的开始时间记为 T_{ijk} , 所需时间记为 F_{ij} , 其中, 若 j 为奇数则表示第 i 层轮询层中第 $(j+1)/2$ 个流水线阶段的计算过程; 若 j 为偶数则表示第 i 层轮询层中第 $j/2$ 个流水线阶段的通信过程。则每一层轮询层 (包含 n 个流水线阶段) 包含了 $2n$ 个合并计数的流水线过程, 其中假设最后一层轮询层的最后一个流水线阶段的通信时间为 0。根据定义设置可知: 第 i 个节点中的流水线阶段组成集合为 $\langle S_i, S_{i+n}, \dots, S_{i+(v-1)n} \rangle$ 。

根据流水线并行模式和双维数据并行模式的流程顺序设定, 第 i 层轮询层中第 j 个合并计数的流水线过程第 k 个子数据块具有两个前序过程, 分别为上一个流水线过程 (第 $j-1$ 个过程) 和第 k 个子数据块的处理过程 (前序流水线过程), 同一个流水线过程 (第 j 个过程) 前一个子数据块 (第 $k-1$ 个数据块) 的处理过程 (前序数据过程), 其中: 若 $j-1=0$ 则对应于上一层轮询层 (第 $i-1$ 层) 最后一个流水线过程; 若 $k-1=0$ 则对应于上一层轮询层第 j 个流水线过程对最后一个子数据块的处理过程。则有本文所提出的 DRoPipe 并行策略在流水线过程层级的递推关系如下所示。

当 $k > 1$ 时,

$$T_{ijk} = \max(T_{ij(k-1)} + F_{ij}, T_{i(j-1)k} + F_{i(j-1)}) \quad (1)$$

当 $k = 1$ 时,

$$T_{ijk} = \max(T_{(i-1)j\rho} + F_{(i-1)j}, T_{i(j-1)k} + F_{i(j-1)}) \quad (2)$$

为了后续通过数学归纳法推导一般形式, 首先以第一层轮询层的流水线过程作为特例进行推导并观察其中规律。通过数学归纳法可得, 第一层轮询层各流水线过程中每个子数据块的开始时间为:

$$T_{1jk} = \sum_{l=1}^{j-1} F_{1l} + (k-1) \max_{l=1}^j (F_{1l}) \quad (3)$$

其中, $j \in 1, 2, \dots, 2n, k \in 1, 2, \dots, \rho$ 。式(3)与文献[37]中针对单层流水线并行的公式一致。

然而, 轮询流水线并行模式中, 存在多层轮询层, 后层轮询层中各流水线过程的开始时间还受到前一层轮询层中同一个算力节点的流水线过程最后一个子数据块的结束时间的限制。为了便于后续公式表示, 将总序号为 i 的流水线过程对于第 k 个子数据块的开始时间记为 ϕ_{ik} , 处理一个子数据块的时间为 H_i , 则有对应关系为:

$$\begin{cases} T_{ijk} = \phi_{((i-1)2n+j)k} \\ F_{ij} = H_{(i-1)2n+j} \end{cases} \quad (4)$$

将总序号为 i 的流水线过程最后一个子数据块的结束时间记为 X_i , 则满足 $X_i = \phi_{i\rho} + H_i$ 。代入式(3)可得第一层轮询层中 X_i 满足:

$$X_i = \sum_{l=1}^i H_l + (\rho-1) \max_{l=1}^i (H_l) \quad (5)$$

其中 $\rho = pq, 1 \leq i \leq 2n$ 。取 $i = 2n, X_{2n}$ 等效于现有单层流水线下的并行策略的理论损失^[37]。

在第一层轮询层理论公式的基础上, 可进一步推导后序轮询层的递推公式。将式(4)、式(5)代入递推关系式一般形式(式(1)和式(2)), 可得任意轮询层各个流水线过程中每个子数据块的开始时间与前序轮询层的结束时间之间的递推关系为:

$$\phi_{ik} = \max_{l=1}^i \left(X_{l-2n} + \sum_{a=l}^{i-1} H_a + \bar{k} \max_{a=l}^i (H_a) \right) \quad (6)$$

其中, $\bar{k} = k-1, 1 < i \leq 2vm$; 若 $l \leq 0$, 则 $X_l = 0$ 。

在式(6)的基础上, 运用数学归纳法进行推导可得对于任意流水线过程(不妨设为第 $2n(i-1)+j$ 个流水线过程, 即第 i 层轮询层第 j 个流水线过程)最后一个子数据块的结束时间的解析表达式为:

$$X_{(i-1)2n+j} = \max_{1 \leq m \leq i} \left(\max_{\Omega_{mij}} Y_{mij} \right) \quad (7)$$

其中,

$$Y_{mij} = \sum_{d=0}^{m-1} \left(\sum_{a=h_d+2n}^{h_{d+1}} H_a + \bar{\rho} \max_{a=h_d+2n}^{h_{d+1}+2n} H_a \right) \quad (8)$$

其中, $\bar{\rho} = \rho-1$; Ω_{mij} 表示满足以下不等式关系的所有 $h_d (d=0, 1, \dots, m-1)$ 的组合形式:

$$\Omega_{mij} = \begin{cases} h_d + 2n \leq h_{d+1} \\ 1 \leq h_d \leq 2n(i-1)+j \end{cases} \quad (9)$$

其中, $h_m = (i-1)2n+j, h_0 = 1-2n$ 。式(7)是以流水线过程的基础时间函数为变量的理论损失模型的解析表达式。为了直观呈现, 图5对解析表达式中 Y_{mij} 的计算过程进行举例说明。

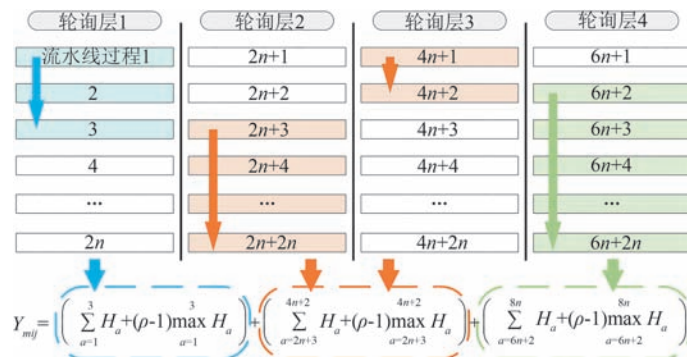


图5 理论损失模型解析表达式计算过程示意图(4层流水线轮询层, 4n个流水线阶段(8n个流水线过程, 包含计算过程和通信过程), 取 $h_1=3, h_2=4n+2, h_3=8n$ 时, $Y_{m,4,2n}$ 的计算过程示意图)

从图5可以看到, $Y_{m,4,2n}$ 的计算结果等于其中三段彩色标记的流水线过程分别按照单层流水线排列后的理论损失值之和。其他 h_d 的组合形式所对应的 Y_{mij} 的计算过程类似。

可得正向传播过程的完工时间如式(10)所示。

$$X_{2vm} = \max_{1 \leq m \leq n} \max_{\Omega_{mnc}} Y_{mnc} \quad (10)$$

由于分布式并行方案需要同时考虑流水线划分、数据划分和张量模型划分, 因此需要进一步将

其改写为关于流水线划分、数据划分和张量模型划分决策变量的形式。以正向传播为例,其表达式为

$$\begin{cases} H_{2i-1}(\lambda, p, q, z) = \sum_{j=a_{i-1}+1}^{a_i} f_j^{(C)}(p, q, z) \\ H_{2i}(\lambda, p, q, z) = f_{a_i}^{(M)}(p, q, z) \end{cases} \quad (11)$$

$$\begin{aligned} T^{(DRoPipe)}(\lambda, p, q, z) = & \max_{1 \leq m \leq n} \max_{\Omega_{muv}} \sum_{d=0}^{m-1} \left(\sum_{a=h_d+2n}^{h_{d+1}} H_a + (\rho-1) \max_{a=h_d+2n}^{h_{d+1}} H_a \right) \rightarrow \text{forward} + \\ & \max_{1 \leq m \leq n} \max_{\Omega_{muv}} \sum_{d=0}^{m-1} \left(\sum_{a=h_d+2n}^{h_{d+1}} B_a + (\rho-1) \max_{a=h_d+2n}^{h_{d+1}} B_a \right) \rightarrow \text{backward} \end{aligned} \quad (12)$$

其中, B_i 表示反向传播过程中第 i 个流水线过程处理一个子数据块的基础时间消耗。该损失模型解决了复杂策略 DRoPipe 的性能表征困难的问题。

4.2 层间递推并行计算模型

式(12)所示理论损失模型是直接计算时间损失的解析表达式。然而,在实际评估和优化过程中,遍历上述所有组合形式的总次数 κ 满足下式:

$$\begin{aligned} \kappa = & \sum_{j_1=1}^{2n \times 1} \cdots \sum_{j_{v-1}=1}^{j_{v-2}} j_{v-1} + \sum_{j_1=1}^{2n \times 2} \cdots \sum_{j_{v-2}=1}^{j_{v-3}} j_{v-2} + \cdots + \\ & \sum_{j_1=1}^{2n \times (v-1)} j_1 + 2nv \end{aligned} \quad (13)$$

其复杂度为 $O((2n)^{v-1})$, 是关于流水线轮询层数的指数增长复杂度,其中 $2n$ 为底的原因是每一层轮询层中包含 $2n$ 个流水线过程(计算和通信各 n 个)。另外,在 DRoPipe 的并行方案的搜索过程中,需要多次进行时间损失的计算,总计算量较大。

通过理论分析,可观察到在式(12)理论损失模型的计算中,实际上会重复部分计算过程。因此,若是能够消除冗余计算过程,则可以减少对时间损失进行计算所需的复杂度。为了进一步改进 DRoPipe 理论时间损失的计算复杂度以加速并行方案的评估和优化过程,结合递推公式和理论损失模型的解析表达式,本文建立了流水线过程间的并行递推计算模型。所有流水线过程中所有子数据块的结束时间可构成一个二维矩阵,记为 $E = \{E_{ik}\}_{2nv \times \rho}$ 。假设已知第 $j-2n$ 个到 $j-1$ 个流水线过程中各个子数据块的结束时间,即 $E[j-2n:j-1]$ 已知。此时可探讨第 j 个流水线过程各子数据块的结束时间 $E[j]$ 。由轮询流水线的递推公式可得,第 j 个流水线过程对于第一个子数据块的结束时间 E_{j1} 满足:

$$E_{j1} = \max(E_{(j-2n)\rho}, E_{(j-1)1}) + H_j \quad (14)$$

第 k 个子数据块(其中 $k > 1$)的结束时间 E_{jk} 为:

其中, $f_j^{(C)}$ 表示第 j 层神经网络在双维度数据划分分数为 $\eta = (p, q)$ 、张量划分数为 $\vartheta = z$ 时计算时间消耗, $f_j^{(M)}$ 表示通信时间消耗。继续推导,可得到 DRoPipe 并行策略同时考虑正向传播和反向传播过程的总完工时间,其关于多维度并行模式决策变量的理论损失模型的解析表达式为

$$E_{jk} = \max(E_{j(k-1)}, E_{(j-1)k}) + H_j \quad (15)$$

对 k 进行递推可得:

$$E_{jk} = \max \left(E_{(j-2n)\rho} + (k-1)H_j, \max_{l=1}^k (E_{(j-1)l} + (k-l)H_j) \right) + H_j \quad (16)$$

设顺序数组为 $R = [1, 2, \dots, \rho]$, 可构造数组 V_1 和 V_2 :

$$\begin{cases} V_1 = E[j-1] - H_j R \\ V_2 = E[j-2n, \rho] + H_j R \end{cases} \quad (17)$$

V_2 第 k 个元素为 $E_{(j-2n)\rho} + kH_j$, V_1 中第 l 个元素为 $E_{(j-1)l} - lH_j$ 。对数组 V_1 求累积最大值可得 V_3 :

$$\begin{aligned} V_3 = & \text{cmax}(V_1) + H_j(R+1) = \\ & \left\{ \max_{l=1}^k (E_{(j-1)l} + (k-l+1)H_j) \right\}_{1 \leq k \leq \rho} \end{aligned} \quad (18)$$

V_3 中的元素正好与式(16)中第二行一致。现有的人工智能框架 Pytorch、MindSpore 等均已支持累积最大值的矩阵并行运算(若不支持,可利用上三角矩阵实现快速计算)。因此,可得关于第 j 个流水线过程各子数据块的结束时间 $E[j]$ 的递推表达式:

$$\begin{aligned} E[j] = & \max(V_2, V_3) = \\ & \max \left(E[j-2n, \rho] + H_j R, \text{cmax}(E[j-1] + H_j) \right) \end{aligned} \quad (19)$$

式(19)即为 DRoPipe 并行策略理论损失模型的递推计算公式。由于式(19)是基于流水线过程间的递推计算公式,因此其计算复杂度主要由递推次数和每次递推的计算复杂度决定。其中递推次数等于流水线过程数 $2nv$, 每次递推计算复杂度主要在于累积最大值函数所需复杂度(为 ρ)。因此基于层间递推计算公式求解 DRoPipe 的时间损失所需总计算复杂度仅为 $O(2nv\rho)$, 亦为轮询多层流水线中所有子数据处理过程的总数。相比于基于式(12)理论损失模型的解析表达式的计算复杂度 $O((2n)^{v-1})$, 层间递推计

算公式大大降低了计算复杂度,可实现整体优化并行方案求解的加速。而在GPU等并行计算单元中运行式(19),可同时计算流水线过程中所有子数据块的结束时间,计算时间复杂度降低为 $O(2nv)$ 。

考虑到轮询流水线模式中层间递推公式的递推结构是固定的,因此可引入矩阵并行计算以进一步同时计算多个并行方案对应的时间损失。接下来从两个方面的时间损失并行计算模型进行论述:相同数据划分方案下多个流水线划分方案;不同数据划分方案下不同流水线划分方案。

针对相同数据划分方案下的多个流水线划分方案,其递推关系和理论损失模型中不同之处在于不同流水线划分方案之间基础时间函数 H_j 存在不同。对于 K 个流水线划分方案 $2vm$ 个流水线过程的基础时间函数可构成一个二维数组(矩阵),不妨记为 $W = \{w_{ki}\}_{K \times 2vm}$; K 个流水线划分方案流水线过程的子数据块结束时间组成三维数组,记为 $G = \{g_{kij}\}_{K \times 2vm \times \rho}$ 。则对于第 k 个流水线划分方案第 j 个流水线过程各子数据块的结束时间为 $G[k, j, :]$,对应于 $E[j]$;第 j 个流水线过程的基础时间函数为 $W[k, j]$,对应于 H_j 。用 $G[k, j, :]$ 替换 $E[j]$, $W[k, j]$ 替换 H_j ,并代入式(19),可得第 k 个流水线划分方案的层间并行递推公式:

$$G[k, j, :] = \max \left(G[k, j-2n, \rho] + W[k, j]R, \text{cmax}(G[k, j-1, :] + W[k, j]) \right) \quad (20)$$

在式(20)的基础上进一步进行推导,可得到多个流水线划分方案的层间并行递推计算模型:

$$G[:, j, :] = \max \left(G[:, j-2n, \rho] - W[:, j]R, \text{cmax}_{dim=2} \left(G[:, j-1, :] - W[:, j]R \right) + W[:, j](R+1) \right) \quad (21)$$

其中, $\text{cmax}_{dim=2}(Z)$ 表示在数组 Z 第2个维度求累积最大值。在并行运算单元中,式(21)可实现同时计算多个流水线划分方案对于时间损失的层间递推关系,则相比于式(21)的层间递推计算模型,进一步将DRoPipe策略多个并行方案(同数据划分方案、不同流水线划分方案)的性能表征时间复杂度降低至 $1/K$ (K 为同时计算的流水线并行方案个数)。

相同流水线划分方案不同数据划分方案的理论损失并行计算与不同流水线划分方案下不同数据划分方案相似。故此处直接梳理后者。

对于不同数据划分方案,其中同一个流水线过程中的子数据块的数量不同,且基础时间函数亦不同,此时式(21)将不再适用。参考文献[16]中理论损失模型递推关系并行计算模型的推导方式,可得对于DRoPipe的不同数据划分方案的并行计算模型。设一个第 x 组并行方案中第 i 个流水线过程中第 j 个数据序列中第 k 个数据元素(共 $P \times Q$ 个数据最小颗粒度切片)的结束时间为 e_{xijk} ,构成一个四维数组 $E = \{e_{xijk}\}_{K \times (2vm) \times P \times Q}$;由流水线划分和数据划分构成的第 x 个解中第 i 个流水线过程的基础时间为 w_{xi} ,构成二维数组 $W = \{w_{xi}\}_{K \times 2vm}$;所有方案组中数据划分方案解对应微批次尺寸构成一维数组 P ,微序列长度构成一维数组 Q 。通过理论推导可得同时计算多个并行方案组(数据划分方案或流水线划分方案均可能不同)的并行计算模型为式(22),其中 $L^{(b)}$ 和 $R^{(b)}$ 分别表示当前时间损失计算步骤中微批次维度和序列元素维度的数据索引起始值,其初始值均为 $[1, 1, \dots, 1]_K$; $L^{(e)}$ 和 $R^{(e)}$ 表示一个子数据块中最后一个数据的索引值;运算符“*”表示两个矩阵对应位置的元素相乘。

$$\begin{aligned} R^{(b)} &= \min(R^{(b)} + (L^{(b)} == Q) * Q, P) \\ R^{(e)} &= \min(R^{(b)} + Q, Q) \\ L^{(b)} &= (L^{(b)} == P) + (L^{(b)} \neq P) * L^{(b)} \\ L^{(e)} &= \min(L^{(b)} + P, P) \\ E[:, i, L^{(b)}[:, :], L^{(e)}[:, :], R^{(b)}[:, :], R^{(e)}[:, :]] &= W[:, i] + \\ &\max \left(E[:, i-1, L^{(e)}[:, :], R^{(e)}[:, :]], \right. \\ &\quad \left. E[:, i, L^{(b)}[:, :]-1, R^{(b)}[:, :]-1] \right) \\ L^{(b)} &= L^{(e)} \end{aligned} \quad (22)$$

同时使用式(21)和式(22),即可得本文所提出的DRoPipe的理论损失模型的层间并行递推计算模型。上述计算模型同样适用于:相同流水线划分方案下多个数据划分方案对应的时间损失并行计算;以及不同流水线划分方案下不同张量划分方案 and 不同数据划分方案对应的时间损失并行计算。

基于式(21)和式(22)的层间并行递推计算模型,将DRoPipe策略多并行方案(不同数据划分、不同流水线划分)的性能表征时间复杂度降低至 $1/K$,实现了对于并行策略不同决策变量组的性能表征并行计算。为了直观呈现不同计算模型的速度,表2列出不同计算形式对于 K_1 种不同流水线划分方案

组、 K_2 种不同数据并行与模型并行联合划分方案组的时间复杂度。从表2可知,结合式(21)和式(22)的多方案组并行计算模型的时间复杂度仅为 $2vn$,计算速度是解析表达式(式(12))的 $K_1K_2(2n)^{v-2}/v$ 倍,是子数据块递推公式(式(1))的 $K_1K_2\rho$ 倍。在多方案组层间并行递推计算模型的基础上,可大幅度改进多维度并行策略优化搜索速度。

表2 理论损失模型不同计算方式下的时间复杂度

计算模型	对应公式	时间复杂度
解析表达式	式(12)	$O(K_1K_2(2n)^{v-1})$
子数据块递推公式	式(1)、(2)	$O(2K_1K_2nv\rho)$
流水线过程间递推公式	式(19)	$O(2K_1K_2nv)$
多流水线方案并行递推	式(21)	$O(2K_2nv)$
多方案组并行计算模型	式(21)、(22)	$O(2nv)$

5 DRoPipe 多维并行策略搜索算法

5.1 算法整体框架:多维度并行策略交叉优化

根据式(12)中所描述的理论损失模型可知,可将DRoPipe并行策略中多维度并行模式的优化拆分为两个层面:轮询层内和轮询层间各流水线过程的负载均衡。根据理论损失模型,当每个子数据块的处理过程的两个前序过程恰好同时结束时,DRoPipe并行策略中轮询多层流水线并行模式的优势可得到最大程度的体现,即整个并行过程应尽可能是对齐状态。这就需要对DRoPipe并行策略中三维并行模式的决策变量进行寻优。

由于流水线并行模式、双维度数据并行模式、张量并行模式属于异构并行模式,其对应的决策变量属于异构决策变量。为了实现对于三种维度决策变量的联合优化,本文借鉴文献[37]中提出的交叉搜索框架,并进一步建立三维并行模式的交叉搜索算法,其框架如图6所示,其伪代码如算法1所示。

算法1. 三维并行模式的交叉搜索算法

输入:神经网络层关于数据划分数和张量划分的基础

时间函数 $f_j^{(C)}(p, q, z), f_{a_i}^{(M)}(p, q, z), b_j^{(C)}(p, q, z), b_{a_i}^{(M)}(p, q, z)$

输出:三维并行决策变量 λ, η, ϑ 与时间损失 $T^{(DRoPipe)}$ //初

始化三维并行模式的决策变量和布尔值

1. 初始化 $\lambda, \eta, \vartheta, check = \text{False}$
2. WHILE $check == \text{False}$ //固定DDP和TMP的决策变量,搜索RPMP决策变量
3. 利用隔板并行邻域搜索算法(调用算法3)求解 $\bar{\lambda}$ //固定RPMP决策变量,搜索DDP和TMP决策变量
4. 利用拐点检验搜索算法(调用算法2)求解 $\bar{\eta}$ 和 $\bar{\vartheta}$ //检查搜索前后的决策变量是否变化
5. IF $(\lambda, \eta, \vartheta) == (\bar{\lambda}, \bar{\eta}, \bar{\vartheta})$
6. $check = \text{True}$
7. END IF//更新当前决策变量组
8. $(\lambda, \eta, \vartheta) \leftarrow (\bar{\lambda}, \bar{\eta}, \bar{\vartheta})$
9. END WHILE//输出当前决策变量组并记录下时间损失值
10. 输出 $(\lambda, \eta, \vartheta)$ 和 $T^{(DRoPipe)}(\lambda, \eta, \vartheta)$

在交叉搜索算法中,不断切换当前进行寻优的并行模式的决策变量,最终实现对于多维度并行模式的同时优化。其中数据划分方案和张量划分方案,可在一定程度上进行组合枚举或者动态规划求解。另外,由于单个张量模型分量处理单个子数据块的基础时间函数均存在拐点,模型分量尺寸或子数据块数据量大于拐点时基础时间函数近似线性,小于拐点时基础时间函数一般在某个特定值附近波动。而最优的张量模型划分方案和双维度数据划分方案往往与基础时间函数的拐点相关。因此本文主要采取基于基础时间函数拐点检验的快速数据划分算法和张量模型划分算法,如算法2所示。

算法2. 基础时间函数拐点检验的快速划分算法

输入:流水线并行决策变量 λ ,神经网络层基础时间函数 $f_j^{(C)}(p, q, z), f_{a_i}^{(M)}(p, q, z), b_j^{(C)}(p, q, z), b_{a_i}^{(M)}(p, q, z)$,数据划分和张量划分的决策变量 (η, ϑ)

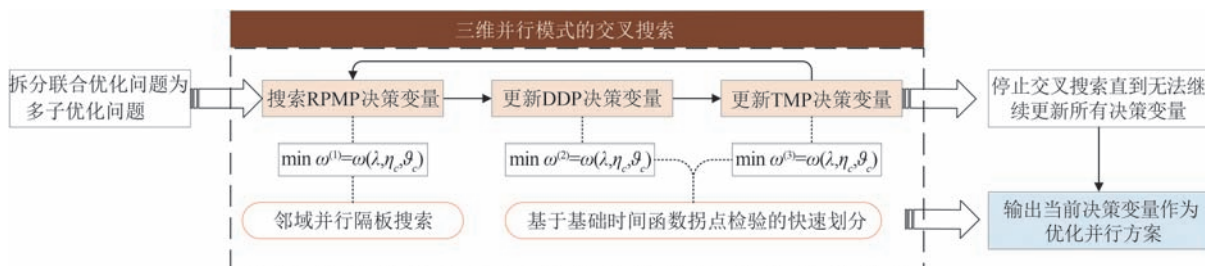


图6 针对DRoPipe策略三维并行模式的交叉搜索算法框架

输出: 数据划分决策变量 η 、张量划分 ϑ 、与时间 $T^{(DRoPipe)}$

1. 将 λ 代入式 11 得到 $H_i(p, q, z)$ 不同自变量的值//计算流水线过程基础时间函数的拐点
2. 获取 $H_i(p, q, z)$ 关于 p, q, z 的拐点值
3. 结合拐点组合, 得到多数据划分方案和多张量划分方案
4. 代入式 21 和式 22 的多方案组并行计算模型, 计算所有方案对应的时间损失
5. 取最小时间损失的 $(\eta = (p, q), \vartheta = z)$ 组合为数据并行和张量并行决策变量组, 并记录时间损失

$T^{(DRoPipe)}$

上述算法利用了 DRoPipe 策略中最优数据划

分数和张量划分数必然是某一个流水线过程以数据划分和张量划分为自变量的基础时间函数的拐点(或极值点)。因此, 将原本需要枚举所有数据划分和张量划分的组合的方式, 改进为仅需要枚举所有的拐点, 大大降低了总计算复杂度。

在多维并行模式交叉搜索算法框架和拐点快速检验划分算法基础上, 接下来将着重介绍轮询多层流水线并行模式的优化并行策略自动搜索算法。

5.2 流水线方案优化: 隔板并行邻域搜索

针对轮询多层流水线并行模式的并行方案寻优的隔板并行邻域搜索算法流程如图 7 所示。

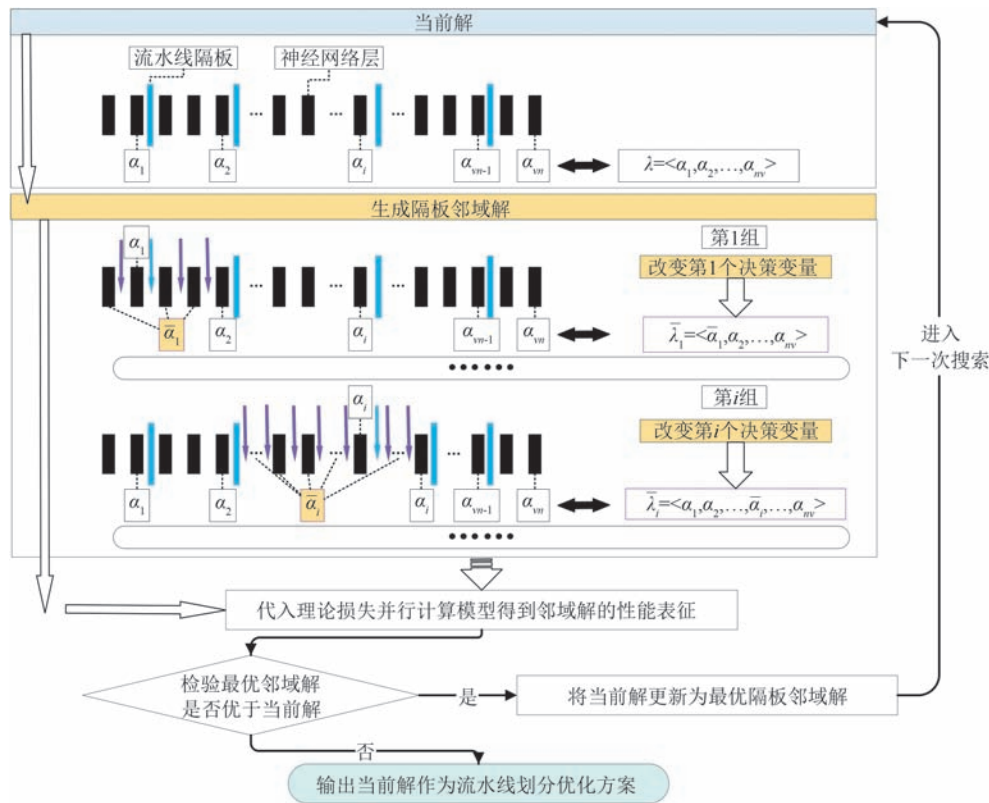


图 7 隔板并行邻域搜索算法流程结构图

在 DRoPipe 并行策略中, 相比于数据划分和张量模型划分, 轮询流水线的模型划分复杂度更高。这是因为 DRoPipe 中不仅需要考虑每层神经网络在不同流水线阶段中对于整体并行效率的影响, 还需要考虑各个流水线轮询层之间的影响。不同流水线阶段可能处于同一个算力节点, 因此还需要额外考虑轮询层的划分对于算力节点的显存占用的影响。另外, 轮询流水线模式中需要权衡流水线阶段内和流水线阶段间的通信过程。因此现有优化方法(如改进多维度二分法)无法直接对轮询流水线阶段的划分方案进行求解。

为解决 DRoPipe 中多轮询层的流水线阶段优化划分问题, 本文结合改进多维度二分法和邻域搜索算法, 提出隔板并行邻域搜索算法, 并建立其并行计算流程, 实现邻域并行隔板搜索负载均衡算法。

文献[37]中提出的改进多维度二分法主要针对单层流水线并行模式, 虽然无法给出轮询流水线并行的最终优化方案, 但可生成作为隔板邻域搜索的初始方案。在隔板邻域搜索算法中, 首先基于改进多维度二分法计算所有流水线阶段在单层轮询层中的优化划分方案。在初始方案的基础上, 通过隔板搜索依次对每两个相邻流水线阶段

进行重划分:在前一个(第 i 个)流水线阶段网络层最小序号 β_i 与后一个流水线阶段网络层最大序号 α_{i+1} 之间重新搜索第 i 个流水线阶段网络的最大序号 α_i 。将 α_i 在区间 $[\beta_i, \alpha_{i+1})$ 之间的搜索称为隔板邻域搜索。由于每个流水线阶段中网络层的最大序号 α_i 可唯一确定流水线阶段的划分方案,因此可用 α_i 表示流水线并行模式的决策变量。在此基础上,可得对于隔板邻域的定义:将一个流水线并行模式划分方案记为 $\lambda = \langle \alpha_1, \alpha_2, \dots, \alpha_m \rangle$,对于任意流水线并行模式划分方案 $\bar{\lambda} = \langle \bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_m \rangle$,若仅存在一个元素与 λ 中对应元素不同,其他均相同,则 $\bar{\lambda}$ 是 λ 的半径为1的隔板邻域中的解。其数学表达式为: $\exists i \in \{1, 2, \dots, m\}$ 使得对 $\forall j \neq i$ 有 $\alpha_j = \bar{\alpha}_j$, 且 $\bar{\alpha}_i \neq \alpha_i$, 其中 $\alpha_{i-1} < \bar{\alpha}_i < \alpha_{i+1}$ 。则邻域解 $\bar{\lambda}$ 亦可表示为: $\bar{\lambda} = \langle \alpha_1, \dots, \alpha_{i-1}, \bar{\alpha}_i, \alpha_{i+1}, \dots, \alpha_m \rangle$ 。

在隔板邻域定义的基础上,对于DRoPipe中轮流流水线并行模式的隔板邻域单步循环搜索方式可设置为:选取当前解 $\lambda = \langle \alpha_1, \alpha_2, \dots, \alpha_m \rangle$ 中一个变量 α_i ,将 $(\alpha_{i-1}, \alpha_{i+1})$ 区间内的所有值分别替换 α_i ,代入DRoPipe的理论损失模型,将其中最优化结果对应的变量值替换 α_i ,得到新的邻域搜索解。切换到下一个隔板,即设置 $i = i + 1$;若 $i = m + 1$ 则调整为 $i = 1$ 。继续上述隔板邻域搜索,直到连续 m 次未更新 α_i ,则停止搜索。

上述单步循环搜索方式,虽然能够在一定程度上对流水线并行模式的决策变量进行优化,但该方式实际上缩小了邻域搜索范围,且对隔板搜索顺序进行了限制。为了进一步改进搜索效率、提升收敛解的优化度,本文提出隔板并行邻域搜索算法,即同时对多个决策变量的可行解进行评估和搜索。其流程为:对所有 $i \in \{1, 2, \dots, m-1\}$,计算其对应邻域 $\bar{\lambda} = \langle \alpha_1, \dots, \alpha_{i-1}, \bar{\alpha}_i, \alpha_{i+1}, \dots, \alpha_m \rangle$ 的理论损失结果,选取最优邻域解作为新的解,进入下一次搜索。继续上述隔板邻域搜索,直到不存在任意邻域解优于当前解,则停止搜索。

上述并行隔板搜索的方式,允许流水线阶段划分的所有决策变量在相同优先级下进行优化更新,其搜索邻域大于单步循环搜索方式,因此其搜索效率和局部收敛解均优于单步循环搜索方式。在隔板邻域数学定义的基础上,可给出隔板并行搜索算法的伪代码,如算法3所示(流程图如图7所示)。

算法3. 隔板并行邻域搜索算法

输入:神经网络层关于数据划分和张量划分的基础时

间函数 $f_j^{(C)}(p, q, z), f_{a_i}^{(M)}(p, q, z), b_j^{(C)}(p, q, z), b_{a_i}^{(M)}(p, q, z)$

输出:流水线划分决策变量 λ 与时间损失 $T^{(DRoPipe)}$ //初始化流水线模型划分决策变量

1. 初始化 λ 作为当前解并计算对应时间损失(记为 T),
 $check = \text{False}$ //隔板并行邻域搜索循环过程
2. WHILE $check == \text{False}$ //生成当前解的隔板邻域解并整合所有邻域解
3. $\bar{\lambda}_i = \langle \alpha_1, \dots, \alpha_{i-1}, \bar{\alpha}_i, \alpha_{i+1}, \dots, \alpha_m \rangle, i \in \{1, 2, \dots, m\}$ //利用并行计算模型计算所有邻域解对应的
时间损失
4. 代入式21计算邻域解对应时间损失,并获取最优邻域解 $\bar{\lambda}$ 及其的时间损失 $\bar{T}$ //检验搜索停止
条件
5. IF $\bar{T} \geq T$ //若无邻域解优于当前解,则设置跳出循环的标记
6. $check = \text{True}$
7. ELSE//若最优邻域解优于当前解,则更新当前解
8. $T = \bar{T}, \lambda = \bar{\lambda}$
9. END IF
10. END WHILE//输出当前流水线并行模式决策变量组与时间损失值
11. 输出 λ 和 $T^{(DRoPipe)} = T$

结合算法1和3,可得DRoPipe的优化算法:基于隔板并行邻域搜索的交叉搜索(CSPBS, cross search with parallel bulkhead neighborhood search)。

5.3 复杂度分析

隔板邻域搜索的总复杂度主要包含三个部分:每次迭代过程中所有邻域解的个数(记为 c_1)、代入理论损失模型评估一个邻域解性能的计算复杂度(记为 c_2)、总迭代次数(记为 c_3)。则隔板邻域搜索算法的计算复杂度($c^{(PN)}$)为:

$$c^{(PN)} = c_1 \cdot c_2 \cdot c_3 \quad (23)$$

邻域中的解个数(c_1)主要由每个决策变量的可选解个数确定。对于某个 α_i 在邻域中的可选解个数为 $\alpha_{i+1} - \alpha_{i-1} - 1$,则所有 $\alpha_i (i \in \{1, 2, \dots, m-1\}, \alpha_m = \sigma, \alpha_0 = 0)$ 构成的邻域解总数为:

$$c_1 = \sum_{i=1}^{m-1} (\alpha_{i+1} - \alpha_{i-1} - 1) = O(2\sigma) \quad (24)$$

根据式12所示理论损失模型的解析表达式,评估一个流水线并行模式的划分方案所需复杂度为 $O((2n)^{v-1})$,而在引入多方案组并行计算模型后其计算复杂度仅为 $c_2 = O(2nv)$ 。

由于隔板邻域搜索达到收敛需要的次数暂无法通过理论推导确定,因此,此处仍然沿用 c_3 进行表示。在实际隔板邻域搜索中可以设置算法早停条

件,即当迭代次数达到给定的值后,即可停止隔板邻域搜索。另一方面,隔板邻域搜索达到收敛所需的迭代次数 c_3 的统计值一般与 σ 和 vn 相关,则可以通过大量的实验统计对其平均复杂度的关系式进行拟合。将 c_1, c_2, c_3 代入式(23)可得隔板邻域搜索算法的总计算复杂度为:

$$c^{(PN)} = O(4c_3\sigma vn) \quad (25)$$

上述复杂度远远小于流水线并行模式中所有可求解的个数(复杂度为 $O(\sigma^m)$)。因此可知隔板邻域搜索算法属于多项式算法,具有可实施性。当引入并行评估和性能表征重用后,隔板搜索算法的时间复杂度 $\bar{c}^{(PN)}$ 可降低为:

$$\bar{c}^{(PN)} = c_2c_3 = O(2c_3vn) \quad (26)$$

对于基于拐点的快速划分算法复杂度进行分析。对于各个数据划分或张量划分每个 H_i 仅有一个拐点,因此最多可生成 $c_4 = O(\min((vn)^3, PQn))$ 组数据划分和张量划分的方案组。因此其计算复杂度为 $c^{(IP)} = c_4c_2$ 。由于多方案组时间损失并行计算模型的引入,所有的方案组的时间损失可并行计算,因此其时间复杂度仅为 $\bar{c}^{(IP)} = c_2 = O(2nv)$ 。

令交叉搜索的轮数为 ξ ,则可得针对 DRoPipe 并行策略的三维并行模式交叉搜索算法(结合拐点检验快速划分法和隔板并行邻域搜索算法)的总计算复杂度 $c^{(\alpha)}$ 和时间复杂度 $\bar{c}^{(\alpha)}$ 分别为:

$$\begin{cases} c^{(\alpha)} = \xi(c^{(IP)} + c^{(PN)}) = O(4\xi c_3\sigma vn) \\ \bar{c}^{(\alpha)} = \xi(\bar{c}^{(IP)} + \bar{c}^{(PN)}) = O(2\xi c_3vn) \end{cases} \quad (27)$$

在理论损失模型并行评估和性能表征重用的基础上,结合基于基础时间函数拐点检验的快速数据划分算法、张量模型划分算法,以及在交叉搜索框架中调用本文所设计的隔板邻域搜索算法,从而形成了针对本文 DRoPipe 的整体的优化方法,其框架结构如图6所示。本文所提出的整体优化方法,融合了不同并行模式的决策变量组所适用的优化算法,并通过交叉搜索进行灵活地联合,有助于同时求解双维度数据并行模式、张量模型并行模式、轮询流水线并行模式的优化方案。这在一定程度上也反映了本文不仅为深度学习模型提供了新型并行策略 DRoPipe 的基础结构,也构建了其理论损失模型、优化方法等关键技术,以形成完善的并行训练体系。因此,在后续的实验验证中,本文不仅对 DRoPipe 并行策略的性能优势进行验证,也将验证理论损失计算模型和并行方案优化方法的性能。

6 实验结果与分析

6.1 实验设置

为了综合验证本文所提 DRoPipe 并行训练策略及其时间损失计算模型与优化算法的性能,本文结合仿真数据驱动实验和真实显卡算力集群实验,其中真实算力节点的显存限制设置为峰值显存占用小于算力节点最大显存。实验选取 GPT 系列和 Llama 系列的典型 DNN 结构进行实验,其详细信息如表3所示。实验中真实算力集群环境配置如表4所示。

表3 实验中大规模深度学习模型的详细信息。

	LLMs	Transformer	D_model	Heads
Llama 系列	Llama2-7B	32 Layers	4096	32
	Llama2-13B	40 Layers	5120	40
	Llama2-70B	80 Layers	8192	64
GPT 系列	GPT2 L-774M	36 Layers	1280	20
	GPT2 XL-1.56B	48 Layers	1600	25
	GPT3-175B	96 Layers	12 288	96

表4 实验中的算力集群环境配置。

编号	集群类型	显卡配置信息
环境1	同构集群	$\langle \underbrace{A, A, \dots, A}_n \rangle$
环境2	同构集群	$\langle \underbrace{B, B, \dots, B}_n \rangle$
环境3	异构集群	$\langle \underbrace{B, A, B, A, \dots, B, A}_n \rangle$
环境4	异构集群	$\langle \underbrace{B, B, \dots, B}_{n/2}, \underbrace{A, A, \dots, A}_{n/2} \rangle$
A: RTX 5060 Ti @ 16 GB; B: RTX 3090 @ 24 GB		

对比实验主要包含三个方面:(1)时间损失计算模型对比,(2)并行方案优化方法对比,(3)并行训练策略对比。为了便于实验结果的呈现和讨论,表5列出参与对比实验的各计算模型、优化方法、并行策略的缩写及其配置说明对照。

实验中大规模深度学习模型主要在 WikiText、Chinese Web Text、Common Crawl 等文本数据集上进行实验测试。由于在相同序列长度和小批次尺寸下,文本数据集的类型不同并不影响深度学习模型并行计算的速度和显存占用,因此后续实验中不再特别区分深度学习模型输入数据集,主要呈现训练 WikiText-103 数据集的实验结果。另外,在计算模型快速性统计验证实验中,结合仿真实验数据以便进行多次实验。对于包含大规模算力集群的实验

表 5 参与对比实验的各计算模型、优化方法、并行策略的英文缩写和配置对照表

类别	英文缩写与全称	说明
计算模型	MSSD (multiple schemes and single data block)	多方案组单子数据块递推计算模型
	SSMD (single scheme and multiple data blocks)	单方案组多子数据块递推计算模型
	SDRC (sub data block-based recursive computing)	子数据递推计算模型
	MSPC (ours) (multiple schemes-based parallel recursive computing)	多方案组并行递推计算模型
优化方法	CSSBS (cross search with single step bulkhead neighborhood search)	基于单步隔板邻域搜索的交叉搜索
	CSGS (cross search with greedy search)	结合贪心搜索的交叉搜索
	CSIMD (cross search with improved multi-dimensional dichotomy)	基于改进多维度二分法的交叉搜索
	EDBalance (enumeration data partitioning and balanced pipeline segmentation)	枚举数据划分+均衡流水线分段
	CSPBS (ours) (cross search with parallel bulkhead neighborhood search)	基于隔板并行邻域搜索的交叉搜索
现有先进并行策略	SPM (single layer pipeline parallelism with micro-batch-based data partitioning)	单层流水线并行+微批次数据划分→GPipe ^[11] 、DeepSpeed ^[23] 等
	SPT (single layer pipeline parallelism with token-based data partitioning)	单层流水线并行+微词条数据划分→TeraPipe ^[6] 等
	SPD (single layer pipeline parallelism with single-dimensional data partitioning)	SPM 和 SPT 的最佳方案
	SPD+1F1B (SPD adding one forward one backward strategy)	SPD 基础上增加时序调度模式→PipeDream ^[22] 、DAPPLE ^[15] 等
	SPUD (single layer pipeline parallelism with unequal data partitioning)	单层流水线并行+非均等数据划分→UMPIPE ^[16] 等
本文策略下的 消融并行策略	SPDD (single layer pipeline parallelism with dual-dimensional data partitioning)	单层流水线并行+双维度数据划分
	MPM (multi-layer pipeline parallelism with micro-batch-based data partitioning)	轮询多层流水线+微批次数据划分
	MPT (multi-layer pipeline parallelism with token-based data partitioning)	轮询多层流水线+微词条数据划分
	MPD (multi-layer pipeline parallelism with single-dimensional data partitioning)	MPM 和 MPT 的最佳方案
本文策略	DRoPipe (ours)	轮询多层流水线+双维度数据划分

证,本文参考文献[37]中串行模拟并行的方式,并结合使用本文所提出的轮询流水线并行,通过少量算力节点近似模拟大规模算力节点的并行性能,以克服算力节点有限、降低实验周期、增加实验次数。本文实验执行过程本身亦能说明本文所提轮询流水线并行模式能够提供有效的实验依托方式。

6.2 时间损失计算模型对比

为了验证本文提出的针对 DRoPipe 策略的多方案组并行递推计算模型 MSPC 的快速性,本文通过控制变量法,在不同节点数、流水线轮询层数和子数据块数量等配置下进行时间损失的计算实验。实验通过仿真程序随机生成各流水线过程的基础时间函数。参与对比的方法包括:子数据递推算式 SDRC、单方案组多子数据块递推算式 SSMD、多方案组单子数据块递推模型 MSSD。所有计算模型均在同一台式机中执行,配置如下:

- 编程版本:Python 3.7 + Pytorch 1.13.1;

- CPU: Intel i5-13600KF @ 3.50 GHz;
- GPU: NVIDIA GeForce RTX 3060 Ti @ 8 GB。

为了体现对于广泛基础时间函数的适应性,实验呈现基础时间函数服从均匀分布 $U \sim [20, 100]$ 的实验结果。分别将子数据块、流水线轮询层数、节点数量作为控制变量,将计算模型代入实验组: $\rho \in [5, 500], n = 20, v = 20; v \in [1, 20], \rho = 64, n = 10; n \in [2, 10], \rho = 64, v = 5$ 。上述实验组中,每组实验随机产生 30 个实例,每个实例中计算模型需要计算 100 组方案对应的时间损失。以每个场景中对应变化的参数作为横坐标,以各计算模型的平均执行时间作为纵坐标,各计算模型的执行时间变化趋势曲线如图 8 所示。

图 8 中多方案组并行递推计算模型的平均执行时间在数量级上明显小于对比计算模型:SDRC、SSMD、MSSD。具体而言,对于图 8(b)和图 8(c)的场景中,子数据块的数量为 64 时,多方案组并行递

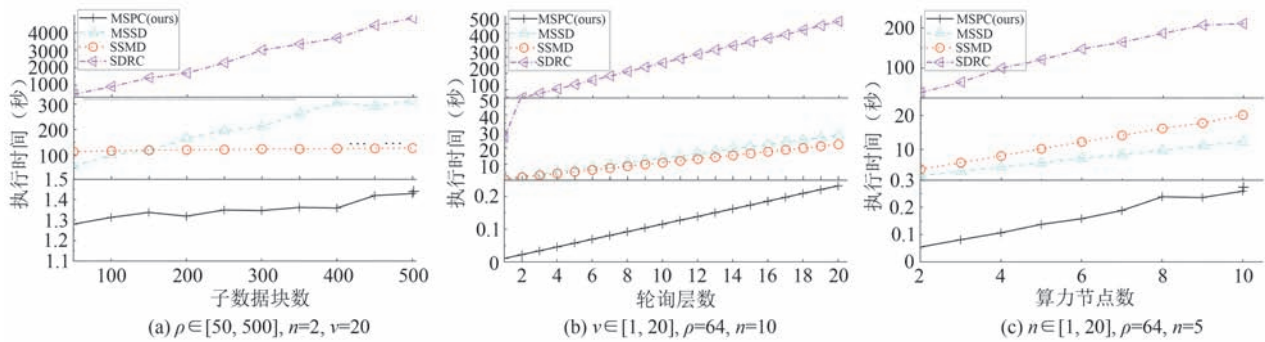


图8 理论时间损失的不同计算模型的平均执行曲线图

推算模型计算100组方案的时间损失所需执行时间在小于0.3s的范围内;而子数据递推算模型(SDRC, 现有常用方式)所需执行时间范围为[30s, 500s];单方案多子数据块递推算模型(SSMD)和多方案单子数据块递推算模型(MSSD)的执行时间范围接近,为[1s, 30s]。此外,SSMD与SDRC之间的对比还可以证明式19所示多子数据块并行计算对于时间损失计算的改进是有效的;比较多方案并行递推和单方案并行递推可以证明式21所示并行计算形式对于时间损失计算的加速亦是有效的。根据图8可得,多方案组并行递推算模型求解100组方案时间损失的执行速度分别是SDRC、SSMD、MSSD的1853、89、109倍。

图8实验结果验证了本文所提出的多方案组并行递推算模型MSPC的快速性优势,其以秒级执行时间足以满足当前现实的大规模深度神经网络在DRoPipe下进行并行方案寻优的性能表征需求。图8实验结果也在一定程度上印证了表2所示理论计算模型的时间复杂度理论分析结果。然而,需要注意的是:表2所分析的时间复杂度是基于计算设备具有足够或无限并行能力,而真实设备的并行计算能力有限,因此本文所提出的理论损失快速计算模型(即多方案组并行递推算模型MSPC)在真实设备运行过程中并未达到表2所示子数据递推算式(SDRC)的 100ρ 倍速度(实验中 $100\rho \geq 5000$)。

在理论损失计算模型快速性实验验证的基础上,进一步验证计算模型的准确性。在实际工程应用中,为了尽可能发挥理论计算模型的优势并提前准确评估多维度并行策略的各划分方案的性能,需要结合性能数据采集分析(profiling)过程中获取的基础时间函数。以10次采集单层Transformer层性能数据的平均时间作为对应基础时间函数,代入理论损失模型计算DRoPipe训练1个小批次数据的时

间,理论模型计算结果与真实集群中四个场景各20次训练实例对比如图9所示。其中采用均衡流水线划分,选取Llama2-7B和GPT2 XL-1.56B典型神经网络在环境2(RTX 3090 @ 24 GB同构集群)10个节点中进行真实训练,图9(a)和图9(b)中每组实例为固定双维度数据划分数(两个维度均为32划分数),图9(c)和图9(d)中每组实例为随机双维度数据划分数。相对误差率 ϕ 计算公式为:

$$\phi = \frac{|T^{(real)} - T^{(theory)}|}{T^{(real)}} \quad (28)$$

其中, $T^{(real)}$ 表示真实集群场景中训练1个小批次数据的时间, $T^{(theory)}$ 表示MSPC所求得的结果。

从图中可以观察到,本文所提出的理论损失计算模型的计算结果与真实训练时间之间最大误差率不超过4%,平均误差率为0.64%,说明本文理论计算模型能够以很小的误差对真实训练过程进行表征。其根本原因在于:理论损失模型充分必要地模拟了并行训练过程及其前序后序过程之间的逻辑对应关系,且通过性能数据采集分析(profiling)过程中端到端的统计时间损失并取多次平均的方式,已经包含了训练过程中数据操作、进程线程控制等复杂过程,从而可以较为准确地反映真实运行过程。

6.3 优化算法对比

为了验证优化算法的综合性能,本文采用单位执行时间下求得的优化结果(对应DRoPipe的训练时间)作为评价指标。由于DRoPipe并行策略是本文首次在深度学习模型分布式并行计算中提出,目前暂无相应的优化算法,因此,本文选取现有具有一定普适性的典型算法作为对比算法,并补充基于单步隔板邻域搜索的交叉搜索算法(CSSBS)参与对比。相关算法配置说明如表5所示。上述优化算法在同一台式机环境中执行。

选取四组基础时间服从均匀分布 $U \sim [20, 100]$

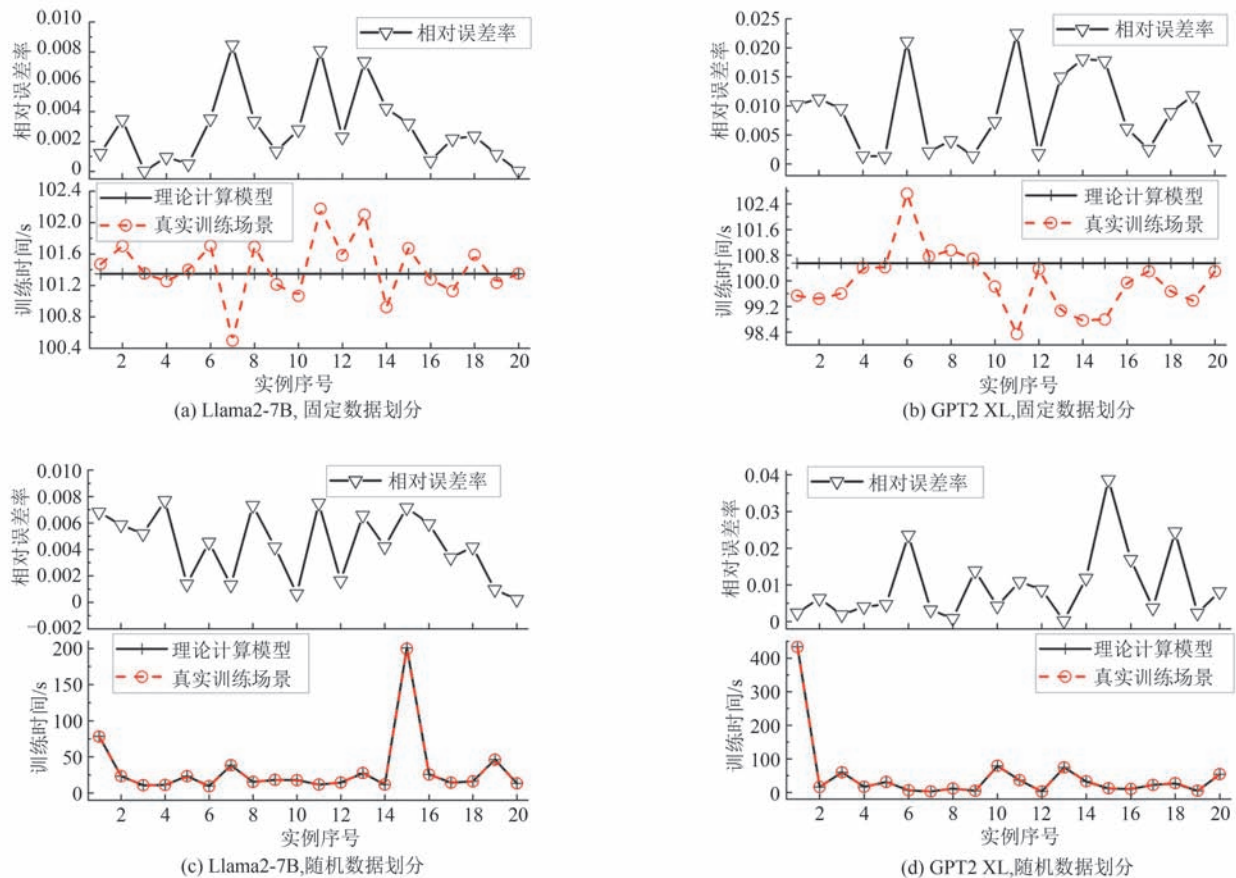


图9 DRoPipe的理论损失模型计算结果与真实环境集群训练时间误差对比图(10个节点,4层流水线轮询层)

的随机仿真场景进行验证:(1) $\sigma=50, n=5, v=5$;(2) $\sigma=64, n=4, v=4$;(3) $\sigma=100, n=5, v=5$;(4) $\sigma=300, n=10, v=10$ 。其中 $P=128, Q=128$ 。为了减小并行策略差异造成的并行计算时间损失的差异,从而直观呈现算法本身的优化效果和搜索速度,在实验场景中均固定 n 和 v 不变。以每个场景中算法执行时间作为横坐标,以各优化算法在执行历程中所求得的多维度并行方案对应的训练时间(即优化结果)作为纵坐标,各优化算法的优化结果变化趋势如图10所示。

如图10结果显示,本文提出的基于隔板并行邻域搜索的交叉优化算法CSPBS求得的优化结果曲线始终低于其他对比算法的曲线。这表示在整个执行过程中,基于隔板并行邻域搜索的交叉优化算法每次迭代搜索所求得的优化解都优于对应执行时间下其他对比算法的优化解。上述结果验证了本文所提基于隔板并行邻域搜索的交叉优化算法对于新型并行策略DRoPipe的可行性和优越性。基于隔板并行邻域搜索的交叉优化算法(CSPBS)的性能优势主要来自两个重要策略:执行一次改进多维度二

分法(IMD)所求得的流水线划分方案作为CSPBS的初始方案;结合并行评估计算模型同时对多个隔板邻域解进行并行验证。为了进一步验证CSPBS算法并考察各种策略对其优化性能的影响,本文对上述两个策略进行消融实验,分别对比随机初始化方案下的隔板并行邻域搜索(Random+PBS)、改进多维度二分法初始方案下的单步隔板邻域搜索(IMD+SBS)、随机初始化方案下的单步隔板邻域搜索(Random+SBS)。以算法执行时间为横坐标、优化结果为纵坐标,结果如图11所示。

根据图11,同时使用改进二分法初始化和隔板并行邻域搜索的方式具有更快的搜索速度和更优的收敛解,而Random+SBS未采用两种策略时搜索速度明显最慢。对比IMD+SBS与Random+PBS:在搜索过程前期,IMD+SBS的曲线明显低于Random+PBS,这是因为IMD+SBS以IMD初始化为起始点,本身更接近最优解;而在搜索过程后期,Random+PBS优于IMD+SBS,且具有更好的收敛解。上述现象说明:在基于IMD+PBS的CSPBS算法中,IMD初始化优势体现在提升搜索速

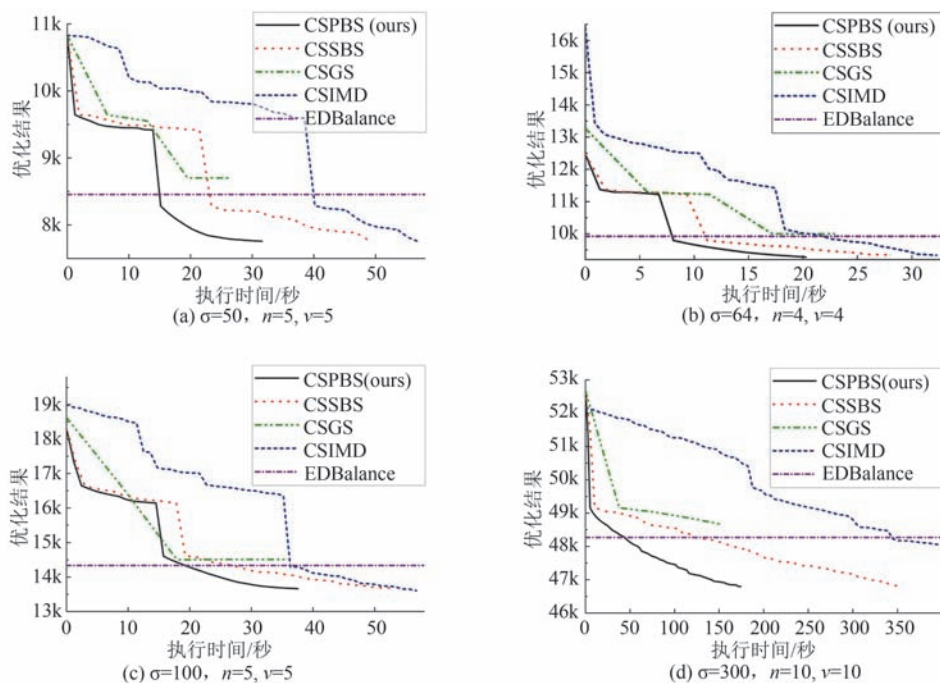


图 10 不同优化算法的优化结果随执行时间的变化趋势图

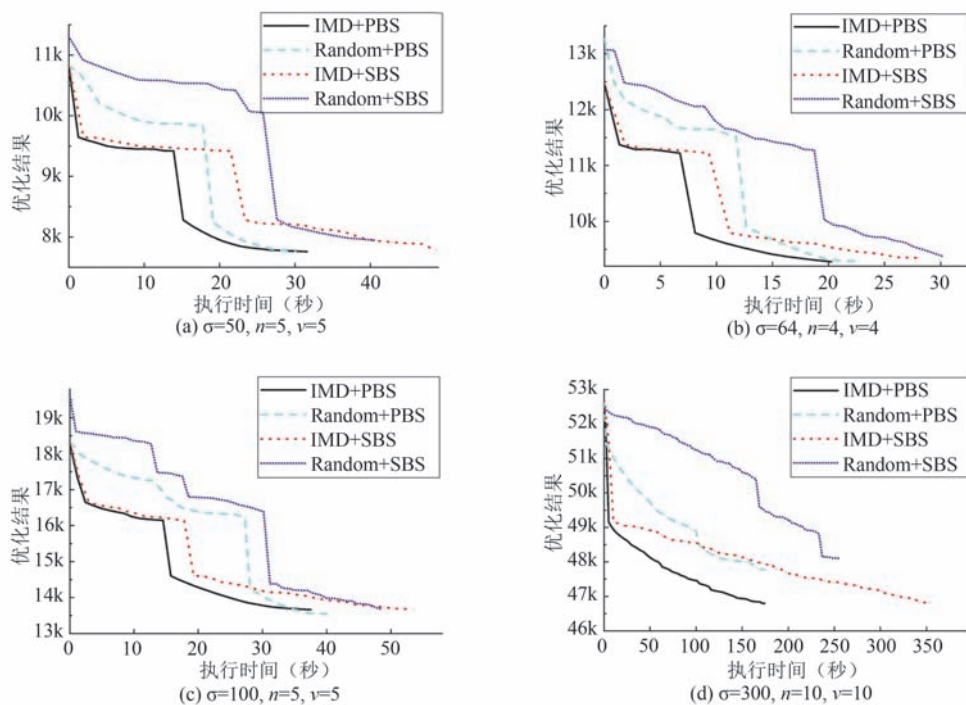


图 11 消融实验(交叉搜索中隔板搜索算法不同策略的优化结果随执行时间的变化趋势曲线图)

度, PBS的优势体现在提升局部收敛解的优化性。

上述实验验证了本文基于隔板并行邻域搜索的交叉搜索算法(CSPBS)的搜索速度和优化性。

6.4 多维度并行训练策略对比

在上述时间损失计算模型和优化算法综合性能的实验验证基础上, 进一步对本文所提出的DRoPipe并行训练策略性能进行多角度实验验证, 参

与对比的并行策略缩写和配置说明如表5所示。

为了控制实验结果的一致性, 本文仅对流水线划分和数据划分两种可控度较高的并行维度进行实验, 在引入张量划分后仅需将单节点流水线改为节点组流水线即可, 相应实验对比结论一致。为了尽可能反映各种并行策略的真实性能, 本文采用以下方法求解各种并行策略的最优并行方案: (1) 对单

层流水线单维数据划分并行策略(SPM和SPT,典型策略包括GPipe、TeraPipe、DeepSpeed等)、单层流水线双维数据划分并行策略(SPDD),采用文献中提出的改进多维度二分法求解最优流水线划分,采用枚举法遍历所有的数据划分方案。(2)对单层流水线非均等数据划分并行策略(SPUD,典型策略包括UMPIPE),采用文献中提出的多染色体遗传算法求解优化的数据划分方案。(3)对引入1F1B时序调度模式的单层流水线单维数据划分并行策略(SPD+1F1B,典型策略包括PipeDream、DAPPLE

等),采用融合动态规划和贪心算法的多路径优化求解时序调度方案。(4)对轮询流水线单维数据划分并行策略(MPM和MPT)、轮询流水线双维数据划分并行策略(DRoPipe),采用基于隔板并行邻域搜索的交叉优化求解并行方案,以表征并行策略性能。

由于典型深度学习模型的神经网络层数已给定,因此实验结果以节点数量作为横坐标,呈现不同并行策略性能关于节点数量的变化趋势。对于表3所示6种神经网络结构在环境1(RTX 5060 Ti @ 16 GB 同构集群)中的训练耗时如图12所示。

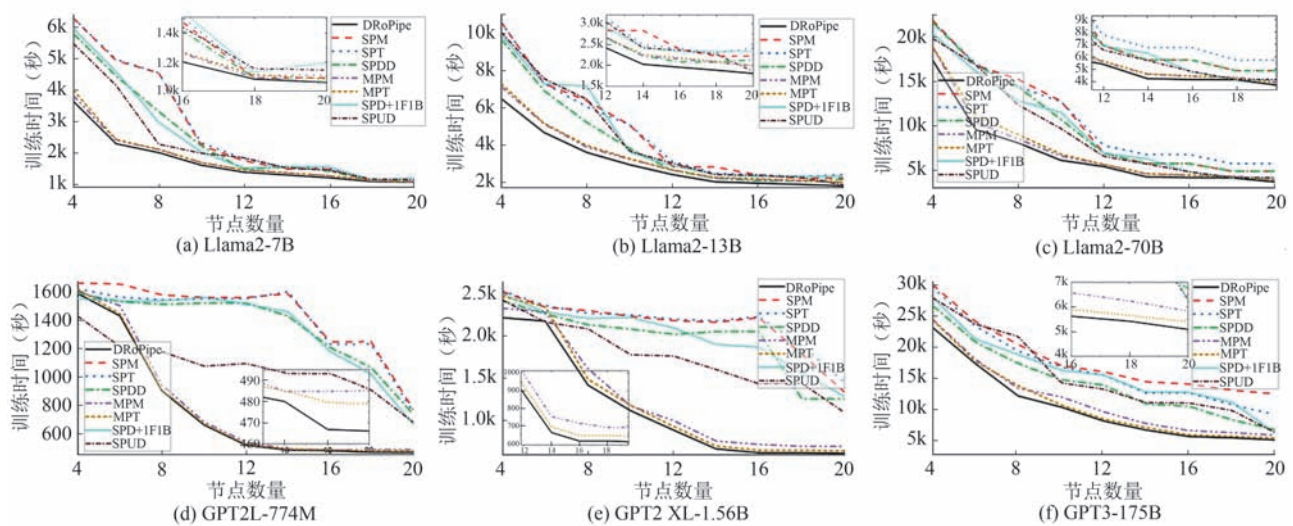


图12 不同分布式并行训练策略训练100组 $P \times Q = 128 \times 128$ 词条数据的性能对比图(其中轮询层数设定为最多4层)

在图12中,DRoPipe并行策略的优化方案对应的时间损失明显低于其他对比并行策略,说明DRoPipe并行策略在各种算力节点数量下均达到了更快的并行训练速度。相比于现有并行策略SPM(典型策略GPipe、DeepSpeed)、SPT(典型策略TeraPipe)、SPD+1F1B(典型策略PipeDream、DAPPLE)、SPUD(典型策略UMPIPE),DRoPipe能够较为稳定地达到更低的训练耗时,显示了DRoPipe作为针对深度学习模型分布并行计算的新型策略的优势。其中,随着算力节点数量的增加,各并行策略的并行训练时间均呈现出整体下降的趋势,而DRoPipe的下降趋势更为明显,说明DRoPipe更能充分发挥给定节点数量的算力,以加快整个并行训练过程。另外,DRoPipe结合轮询多层流水线并行和双维度数据划分,在相同显存限制的条件下,允许更多可行的数据划分方案,降低了对于整体计算时间损失的需求。以SPM和SPT为基准性能,对比DRoPipe、SPD+1F1B、SPUD相对于SPM和

SPT基准性能的提升幅度,可以观察到DRoPipe的提升幅度最大,其次是SPUD,说明了结合轮询多层流水线并行和双维度数据并行对并行训练的提速效果,优于SPUD中的非均等数据并行和SPD+1F1B中的1F1B时序调度模式。其原因可能在于:在上述实验中各大模型的Transformer层均属于同构层,流水线阶段间异构性较低,因此非均等数据划分的效果并不明显;在实验过程中,正向传播过程的整体耗时较短,因此1F1B时序调度模式可以进行时序提前的反向传播过程总数有限。

图12的结果与第4节关于DRoPipe的并行速度的分析一致,实验结果说明了DRoPipe并行策略在并行训练速度方面相比于现有策略具有明显优势。

对图12实验中DRoPipe并行策略的两种基础并行模式(双维度数据并行和轮询多层流水线并行)进行消融对比:SPDD优于SPM和SPT,DRoPipe优于MPM和MPT,验证了双维度数据划分有效地提升了并行速度;MPM优于SPM,MPT优于SPT,

DRoPipe 优于 SPDD, 验证了轮询流水线并行模式亦有利于提升并行速度。对于两种消融对比角度的提升幅度, DRoPipe 相比于 SPDD 的提升幅度大于其相比于 MPT 和 MPM 的提升幅度。比如, 当算力节点数为 4 时, DRoPipe 训练 Llama2-7B 的时间为 3630.5 s, 相比于 SPDD 的 5794.0 s 减少了 2163.5 s, 相比于 MPT (3812.0 s) 和 MPM (3990.5 s) 分别减少了 181.5 s 和 360.0 s。上述现象揭示了在 DRoPipe 并行策略中, 轮询流水线并行模式的提速效果高于双维度数据划分的提升效果。尤其在节点数偏少 (小于 12 个节点) 的场景下, 轮询流水线并行在训练时间方面的提升效果更为明显。这是因为轮询流水线并行模式可以有效进行算力节点复用, 改善流水

线划分颗粒度, 从而提高设备利用率。

为了考察 DRoPipe 的综合性能, 实验统计算力集群总能耗系数 $E^{(c)}$ 和平均气泡时间 $T^{(b)}$ 指标:

$$E^{(c)} = \sum_{i=1}^n T_i^{(e)} = \bar{U} \times T \times n \quad (29)$$

$$T^{(b)} = \frac{1}{n} \left(\sum_{i=1}^n T - T_i^{(e)} \right) = T \times (1 - \bar{U})$$

其中, $\bar{U}$ 表示算力节点平均工作时间与总完工时间之比, T 表示并行训练的完工时间, $T_i^{(e)}$ 表示第 i 个算力节点有效计算时间, n 为算力节点数。当算力节点处于高效工作时, 实时功率和平均功率范围相对稳定, 因此 $E^{(c)}$ 可在一定程度上反映整个集群的能耗水平。此时, 可给出图 12 中部分实验对应的能耗系数和气泡时间, 分别如图 13 和图 14 所示。

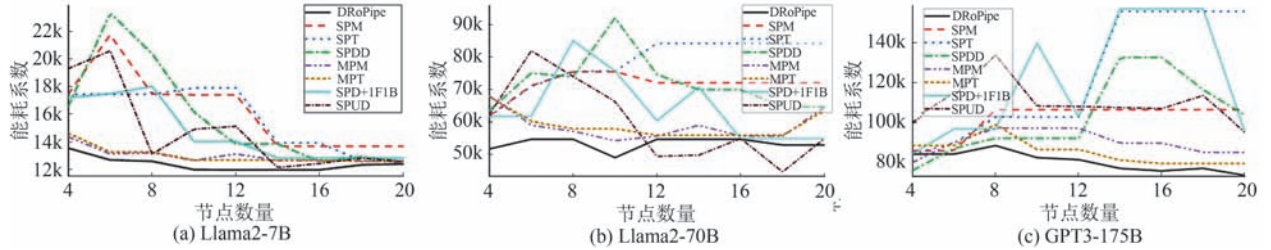


图 13 不同并行训练策略下能耗系数对比图 (其中轮询流水线模式的轮询层数设定为最多 4 层)

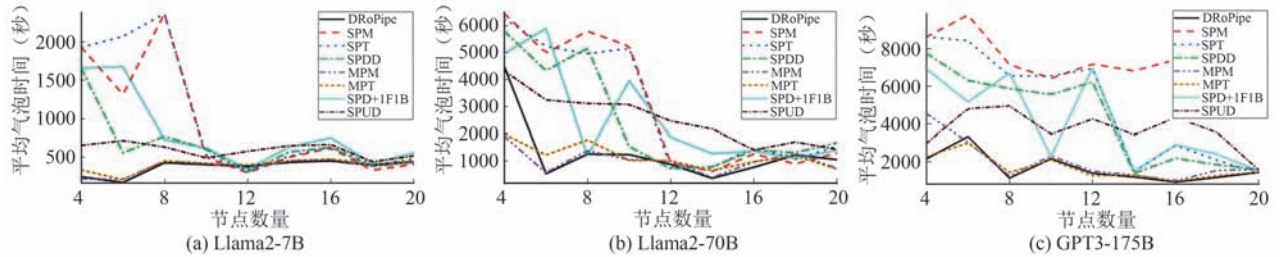


图 14 不同并行训练策略下气泡时间对比图 (其中轮询流水线模式的轮询层数设定为最多 4 层)

从图 13 和图 14 可知: DRoPipe 的能耗系数 (集群有效计算时间总和) 和气泡时间明显低于对比策略, 说明 DRoPipe 在加速并行训练的同时也降低了算力集群总能耗和气泡时间。虽然, 在部分场景中, DRoPipe 的气泡时间略高于部分对比策略, 然而在对场景中 DRoPipe 的计算时间损失更小, 因此可达到更低的训练完工时间。同样, 在部分场景中, DRoPipe 的计算损失高于其他部分并行策略, 但 DRoPipe 的气泡时间更低, 因此亦达到更低完工时间: 如 12 个节点中训练 Llama2-70B, DRoPipe 总计算时间损失为 54 553.60 s (节点平均损失为 4546.13 s), 高于 SPUD 的 49 230.41 (平均损失为 4102.53 s); 而 DRoPipe 气泡时间仅为 861.47 s, 远

低于 SPUD 的 2458.78 s; 此时 DRoPipe 总完工时间为 $4546.13 + 861.47 = 5407.60$ s, 明显低于 SPUD 的 $4102.53 + 2458.78 = 6561.31$ s。结合图 12、图 13 和图 14 可以观察到, DRoPipe 并行训练速度提升在于同时改进了有效计算时间损失和气泡时间。这是因为, DRoPipe 引入轮询流水线并行和双维度数据并行, 使得相同并行方案下的峰值显存需求降低。因此, 在相同算力节点中, DRoPipe 可以选择具有更低计算时间损失的流水线划分和数据划分方案; 单节点中部署多个离散流水线阶段, 使得在执行过程中可以更早地调用后续算力节点, 降低气泡时间。

6.5 消融实验与可视化分析

为了观察不同算力节点数下 DRoPipe 的并行训练时间关于流水线轮询层数的变化趋势,如

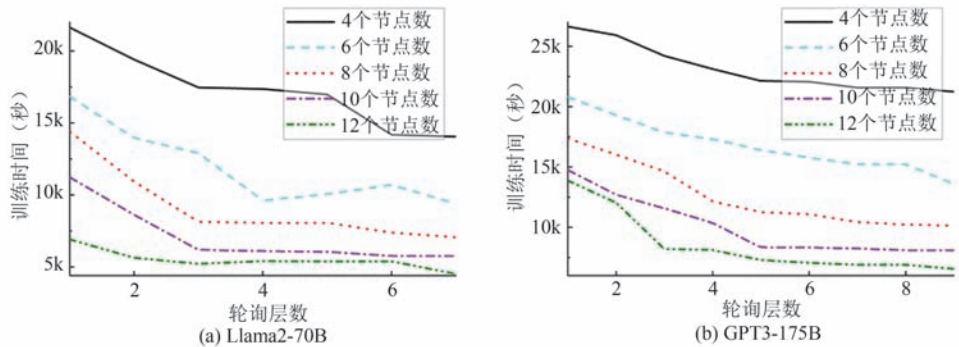


图 15 DRoPipe 并行训练时间关于轮询层数曲线图

从图 15 趋势可以观察到,随着轮询层数的增加,DRoPipe 并行策略的训练所需时间不断减少,即速度不断增加,且其中曲线变化趋势最大的部分出现在从 1 层轮询层(即单层流水线并行模式)到 2 层轮询层处;随着轮询层数不断增加,训练时间的下降趋势减缓。这与第 3 节的理论分析是一致的,说明轮询流水线模式的引入对于深度学习模型分布式并行训练有着显著的影响。对图 15 中不同曲线之间进行对比,可以观察到随着算力节点数增加,其训练时间对应的曲线越低,即增加算力节点数可以提升分布式并行整体训练速度。另外,对于 Llama2-70B,当在 6 个算力节点中采用 2 层流水线轮询层时,其训练时间为 13 970.1 s,低于在 8 个节点中采用单层流水线并行模式的训练时间(14 398.6 s);对于 GPT3-175B,在 6 个算力节点中采用 4 层流水线轮询层的训练时间为 17 311.7 s,低于在 8 个节点中采用单层流水线并行模式的训练时间(17 360.6 s)。上述不同算力节点的对比显示:少量算力节点数通过增加流水线轮询层的方式亦可以达到甚至超过更多算力节点数单层流水线的性能。这说明了轮询流水线并行可以在一定程度上克服算力节点数对于并行训练整体性能的限制。

为了验证上述实验分析,在训练速度、能耗系数、气泡时间等性能指标的基础上,还需观察峰值显存的对比情况。选取图 12 中 Llama2-13B 在 8 个节点中的实验,记录 DRoPipe 和 SPM 的优化并行方案下训练一个小批次过程中显存实时占用情况,如图 16 所示。从图 16 可以观察到,DRoPipe 所需峰值显存明显低于其它并行训练策略,验证了图 13 和图 14 实验中对于 DRoPipe 综合性能的分析。

图 15 所示。以轮询层数作为横坐标,绘制 DRoPipe 策略下训练 Llama2-70B 和 GPT3-175B 的完工时间曲线图。

从图 16 对比可以看出,在显存方面,轮询流水线并行和双维度数据并行均带来一定的收益。一方面,轮询流水线并行允许同一算力节点中部署非连续的神经网络层,使得单个算力节点中正向传播和反向传播的单个运算的神经网络层数降低,因此产生的实时显存占用降低;另一方面,双维度流水线并行,允许并行训练灵活地选择局部子数据块进行计算,有效降低了数据在单个维度中的长度,增加了对于实时产生的数据长度的可控性,从而可以根据算力设备显存容量灵活配置子数据块大小。

为了从详细执行过程可视化观察 DRoPipe 与其他并行策略的区别,图 17 绘制了 DRoPipe、MPD、SPDD、SPD、SPD+1F1B、SPUD 在 10 个算力节点中训练 Llama2-70B 一个小批次的时历甘特图。

从图 17 可观察到,DRoPipe 的优势在于将单个算力节点中的神经网络层离散化,更快地调用后续算力节点,降低算力集群的气泡时间,从而加速整个训练过程。对于训练 Llama2-70B(1 层 Embedding 层+80 层 Transformer 层+1 层 RMSNorm 层+1 层线性层)的反向传播过程进行具体说明:SPD、SPD+1F1B、SPUD 均基于单层流水线并行,平均每个节点大约需要部署 8 层连续的神经网络,因此反向传播的第一个流水线过程(即 G-10 中反向传播过程)需要对大约 8 层 Transformer 层进行梯度下降后,才能开启 G-9 及其后续反向传播过程;而 DRoPipe 基于轮询流水线并行,允许各算力节点中部署离散的神经网络层,在图 17 中 4 层轮询层的实验中,平均每个流水线阶段大约包含 2 层 Transformer 层,因此 G-10 仅需对大约 2 层 Transformer 层进行反向传播即可开启 G-9 及其后

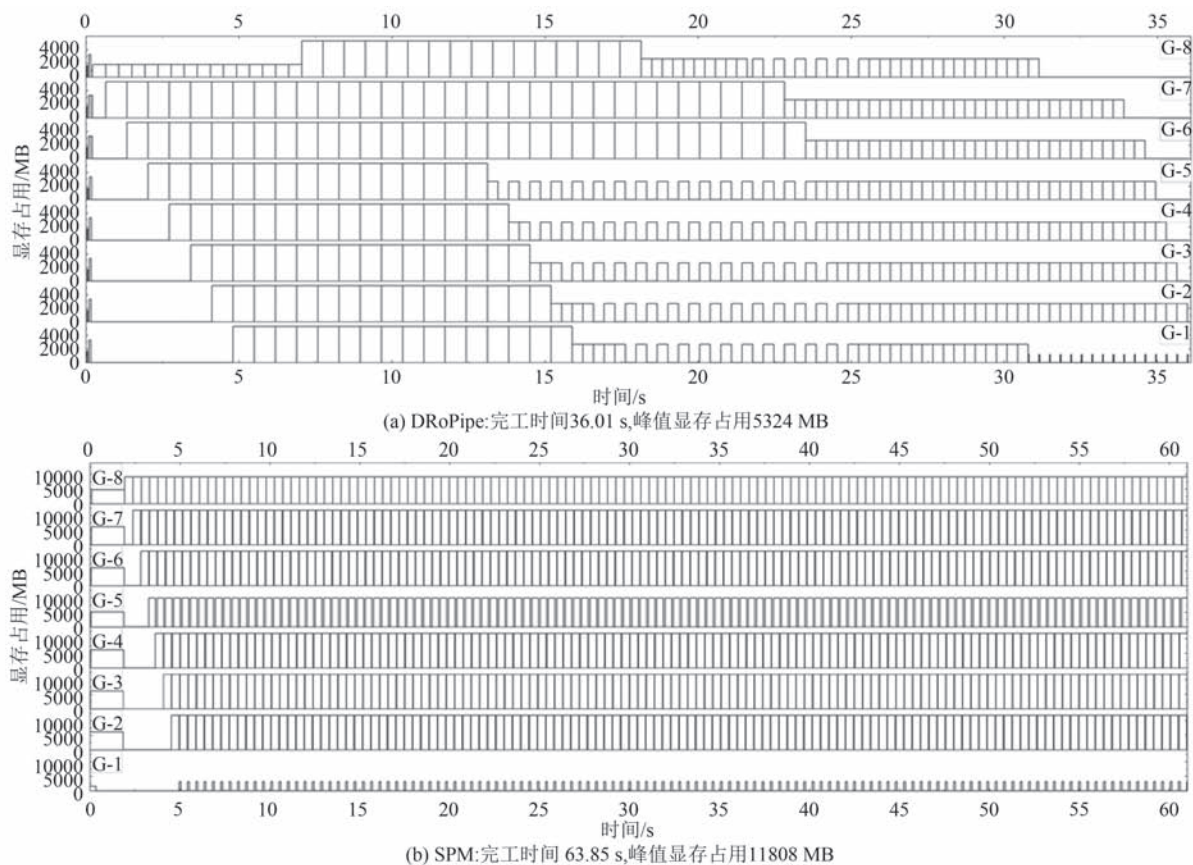


图16 不同并行训练策略训练Llama2-13B一个小批次的显存占用时历变化台阶图(G- i 表示第 i 个GPU)

序的过程。另外,从甘特图可以观察到,正向传播过程耗时明显小于反向传播过程,此时1F1B时序调度策略(更适用于正向反向耗时较为接近的场景)的效果并不明显。理论上DRoPipe的算力设备计算过程的气泡时间(仅考虑GPU计算过程)是SPD的1/4左右。对图17进行统计,在算力集群环境中,DRoPipe并行策略下各节点平均气泡时间为12.20 s,分别是MPD(10.37 s)、SPDD(14.94 s)、SPD(51.53 s)、SPD+1F1B(39.52 s)、SPUD(30.71 s)的117.65%、81.61%、23.68%、30.87%、39.73%,验证了DRoPipe对于降低算力节点气泡时间的先进性,且相比于现有1F1B和非均等数据划分策略,对于正向反向传播耗时不均衡的场景,轮询流水线并行的适应性更好。

对比图17(c)与图17(d),在使用单维度数据划分单层流水线并行的SPD策略(计算时间损失75.60 s)的基础上,引入双维度数据划分的SPDD策略(计算时间损失92.08 s)降低了20.11 s完工时间、气泡时间36.59 s,但增加了16.48 s计算时间损失;对比图17(a)与图17(b),在使用了多层流水线并行模式的MPD的基础上,引入双维度数据划分

的DRoPipe降低了5.19 s完工时间,但增加了12.20 s气泡时间。上述两组对比说明双维度数据划分的主要作用表现在能够稳定降低总训练时间,而对于计算时间损失和空闲气泡时间的优化效果依赖于原有基础策略的性能表现,其原因在于:在SPD中使用了单层流水线并行,导致显存占用需求较高,此时可选的数据划分方案有限,因此引入双维度数据划分后以降低完工时间作为首要优化目标,优化算法将搜索得到计算时间损失更高的数据划分方案;而在MPD中使用了多层流水线并行,降低了并行训练过程的峰值显存,因此可选的双维度数据划分方案更多,使得自动搜索算法在加速训练的同时可以找到计算时间损失更小的并行方案。

图17在一定程度上反映了双维度数据划分和多层流水线并行在优化并行训练速度和气泡时间方面的优势。为了量化分析,表6、表7、表8列出图12中所有实验组SPD、SPDD、MPD、DRoPipe等并行策略在训练速度、气泡时间、计算时间损失(能耗系数)等方面两两对比的平均改进幅度。

双维度数据划分效果:从表6、表7、表8可以观察到,在SPD的基础上引入双维度数据划分,SPDD

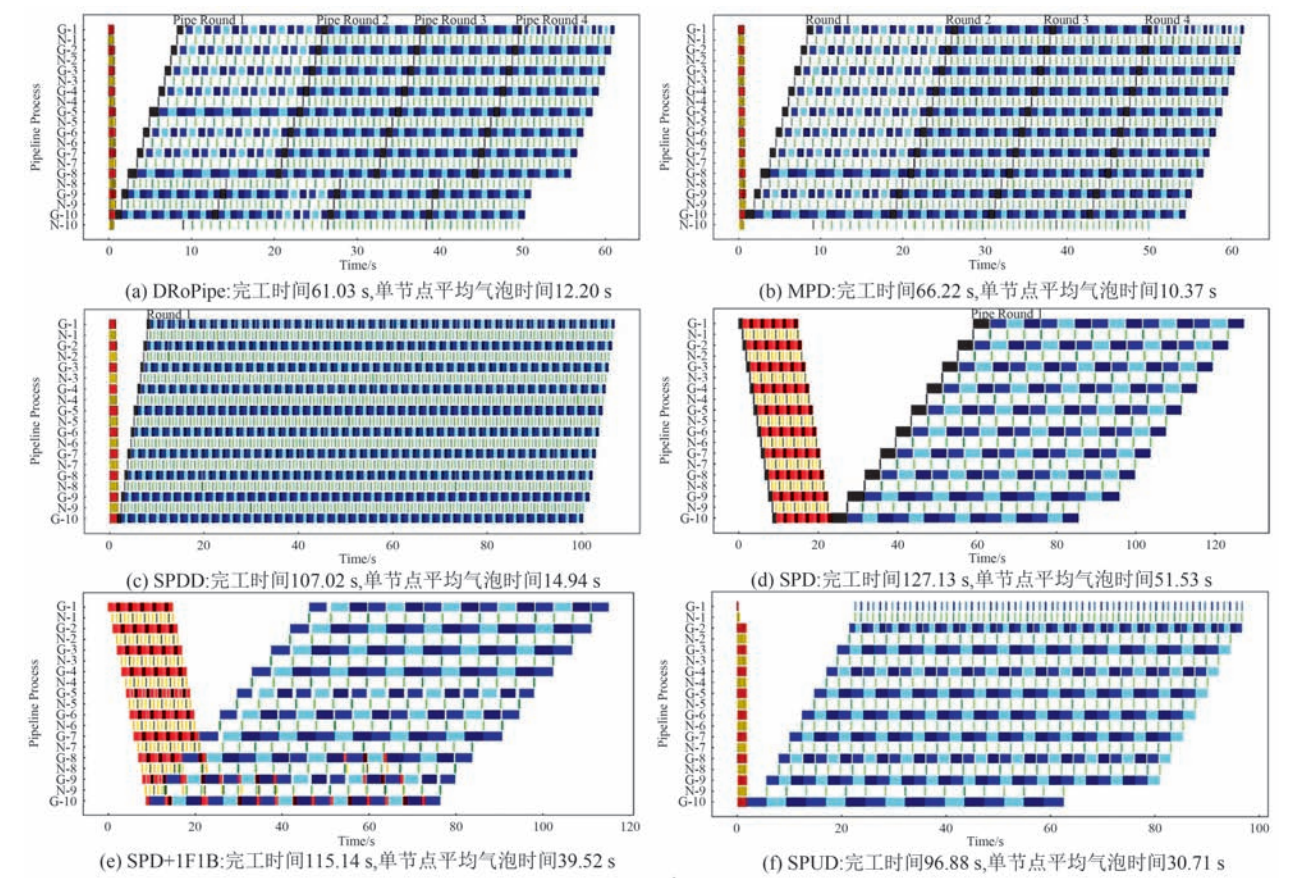


图 17 Llama2-70B 在 10 个 GPU 节点中不同并行策略下训练一个小批次的时历甘特图(G- i 表示 GPU- i 上的计算过程(红色系表示正向传播,蓝色系表示反向传播),N- i 表示节点 GPU- i 与 GPU- $(i+1 \bmod 10)$ 之间的网络通信过程(黄色系表示正向传播,绿色系表示反向传播),黑色矩形块表示各设备上每轮流水线层的首个过程)

表 6 双维度数据划分和多层流水线并行模式在训练速度方面的相对提升比例 $(T_A - T_B)/T_B$					
$B \backslash A$	SPD	MPD	SPDD	DRoPipe	
SPD	-	-0.328	-0.077	-0.362	
MPD	0.671	-	0.527	-0.048	
SPDD	0.093	-0.274	-	-0.311	
DRoPipe	0.752	0.052	0.600	-	

表 7 双维度数据划分和多层流水线并行模式在空闲时间方面的相对下降比例 $(T_A^{(b)} - T_B^{(b)})/T_A^{(b)}$					
$B \backslash A$	SPD	MPD	SPDD	DRoPipe	
SPD	-	-2.013	-0.226	-1.985	
MPD	0.406	-	0.395	0.003	
SPDD	0.058	-1.467	-	-1.427	
DRoPipe	0.396	-0.048	0.385	-	

可提升 9.3% 训练速度、降低 5.8% 气泡时间、降低 1.4% 计算时间损失;在 MPD 的基础上引入双维度数据划分,DRoPipe 可提升 5.2% 训练速度、增加 4.8% 空闲时间、降低 5.6% 计算时间损失。多层流水线并行效果:在 SPD 的基础上引入多层流水线并

表 8 双维度数据划分和多层流水线并行模式在计算时间损失(能耗系数)方面的相对下降比例 $(E_A^{(c)} - E_B^{(c)})/E_A^{(c)}$					
$B \backslash A$	SPD	MPD	SPDD	DRoPipe	
SPD	-	-0.212	-0.001	-0.285	
MPD	0.163	-	0.159	-0.063	
SPDD	0.014	-0.234	-	-0.308	
DRoPipe	0.211	0.056	0.208	-	

行,MPD 可提升 67.1% 训练速度、降低 40.6% 气泡时间、降低 16.3% 计算时间损失;在 SPDD 的基础上引入双维度数据划分,DRoPipe 可提升 60.0% 训练速度、降低 38.5% 空闲时间、降低 20.8% 计算时间损失。虽然双维度数据划分改进幅度较低,但在 MPD 的基础上仍然具备进一步提升效果,对于大模型分布式训练而言,其带来的收益是可观的;另外,双维度数据划分的另一个优势在于为并行训练过程提供更精细化的数据划分,当集群中算力设备和通信能力进一步提升时,可能带来额外的收益(将在环境 2.3.4 中进行进一步实验)。

然而,从图 17 的甘特图可以明显观察到,

DRoPipe需要额外增加流水线阶段间的通信过程。图17中甘特图对应的实验是基于高速通信网络算力集群。为了观察DRoPipe策略可能存在的不足,

本文降低算力集群中通信网络的速率(降低为百兆带宽),进行有效性威胁实验(threats to validity),其训练Llama2-70B的时历甘特图如图18所示。

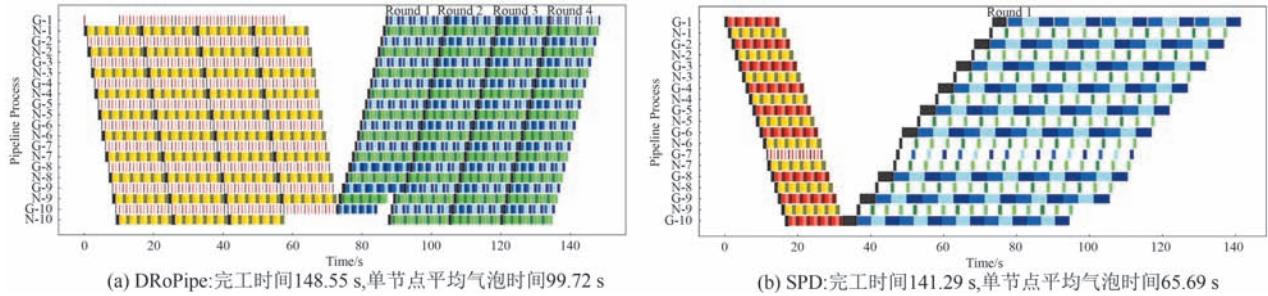


图18 Llama2-70B在低带宽通信状态下的10个GPU节点中训练一个小批次的时历甘特图

从图18可以观察到,当单次通信损失增加后,DRoPipe策略的训练速度低于SPD,其主要耗时增加在于通信过程对于整体时间的影响增大。因此,在实际工程中,应根据集群通信和算力的性能对比来灵活地比选流水线轮询层数,在高速率通信集群中采用轮询流水线并行的方式,或在复杂通信环境中混合使用多层流水线并行和单层流水线并行。

6.6 不同设备集群环境中的实验对比

为了验证DRoPipe对算力集群环境的适应性,本文进一步在多种集群配置中进行并行策略对比实验,相关集群环境配置如表4所示,其中RTX 3090系列显卡较5060 Ti浮点运算速度更高、单卡显存更大,且可支持NVLink等显卡快速通信配

置。实验选取SPD、SPDD、MPD、SPD+1F1B、SPUD作为对比策略,并进一步引入I1F1B(improved interleaved 1F1B,改进交错式1F1B)参与对比实验,其中改进交错式1F1B是针对交错式1F1B进行了改进的时序调度模式,允许任意数据条的正向传播结束后即可开始反向传播过程而无需等待所有数据正向传播结束,且在交错式1F1B的基础上进一步利用先到先服务原则重新优化时序过程。

在环境2(3090显卡同构集群)中对Llama2-13B和GPT2 XL-1.56B进行实验验证,记录不同算力节点数 $n \in [4, 14]$ 下各并行训练策略训练100组 128×128 词条组一次所需训练时间如图19所示,平均每个算力节点的气泡时间如图20所示。

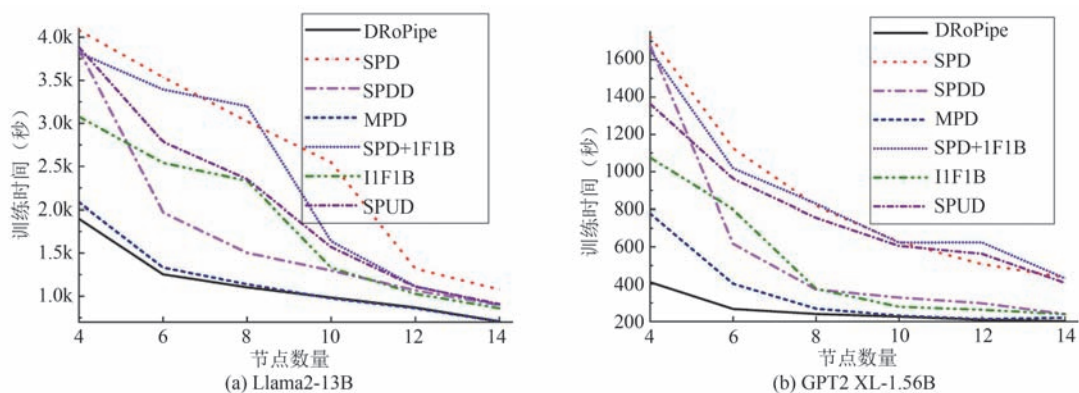


图19 不同分布式并行训练策略在环境2(3090同构集群)中的训练时间对比图

从图19和图20实验结果可以观察到,在其他型号显卡的同构集群中,本文所提出的DRoPipe并行训练策略同样保持着优势,且其对比趋势与图12中RTX 5060 Ti@16GB同构集群中一致。

对比图12(b)与图19(a)中4至14个节点下的实验结果,DRoPipe相比于SPD的平均提速幅度从

53.25%增加到122.58%,说明随着显卡性能的提升,DRoPipe并行训练策略的优势可以得到更好地体现。这可能是因为,DRoPipe提出的多层流水线并行对同一算力节点实现了节点复用,而基础节点性能越好,其带来的收益越大。侧面说明了本文所提DRoPipe策略能有效挖掘算力节点潜在效能。

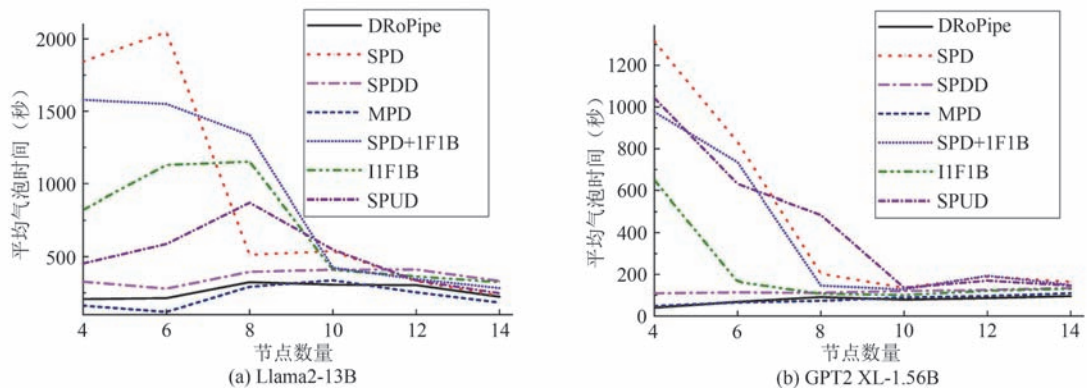


图 20 不同分布式并行训练策略在环境2(3090同构集群)中算力节点平均气泡时间对比图

在环境1和环境2同构集群的实验基础上,进一步对环境3和环境4中的异构集群进行实验。

Llama2-13B在各并行策略下的训练时间如图21所示,平均每个算力节点的气泡时间如图22所示。

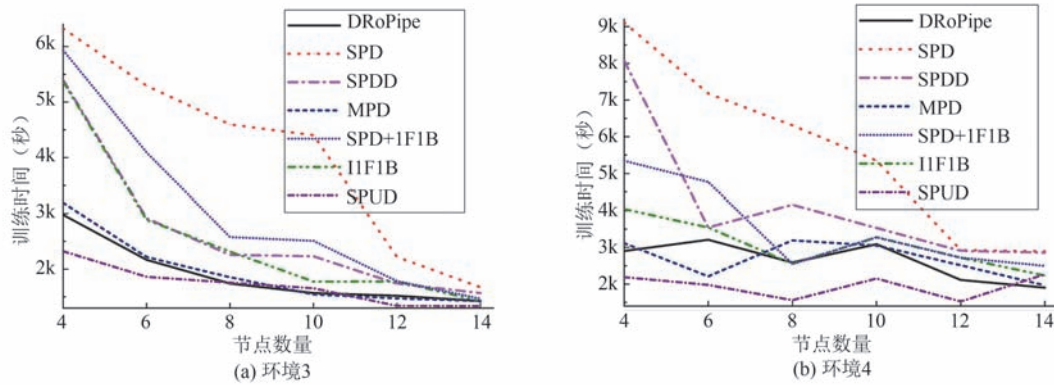


图 21 不同并行训练策略在异构集群环境下训练 Llama2-13B 的训练时间对比图

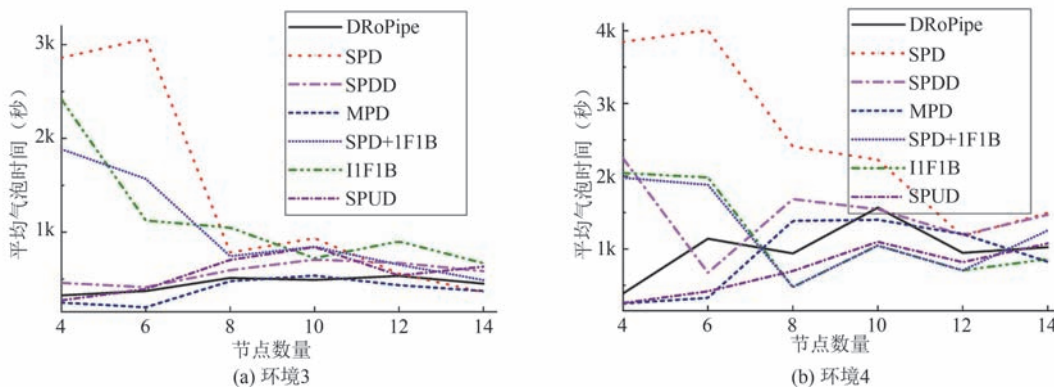


图 22 不同并行训练策略在异构集群环境下训练 Llama2-13B 的算力节点平均气泡时间对比图

图21中各并行训练策略的对比趋势与同构集群一致。本文所提出的DRoPipe并行训练策略在异构集群中仍然保持了最低的训练时间,其次是MPD(轮询流水线并行模式单维度数据划分)、11F1B、SPUD(单层流水线并行模式非均等数据划分),验证了本文所提出的轮询多层流水线并行模式对于不同的算力集群均能体现出较好的适应性。另外,SPUD呈现出与DRoPipe相近的速度且部分场

景中优于DRoPipe,其原因在于SPUD采用非均等数据划分具有更高的数据划分灵活性,可以动态适应异构节点的算力不均衡状态,说明:对于异构集群,DRoPipe仍然具有一定的改进空间。

另外,由于3090较5060 Ti计算速度更快,因此图21中异构集群下的训练速度小于3090同构集群、大于5060 Ti同构集群,说明在异构集群所表现出的整体性能受限于其中各算力设备的性能对比关

系。图 21 实验中集群 3 与集群 4 之间存在性能差异亦启发了对于异构集群,需要进一步考虑更多异构调度和异构并行策略。

为了定量呈现 DRoPipe 的提升效果,表 9 整合集群 1 至集群 4 中实验结果,列出 DRoPipe 相比于其他对比策略在训练速度方面的提升比例。对比现有先进策略 SPD、SPD+1F1B、11F1B、SPUD, DRoPipe 并行策略在并行速度方面分别平均提升了 98.0%、74.9%、43.9%、51.3%;对比两种基于本文轮询流水线并行和双维度数据并行的消融对比策略 SPDD 和 MPD, DRoPipe 分别提速 60.0%、7.5%。

表 9 DRoPipe 相比于其他并行策略在分布式训练速度方面的提升比例。							
集群环境	SPD	SPD+1F1B	11F1B	SPUD	SPDD	MPD	实验组数
环境 1	0.792	0.634	-	0.48	0.6	0.071	54
环境 2	1.711	1.585	0.698	1.335	0.732	0.152	12
环境 3	1.107	0.526	0.192	-0.08	0.353	0.019	6
环境 4	1.077	0.33	0.168	-0.236	0.587	0.035	6
加权平均	0.98	0.749	0.439	0.513	0.6	0.075	78组平均
类别平均	1.172	0.769	0.353	0.375	0.568	0.069	4类平均

7 结 论

分布式并行训练速度的改进有利于促进人工智能和分布式计算等领域的发展,尤其是分布式并行训练结构层级的改进具有更为广泛的影响。

为了提升分布式并行训练的性能,针对现有先进策略中单层流水线并行的局限性,本文提出了轮询多层流水线并行模式。结合轮询流水线并行和双维度数据并行,本文提出了新型多维度分布式并行训练策略 DRoPipe,有利于降低气泡时间、提升并行训练速度。为了解决轮询流水线并行模式带来的多维度并行方案性能表征和联合优化困难等挑战,本文通过理论推导了 DRoPipe 时间损失理论解析式,并在此基础上建立多方案组并行递推快速计算模型 MSPC,以同时计算 DRoPipe 中多组并行方案的时间损失。在时间损失理论计算模型的基础上,本文进一步提出了基于隔板并行邻域搜索的交叉优化算法 CSPBS,以求解 DRoPipe 并行策略中多维度并行模式的联合优化方案。

结合仿真数据驱动实验和真实算力集群实验,本文从多种角度验证了所提出的多方案组并行递

推计算模型 MSPC 的快速性,验证了基于隔板并行邻域搜索的交叉优化算法 CSPBS 的搜索效率和局部收敛解的优化性,验证了 DRoPipe 并行策略相对于现有先进策略在并行速度、显存占用、能耗系数、气泡时间等性能指标方面的综合优势。在本文实验中,相比于单层流水线单维数据划分并行策略 (SPD)、基于 1F1B 时序调度模式的单层流水线并行策略 (SPD+1F1B)、单层流水线非均等数据并行策略 (SPUD) 等现有先进并行训练策略,本文 DRoPipe 并行策略的训练速度分别提升了 98.0%、74.9%、51.3%;相比于两种基于本文所提新型并行模式的消融并行策略,即单层流水线双维数据划分并行策略 (SPDD) 和轮询流水线单维数据划分并行策略 (MPD), DRoPipe 分别提速 60.0%、7.5%;验证了新型并行策略 DRoPipe 的有效性和先进性。

作为在分布式并行训练中对于轮询流水线并行的探索工作,本文形成了针对新型并行策略 DRoPipe 的结构建模、损失模型、优化算法,从而为分布式并行训练提供了新的框架参考和选择。然而,通过有效性威胁实验可以看到,本文 DRoPipe 增加了流水线阶段间的通信过程,对于通信速率较低的集群适应性较差。在未来工作中,将进一步探索超节点或混合算力节点集群中局部多层混合流水线并行的可能性,并将多层流水线并行与非均等数据并行、多交叉流水线并行、1F1B 时序调度相结合以进一步提升并行训练效率。另外,结合多维度并行模式和超节点集群中异构算力资源调度模式,研究并行训练和并行推理过程中并行方案和资源调度方案的联合优化,亦具有实际应用意义。

致 谢 感谢国家自然科学基金项目 (NOs. 62471411, W2433163, U2468207)、四川省科技厅国际创新合作项目 (NO. 2024YFHZ0317)、中央高校基本科研业务费专项资金 (NOs. 2682025CX127, 2682026ZT007) 和华为技术合作项目 (No. R111626H01006) 的资助。

参 考 文 献

[1] Simonyan Karen, Zisserman Andrew. Very deep convolutional networks for large-scale image recognition// Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015). San Diego, USA. 2015

[2] Xue Fanglei, Wang Qiangchang, Guo Guodong. TransFER:

- Learning relation-aware facial expression representations with transformers// Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2021). Montreal, Canada. 2021: 3581-3590
- [3] Zini Julia El, Awad Mariette. On the explainability of natural language processing deep models. *ACM Computing Survey*, 2023, 55(5): 103:1-103:31
- [4] Fu Hao, Tang Shanjiang, He Bingsheng, et al. HGP4CNN: An efficient parallelization framework for training convolutional neural networks on modern GPUs. *Journal of Supercomputing*, 2021, 77(11): 12741-12770
- [5] Narayanan Deepak, Shoeybi Mohammad, Casper Jared, et al. Efficient large-scale language model training on GPU clusters using megatron-LM// Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 21). St. Louis, USA. 2021: 58:1-58:15
- [6] Li Zhuohan, Zhuang Siyuan, Guo Shiyuan, et al. TeraPipe: Token-level pipeline parallelism for training large-scale language models// Proceedings of the 38th International Conference on Machine Learning (ICML 2021). 2021: 6543-6552
- [7] Lee Yunjae, Chung Jinha, Rhu Minsoo. SmartSAGE: Training large-scale graph neural networks using in-storage processing architectures// Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA 22). New York, USA. 2022: 932-945
- [8] Gao Jie, Hu Jia, Min Geyong, et al. XDGNN: Efficient distributed GNN training via explanation-guided subgraph expansion. *IEEE Transactions on Parallel and Distributed Systems*, 2025, 36(11): 2354-2365
- [9] Wang Endong, Yan Ruidong, Guo Zhenhua, et al. A survey of distributed training system and its optimization algorithms. *Chinese Journal of Computers*, 2024, 47(1): 1-28 (in Chinese)
(王恩东, 闫瑞栋, 郭振华, 等. 分布式训练系统及其优化算法综述. *计算机学报*, 2024, 47(1): 1-28)
- [10] Zhang Guipeng, Sun Yuzhong. A survey on hybrid parallelism techniques for large model training. *Chinese Journal of Computers*, 2026, 49(1): 109-131 (in Chinese)
(张贵鹏, 孙毓忠, 大模型训练的混合并行技术综述. *计算机学报*, 49(1): 109-131, 2026)
- [11] Huang Yanping, Cheng Youlong, Bapna Ankur, et al. GPipe: Efficient training of giant neural networks using pipeline parallelism// Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS 2019). Vancouver, Canada. 2019: 103-112
- [12] Li Dongsheng, Li Shengwei, Lai Zhiqian, et al. A memory-efficient hybrid parallel framework for deep neural network training. *IEEE Transactions on Parallel and Distributed Systems*, 2024, 35(4): 577-591
- [13] Zhao Shixiong, Li Fanxin, Chen Xusheng, et al. NASPipe: High performance and reproducible pipeline parallel supernet training via causal synchronous parallelism// Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 22). Lausanne, Switzerland. 2022: 374-387
- [14] Li Fanxin, Zhao Shixiong, Qing Yuhao, et al. Fold3D: Rethinking and parallelizing computational and communicational tasks in the training of large DNN models. *IEEE Transactions on Parallel and Distributed Systems*, 2023, 34(5): 1432-1449
- [15] Fan Shiqing, Rong Yi, Meng Chen, et al. DAPPLE: A pipelined data parallel approach for training large models// Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2021). Republic of Korea. 2021: 431-445
- [16] Zhou Guangyao, Tian Wenhong, Buyya Rajkumar, et al. UMPIPE: Unequal microbatches-based pipeline parallelism for deep neural network training. *IEEE Transactions on Parallel and Distributed Systems*, 2025, 36(2): 293-307
- [17] Ren Lei, Wang Haiteng, Dong Jiabao, et al. Industrial foundation model: architecture, key technologies, and typical applications. *Science China-Information Sciences*, 2025, 54: 2060-2622 (in Chinese)
(任磊, 王海腾, 董家宝, 等. 工业大模型: 体系架构、关键技术与典型应用. *中国科学(信息科学)*, 2025, 54(11): 2060-2622)
- [18] Xu Jinlong, Li Pengfei, Li Jianan, et al. Study on distributed training optimization based on hybrid parallel. *Computer Science*, 2024, 51(12): 120-128 (in Chinese)
(徐金龙, 李鹏飞, 李嘉楠, 等. 基于混合并行的分布式训练优化研究. *计算机科学*, 2024, 51(12): 120-128)
- [19] Lu Yao, Luan Zhongzhi, Li Gen, et al. Multi-level parallel optimization of multi-head attention mechanism for the MT-3000 heterogeneous processor. *Chinese Journal of Computers*, 2025, 48(09): 2049-2063 (in Chinese)
(路瑶, 栾钟治, 李根, 等. 面向迈创3000异构处理器的多头注意力机制多重并行优化. *计算机学报*, 2025, 48(09): 2049-2063)
- [20] Liu Aixin, Feng Bei, Xue Bing, et al. DeepSeek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024
- [21] Yuan Zhengyi, Wang Xiong, Nie Yuntao, et al. DynPipe: Toward dynamic end-to-end pipeline parallelism for interference-aware DNN training. *IEEE Transactions on Parallel and Distributed Systems*, 2025, 36(11): 2366-2382
- [22] Narayanan Deepak, Harlap Aaron, Phanishayee Amar, et al. PipeDream: Generalized pipeline parallelism for DNN training// Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP 2019). Huntsville, Canada. 2019: 1-15
- [23] Rasley Jeff, Rajbhandari Samyam, Ruwase Olatunji, et al. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters// Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. CA, USA. 2020: 3505-3506
- [24] Wang Zixuan, Sim Joonseop, Lim Euicheol, et al. Enabling efficient large-scale deep learning training with cache coherent disaggregated memory systems// IEEE International Symposium on High-Performance Computer Architecture (HPCA 2022). Seoul, Republic of Korea. 2022: 126-140
- [25] Zheng Lianmin, Li Zhuohan, Zhang Hao, et al. Alpa:

- Automating inter-and intra-operator parallelism for distributed deep learning// Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2022). Carlsbad, USA. 2022: 559-578
- [26] Lai Zhiqian, Li Shengwei, Tang Xudong, et al. Merak: An efficient distributed DNN training framework with automated 3d parallelism for giant foundation models. *IEEE Transactions on Parallel and Distributed Systems*, 2023, 34(5): 1466-1478
- [27] Ouyang Shuo, Dong Dezun, Xu Yemao, et al. Communication optimization strategies for distributed deep neural network training: A survey. *Journal of Parallel and Distributed Computing*, 2021, 149: 52-65
- [28] Sun Zhenbo, Cao Huanqi, Wang Yuanwei, et al. AdaPipe: Optimizing pipeline parallelism with adaptive recomputation and partitioning// Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2024). La Jolla, USA. 2024: 86-100
- [29] Zou Xiaofeng, Chen Cen, Lin Peiying, et al. Scalable heterogeneous scheduling based model parallelism for real-time inference of large-scale deep neural networks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024, 8(4): 2962-2973
- [30] Zeng Zihao, Liu Chubo, Tang Zhuo, et al. Training acceleration for deep neural networks: A hybrid parallelization strategy// Proceedings of the 58 th ACM/IEEE Design Automation Conference (DAC 2021). San Francisco, USA 2021: 1165-1170
- [31] Elango Venmugil. Pase: Parallelization strategies for efficient DNN training// Proceedings of the 35 th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2021). Portland, USA. 2021: 1025-1034
- [32] Zhang Jinghui, Niu Geng, Dai Qiangsheng, et al. PipePar: Enabling fast DNN pipeline parallel training in heterogeneous GPU clusters. *Neurocomputing*, 2023, 555: 126661
- [33] Zhang Zechao, Chen Jianfeng, Hu Bing. The optimization of model parallelization strategies for multi-GPU training// Proceedings of the IEEE Global Communications Conference (GLOBECOM 2021). Madrid, Spain. 2021: 1-6
- [34] Li Yajie, Zeng Zebin, Li Jun, et al. Distributed model training based on data parallelism in edge computing-enabled elastic optical networks. *IEEE Communications Letters*, 2021, 25(4): 1241-1244
- [35] Gao Wei, Ouyang Zhuoyuan, Sun Peng, et al. IceFrog: A layer-elastic scheduling system for deep learning training in GPU clusters. *IEEE Transactions on Parallel and Distributed Systems*, 2025, 36(6): 1071-1086
- [36] Lin Changyao, Chen Zhenming, Zhang Ziyang, et al. TOP: Task-based operator parallelism for asynchronous deep learning inference on GPU. *IEEE Transactions on Parallel and Distributed Systems*, 2025, 36(2): 266-281
- [37] Zhou Guangyao, Fu Yiqin, Lan Haocheng, et al. Cross-search with improved multi-dimensional dichotomy-based joint optimization for distributed parallel training of DNN. *IEEE Transactions on Parallel and Distributed Systems*, 2025, 36(7): 1680-1694
- [38] Guo Zhenhua, Tang Yinan, Zhai Jidong, et al. A survey on performance modeling and prediction for distributed DNN training. *IEEE Transactions on Parallel and Distributed Systems*, 2024, 35(12): 2463-2478
- [39] Zhao Shixiong, Li Fanxin, Chen Xusheng, et al. vPipe: A virtualized acceleration system for achieving efficient and scalable pipeline parallel DNN training. *IEEE Transactions on Parallel and Distributed Systems*, 2022, 33(3): 489-506



ZHOU Guang-Yao, Ph. D., lecturer. His research interests include high performance distributed parallel computing, the theory of optimization algorithm and the theory of deep learning

TIAN Wen-Hong, Ph. D., professor. His main research interests include Cloud computing, optimization and resource scheduling of AI large model training and inference.

Background

The increasing need for large-scale deep neural networks has made parallel training a hotspot. To improve the efficiency of distributed parallel training, researchers have established various parallel strategies based on data parallelism, tensor model

ZHANG Jia-Shu, Ph. D., professor. His main research interests include distributed and parallel signal processing, theory and application of artificial intelligence, and communication security.

XUE Rui-Ni, Ph. D., associate professor. His main research interests include distributed system and high performance computing.

LI Tian-Rui, Ph. D., professor. His main research interests include cloud computing, big data, artificial intelligence and granular computing.

parallelism, and pipeline model parallelism.

Although existing parallel training strategies have achieved good parallel efficiency, there are still some gaps that urgently need to be addressed. In the existing pipeline parallelism, all

pipeline stages need to independently occupy one or a group of nodes, without considering node reuse in scenarios with limited computing-power-nodes, which means the number of pipeline stages cannot exceed the number of nodes. Therefore, when the number of nodes is small, multiple continuous neural network layers need to be deployed on a computing-power-node, resulting in more pipeline bubble time and even out-of-memory. In addition, the existing data parallelism only partitions the data in a single-dimension (in the microbatch dimension or token sequence dimension), without considering multi-dimensional data partitioning. The above properties constraint the lower limit of video memory and bandwidth usage for one time of computation or communication process.

In order to further improve the performance of parallel training for large-scale deep neural networks, this paper Proposed a novel pipeline parallelism, i. e. , round robin multi-layer pipeline parallelism. Round robin multi-layer pipeline parallelism allows multiple pipeline stages to be deployed on the same computing- power-node, achieving node reuse and accelerating the parallel training. In theory, even in the case with limited

number of computing-power-nodes, the round robin multi-layer pipeline parallelism allows the number of pipeline stages to reach the number of neural network layers. In addition, we design a dual dimensional data parallelism. This paper also proposes the corresponding fast computing model and optimization algorithm, establishing a comprehensive system of the novel parallel training strategy. This work is the first work to combine round robin multi-layer pipelines with parallel training of deep neural networks, providing a new paradigm for future research on multi-dimensional parallel training strategies. This work is the innovation on the basic parallelism, hence belonging to the improvement on the architecture hierarchy.

This work is supported by the National Natural Science Foundation of China (NOs. 62471411, W2433163, U2468207), International Innovation Cooperation Project of Sichuan Provincial Department of Science and Technology (NO. 2024YFHZ0317), the Fundamental Research Funds for the Central Universities (NOs. 2682025CX127, 2682026ZT007) and the Technical Cooperation Project of Huawei (No. R111626H01006).